A method for incorporating knowledge and communication into decentralized discrete-event systems*

S.L.Ricker
CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

K. Rudie
Department of Electrical
and Computer Engineering
Queen's University
Kingston, Ontario K7L 3N6
Canada

E-Mail: S.L.Ricker@cwi.nl, rudie@ee.queensu.ca

July 2000

Abstract

Using a formal method for reasoning about knowledge, decentralized discrete-event control problems are described. States of the system where supervisors do not have sufficient knowledge to make the correct control decision can be detected. A solution where supervisors may communicate is presented. The relationship between communication and control is complex because (i) control decisions made by one supervisor may depend on communications received from another supervisor and yet (ii) the content of the information that is communicated could be affected by information the communicating supervisor previously received. Procedures are derived for incorporating communication into decentralized discrete-event control. The resulting augmented system gives the supervisors enough knowledge to make the correct control decisions. These procedures yield a control solution while ensuring that supervisors communicate in a consistent manner. Further, these procedures yield a minimal set of communications.

Keywords: Discrete-event systems, supervisory control, modal logic, formal reasoning about knowledge, automata.

1 Introduction

Recent work has explored the relationship between control and communication in decentralized discrete-event systems [1, 7, 15, 16, 19, 21, 22]. The basic idea of this class of problems is that no single agent in a multi-agent system has a complete view of system behavior and a given cooperative task cannot be performed without the agents sending or receiving information that will allow agents to make correct control decisions.

The framework for decentralized discrete-event control that we adopt for this paper is based on the theory of formal languages [13]. In this context, a discrete-event system (DES) is modeled

*This work is partially supported by an ERCIM fellowship for the first author and NSERC Grant OGP0138887 for the second author.

as a generator of a formal language. Decentralized supervisors, based on their partial view of the system and a specified subset of desirable behavior, must determine which sequences in the language should be recognized. Control decisions consist of either allowing an event in a sequence to occur—*enabling* an event—or preventing the event from taking place—*disabling* an event.

Until recently in this theory, decentralized control decisions were based solely on what each supervisor has observed. In such a situation, a control strategy can be formulated as long as there is at least one supervisor that *knows* the correct control decision to make. It is clear when a control solution for a decentralized DES cannot be achieved. What is less clear is the way in which the failure to reach a solution leads to (i) identifying locations where supervisors could communicate; (ii) establishing which supervisor should communicate; and (iii) determining what should be communicated to realize a control solution.

Previously [15] we introduced a knowledge model for discrete-event control whereby supervisors formally reason about the knowledge each requires to achieve a control solution. We used the knowledge model to identify locations where supervisors did not have enough knowledge to make the correct control decision. Here we use the knowledge framework to reconstruct sequences along which supervisors could communicate. A location where communication occurs is identified by a specific communication event, indicating that supervisor i communicates to supervisor j. When a communication event occurs, we assume that a communicating supervisor i sends its partial view of the system to supervisor j. Finally, we continue to use the knowledge model to determine whether or not a control solution exists once communication is incorporated into the description of system behavior.

The relationship between communication and control in decentralized discrete-event systems is complex and co-dependent. On the one hand, control decisions may be affected by information a supervisor receives from another supervisor. On the other hand, the content of the information that is communicated may be affected by information the communicating supervisor has previously received.

A solution to this new class of decentralized discrete-event control problems not only ensures that correct control decisions are made. In addition, since a supervisor has only a partial view of the system, if it communicates at a particular location in the system, it must also communicate at every location it cannot distinguish from that location. This property of supervisor behavior is called *consistency* [16]. In addition, communication may be costly, so we want to eliminate any unnecessary communications without violating consistency yet still ensure that enough information is available to a supervisor making a control decision. A set of communications is called *minimal* if it (a) satisfies consistency, (b) provides enough information for supervisors to solve the control problem and (c) no subset of it satisfies (a) and (b) [16].

Our work on decentralized control and communication uses the concept of minimality from [16] and our motivation for communication is similar to the idea of avoiding the conflict states in [1]. One significant difference in our approach is that our supervisors do not exchange observations: a two-way broadcast would occur only if neither supervisor had sufficient knowledge to make the correct control decision and each needed the information from the other to reach a control solution. In addition, we represent the action of communication as an event in the DES and incorporate these new events into the system.

We begin by summarizing some of the notation from the theory of DESs and from the theory of formal reasoning about knowledge. In section 3 we introduce a new version of the knowledge model first proposed for DES in [15]. Sections 4 and 5 explain our strategy for identifying locations where supervisors could communicate to solve the control problem and satisfy consistency. In section 6 we discuss what it means for our communication protocols to be "well-defined". Once the system model

has been updated to include communication events, section 7 describes how to use the knowledge model to determine whether or not a control solution exists. Algorithms for achieving a minimal set of communication are presented in section 9. We illustrate our strategy for communication in decentralized supervisory control and for generating a set of minimal communications with an example.

2 Review

We present a summary of some relevant concepts from discrete-event control theory and a theory for formal reasoning about knowledge. References that could be consulted for further clarification of possible notational conflicts between the two fields are also noted.

2.1 Discrete-Event Systems

As stated previously, we use the approach to discrete-event systems as developed by Ramadge and Wonham[13]. Other references on discrete-event control theory include [3, 14, 20].

In the discrete-event control theory of Ramadge and Wonham [13], the system requiring control (the *plant*) is described as a generator of a formal language (i.e., an automaton). The behavior of 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 modeled 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 $\delta^G: \Sigma \times Q^G \to Q^G$; and $q_0^G \in Q^G$ is the *initial state*. For any event $\sigma \in \Sigma$ and state $q^G \in Q^G$, if $\delta^G(\sigma, q^G)$ is defined (i.e., there is some state in the plant that we can reach from q^G via event σ), we write $\delta^G(\sigma, q^G)!$. 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:

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

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

For any strings $t, u \in \Sigma^*$, we say that u is a *prefix* of t if $\exists v$ such that $t{=}uv$. Thus every string $t \in \Sigma^*$ (where $t \neq \varepsilon$) has at least two prefixes: ε and t. If $L \subseteq \Sigma^*$, the *prefix-closure* of L is a language, denoted by $\overline{L}$, consisting of all prefixes of strings of L: $\overline{L} := \{u \in \Sigma^* \mid u \text{ is a prefix of } t\}$. Because every string is a prefix of itself, $L \subseteq \overline{L}$. A language is said to be *prefix-closed* if $L = \overline{L}$. By definition, $L(G)$ is prefix-closed.

We assume that the legal behavior of the plant may be described by an automaton $E = (Q^E, \Sigma, \delta^E, q_0^E)$ and the legal language is denoted $L(E)$. We assume that E is a subautomaton

of G as described in the context of supervisory control in [4] and [11]. That is, $Q^E \subseteq Q^G$, $q_0^E = q_0^G$ and $\delta^E(t, q_0^G) = \delta^G(t, q_0^G)$ for all $t \in L(E)$.

When Q^G is finite, the automaton G can be described as a finite-state automaton and can be represented by a directed graph, where the nodes of the graph are the states in Q^G, the arcs of the graph are the transitions defined by the partial function δ^G, and the set of labels for the arcs are the events in Σ. Thus for any event $\sigma \in \Sigma$ and state $q \in Q^G$, $\delta^G(\sigma, q)!$ if there is an arc labeled by σ from q to some other state. In the diagrams in this paper, the initial state is marked with a small entry arrow. Illegal transitions are indicated with a dashed line.

Informally, a supervisor is an agent that has the ability to control some events based on a (partial) view of the plant's behavior. To establish such supervision on G, we partition the set of events Σ into the 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 which cannot be prevented and are deemed permanently enabled. There are some systems where not all events can be seen by the supervisor. A supervisor thus has only a partial view of the system and can see only a subset of events in Σ. The set of *observable* events visible to a supervisor is denoted Σ_o. To describe a supervisor's view of sequences we use the *canonical projection* P, which is a mapping from Σ^* to Σ_o^*. This operator effectively "erases" those events σ from a string t that are not found in the set of observable events Σ_o:

$$\begin{aligned} P(\varepsilon) &= \varepsilon \\ P(\sigma) &= \varepsilon, \quad \sigma \in \Sigma \setminus \Sigma_o \\ P(\sigma) &= \sigma, \quad \sigma \in \Sigma_o \\ P(t\sigma) &= P(t)P(\sigma), \quad t \in \Sigma^*, \sigma \in \Sigma. \end{aligned} \tag{1}$$

Thus if the plant generates sequence t, then $P(t)$ indicates the sequence of events observed by the centralized supervisor. The *inverse projection* of P is the mapping from Σ_o^* to 2^{Σ^*}:

$$P^{-1}(t) = \{u \mid P(u) = t\}.$$

A prefix-closed language L is *observable* [12] with respect to G, P if

$$\begin{aligned} &(\forall t, t' \in \Sigma^*)(\forall \sigma \in \Sigma) \\ &\quad P(t) = P(t') \Rightarrow (t'\sigma \in L \wedge t \in L \wedge t\sigma \in L(G) \Rightarrow t\sigma \in L). \end{aligned} \tag{2}$$

This condition indicates that an observer's view of a string in $L(G)$ is sufficient to determine whether or not σ should be disabled.

The decentralized control problem arises when more than one supervisor is involved in coordinating control actions [5] [18]. 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 $i = 1, 2$. As with the centralized version of projection, if the plant generates sequence t, $P_i(t)$ indicates the sequence of events observed by supervisor i.

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 supervisor i has of the plant, we use an *observer automaton* [3], based on an algorithm in [10] to translate a nondeterministic finite-state automaton into a deterministic finite-state automaton:

$$G^{obs_i} = (Q^{obs_i}, \Sigma_{i,o}, \delta^{obs_i}, q_0^{obs_i}),$$

where $Q^{obs_i} = 2^{Q^G}$ is the set of states, $\Sigma_{i,o} \subseteq \Sigma$ is the set of events *observable* to supervisor i (and the set of events unobservable to supervisor i is $\Sigma_{i,uo}$). The transition function δ^{obs_i} and initial state $q_0^{obs_i}$ are defined as follows:

$$\begin{aligned} q_0^{obs_i} &:= \{q_k^G \mid \delta^G(t, q_0^G) = q_k^G \text{ and } t \in (\Sigma \setminus \Sigma_{i,uo})^*\}; \\ \delta^{obs_i}(\sigma_i, q_j^{obs_i}) &:= \{q_h^G \mid \delta^G(\sigma_i t, q_{h'}^G) = q_h^G \ , \ \sigma_i \in \Sigma_{i,o}, \ t \in (\Sigma \setminus \Sigma_{i,uo})^* \text{ and } q_{h'}^G \in q_j^{obs_i}\}. \end{aligned}$$

The initial state $q_0^{obs_i}$ of the automaton captures all the states reachable by unobservable events (to supervisor i) from the initial state of the plant. Subsequent states in the projection automaton are generated by considering which states can be reached next via an observable event $\sigma \in \Sigma_{i,o}$ from the current state. The resulting set of states includes all states reached by unobservable sequences from the state to which the observable event σ leads.

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 projection automaton). Such a structure, which we call the *monitoring automaton* A, is a deterministic version of the nondeterministic automaton M described in [17]. The monitoring automaton is formally defined as follows: $A = (Q^A, \Sigma, \delta^A, q_0^A)$, where $Q^A \subseteq Q^G \times Q^{obs_1} \times Q^{obs_2}$ (Q^A will be fully defined below), the initial state is $q_0^A = (q_0^G, q_0^{obs_1}, q_0^{obs_2})$, and δ^A is defined below. When $\delta^G(\sigma, q^G)$ is defined, we have four cases to consider for the construction of the transition function:

- $\sigma \notin \Sigma_{1,o}$, $\sigma \notin \Sigma_{2,o}$:

$$\delta^A(\sigma, (q^G, q^{obs_1}, q^{obs_2})) = (\delta^G(\sigma, q^G), q^{obs_1}, q^{obs_2}),$$

- $\sigma \in \Sigma_{1,o}$, $\sigma \notin \Sigma_{2,o}$:

$$\delta^A(\sigma, (q^G, q^{obs_1}, q^{obs_2})) = (\delta^G(\sigma, q^G), \delta^{obs_1}(\sigma, q^{obs_1}), q^{obs_2}),$$

- $\sigma \notin \Sigma_{1,o}$, $\sigma \in \Sigma_{2,o}$:

$$\delta^A(\sigma, (q^G, q^{obs_1}, q^{obs_2})) = (\delta^G(\sigma, q^G), q^{obs_1}, \delta^{obs_2}(\sigma, q^{obs_2})),$$

- $\sigma \in \Sigma_{1,o}$, $\sigma \in \Sigma_{2,o}$:

$$\delta^A(\sigma, (q^G, q^{obs_1}, q^{obs_2})) = (\delta^G(\sigma, q^G), \delta^{obs_1}(\sigma, q^{obs_1}), \delta^{obs_2}(\sigma, q^{obs_2})),$$

where $q^G \in Q^G, q^{obs_1} \in Q^{obs_1}, q^{obs_2} \in Q^{obs_2}$. When $\delta^G(\sigma, q^G)$ is not defined, $\delta^A(\sigma, (q^G, q^{obs_1}, q^{obs_2}))$ is also not defined. The set of states Q^A is the set of states in $Q^G \times Q^{obs_1} \times Q^{obs_2}$ reachable from the initial state via the δ^A defined above.

By the way in which A is constructed, we have $L(A) = L(G)$. Note that a state $(q^G, q^{obs_1}, q^{obs_2}) \in Q^A$ is reachable if there exists $t \in L(G)$ such that $\delta^G(t, q_0^G) = q^G$.

2.2 Formal Reasoning about Knowledge

The framework for modeling knowledge that we use is based on a theory of formal reasoning about knowledge for distributed systems [9], where multiple agents reason about their knowledge of the world. An agent could be a human, a machine (e.g., a robot) or even a component of a machine (e.g., an electrical circuit). Unless otherwise indicated, the definitions and results in this section are adopted from [8]. The model assumes that if an agent does not have complete knowledge of the true state of the world, it assumes a number of worlds are possible. Worlds are described in terms of a non-empty set Φ of facts or *primitive propositions*. More complicated formulas are constructed using expressions from propositional calculus: $\neg$ (negation) and $\wedge$ (conjunction). In addition, $\varphi \vee \psi$ represents $\neg(\neg\varphi \wedge \neg\psi)$.

The system model is conceptually divided into two components: the agents and the environment. The latter captures the relevant aspects of the system that are not part of the description of agent behavior. We assume that there is a set of agents $\mathsf{G} = \{1, \ldots, n\}$ to which we ascribe knowledge about the system.

The system behavior is captured by a *global* state. A global state is an (n+1)-tuple, denoted w, that records the state of the environment and the *local* state—an agent's set of possible worlds—for each of the n agents. Formally $w = (w_e, w_1, \ldots, w_n)$. We can further refer to individual components of w: w_e and w_i represent the state of the environment and the local state of agent i (for $i \in \{1, \ldots, n\}$), respectively. We will use the terms "world" and "global state" interchangeably.

We will reason about what an agent knows about the truth of facts in the system at global states. Knowledge of a fact is expressed using modal operators (one for each agent) $K_1, \ldots, K_n$. Thus $K_1 p$, where $p \in \Phi$, is interpreted as "agent 1 knows p".

The semantics of the possible-worlds model is formalized using *Kripke structures*. A Kripke structure is an (n+2)-tuple containing a set of worlds (e.g., global states), an *interpretation* function π that assigns truth values at each world w to the primitive propositions in Φ (e.g., $\pi(w)(p) = \textbf{false}$), and *possibility relations*, one for each agent, that define binary relations on the set of worlds. That is, the relation defines the (set of) worlds that look alike to an agent at any world of the system. For purposes of this discussion, the possibility relation is always an equivalence relation and therefore, it is always the case that reflexivity and symmetry hold. The possibility relation is typically not defined for the environment since we are not interested in ascribing knowledge to the environment.

A Kripke structure is also expressible as a labeled graph. In particular, nodes are worlds and edge labels (sets of agents) capture the possibility relation. For instance, worlds that look alike to agent i are joined by an edge with a label "i". Each world is also labeled with the truth values of all primitive propositions $p \in \Phi$, where we use the notation "$\neg p$" to indicate that the truth value of p is **false** and "p" corresponds a value of **true**.

We now have all the components we need to reason about knowledge: a set of worlds describing the behavior of the system and an interpretation π to analyze truth values of the propositions at states of the system. Together the set of worlds and π define an *interpreted system* and is denoted by $\mathcal{I}$.

To discuss knowledge in an interpreted system, we assume that the possibility relation is defined as follows. Let w, w' be two global states in $\mathcal{I}$. We say w and w' are *indistinguishable to agent i* if the local state according to agent i is the same at both global states:

$$w \sim_i w' \text{ if } w_i = w'_i. \tag{3}$$

To discuss what it means for a fact p to be true at a particular global state in $\mathcal{I}$, we use the notation $(\mathcal{I}, w) \models p$, which can be read as *"p is true at $(\mathcal{I}, w)$"* or *"p holds at $(\mathcal{I}, w)$"*. A fact p holds at a world w if the truth value as defined by π is true at w.

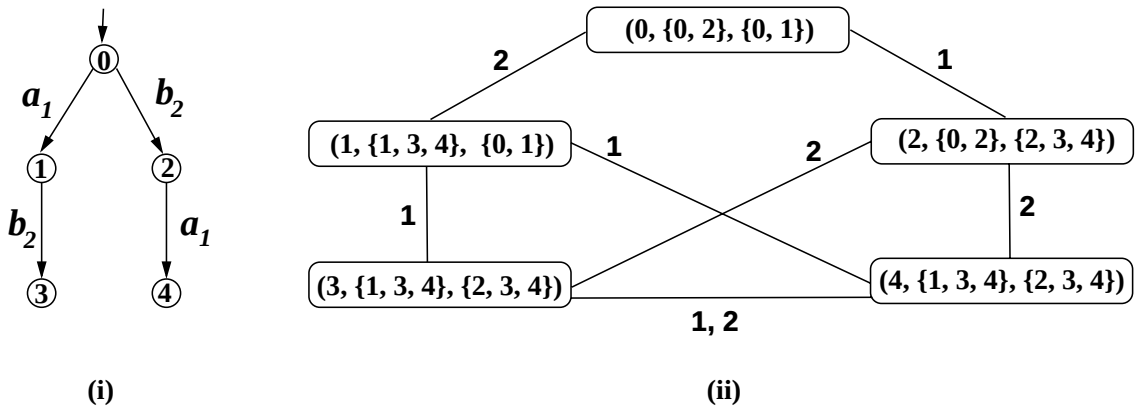

Figure 1: (i) a plant G; (ii) a Kripke structure for the plant in (i).

What does it mean for an agent to know facts in the system? An agent knows a fact p at w if p holds at all worlds that the agent cannot distinguish from w:

$$(\mathcal{I}, w) \models K_i p \text{ iff } (\mathcal{I}, w') \models p \text{ for all } w' \text{ such that } w \sim_i w'. \tag{4}$$

It follows that if an agent knows p at w, it also knows p at all other worlds it considers possible at w:

$$(\mathcal{I}, w) \models K_i p \text{ iff } (\mathcal{I}, w') \models K_i p \tag{5}$$

for all w' such that $w \sim_i w'$.

3 State-Based Knowledge Model

Our previous knowledge model [15] assumed that agents made decisions based on their recorded observations of *event sequences* generated by the plant. If the language requiring control has an infinite number of strings, the associated Kripke structure would have an infinite number of worlds. To exploit the finite representation of a regular language, in this section we introduce a model where agents now monitor the set of *states* the plant could be in.

We construct our "state-based" interpreted system $\mathcal{I}^{DES}$ as follows. The environment component of this interpreted system is the set of plant states Q^G, while the agents are a slight variation of the DES local supervisors $\mathcal{S}_1$ and $\mathcal{S}_2$. In the "sequence-based" system of [15] each supervisor has only a partial view of the complete system behavior, based on the projection operator P_i. Here we present a system where the partial view of the plant is based on the state estimates of the observer automaton from section 2.1.

In [15] we assumed that the worlds of the interpreted system were composed of sequences from the plant language $L(G)$. The worlds in the state-based system describe plant states in Q^G and the respective views of those states for the group of agents G. The global states are constructed according to the strategy for generating states in Q^A of the monitoring automaton A as described in section 2.1. Consequently, a global state w has the form (w_e, w_1, w_2) where $w_e \in Q^G$, $w_1 \in Q^{obs_1}$ and $w_2 \in Q^{obs_2}$. Figure 1 shows a sample DES plant (where $\Sigma_{1,o} = \{a_1\}$ and $\Sigma_{2,o} = \{b\}$) and its corresponding Kripke structure, with global states as described above.

The propositions in our knowledge model are the events in Σ. Each event is translated into two propositions: one corresponding to whether or not the event is defined at a particular state of the

plant, and the other indicating whether the event is defined in the legal automaton. The following definition summarizes some concepts from [15].

Definition 1 *(i) The proposition σ_G is "event σ can occur" and σ_E is "event σ is legal". (ii) A proposition σ_G is* **true** *at a world w if the event σ happens at the true plant state described by w. A proposition σ_G is* **false** *(denoted $\neg\sigma_G$) at a world w if the event σ is not defined at the true plant state described by w. (iii) A proposition σ_E is* **true** *at a world w if the event σ happens at the true plant state described by w and is part of the legal behavior of the plant. A proposition σ_E is* **false** *(denoted $\neg\sigma_E$) at a world w if either $\sigma_G =$* **false** *or if the event σ is part of the illegal behavior of the plant.*

Previously, the interpretation π^{DES} returned a truth value based on whether or not event σ occurs after a sequence $t \in L(G)$. In our new model, the interpretation returns a truth value based on whether σ is defined at the current plant state (i.e., at w_e):

$$\pi^{DES}(w)(\sigma_G) := \begin{cases} \textbf{true} & \text{if } \delta^G(\sigma, w_e)!, \\ \textbf{false} & \text{otherwise.} \end{cases} \tag{6}$$

$$\pi^{DES}(w)(\sigma_E) := \begin{cases} \textbf{true} & \text{if } \delta^E(\sigma, w_e)!, \\ \textbf{false} & \text{otherwise.} \end{cases} \tag{7}$$

Since we assume that the legal automaton can always be expressed as a subautomaton of the plant automaton, the seemingly ambiguous reference to w_e (which is a plant state) in (7) is a consistent reference to the same state in both automata.

For the decentralized DES we consider, when we say that a supervisor $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ solves a control problem, we mean that when G is under the control of $\mathcal{S}$ (i.e., when $\mathcal{S}$ disables or enables events of G), the resultant language generated equals the legal language. When we solve the decentralized problem with a knowledge model we want to construct a *knowledge-based protocol* that will ensure that all legal sequences and only legal sequences are generated. A knowledge-based protocol maps a supervisor's knowledge of event σ at its local state to a control action (e.g., disable, enable). If the system satisfies a condition called *Kripke observability* then such a protocol can be constructed. This is formally stated and proven as Theorem 2 in [15].

If the system is not *Kripke-observable*, then there is some world w' where an illegal event σ is about to occur but none of the agents capable of controlling σ knows that σ should be disabled. The lack of knowledge arises from the existence of another world w where σ is also about to occur but is a legal event and this same set of agents cannot distinguish w from w'. More formally, there exists $w' \in \mathcal{I}$ and a pair of primitive propositions (σ_G, σ_E) where

$$(\mathcal{I}, w') \not\models \neg\sigma_G \vee \sigma_E \tag{8}$$

and for all $i \in \mathsf{G}_\sigma$

$$(\mathcal{I}, w') \not\models K_i \neg\sigma_E. \tag{9}$$

When we describe a knowledge model for a decentralized DES with communication, we assume (by Theorem 2 noted above) that a control solution exists if the system satisfies Kripke observability.

4 Communication for Control

In [15], solving the control problem in our knowledge model amounted to each agent having enough information to make the correct control decisions. We characterized an agent's inability to make such a decision as a state in the interpreted system that contributes to the system not being Kripke-observable. We then speculated about the role "pooling information" might play in providing agents with more information to make the correct control decisions. Our previous work suggested that pooling should take place just before a control decision needed to be made. However, it is possible to come up with examples where a control solution exists but where pooling possible worlds under these conditions does not lead to an agent having the knowledge to make the correct control decision. Such examples indicate that pooling information at the point in time proposed in [15] is too late. Therefore, in our strategy for communication, we identify locations where pooling information *is* helpful.

This section introduces our approach to incorporating communication into decentralized discrete-event control problems. The problem that we are interested in concerns interpreted systems that do not satisfy Kripke-observability. Since an agent bases its actions on the information it has, if an agent does not have "enough" information to know that event σ should be prevented from occurring, under what conditions would information from other agents give that agent the knowledge to make the correct control decision about σ?

In a Kripke structure, a state where Kripke-observability is not satisfied corresponds to a world w where for all $i \in \mathsf{G}_\sigma$ the following holds: $(\mathcal{I}^{DES}, w) \models \sigma_G \wedge \neg\sigma_E \wedge \neg K_i \neg\sigma_E$. That is, supervisor i does not have the knowledge to disable event σ at plant state w_e. This lack of knowledge means that there exists a state w' that is indistinguishable from w to supervisor i where σ is allowed to happen at plant state w'_e, i.e., $(\mathcal{I}^{DES}, w') \models \sigma_G \wedge \sigma_E$.

Thus, using the Kripke structure and the corresponding monitoring automaton we can identify a world and subsequently a state in the plant, say q^G, where without communication a decentralized supervisor might not be able to make the correct control decision about event σ. This is because to agent i, q^G is indistinguishable from state q'^G and, as noted above, σ is allowed to occur at one state and should be disabled at the other state. If communication from supervisor j to i occurs somewhere along the paths to state q^G or along the paths to q'^G, supervisor j could give supervisor i the knowledge to distinguish states where σ must be disabled from other states where σ is enabled. Thus we need to identify those paths along which communication should occur. We use the monitoring automaton to reconstruct such paths.

First of all, we identify all pairs of global states w, w' in the Kripke structure where for some $\sigma \in \Sigma$, the propositions σ_G and $\neg\sigma_E$ are true at w but σ_G and σ_E are true at w'. Suppose that $w_e = y$ and $w'_e = y'$, i.e., y and y' are the plant states associated with global states w and w'. The idea is that we want to insert communication at plant states to distinguish every sequence that leads to y from every sequence that leads to y'. The identification of these sequences is performed with the monitoring automaton A. That is, we want to reconstruct sequences that lead to worlds w and w'. (Recall that the states of the Kripke structure are the reachable states of A.) Since there may be infinitely many sequences leading to w (corresponding to cycles in the directed graph representing the plant), it appears on the face of it that comparing all pairs t, t' that lead to states w, w', respectively, is an intractable task. We conjecture that when there are infinitely many sequences leading to world w, we can exploit the finite-state representation of a Kripke structure and do not need to reconstruct all sequences involving cycles.

We describe our intuition via the example in figures 2 and 3. Suppose that supervisor 1 sees and controls events a and b while supervisor 2 sees events b and c. The states of the Kripke structure for

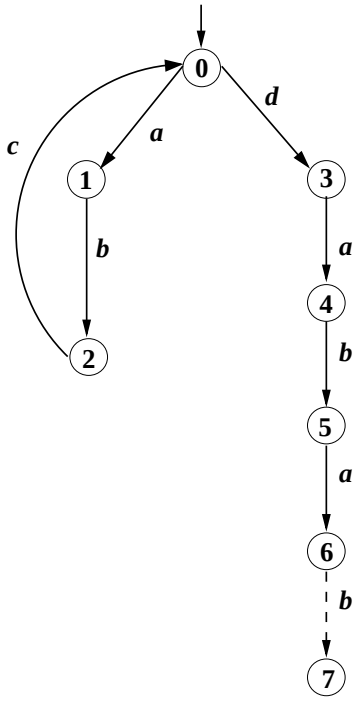

Figure 2: Finding locations to communicate in the presence of cycles.

the plant shown in figure 2 are simply the states of the corresponding monitoring automaton (shown in figure 3). In the associated Kripke structure (not illustrated here) Kripke-observability fails at state $(6, \{1,4,6\}, \{2,5,6\})$ because supervisor 1 does not have enough knowledge to make the correct control decision about event b. At this state the truth values of the primitive propositions associated with event b are $b_G = \textbf{true}$ and $b_E = \textbf{false}$. There are two other states—$(1, \{1,4,6\}, \{0,1,3,4\})$ and $(4, \{1,4,6\}, \{0,1,3,4\})$—that supervisor 1 cannot distinguish from $(6, \{1,4,6\}, \{2,5,6\})$. At both these states, namely $(1, \{1,4,6\}, \{0,1,3,4\})$ and $(4, \{1,4,6\}, \{0,1,3,4\})$, the truth values of the primitive propositions associated with event b are $b_G = \textbf{true}$ and $b_E = \textbf{true}$, thereby giving rise to supervisor 1 not knowing $\neg b_E$ at state $(6, \{1,4,6\}, \{2,5,6\})$.

We use the monitoring automaton of figure 3 to find paths t to state $(6, \{1,4,6\}, \{2,5,6\})$ such that $tb \in L(G)$ but $tb \notin L(E)$. Similarly we want to find paths t' to states $(1, \{1,4,6\}, \{0,1,3,4\})$ and $(4, \{1,4,6\}, \{0,1,3,4\})$ where $t'b \in L(G), L(E)$ and $P_1(t) = P_1(t')$. We conjecture that we need only consider a finite number of these paths.

We begin by looking at state $(6, \{1,4,6\}, \{2,5,6\})$. We suggest that we need only consider the following paths in the monitoring automaton: $t = daba$ or $t = abcdaba$ or $t = abcabcdaba$ or $t = abcabcabcdaba$. These paths either contain no cycles (the first three candidates) or one iteration of a cycle (the last sequence). We suggest that these sequences capture sufficient detail of the family of regular expressions that lead to the states where Kripke-observability fails. For example, since Kripke-observability fails at state $(6, \{1,4,6\}, \{2,5,6\})$ it does not matter whether it is reached via $t = abcabcabcdaba$ or by some other $t \in abc(abc)^*daba$. We can find corresponding t' for each such t where $P_i(t) = P_i(t')$ as described above. Our strategy for introducing communication events transforms t and t' such that the projections of these transformed sequences are no longer identical. We conjecture that by transforming $t = abcabcabcdaba$ we also inadvertently transform any t in $abc(abc)^*daba$ and therefore once the former sequence is changed, there is no need to check the

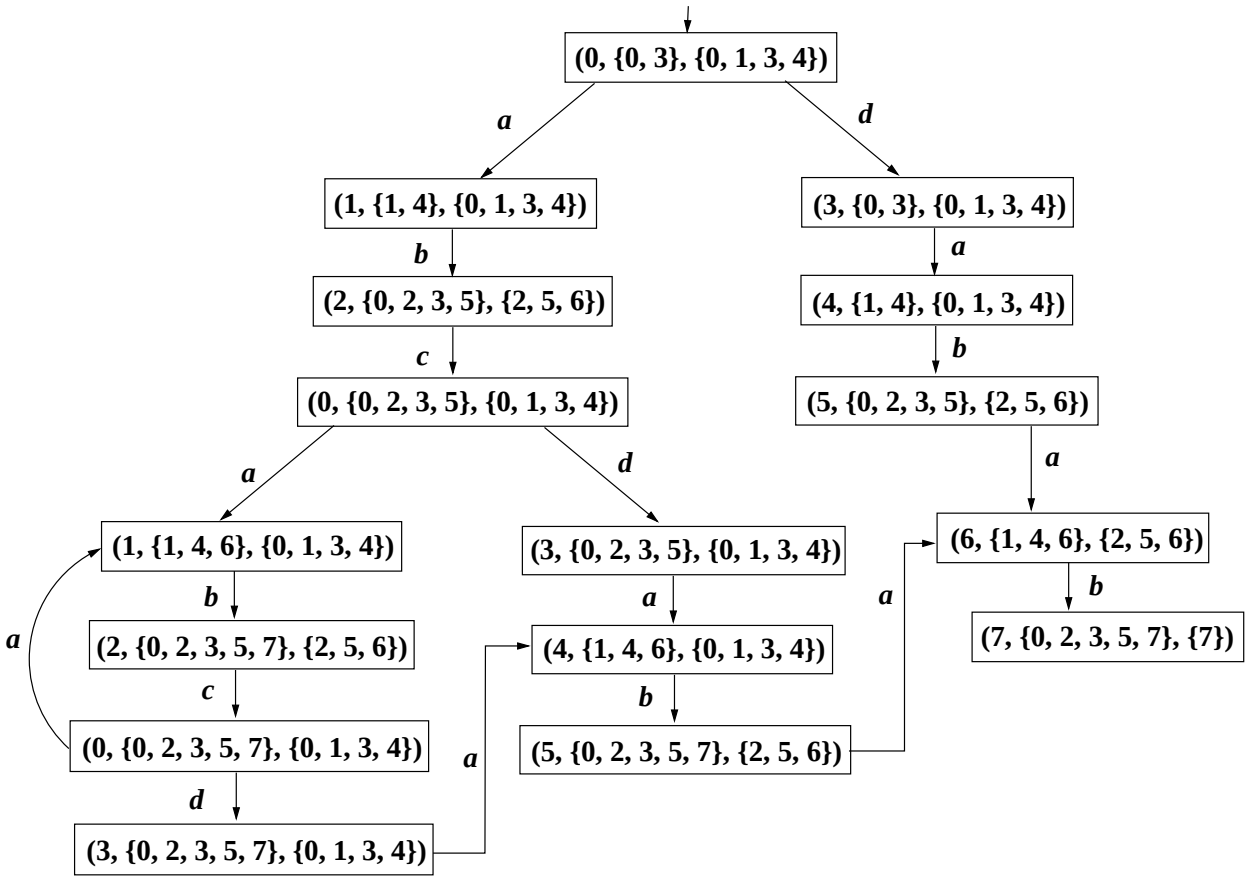

Figure 3: The monitoring automaton for the plant in figure 2.

infinite number of sequences represented by the latter expression.

Similarly we examine the paths t' to states $(1, \{1,4,6\}, \{0,1,3,4\})$ and $(4, \{1,4,6\}, \{0,1,3,4\})$ such that $P_1(t) = P_1(t')$ for the t noted above. Our claim is that the identification of these sequences is sufficient to determine where communication should be added for purposes of solving the control problem. The formal proof of this conjecture constitutes future work.

4.1 Finding a location to communicate: picking control communication pairs

We have established that an supervisor requires extra information (e.g., communicated from another supervisor) when it cannot distinguish an illegal sequence from a legal sequence and the supervisor must make the correct control decision. We have yet to establish how to identify specific locations where communication will give supervisors the knowledge to solve the control problem. In this section we claim that, subject to certain conditions, we can always find a location for supervisors to communicate so that a control solution will eventually be reached. At such a location in the interpreted system, the communicating supervisor i can provide supervisor j with information that allows j to determine whether the system is along a sequence where j will have to make a control decision. We begin by introducing some terminology we will need to identify locations where communication occurs to solve the control problem.

Definition 2 *A* **communication state** *is a state $q \in Q^G$ where supervisor i communicates to supervisor j (for $i, j \in \{1,2\}$ and $i \neq j$) so that supervisor j will know whether it is observing states along a legal sequence or an illegal sequence.*

This definition is intentionally imprecise at this stage and will be updated later. For now, we consider a communication state to be a state where information from one supervisor is imparted to the supervisor responsible for making a control decision. The communicated information allows the latter supervisor to enable or disable the appropriate event at a subsequent point in the system.

Definition 3 *A* **maximal-P pair** *(t, t') is a pair of sequences $t, t' \in \Sigma^*$ where $P(t) = P(t')$ and $\nexists \sigma \in \Sigma$ such that $P(t\sigma) = P(t')$ or $P(t) = P(t'\sigma)$.*

Recall that the canonical projection operator P in (1) effectively erases the unobservable events in a sequence t. In this case P is a mapping from Σ^* to $(\Sigma_{1,o} \cup \Sigma_{2,o})^*$. Thus, a maximal-P pair pinpoints the last place two sequences look alike to an observer that sees all observable events. We will use maximal-P pairs to identify communication states by locating the worlds in the interpreted system where an illegal and a legal sequence in $L(G)$ look alike using canonical projection.

Definition 4 *The* **local view** *ℓ_i of a state $\ell \in Q^G$ reached via sequence t (i.e.,$\exists t \in \Sigma^*$ where $\delta^G(t, q_0^G) = \ell$) is the set of all the states in the plant that supervisor i considers the plant could be in upon seeing $P_i(t)$:*

$$\ell_i := \{q^G \mid q^G \in Q^G \wedge \exists u \in P_i^{-1}(P_i(t)) \text{ such that } \delta^G(u, q_0^G) = q^G\}.$$

Thus if supervisor i cannot determine if t or t' has occurred in the plant (i.e., $P_i(t) = P_i(t')$) and if $\delta^G(t, q_0^G) = q$ while $\delta^G(t', q_0^G) = q'$, the local view of supervisor i at state q will contain q and q'.

Definition 5 *If $t \in \Sigma^*$ and $\sigma \in \Sigma$, state $q^G \in Q^G$ is called a* **good state with respect to $t\sigma$** *if $\exists u, v \in \Sigma^*$ such that $t = uv$, $\delta^G(u, q_0^G) = q^G$ and $\delta^E(t\sigma, q_0^E)$ is defined.*

That is, a good state is one that occurs along a path of a legal sequence.

Definition 6 *If $t \in \Sigma^*$ and $\sigma \in \Sigma$ state $q^G \in q^G$ is called a* **bad state with respect to $t\sigma$** *if $\exists u, v \in \Sigma^*$ such that $t = uv$, $\delta^G(u, q_0^G) = q^G$ and $\delta^G(t\sigma, q_0^G)$ is defined but $\delta^E(t\sigma, q_0^E)$ is not defined.*

Similarly, a bad state is one that occurs along a path of an illegal sequence.

We will want to be able to draw conclusions about what a supervisor sees if the canonical projections of two sequences are equal. For instance, if $P(t) = P(t')$, we want to conclude that $P_i(t) = P_i(t')$. In the lemma that follows, we use P^A to identify a canonical projection operator from Σ^* to A^*, where A is a subset of Σ.

Lemma 1 *Let $B \subseteq A \subseteq \Sigma$. For canonical projection operators $P^A : \Sigma^* \to A^*$ and $P^B : \Sigma^* \to B^*$, if $P^A(t) = P^A(t')$, where $t, t' \in \Sigma^*$ then $P^B(t) = P^B(t')$.*

The result can be shown by using induction on the length of strings. Thus sequences that are indistinguishable to a supervisor are also indistinguishable to other supervisors that observe fewer events.

We prove here that under a certain condition we can find locations where a communicating supervisor can eliminate the confusion of the supervisor incapable of making the correct control decision. The confused supervisor simply needs to be able to tell bad states from good states.

In the following theorem, observability is a hypothesis because observability means that a centralized observer (one that could see all the events that both supervisors see) could solve the control problem. Otherwise one supervisor lacks observations that could not necessarily be supplied by the other supervisor.

Theorem 1 *Given G, E and let $i \in \{1, 2\}$. If E is observable with respect to G, P and $\exists \hat{\sigma} \in \Sigma_{i,c}, t, t' \in L(G)$ such that $t\hat{\sigma} \notin L(E)$ and $t'\hat{\sigma} \in L(E)$ and $P_i(t) = P_i(t')$ then $\exists \ell \in Q^G$ where ℓ is either a good state with respect to $t'\hat{\sigma}$ or a bad state with respect to $t\hat{\sigma}$ and $\nexists y, y' \in \ell_1 \cap \ell_2$ (for $y \neq y'$) where y is a bad state with respect to $t\hat{\sigma}$ and y' is a good state with respect to $t'\hat{\sigma}$.*

Proof: There exists $u, u' \in \Sigma^*$ such that $u \in \overline{t}$, $u' \in \overline{t'}$ and (u, u') is a maximal-P pair. Since E is observable with respect to G, (t, t') is not a maximal-P pair and therefore either u is a proper prefix of t (i.e., $u \neq t$) or u' is a proper prefix of t'. Without loss of generality, let u be a proper prefix of t (i.e., $\exists \sigma \in \Sigma, v \in \Sigma^*$ such that $t = u\sigma v$). Let $\delta^G(u, q_0^G) = z$ and $\delta^G(u\sigma, q_0^G) = x$.

We consider the following two cases:

Case A: (u, t') is a maximal-P pair.

Let $\delta^G(t', q_0^G) = z'$. Refer to figure 4 (a) for a graphical representation of this case.

(i) $\sigma \in \Sigma_{uo}$

The next event after u cannot be unobservable. If σ is unobservable, then (u, t') would not be a maximal-P pair because we could extend u by σ.

(ii) $\sigma \in \Sigma_{i,o}$

We will argue that this scenario is not possible. It suffices to argue as follows:

$$\begin{aligned} P_i(t') &= P_i(t) \\ &= P_i(u\sigma v) \\ &= P_i(u)P_i(\sigma)P_i(v) \end{aligned} \tag{10}$$

Since (u, t') is a maximal-P pair, $P(u) = P(t')$. Since $\Sigma_{i,o} \subseteq \Sigma_{i,o} \cup \Sigma_{j,o}$, by Lemma 1 $P_i(u) = P_i(t')$. Therefore (10) holds only if $P_i(\sigma)P_i(v) = \varepsilon$. This leads to a contradiction as $P_i(\sigma) \neq \varepsilon$.

(iii) $\sigma \in \Sigma_{j,o}$

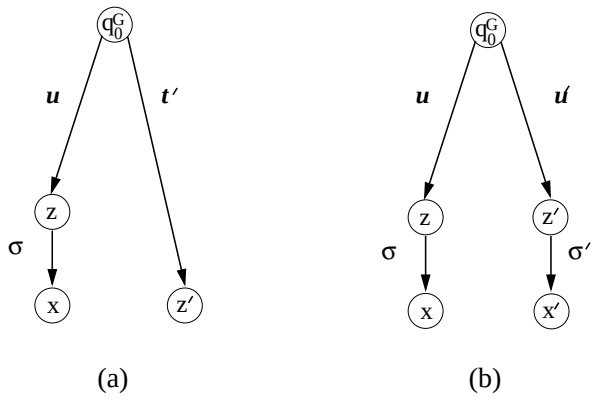

Figure 4: Identifying a communication state.

Claim 1 *State x is a state where $x_i \cap x_j$ does not contain distinct states y and y' where y' is a good state with respect to $t'\hat{\sigma}$ and y is a bad state with respect to $t\hat{\sigma}$.*

Note that $x_i \cap x_j$ already contains a bad state, namely x. Therefore we just have to show that there is no prefix of t' that has the same projection as some prefix of $u\sigma$.

At x, x_j already contains bad state x. The only way x_j could also contain a different good state with respect to $t'\hat{\sigma}$ is if there is some prefix of t', say w', where $P_j(w') = P_j(u\sigma)$. If it did, x_j would additionally contain the good state $\delta^G(w', q_o^G)$. Assume $t' = w'v'$:

$$\begin{aligned} P_j(w') &= P_j(u\sigma) \\ &= P_j(u)P_j(\sigma) \\ &= P_j(t')P_j(\sigma) \quad (\text{since } \Sigma_{j,o} \subseteq \Sigma_{i,o} \cup \Sigma_{j,o} \text{ and } P(u) = P(t'), \\ &\qquad\qquad \text{by Lemma 1, } P_j(u) = P_j(t')) \\ &= P_j(w')P_j(v')P_j(\sigma) \end{aligned} \tag{11}$$

For (11) to hold, $P_j(v')P_j(\sigma) = \varepsilon$ which leads to a contradiction as $P_j(\sigma) \neq \varepsilon$ (since $\sigma \in \Sigma_{j,o}$).

□ **Claim 1**

Case B: (u, t') is not a maximal-P pair.

Then u' is a proper prefix of t'.

Let $t' = u'\sigma'v'$ for $\sigma' \in \Sigma$, $v' \in \Sigma^*$ and $\delta^G(u', q_0^G) = z'$ and $\delta^G(u'\sigma', q_0^G) = x'$, as shown in figure 4 (b).

(i) $\sigma, \sigma' \in \Sigma_{uo}$

This scenario is not possible. A next event along t after u (respectively, along t' after u') cannot be unobservable, otherwise we would be able to extend u or u' and violate the fact that (u, u') is a maximal-P pair.

(ii) $\sigma, \sigma' \in \Sigma_o$ and $\sigma = \sigma'$

This scenario is not possible. The next event along t after u cannot be identical to the next event along t' after u', otherwise (u, u') would not be a maximal-P pair.

(iii) $\sigma \in \Sigma_{i,o}, \sigma' \in \Sigma_{j,o}$

Claim 2 *State x' is a state where $x'_i \cap x'_j$ does not contain distinct states y and y' where y' is a good state with respect to $t'\hat{\sigma}$ and y is a bad state with respect to $t\hat{\sigma}$.*

We will first show that from state z there is no sequence $v = \sigma w$, where $v \in \Sigma^*$, such that $P_i(uv) = P_i(u'\sigma')$. That is, if x'_j contains any bad states (distinct from x') with respect to $t\hat{\sigma}$ that occur after z along t, these states are not in x'_i.

We need only show that $P_i(uv) \neq P_i(u'\sigma')$. Suppose it were. Then

$$\begin{aligned} P_i(uw) &= P_i(u'\sigma') \\ P_i(u)P_i(w) &= P_i(u')P_i(\sigma') \\ &= P_i(u)P_i(\sigma') \quad (\text{ since } \Sigma_{i,o} \subseteq \Sigma_{i,o} \cup \Sigma_{j,o}, \\ &\qquad \text{by Lemma 1, } P_i(u) = P_i(u')) \end{aligned} \tag{12}$$

For (12) to hold it must be the case that $P_i(\sigma w) = P_i(\sigma') = \varepsilon$ (since $\sigma' \in \Sigma_{j,o}$), which leads to a contradiction since $P_i(\sigma) \neq \varepsilon$.

As for Case A (iii), we have a situation where at state x', x'_i already contains the states z, z', x' since $P_i(u) = P_i(u')$ and $P_i(u'\sigma') = P_i(u)$ (because $\sigma' \in \Sigma_{j,o}$). At x', x'_j already contains good state x'. Now, the only way x'_j could also contain a bad state (distinct from x') with respect to $t\hat{\sigma}$ is if there is some prefix of u, say $\hat{w}$, where $P_j(\hat{w}) = P_j(u'\sigma')$. Thus x'_j would also contain a bad state $\delta^G(\hat{w}, q_o^G)$. Suppose that such a $\hat{w}$ exists (i.e., $u = \hat{w}\hat{v}$ and $\hat{v} \in \Sigma^*$). Then

$$\begin{aligned} P_j(\hat{w}) &= P_j(u'\sigma') \\ &= P_j(u')P_j(\sigma') \\ &= P_j(u)P_j(\sigma') \quad (\text{since } \Sigma_{j,o} \subseteq \Sigma_{i,o} \cup \Sigma_{j,o}, \text{ by Lemma 1, } P_j(u) = P_j(u')) \\ &= P_j(\hat{w}\hat{v})P_j(\sigma') \\ &= P_j(\hat{w})P_j(\hat{v})P_j(\sigma') \end{aligned} \tag{13}$$

which leads to a contradiction since $P_j(\sigma') \neq \varepsilon$ (since $\sigma' \in \Sigma_{j,o}$).

□ Claim 2

(iv) $\sigma \in \Sigma_{j,o}, \sigma' \in \Sigma_{i,o}$

Analogous to Case B (ii). In the current scenario, the claim to be proven becomes: The state x is a state where $x_i \cap x_j$ does not contain a good state y' with respect to $t'\hat{\sigma}$ and a bad state y with respect to $t\hat{\sigma}$.

(v) $\sigma, \sigma' \in \Sigma_{i,o}$ and $\sigma \neq \sigma'$

We will argue that this scenario is not possible. We have that $P_i(t) = P_i(t')$ and substituting for t and t':

$$\begin{aligned} P_i(u\sigma v) &= P_i(u'\sigma'v') \\ P_i(u)P_i(\sigma)P_i(v) &= P_i(u')P_i(\sigma')P_i(v') \\ &= P_i(u)P_i(\sigma')P_i(v') \quad (\text{since } \Sigma_{i,o} \subseteq \Sigma_{i,o} \cup \Sigma_{j,o}, \\ &\qquad \text{by Lemma 1, } P_i(u) = P_i(u')) \end{aligned}$$

which leads to a contradiction because $P_i(\sigma) \neq P_i(\sigma')$ (since $\sigma \neq \sigma'$).

(vi) $\sigma, \sigma' \in \Sigma_{j,o}$ and $\sigma \neq \sigma'$

Claim 3 *States x and x' are both states where $x_i \cap x_j$ and $x'_i \cap x'_j$ do not contain distinct states y and y' where y is a good state with respect to $t'\hat{\sigma}$ and y' is a bad state with respect to $t\hat{\sigma}$.*

We want to first illustrate the case for x' by showing after state z there is no sequence $v = \sigma w$, where $v \in \Sigma^*$, such that $P_j(uv) = P_j(u'\sigma')$.

Suppose that such a v exists. Then

$$\begin{aligned} P_j(uv) &= P_j(u'\sigma') \\ P_j(u)P_j(v) &= P_j(u')P_j(\sigma') \\ &= P_j(u)P_j(\sigma') \quad (\text{ since } \Sigma_{j,o} \subseteq \Sigma_{i,o} \cup \Sigma_{j,o}, \\ &\qquad\qquad \text{by Lemma 1, } P_j(u) = P_j(u')) \end{aligned} \tag{14}$$

Since $v = \sigma w$, for (14) to hold it must be the case that $P_j(\sigma w) = P_j(\sigma')$, which leads to a contradiction since $P_j(\sigma) \neq P_j(\sigma')$.

To show that there is no sequence $\hat{w}$ leading to state z where $P_j(\hat{w}) = P_j(u'\sigma')$, we follow the same steps in (13).

Similar reasoning shows that if we instead select x as our state, that there is (a) no $v' = \sigma' w'$ such that $P_j(u'v') = P_j(u\sigma)$; and (b) that there is no $\hat{w}$ along t' leading to state x' such that $P_j(\hat{w}) = P_j(u\sigma)$.

□ Claim 3

□ THEOREM 1

The idea of Theorem 1 is that when supervisor i cannot make the correct control decision about $\sigma \in \Sigma_{i,c}$, (i.e., $\exists t, t' \in L(G)$ such that $t'\sigma \in L(E)$, $t\sigma \notin L(E)$ and $P_i(t) = P_i(t')$) we can always find a location—somewhere along either t or t'—where supervisor j can distinguish between t and t'. At this location or communication state, supervisor j sends its local state or local view to supervisor i. Prior to receiving the communication, supervisor i does not know whether or not the current state of the system leads to an illegal sequence or a legal sequence. When supervisor i updates its own local state with the communicated information, supervisor i can tell the difference between the legal and the illegal sequence.

The proof of Theorem 1 yields the following update to our definition of a communication state:

Definition 7 *Given t, t' satisfying the hypotheses of Theorem 1, sequences u, u' where $u \in \overline{t}$, $u' \in \overline{t'}$ and (u, u') is a maximal-P pair, define a* **communication state q** *to be*

(a) $\delta^G(u'\sigma_j, q_0^G)$ *if $t' = u'\sigma_j v'$ for some $\sigma_j \in \Sigma_{j,o}, v' \in \Sigma^*$ and $t = u\sigma_i v$ for some $\sigma_i \in \Sigma_{i,o}, v \in \Sigma^*$ (by Claim 1);*

(b) $\delta^G(u\sigma_j, q_0^G)$ *if $t = u\sigma_j v$ for some $\sigma_j \in \Sigma_{j,o}, v \in \Sigma^*$ and $t' = u'\sigma_i v'$ for some $\sigma_i \in \Sigma_{i,o}, v' \in \Sigma^*$ (by Claim 2);*

(c) $\delta^G(u\sigma_j, q_0^G)$ *or* $\delta^G(u'\hat{\sigma}_j, q_0^G)$ *if $t = u\sigma_j v$, and $t' = u'\hat{\sigma}_j v'$ for some $\sigma_j, \hat{\sigma}_j \in \Sigma_{j,o}, v, v' \in \Sigma^*$ and $\sigma_j \neq \hat{\sigma}_j$ (by Claim 3).*

In the definitions that follow, the sequences t and t' are those satisfying the hypothesis of Theorem 1.

Definition 8 *A* **communication sequence s** *for a* **communication state q** *and a sequence* **t** *is a sequence that leads to q (i.e., $\delta^G(s, q_0^G) = q$) and is a prefix of t, where q is a bad state with respect to $t\sigma$. (Equivalently, for q and a sequence t' where s is a prefix of t' and q is a good state with respect to $t'\sigma$). A* **control sequence for communication state q** *is the sequence along which a communication state has been identified. If t is a control sequence for q then t' is the* **control twin** *for t (and vice versa).*

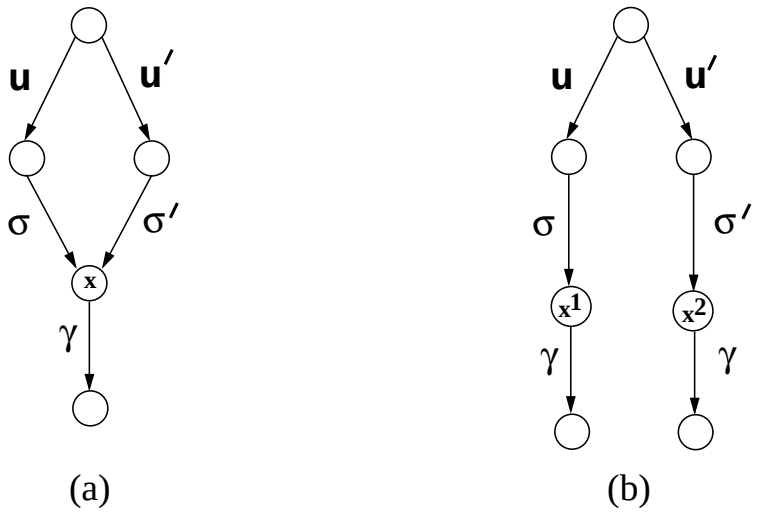

Figure 5: Splitting G: (a) intention is for communication to occur at state x after $u\sigma$; (b) redraw G and split state x to find a definitive communication state x^1.

That is, there are two sequences that a supervisor cannot distinguish but one leads to an illegal sequence and the other leads to a legal sequence. Communication that will allow a supervisor to distinguish between these two sequences and make the correct control decision occurs along the "control sequence".

We can now uniquely identify when and where supervisors communicate to solve the control problem: communication from one supervisor to another occurs at a communication state q, after the communication sequence s is observed by the communicating supervisor, say supervisor i.

Definition 9 *A* **control communication pair** *for supervisor i is a pair (q, t) and consists of a communication state q and a control sequence t.*

The idea is that supervisor i communicates its local view q_i to supervisor j when the plant is at state q. The sets $\mathcal{C}_{12}$ and $\mathcal{C}_{21}$ will store the control communication pairs for supervisors 1 and 2, respectively.

Definition 10 *The* **communication event** *associated with a control communication pair $(q, t) \in \mathcal{C}_{ij}$ is com_{ij}:q.*

This notation represents the action of supervisor i communicating its local state to supervisor j at communication state q. That is, communication occurs after the communication sequence s for (q, t) occurs. A communication event com_{ij}:q is observable by both supervisors i and j but is controllable only by supervisor i.

4.2 Avoiding futile communication

The definitions regarding communication locations presented thus far have hidden a subtle possibility. It is possible that the selection of a communication state results in the reception of information that will not lead to a supervisor knowing definitively whether it is along an illegal or a legal sequence. If after both u and u', there are events σ and σ' (as in figure 5(a)), leading to *the same state* (i.e., $x = x'$ in figure 5(a)) or if after u there is an event σ leading to the same state that

t' leads to (i.e., $x = z'$ in figure 5(a)), then state x itself is such that x is good with respect to $t'\hat{\sigma}$ and is bad with respect to $t\hat{\sigma}$. So, a communication from supervisor j to supervisor i that it is at state x would not yield any helpful information for supervisor i. Consequently, for those cases, we "split the state" x into two different states with distinct labels. That is, we make two copies of x. An illustration of what we mean is shown in figure 5. In figure 5(a), assume that the intention is for communication to occur at state x either after $u\sigma$ or $u'\sigma'$ but not after both. Suppose it had been determined that supervisor j—after seeing $P_j(u\sigma)$—communicates its local view of state x, with the intention of allowing supervisor i to distinguish between $u\sigma$ and $u'\sigma'$. But communication occurs whenever supervisor j believes the plant to be at state x. This happens not only when supervisor j sees $P_j(u\sigma)$ but also when it sees $P_j(u'\sigma')$. Yet we only want supervisor j to communicate after $P_j(u\sigma)$ and not after both $P_j(u\sigma)$ and $P_j(u'\sigma')$. In figure 5(b) we redraw the graph we use to represent G and split state x to find a definitive communication state x^1.

Now, either state x^1 or state x^2 in figure 5(b) would be a state that does not contain both a good state with respect to $t'\hat{\sigma}$ and a bad state with respect to $t\hat{\sigma}$.

We identify a finite number of states, say n, where communication is necessary to solve the control problem. As a result, if it is necessary to redraw the graph we use to represent G so that the only state shared by t and t' is q_0^G, the strategy of splitting states is one that terminates. In the worst case, if we have to perform a split at every state (where a split would entail two copies of the plant to be created) there would be 2^n iterations of the process (a finite number since n is finite).

Note that the language generated by an automaton where some states have been split as described above is the same as the language generated by the original automaton. From here on, we assume that the automaton representing the plant G has been redrawn to accommodate all occurrences of the above scenario.

4.3 How to incorporate communication into a plant

We represent the action of communication from one supervisor to another as a new event that is added to the plant. To this end we define a set Σ^{com} to store events that represent communication and a set Q^{com} to keep track of all new states we will need to incorporate the events of Σ^{com} into the plant.

Formally, to incorporate communication into our system, we create a new automaton:

$$G^{com} = (Q^{G^{com}}, \Sigma \cup \Sigma^{com}, \delta^{G^{com}}, q_0^{G^{com}}),$$

where the set of states $Q^{G^{com}} := Q^G \cup Q^{com}$, the alphabet is $\Sigma \cup \Sigma^{com}$ and the initial state $q_0^{G^{com}} := q_0^G$. The identification of a communication state $q \in Q^G$ (where supervisor i communicates to supervisor j) and a communication sequence s gives rise to the creation of a new state $q^{\mathbf{c}}$ which is added to Q^{com} and a new event com_{ij}:q which is added to Σ^{com}. We will sometimes want to refer to those communication events where supervisor i communicates to supervisor j. Thus we partition Σ^{com} into disjoint sets Σ_{ij}^{com}, for $i, j \in \{1, 2\}$ and $i \neq j$. Prior to incorporating communication, G^{com} is simply a copy of G. That is, $Q^{com} = \emptyset$, $\Sigma^{com} = \emptyset$ and $\delta^{G^{com}} = \delta^G$.

Observation 1 *Suppose that a sequence $v \in L(G^{com})$ leads to a state $q \in Q^{G^{com}}$ but $q \notin Q^{com}$. That is, $\delta^{G^{com}}(v, q_0) = q$. Then by the way in which G^{com} is constructed from G, the version of this sequence that appears in $L(G)$, say v', (i.e., all the communication events have been removed from v) also leads to state q. That is, $\delta^G(v', q_0) = q$.*

We present the first of three main procedures that transform G into G^{com}. Procedure 1 describes how to identify control communication pairs using the knowledge model $\mathcal{I}^{DES}$ and the monitoring automaton A.

Procedure 1 : Identifying Communication for Control

1. Initially $G^{com} = G$, $\Sigma^{com} = \emptyset$, $Q^{G^{com}} = Q^G$ and $\delta^{G^{com}} = \delta^G$. We also initialize $\mathcal{C}_{12} = \mathcal{C}_{21} = \emptyset$.
2. Identify those states at which Kripke-observability fails for $\mathcal{I}^{DES'}(G, E)$, i.e., a state in the monitoring automaton A.
3. Using Theorem 1, identify control communication pairs (q, t) and their corresponding control twins t' for supervisor 1 and for supervisor 2. We use the monitoring automaton A to identify t and t'. Update the appropriate set of control communication pairs $\mathcal{C}_{ij} = \mathcal{C}_{ij} \cup \{(q, t)\}$, for $i, j \in \{1, 2\}$ and $i \neq j$.

□ Procedure 1

Procedure 1 identifies the control communication pairs (q, t) that indicate where a supervisor discloses its local state to another supervisor. This information must now be translated into places where we add communication events to the augmented plant G^{com}. We call the function *Build*G^{com} with parameters $\mathcal{C}_{12}$ and $\mathcal{C}_{21}$ incorporating the communication event associated with each $(q, t) \in \mathcal{C}_{12} \cup \mathcal{C}_{21}$. That is, a communication event is added, after sequence s occurs, at state q in G^{com}.

Function *Build*G^{com} ($\mathcal{C}_{12}$, $\mathcal{C}_{21}$)
Input: G^{com}, $\mathcal{C}_{12}$, $\mathcal{C}_{21}$
Output: G^{com}

1. For each $(q, t) \in \mathcal{C}_{ij}$, for $i, j \in \{1, 2\}$ and $i \neq j$:
   (a) Create a new state $q^{\mathbf{c}}$. If $q^{\mathbf{c}} \notin Q^{com}$, update the state set: $Q^{com} = Q^{com} \cup \{q^{\mathbf{c}}\}$.
   (b) Create a new event called com_{ij}:q which represents the action of supervisor i communicating local view q_i to supervisor j. If com_{ij}:$q \notin \Sigma^{com}$, update the alphabet: $\Sigma^{com} = \Sigma^{com} \cup \{com_{ij}$:$q\}$.
   (c) Update the transition function $\delta^{G^{com}}$. Suppose the communication sequence for state q has the form $s = u\sigma$ where $\delta^G(u, q_0^G) = q'$ and $\delta^G(\sigma, q') = q$. Then if $\delta^{G^{com}}(\sigma, q') = q$ (i.e., no communication has been added at state q yet) we must first remove this transition from $\delta^{G^{com}}$. The following transitions are then added to $\delta^{G^{com}}$:

$$\begin{aligned} \delta^{G^{com}}(\sigma, q') &= q^{\mathbf{c}}, \\ \delta^{G^{com}}(com_{ij}{:}q, q^{\mathbf{c}}) &= q. \end{aligned}$$

   It could be the case that a communication event representing communication from supervisor i to supervisor j has already been added to state q in G^{com}. That is, more than one communication sequence associated with the elements of $\mathcal{C}_{ij}$ leads to state q. A communication event com_{ij}:q is added to state q only once. Or it could be the case that a communication event representing communication from supervisor j to supervisor i has already been added to state q in G^{com}. If a communication event from supervisor

j to supervisor i has been added to state q already (i.e., $\delta^{G^{com}}(\sigma, q') \neq q$), we create a new state $q^{\mathbf{cc}}$ and update Q^{com}:

$$Q^{com} = Q^{com} \cup \{q^{\mathbf{cc}}\}.$$

This situation arises if $(q,t) \in \mathcal{C}_{ji} \cap \mathcal{C}_{ij}$, where the communication sequence is $s = u\sigma$ such that $\sigma \in \Sigma_{i,o} \cap \Sigma_{j,o}$, since a communication state where supervisor i communicates to supervisor j occurs only after an event that supervisor i sees. Then we remove the following transition from G^{com}:

$$\delta^{G^{com}}(\sigma, q') = q^{\mathbf{c}}.$$

Add the following transitions to $\delta^{G^{com}}$:

$$\delta^{G^{com}}(\sigma, q') = q^{\mathbf{cc}}$$
$$\delta^{G^{com}}(com_{ij}{:}q, q^{\mathbf{cc}}) = q^{\mathbf{c}}.$$

2. Return.

□ **Function** *BuildG*^{com}

We interpret the appearance of two consecutive communication events in G^{com} as a two-way broadcast between supervisors i and j. That is, each supervisor communicates its local state to the other at the same time. Note that, by construction of G^{com}, one event will always correspond to a control communication pair in $\mathcal{C}_{ij}$ and the other to an element of $\mathcal{C}_{ji}$. In section 6 we elaborate on the effect this has on the construction of a communication protocol.

The time complexity of Procedure 1 is dominated by step 3: finding the control communication pairs. The other steps in the procedures can be accomplished in constant time. We use our knowledge model to identify states where Kripke-observability fails, and thus where we can reconstruct sequences that give rise to control communication pairs. The identification of such sequences can be performed using a dynamic-programming algorithm that takes $O(n^3)$ time, where n is the number of states in the monitoring automaton [6]. All the steps in function *BuildG*^{com} can be performed in constant time and therefore, when iterated over the total number of communication pairs, the time complexity is $O(|\mathcal{C}_{12} \cup \mathcal{C}_{21}|)$.

4.4 Communication that solves the control problem

We must formally show that when supervisor i finds a control sequence indistinguishable from its corresponding control twin, the addition of a communication event along the communication sequence allows supervisor i to distinguish these two sequences in G^{com}. We begin by describing what it means for a sequence in G to be translated into G^{com}.

We define an operation that "erases" communication events and extend our definition of P_i as follows. Let $\hat{P}$ be the projection from $(\Sigma \cup \Sigma^{com})^*$ to Σ^* and therefore $(\Sigma^{com})^* \to \varepsilon$. Similarly, $\hat{P}_i$ is the projection from $(\Sigma \cup \Sigma^{com})^*$ to $\Sigma_{i,o}^*$ and again $(\Sigma^{com})^* \to \varepsilon$. Despite expanding the domain of P_i, $\hat{P}_i$ recognizes the same set of sequences as its predecessor. The only difference is that now $\hat{P}_i$ "erases" not just the events in $\Sigma \setminus \Sigma_{i,o}$ but also those events in Σ^{com} from a sequence t.

We will want to describe a sequence in $L(G)$ when it is transformed by communication events and appears in $L(G^{com})$ after following Procedure 1 and *BuildG*^{com}.

Definition 11 *For two sequences $t \in L(G)$ and $t^{\mathbf{c}} \in L(G^{com})$, we say $\boldsymbol{t^{\mathbf{c}}}$ is a* **communication-equivalent sequence for $\boldsymbol{t}$** *if $L(G^{com})$ is the language generated by the G^{com} that results from the completion of Procedure 1 and BuildG^{com} and*

$$\delta^{G}(t, q_0^{G}) = \delta^{G^{com}}(t^{\mathbf{c}}, q_0^{G^{com}})$$

and

$$\hat{P}(t^{\mathbf{c}}) = t.$$

Thus, a communication-equivalent sequence contains any communication events that occur along t and any communication events that occur directly after t. From now on we use $t^{\mathbf{c}}$ to refer to a communication-equivalent sequence for t generated by the G^{com} produced by completing Procedure 1 and *Build*G^{com}.

If we add a communication event along a control sequence t and not along its control twin t' (according to Theorem 1) the two sequences will no longer look alike to the supervisor making the control decision at t or t'. This is formalized in the following lemma. In this lemma, since we will want to describe what a supervisor sees in G^{com}, we define $P_i^{\mathbf{c}}$ (for $i = 1, 2$) to be the projection from $(\Sigma \cup \Sigma^{com})^*$ to $(\Sigma_{i,o} \cup \Sigma^{com})^*$.

Lemma 2 *For a control sequence t and its control twin t' defined with respect to supervisor i (i.e., $P_i(t) = P_i(t')$), after following Procedure 1 and BuildG^{com}, $P_i^{\mathbf{c}}(t^{\mathbf{c}}) \neq P_i^{\mathbf{c}}(t'^{\mathbf{c}})$.*

Proof. Since $P_i(t) = P_i(t')$ we know that $\mathcal{C}_{ji}$ will contain at least one element. Let com_{ji}:q be such a communication event added to t. Since the plant has been redrawn such that t and t' do not share communication states, the state q does not appear along t'. Therefore, after Procedure 1 and *Build*G^{com} are completed, com_{ji}:q will not be added along t'. Therefore $P_i^{\mathbf{c}}(t^{\mathbf{c}}) \neq P_i^{\mathbf{c}}(t'^{\mathbf{c}})$.

□ Lemma 2

In the next section, we show that after adding the remaining communication events to the rest of the plant (i.e., along sequences that are indistinguishable from communication sequences identified by Procedure 1), the communication-equivalent sequence for control sequence t remains distinguishable from the updated communication-equivalent sequence for the control twin t'.

5 Communication for Consistency

Our communication goal is two-fold: (i) to have supervisors communicate at some place that will lead to a control solution—we identified this place as the state after a communication sequence occurs; and (ii) to have the plant reflect the intent of each supervisor to communicate at all places that they cannot distinguish from the communication state.

This seems like a straightforward process. We proceed naïvely and add a communication event com_{ij}:q to the plant after each control communication sequence s associated with each $(q, t) \in \mathcal{C}_{ij}$. Additionally, we add com_{ij}:q after each sequence v that is indistinguishable to supervisor i from s. But we must take into consideration that as we take care of adding communication events with respect to one control sequence, the addition of a new communication event may alter the situation for other control sequences. This was a point that was first raised in [16]. We will return to this observation shortly.

We formally define what we mean for G^{com} to satisfy consistency:

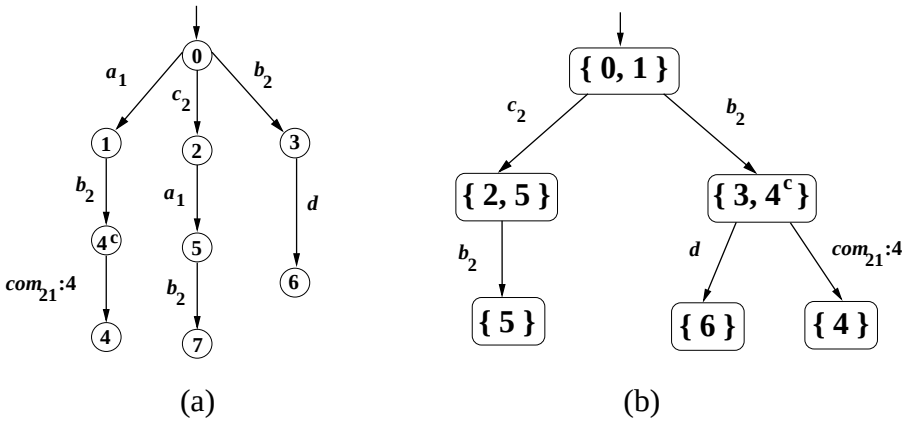

Figure 6: A G^{com} that does not satisfy consistency.

Definition 12 *A system G^{com} is said to be* **consistent** *if for all $(q,t) \in \mathcal{C}_{ij}$ (where $i,j \in \{1,2\}$ and $i \neq j$), and for all $q^{\mathbf{c}} \in Q^{com}$ such that $\delta^{G^{com}}(q^{\mathbf{c}}, com_{ij}{:}q) = q$, and for all $y \in Q^{G^{com}}$ such that $y_i = q_i^{\mathbf{c}}$, $\delta^{G^{com}}(y, com_{ij}{:}q)$ must be defined, where $y_i, q_i^{\mathbf{c}}$ are supervisor i's local views of states y and $q^{\mathbf{c}}$, respectively.*

That is, whenever we identify a communication state q from a control communication pair (q,t) for a supervisor, not only does a communication event exit from state $q^{\mathbf{c}}$ (e.g., $\delta^{G^{com}}(q^{\mathbf{c}}, com_{ij}{:}q) = q$)) it must also exit all states $y \in Q^{G^{com}}$ when supervisor i's local view of y is equal to supervisor i's local view of $q^{\mathbf{c}}$.

Figure 6 illustrates a scenario we must preclude. Suppose that supervisor 1 sees and controls events a_1 and d while supervisor 2 sees and controls events b_2, c_2 and d. The G^{com} in figure 6(a) does not satisfy consistency. The observer automaton of G^{com} with respect to supervisor 2 is shown in figure 6(b). Note that in figure 6(b) the local view of communication state $4^{\mathbf{c}}$ for supervisor 2 is $\{3, 4^{\mathbf{c}}\}$. Similarly, the local view of state 3 for supervisor 2 is $\{3, 4^{\mathbf{c}}\}$. Our definition of consistency says that the communication event $com_{21}{:}4$ must exit from every state in G^{com} that shares that same local view as the communication state $4^{\mathbf{c}}$. There is no communication event defined at state 3, thus violating consistency.

The reason that we will want to preclude this type of scenario (as in figure 6(b)) is because the observer automaton will form the basis of an supervisor's communication protocol. The idea is that if a communication event occurs at a particular state, a supervisor must communicate. If more than one event is defined at that state, a supervisor would not have a clear directive as to when communication should happen. For example, when supervisor 2 is at state $\{3, 4^{\mathbf{c}}\}$ in figure 6(b) either event d may occur *or* a communication event $com_{21}{:}4$ may occur. We clarify this notion, which we refer to as a *well-defined* communication protocol, in section 6.

5.1 Refining local views of control communication pairs

As noted earlier, one option for making G^{com} consistent would be to add communication events to G^{com} at those states that are indistinguishable from state $\delta^G(s, q_0^G)$. However, the following scenario could unfold: suppose that supervisor j must communicate for control to supervisor i at state x and suppose that its local view of x is $x_j = \{x, y, z\}$. Thus, in G, supervisor j is unable to distinguish plant states x, y and z. Further suppose that because of some prior communication from supervisor i, supervisor j can distinguish x and y in G^{com}. In this case, x_j really just consists

of the plant states x and z. An intent to communicate at plant state y constitutes a communication that is unnecessary.

Our strategy refines the local views of communication states calculated for each supervisor with respect to the original plant G. This approach considers the effects of prior communication along a communication sequence before determining where to add communication events to satisfy consistency.

We begin by introducing some terminology we will need for describing how we refine the supervisors' local views of G^{com}.

Definition 13 *A pair* (x, v) *consisting of a state* $x \in Q^G$ *and a sequence* $v \in \Sigma^*$, *such that* $\delta^G(v, q_0^G) = x$, *is* **compatible with** *a control communication pair* $(q, t) \in \mathcal{C}_{ij}$, *for* $i, j \in \{1, 2\}$ *and* $i \neq j$, *if*

$$P_i(v) = P_i(s),$$

where s *is the communication sequence for* (q, t) *and* $v \neq s$.

That is, prior to incorporating communication events into G^{com}, we identify any sequence v that leads to state x and is indistinguishable to supervisor i from communication sequence s. Note that by not permitting $v = s$, we eliminate (q, s) from being compatible with (q, t).

We want to be able to identify places in the plant where we add communication events to satisfy consistency: sequences that are indistinguishable to the supervisor sending a communication for control after it observes s. Let $\mathcal{X}(q, t) = \{(x, v) \mid (x, v) \text{ is compatible with } (q, t) \in \mathcal{C}_{ij}\}$.

We state an assumption regarding where we place communication events along sequences that are indistinguishable from a communication sequence to a communicating supervisor.

Assumption 1 *If the system is at a communication state, we assume that communication from one supervisor to another happens the instant the communication sequence occurs and thus before the system makes any more transitions—including transitions that are unobservable to the communicating supervisor.*

Thus, we want to narrow down our set of locations where supervisors communicate and omit any pairs (x, v) such that v ends in a sequence unobservable to the communicating supervisor. We remove these pairs because we assume that a supervisor communicates the instant it observes the communication sequence.

Definition 14 *A pair* $(x, v) \in \mathcal{X}(q, t)$ *is called a* **compatible communication pair for** $(q, t) \in \mathcal{C}_{ij}$, *for* $i, j \in \{1, 2\}$ *and* $i \neq j$, *if* $\nexists w \in \Sigma^* \setminus \Sigma_{i,o}^*$ *such that* $v = uw$ (*i.e., the last event in* v *is in* $\Sigma_{i,o}$).

We want to describe whether or not a communication sequence s could contain other communication events. That is, does s contain a prefix v that is indistinguishable to supervisor i from some other communication sequence s'? Or is prefix v itself a communication sequence for another control communication pair?

Definition 15 *We say that a control communication pair* (q, t) **depends on** *control communication pair* (q', t') *if* (i) *we can find a compatible communication pair* (x, v) *for* (q', t') *such that* $v \in \overline{s}$ *and* $\delta^G(v, q_0^G) = x$, *where* s *is the communication sequence for* (q, t); *or* (ii) $s' \in \overline{s}$, *where* s' *and* s *are the communication sequences for* (q', t') *and* (q, t), *respectively.*

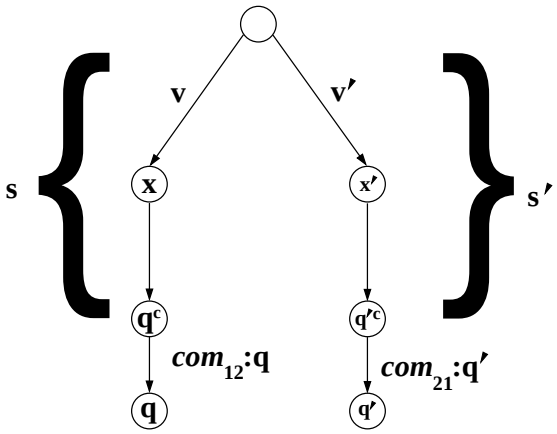

Figure 7: Communication sequences that contain other communication events. Let $P_1(v') = P_1(s)$ and $P_2(v) = P_2(s')$: event $com_{21}{:}q'$ could occur directly after v and thus prior to event $com_{12}{:}q$; event $com_{12}{:}q$ could be added directly after v' and thus prior to event $com_{21}{:}q'$.

That is, a communication sequence for (q,t) potentially contains another communication event, namely the event associated with the control communication pair (q',t'). For example, in figure 7, suppose that s is the communication sequence for (q,t) and s' is the communication for (q',t'). Further suppose that the communication sequence s for (q,t) contains a prefix v and supervisor 2 cannot distinguish v from communication sequence s'. That is, in G, state x is contained in supervisor 2's local view of state q', i.e., $x \in q'_2$. Then if the communication event $com_{21}{:}q'$ was added to the sequence on the left-hand side of figure 7 just after v happens, the updated version of communication sequence s now contains a communication event. Therefore, (q,t) depends on (q',t').

To detect some of the potential "dependencies" between control communication pairs, we build a *dependency graph* D. A dependency graph of an object illustrates all of its relations to other objects. The objects of interest are the control communication pairs. The relationship of interest here is whether communication sequence s for a control communication pair (q,t) contains a prefix that either looks like another communication sequence to the appropriate supervisor or is itself another communication sequence.

We use D to clarify the form of the communication sequences for the control communication pairs. That is, we want to determine how many (if any) and in which order other communication events could occur along a communication sequence. The nodes of the graph correspond to the set of the control communication pairs in $\mathcal{C}_{12} \cup \mathcal{C}_{21}$. There is a directed edge from $(q,t) \in \mathcal{C}_{ji}$ to $(q',t') \in \mathcal{C}_{ij}$ if (q,t) depends on (q',t'). The edge is labeled "(x,v)" if state x occurs somewhere along s (the communication sequence for (q,t)), $v \in \overline{s}$ and supervisor i cannot distinguish state x from state q' (i.e., (x,v) is a compatible communication pair for (q',t')). This edge labeling is unique for each pair of control communication pairs. It is not possible to have both (x,v) and (x',v') compatible with (q',t') such that v and v' are prefixes of s. That is, if $P_i(v) = P_i(v')$ and $v, v' \in \overline{s}$, such that $\delta^G(v, q_0^G) = x$ and $\delta^G(v', q_0^G) = x'$, then $x = x'$ and $v = v'$ (since by definition of a compatible communication pair both v and v' must end in events that are observable to supervisor i).

For the remainder of this discussion, we represent D as an adjacency matrix. The dependency graph contains $n_1 + n_2$ nodes, where $|\mathcal{C}_{12}| = n_1$ and $|\mathcal{C}_{21}| = n_2$. Thus D is an $(n_1 + n_2) \times (n_1 + n_2)$

matrix. The first n_1 row and column entries contain information pertaining to the dependencies of the control communication pairs in $\mathcal{C}_{12}$. The next n_2 rows and columns contain dependency information about the control communication pairs in $\mathcal{C}_{21}$.

For convenience, we do not refer to the entries of the matrix by the numerical row and column (i.e., $D[3,4]$ indexes the entry in row 3 and column 4 of D). Instead, we use the notation $D[(q,t),(q',t')]$ to refer to the row and column in D that contains information about the control communication pairs (q,t) and (q',t'), respectively. The non-empty entries in a row (q,t) of D indicate possible communication events that could occur along the communication sequence s for (q,t). For $D[(q,t),(q',t')]$ corresponding to $D[i,j]$, if $i \leq n_1$ then $(q,t) \in \mathcal{C}_{12}$ (otherwise $(q,t) \in \mathcal{C}_{21}$) and if $j \leq n_1$ $(q',t') \in \mathcal{C}_{12}$ (otherwise $(q',t') \in \mathcal{C}_{21}$). If $D[(q,t),(q',t')] = \emptyset$, then (q,t) is not dependent on (q',t'). That is, none of the states x that occur along the path to sequence s at state q coupled with any of the prefixes of s (i.e., $v \in \overline{s}$) forms a pair (x,v) that is compatible with (q',t'). If $D[(q,t),(q',t')] \neq \emptyset$, then (q,t) depends on (q',t') and $D[(q,t),(q',t')]$ contains a compatible communication pair for (q',t') that satisfies Definition 14.

The dependency graph could contain cycles (i.e., a path that begins at some state and returns to the same state). If G^{com} is to satisfy consistency we want the communication event $com_{ij}{:}q$ (which has been incorporated into G^{com} at state $q^{\mathbf{c}}$) to appear after all states $y^{\mathbf{c}}$ that are elements of $q_i^{\mathbf{c}}$. In the presence of a cycle in D, it is not clear when a state $y^{\mathbf{c}}$ is actually an element of $q_i^{\mathbf{c}}$. We return again to figure 7 for an illustration of the effect a cycle in D has on the construction of G^{com}. The figure shows part of a G^{com} after performing Procedure 1 and function *BuildG*^{com}. The (q,t) associated with s depends on the (q',t') associated with s' (via the compatible communication pair (x,v)) and that (q',t') depends on (q,t) (via the compatible communication pair (x',v')). If we added event $com_{21}{:}q'$ to state x in figure 7, then $x^{\mathbf{c}}$ is certainly in $q_2'^{\mathbf{c}}$ (since $P_2^{\mathbf{c}}(s') = P_2^{\mathbf{c}}(v)$). Similarly, if instead event $com_{12}{:}q$ is added to state x' in figure 7, then $x'^{\mathbf{c}}$ is certainly in $q_1^{\mathbf{c}}$ (since $P_1^{\mathbf{c}}(s) = P_1^{\mathbf{c}}(v)$). Note, though, that if *both* communication events are added to states x and x' then neither $x^{\mathbf{c}}$ nor $x'^{\mathbf{c}}$ are in $q_2'^{\mathbf{c}}$ or $q_1^{\mathbf{c}}$, respectively, and therefore we do not add communication at either state—whereby we end up where we started. To break this impasse, we add a communication event to only one of the states. That is, if we choose to only add $com_{12}{:}q'$ to G^{com} at state x, then $x^{\mathbf{c}} \in q_2'^{\mathbf{c}}$ and now $x' \notin q_1^{\mathbf{c}}$. Therefore, we break cycles in D to provide a systematic way of dealing with such situations.

Note that there is more than one way to break a cycle. Here we choose to break cycles at the node in the cycle corresponding to the control communication pair that depends on the fewest number of other control communication pairs. This means that once we detect a cycle in D we select the element, say (q,t), such that of all the elements involved in the cycle, row (q,t) of D has the fewest non-empty entries. In the event that each pair in the cycle depends on the same number of other pairs, the choice of a pair where communication is fixed is made at random. Different versions of D simply means that there is more than one way to arrange communication dependencies for the control communication pairs. A strategy for breaking cycles in D is described in function *DetectAndBreakCycles*. This function is used in conjunction with any algorithm for detecting cycles (e.g., substitute non-empty entries of D for "1", empty entries for "0"and raise D to the n^{th} power to determine which elements can be reached from themselves in n steps).

Function *DetectandBreakCycles*
Input:
Output: D

1. While cycles of length i (where $i = 2, \ldots, n$) exist in D

(a) For each cycle let $Cycle(q,t) := \{(q',t') \mid (q',t')$ occurs along a cycle from (q,t) to (q,t) in $D\}$

(b) Choose $(q',t') \in Cycle(q,t)$ such that row (q',t') in D has the fewest number of non-empty entries. If more than one (q',t') satisfies this criteria, randomly select one.

(c) Set $D[(q',t'),(q'',t'')] = \emptyset$, where $(q'',t'') \in Cycle(q,t)$ and $(q'',t'') \neq (q',t')$.

2. Return.

□ **Function** *DetectandBreakCycles*

If a cycle is detected in D, then at step 1(a) of the function we want to keep track of all the control communication pairs involved in the cycle. We arbitrarily select a communication pair to mark the start and end of the cycle, (q,t), and denote the set of all elements in a given cycle as $Cycle(q,t)$. In step 1(b), the element of $Cycle(q,t)$ that depends on the fewest number of other control communication pairs is selected as a place where the cycle will be broken. If more than element satisfies this criteria, then randomly select one. The cycle is broken in step 1(c) by removing the edge between the selected element and the next element in the cycle.

In Procedure 2 we restrict our attention to whether a control communication pair depends on any other control communication pair. The following procedure identifies states in the plant where a communication event (other than the one associated with control communication pair (q,t)) occurs along the path to state q via communication sequence s. The set $\mathcal{C}_{ij}^{compat}$, for $i,j \in \{1,2\}$ and $i \neq j$, stores the compatible communication pairs (x,v) for (q,t) where the communication event com_{ij}:q will be added to state x in G^{com}.

This procedure is used to determine which communication sequences will contain additional communication events. Procedure 2 terminates when all communication sequences have been examined, i.e., when all control communication pairs are "marked" (Steps 5 and 6(c)). Note that in Step 5, after the initialization of Steps 1 to 4, communication sequences that will not contain any additional communication events in G^{com} (i.e., the corresponding row in D contains all $\emptyset$ entries) do not need to be examined any further and are therefore labeled "marked".

Procedure 2

1. Let $\mathcal{XV} = \{(x,v) \mid \exists (q,t) \in \mathcal{C}_{12} \cup \mathcal{C}_{21}$ where (x,v) is a compatible communication pair for $(q,t)\}$ be the set of all compatible communication pairs for the control communication pairs of $\mathcal{C}_{12} \cup \mathcal{C}_{21}$. Let $\mathcal{XV}_2 = \{(x,v) \mid (x,v) \in \mathcal{XV}$ and $v \in \overline{s'}$, where s' is a communication sequence for some $(q',t') \in \mathcal{C}_{12} \cup \mathcal{C}_{21}\}$. That is, $\mathcal{XV}_2$ is the set of all (x,v) that give rise to a dependency of a control communication pair (q',t') on another control communication pair (q,t).

2. Initialize D, for all $(q,t),(q',t') \in \mathcal{C}_{12} \cup \mathcal{C}_{21}$, as follows:

$$D[(q,t),(q',t')] = \begin{cases} (x,v) & \text{if } (q,t) \text{ depends on } (q',t') \text{ via } (x,v) \in \mathcal{XV}_2; \\ \emptyset & \text{otherwise.} \end{cases}$$

3. Detect and resolve cycles in D using function *DetectAndBreakCycles*.

4. Denote all control communication pairs $(q,t) \in \mathcal{C}_{12}$ and $(q',t') \in \mathcal{C}_{21}$ "unmarked".

5. If all entries in a row of D are $\emptyset$, denote the corresponding control communication pair "marked".

6. While there still exist "unmarked" control communication pairs:

   (a) Choose "unmarked" $(q,t) \in \mathcal{C}_{12} \cup \mathcal{C}_{21}$ such that all the non-empty elements in the corresponding row of D are "marked". At least one such (q,t) exists since all cycles in D have been broken.

   (b) For each non-empty entry in column (q',t') of row (q,t), compare the pattern of non-empty and empty elements in row (q,t) and row (q',t') of D—omitting the elements in columns (q,t) and (q',t'). If the occurrence of non-empty and empty elements in two rows does not coincide then
   $$D[(q,t),(q',t')] = \emptyset.$$

   (c) Denote (q,t) as "marked".

7. Initialize $\mathcal{C}_{12}^{compat} = \mathcal{C}_{21}^{compat} = \emptyset$. These are sets that store the compatible communication pairs (x,v) that will be added to G^{com}. Update $\mathcal{C}_{ij}^{compat}$ (for $i,j \in \{1,2\}$ and $i \neq j$) as follows: if $D[(q,t),(q',t')] = (x,v)$ and $P_i^{\mathbf{c}}(v^{\mathbf{c}}) = P_i^{\mathbf{c}}(s'^{\mathbf{c}})$, where s' is the communication sequence for (q',t'), then
   $$\mathcal{C}_{ij}^{compat} = \mathcal{C}_{ij}^{compat} \cup \{(x,v)\}.$$

□ Procedure 2

Procedure 2 identifies compatible communication pairs (x,v), for each control communication pair (q,t) found in Procedure 1, that occur *along* control communication sequences. The purpose of this procedure is to refine—if necessary—a supervisor's local view of communication states in light of any communication it receives from another supervisor prior to reaching a communication state.

This procedure will always terminate because we break any cycles that occur in the dependency graph. In addition, we only *remove* dependencies from D. Thus we do not need to worry about inadvertently introducing new cycles into D when we break existing cycles. At the conclusion of Procedure 2, we have the sets of compatible communication pairs that give rise to dependencies between control communication pairs. We update G^{com}, in light of the entries in $\mathcal{C}_{12}^{compat}$ and $\mathcal{C}_{21}^{compat}$, using $ConsistentG^{com}$, a variation of $BuildG^{com}$ that incorporates the compatible communication pairs into G^{com}.

Function $ConsistentG^{com}$ ($\mathcal{C}_{12}^{compat}$, $\mathcal{C}_{21}^{compat}$)
Input: G^{com}, $\mathcal{C}_{12}^{compat}$, $\mathcal{C}_{21}^{compat}$
Output: G^{com}

1. For each $(x,v) \in \mathcal{C}_{ij}^{compat}$ compatible with $(q,t) \in \mathcal{C}_{ij}$, for $i,j \in \{1,2\}$ and $i \neq j$:

   (a) Create a new state $x^{\mathbf{c}}$. If $x^{\mathbf{c}} \notin Q^{com}$, update the state set: $Q^{com} = Q^{com} \cup \{x^{\mathbf{c}}\}$.

   (b) Update the transition function $\delta^{G^{com}}$. Suppose sequence v has the form $v = u\sigma$ where $\delta^G(u, q_0^G) = x'$ and $\delta^G(\sigma, x') = x$. Then if $\delta^{G^{com}}(\sigma, x') = x$ (i.e., no communication has been added at state x yet) we must first remove this transition from $\delta^{G^{com}}$. The following transitions are then added to $\delta^{G^{com}}$:
   $$\begin{aligned} \delta^{G^{com}}(\sigma, x') &= x^{\mathbf{c}}, \\ \delta^{G^{com}}(com_{ij}{:}q, x^{\mathbf{c}}) &= x. \end{aligned}$$

It could be the case that a communication event representing communication from supervisor i to supervisor j has already been added to state x in G^{com}. That is, more than one communication sequence associated with the elements of $\mathcal{C}_{ij}$ leads to state x. A communication event com_{ij}:q is added to state x only once. Or it could be the case that a communication event representing communication from supervisor j to supervisor i has already been added to state x in G^{com}. If a communication event from supervisor j to supervisor i has been added to state x already (i.e., $\delta^{G^{com}}(\sigma, x') \neq x$), we create a new state $x^{\mathbf{cc}}$ and update Q^{com}:

$$Q^{com} = Q^{com} \cup \{x^{\mathbf{cc}}\}.$$

This situation arises if $(x, v) \in \mathcal{C}_{ji}^{compat} \cap \mathcal{C}_{ij}^{compat}$. Then we remove the following transition from G^{com}:

$$\delta^{G^{com}}(\sigma, x') = x^{\mathbf{c}}.$$

Add the following transitions to $\delta^{G^{com}}$:

$$\delta^{G^{com}}(\sigma, x') = x^{\mathbf{cc}}$$
$$\delta^{G^{com}}(com_{ij}{:}q, x^{\mathbf{cc}}) = x^{\mathbf{c}}.$$

2. Return.

□ **Function** *ConsistentG*^{com}

We make a similar assumption regarding the structure of G as described in section 4.2. We want to add a communication event com_{ij}:q at state x in G^{com} corresponding to an (x, v) identified in Procedure 2. If sequences other than v lead to state x (i.e., there exists $v' \in L(G)$ such that $\delta^G(v', q_0^G) = x$) and these sequences are not associated with a compatible communication pair for (q, t), we want to split state x into x^1 and ^{2}. We split x as follows: for all v such that $\delta^G(v, q_0^G) = x$, if (x, v) is a compatible communication pair for (q, t), update δ^G so that $\delta^G(v, q_0^G) = x^1$; otherwise $\delta^G(v, q_0^G) = x^2$. We assume that the plant G has been redrawn to accommodate all occurrences of the above scenario.

The time complexity for Procedure 2 is dominated, as was Procedure 1, by step 1: finding the set of compatible communication pairs. Once again we can use an $O(n^3)$ dynamic-programming algorithm to reconstruct the paths of these sequences, where n is the number of states in the monitoring automaton. Initializing the matrix in step 2 takes $O(n^2)$ time and we can use a depth-first search algorithm ($O(n+e)$ where e is the number of transitions in the plant) to detect cycles. Breaking cycles simply involves removing an edge. Steps 6 and 7 also take $O(n^2)$ time. Overall, the procedure is, because of step 1, $O(n^3)$. The time complexity of function *ConsistentG*^{com}, like function *BuildG*^{com} on which it is based, is $O(|\mathcal{C}_{12}^{compat} \cup \mathcal{C}_{21}^{compat}|)$.

We want to describe a sequence in $L(G)$ as it appears after completing Procedure 2 followed by *ConsistentG*^{com}. At this point $L(G^{com})$ is the language generated by the G^{com} that results from the completion of Procedure 2 and *ConsistentG*^{com}. We write $\hat{t^{\mathbf{c}}}$ for the sequence in $L(G^{com})$ such that

$$\delta^G(t, q_0^G) = \delta^{G^{com}}(\hat{t^{\mathbf{c}}}, q_0^{G^{com}})$$

and

$$\hat{P}(\hat{t^{\mathbf{c}}}) = t.$$

We abuse terminology and also refer to $\hat{t^{\mathbf{c}}}$ as a communication-equivalent sequence for t.

In Lemma 2 we showed that any communication event added to G^{com} after following Procedure 1 is sufficient to distinguish a control sequence t from its control twin t'. We want to make a similar statement about the distinguishability of t and t' after Procedure 2 is completed. Note that after Procedures 1 and 2, the only sequences affected by the addition of communication events are communication sequences.

Lemma 3 *For a control sequence t and its control twin t' defined with respect to supervisor i (i.e., $P_i(t) = P_i(t')$), after following Procedure 2 and ConsistentG^{com}, $P_i^{\mathbf{c}}(\widehat{t^{\mathbf{c}}}) \neq P_i^{\mathbf{c}}(\widehat{t'^{\mathbf{c}}})$.*

Proof. (By contradiction) Let $P_i(t) = P_i(t')$ and assume $P_i^{\mathbf{c}}(\widehat{t^{\mathbf{c}}}) = P_i^{\mathbf{c}}(\widehat{t'^{\mathbf{c}}})$.

By Lemma 2 there is a com_{ji}:q along $t^{\mathbf{c}}$ that does not appear along $t'^{\mathbf{c}}$ (respectively, the event appears along $t'^{\mathbf{c}}$ and not along $t^{\mathbf{c}}$), i.e., (q, t) is an element of $\mathcal{C}_{ji}$ identified in step 3 of Procedure 1.

Suppose that we added com_{ji}:q along $t'^{\mathbf{c}}$ according to step 7 of Procedure 2. Note that an event com_{ji}:q could only get added in one place along $t'^{\mathbf{c}}$ according to Procedure 2. Then the matrix D (representing the dependency graph) has the following entry:

$$D[(q', t'), (q, t'')] = (x, b),$$

where $b \in \overline{t'}$, $\delta^G(b, q_0^G) = x$, t'' is some sequence that passes through q, (x, b) is a compatible communication pair for (q, t'') and therefore $P_j(b) = P_j(s)$, where s is the communication sequence for (q, t'').

We must first determine if we can find such a prefix b of t'. Should b exist, we would add the communication event along t' at state x.

We begin by finding $b \in \overline{t'}$ such that $P_j(s) = P_j(b)$ and such that b satisfies Definition 14. By Procedure 1, a com_{ji}:q added along t'' right after s implies that $s = u\sigma_j$ for some $u \in \Sigma^*$, $\sigma_j \in \Sigma_{j,o}$. From Definition 7 there are two forms for t' we consider when $t'' = u\sigma_j v$.

Case 1. $t'' = u\sigma_j v$ and $t' = u'\sigma_i v'$, where $\sigma_i \in \Sigma_{i,o}$.

Since $s = u\sigma_j$, we want to find $b \in \overline{t'}$ such that $P_j(b) = P_j(u\sigma_j)$.

Claim 4 $b \notin \overline{u'}$.

Proof. This is because if $b \in \overline{u'}$, then $u' = bb'$ for some $b' \in \Sigma^*$:

$$\begin{aligned} P_j(u) &= P_j(u') && \text{(since } (u, u') \text{ is a maximal-P pair)} \\ &= P_j(bb') \\ &= P_j(u\sigma_j)P_j(b') && \text{(since } P_j(s) = P_j(b)) \\ &= P_j(u)P_j(\sigma_j)P_j(b'). \end{aligned}$$

This is only possible if $P_j(\sigma_j)P_j(b') = \varepsilon$, but $\sigma_j \in \Sigma_{j,o}$ so $P_j(\sigma_j) \neq \varepsilon$.

□ *Claim 4*

Since $b \in \overline{t}$ and $P_j(b) = P_j(u\sigma_j)$, we can rewrite this equation as follows:

$$\begin{aligned} P_j(b) &= P_j(u)\sigma_j \\ &= P_j(u')\sigma_j. \end{aligned} \tag{15}$$

Since $b \notin \overline{u'}$, we know from (15) that $b = u'v''\sigma_j v'''$ for some $v'', v''' \in (\Sigma \setminus \Sigma_{j,o})^*$.

For b to satisfy definition 14, it must be the case that $b = u'v''\sigma_j$ since $v''' \in (\Sigma \setminus \Sigma_{j,o})^*$.

Since $b \in \overline{t'}$, $\exists b'' \in \Sigma^*$ such that $t' = bb''$. Therefore

$$t' = u'v''\sigma_j b''. \tag{16}$$

Previously we assumed that

$$t' = u'\sigma_i v'. \tag{17}$$

Equating (16) and (17) we have

$$v''\sigma_j b'' = \sigma_i v'$$

Therefore, the first event in v'' must be σ_i, i.e., $\exists v'''$ such that $v'' = \sigma_i v'''$.

Therefore, $b = u'\sigma_i v'''\sigma_j$ and $t' = u'\sigma_i v'''\sigma_j b''$.

Case 2. $t'' = u\sigma_j v$ and $t' = u'\hat{\sigma}_j v'$, where $\hat{\sigma}_j \in \Sigma_{j,o}$ and $\hat{\sigma}_j \neq \sigma_j$ (since (u, u') is a maximal-P pair). Since $s = u\sigma_j$, we want to find a prefix b of t' such that $P_j(b) = P_j(u\sigma_j)$. As in Claim 4, it can be shown that $b \notin \overline{u'}$.

As in Case 1, since $b \notin \overline{u'}$, $b = u'v''\sigma_j v'''$ for some $v'', v''' \in (\Sigma \setminus \Sigma_{j,o})^*$. Additionally, as in Case 1, we truncate b to satisfy definition 14 so that $b = u'v''\sigma_j$.

Since $b \in \overline{t'}$, $\exists b'' \in \Sigma^*$ such that $t' = bb''$. Therefore

$$t' = u'v''\sigma_j b''. \tag{18}$$

Previously we assumed that

$$t' = u'\hat{\sigma}_j v'. \tag{19}$$

Equating (18) and (19) we have

$$v''\sigma_j b'' = \hat{\sigma}_j v'$$

Thus, the first event in v'' must be $\hat{\sigma}_j$ but $v'' \in (\Sigma \setminus \Sigma_{j,o})^*$. Therefore, no such b can be constructed. Since no such b exists we do not consider this case any further.

By Case 1, we do have a place where the communication event $com_{ji}{:}q$ could be added to t'. We add a communication event just after b occurs. That is, the communication event is added after the last event observable to supervisor j, e.g., after $u'v''\sigma_j$. It remains to be shown that this additional event now leads to a contradiction to the assumption that $\widehat{t''^{\mathbf{c}}}$ and $\widehat{t'^{\mathbf{c}}}$ are indistinguishable.

After Procedure 1 either no communication events were added to t' or some communication events were added to t'—but not the event $com_{ji}{:}q$ since it was only added to t. We consider the effect of adding $com_{ji}{:}q$ after $\hat{b}^{\mathbf{c}}$ in $\widehat{t'^{\mathbf{c}}}$ (i.e., add the event to state x).

Let $\hat{t}^{\mathbf{c}} = \widehat{u^{\mathbf{c}}}\sigma_j com_{ji}{:}q\widehat{v^{\mathbf{c}}}$ and $\widehat{t'^{\mathbf{c}}} = \widehat{u'^{\mathbf{c}}}\sigma_i\widehat{v'''^{\mathbf{c}}}\sigma_j\ com_{ji}{:}q\ \widehat{b''^{\mathbf{c}}}$.

$$P_i^{\mathbf{c}}(\widehat{u^{\mathbf{c}}}\sigma_j\ com_{ji}{:}q\ \widehat{v^{\mathbf{c}}}) = P_i^{\mathbf{c}}(\widehat{u'^{\mathbf{c}}}\sigma_i\widehat{v'''^{\mathbf{c}}}\sigma_j\ com_{ji}{:}q\ \widehat{b''^{\mathbf{c}}})$$

$$P_i^{\mathbf{c}}(\widehat{u^{\mathbf{c}}})P_i^{\mathbf{c}}(\sigma_j)P_i^{\mathbf{c}}(com_{ji}{:}q)P_i^{\mathbf{c}}(\widehat{v^{\mathbf{c}}}) =$$

$$P_i^{\mathbf{c}}(\widehat{u'^{\mathbf{c}}})P_i^{\mathbf{c}}(\sigma_i)P_i^{\mathbf{c}}(\widehat{v'''^{\mathbf{c}}})P_i^{\mathbf{c}}(\sigma_j)P_i^{\mathbf{c}}(com_{ji}{:}q)P_i^{\mathbf{c}}(\widehat{b''^{\mathbf{c}}})$$

$$P_i^{\mathbf{c}}(\widehat{u^{\mathbf{c}}})\ com_{ji}{:}q\ P_i^{\mathbf{c}}(\widehat{v^{\mathbf{c}}}) =$$

$$P_i^{\mathbf{c}}(\widehat{u'^{\mathbf{c}}})\sigma_i P_i^{\mathbf{c}}(\widehat{v'''^{\mathbf{c}}})\ com_{ji}{:}q\ P_i^{\mathbf{c}}(\widehat{b''^{\mathbf{c}}})\ \ (\text{since } \sigma_j \notin \Sigma_{i,o})$$

If $P_i^{\mathbf{c}}(\widehat{u^{\mathbf{c}}}) = P_i^{\mathbf{c}}(\widehat{u'^{\mathbf{c}}})$ then we have a contradiction because $P_i^{\mathbf{c}}(\sigma_i) \neq \varepsilon$. However, if $P_i^{\mathbf{c}}(\widehat{u^{\mathbf{c}}}) \neq P_i^{\mathbf{c}}(\widehat{u'^{\mathbf{c}}})$ then we must show it is not possible for $P_i^{\mathbf{c}}(\widehat{u^{\mathbf{c}}}) = P_i^{\mathbf{c}}(\widehat{u'^{\mathbf{c}}})\sigma_i P_i^{\mathbf{c}}(\widehat{v''''^{\mathbf{c}}})$.

Suppose that $P_i^{\mathbf{c}}(\widehat{u^{\mathbf{c}}}) = P_i^{\mathbf{c}}(\widehat{u'^{\mathbf{c}}\sigma_i v''''^{\mathbf{c}}})$. It should be the case, by Lemma 1, that these sequences look the same with the communication events "erased":

$$\hat{P}_i(\widehat{u^{\mathbf{c}}}) \quad = \quad \hat{P}_i(\widehat{u'^{\mathbf{c}}\sigma_i v''''^{\mathbf{c}}}).$$

By the definition of $\hat{P}_i$ we have

$$\begin{aligned} \hat{P}_i(\widehat{u^{\mathbf{c}}}) &= P_i(u), \\ \hat{P}_i(\widehat{u'^{\mathbf{c}}\sigma_i v''''^{\mathbf{c}}}) &= P_i(u'\sigma_i v''''). \end{aligned}$$

Therefore, we have

$$\begin{aligned} P_i(u) &= P_i(u'\sigma_i v'''') \\ P_i(u) &= P_i(u')P_i(\sigma_i)P_i(v'''') \\ &= P_i(u)P_i(\sigma_i)P_i(v'''') \quad \text{(since } (u, u') \text{ is a maximal-P pair)} \end{aligned}$$

However, this implies that $P_i(\sigma_i) = \varepsilon$ which is not possible since $\sigma_i \in \Sigma_{i,o}$.

□ Lemma 3

Procedure 2 finds compatible communication pairs that lie along communication sequences. The final step, as given further on in Procedure 3, is to find the remaining compatible communication pairs (that may not lie along communication sequences) for the updated version of the control communication pairs.

5.2 Refining local views of compatible communication pairs

If a supervisor's local view of a communication state in the original plant includes states that do not lie along a communication sequence, then we need to determine whether or not these states are still part of the supervisor's local view of the communication state in G^{com}. We identify the compatible communication pairs for each control communication pair and determine whether or not any prior communication along the communication sequence (as identified in Procedure 2) affects the supervisor's view of the compatible communication pairs.

In the course of finding the remaining compatible communication pairs of G^{com}, we will want to discuss dependencies between compatible communication pairs and control communication pairs:

Definition 16 *A compatible communication pair* $(\boldsymbol{x}, \boldsymbol{v})$ *for control communication pair* (q, t) **depends on** *control communication pair* $(\boldsymbol{q'}, \boldsymbol{t'})$ *if we can find a compatible communication pair* (x', v') *for* (q', t') *such that, for some* $w \in \Sigma^*$, $v = v'w$ *and* $\delta^G(w, x') = x$.

Figure 8 illustrates the "depends on" relationship.

Our strategy amounts to identifying all the remaining compatible communication pairs (x, v) for all control communication pairs (q, t). We subsequently determine if a given compatible communication pair depends on any control communication pairs. If the dependencies for (x, v) match the dependencies in row (q, t) of D, then we add the appropriate communication event to state x in G^{com}.

We build another dependency graph $\hat{D}$ and refer to it only in its adjacency matrix form. $\hat{D}$ is an $n_3 \times (n_1 + n_2)$ matrix where n_3 is the number of compatible communication pairs in $\mathcal{XV} \setminus \mathcal{XV}_2$

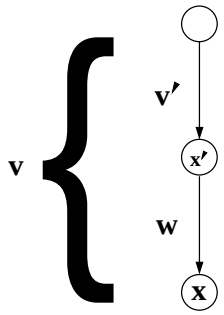

Figure 8: Dependency in the context of compatible communication pairs: (v, x) and (v', x') are compatible communication pairs for (q, t) and (q', t'), respectively. We say (v, x) depends on (v', x').

and n_1 and n_2 are still the number of control communication pairs in $\mathcal{C}_{12}$ and $\mathcal{C}_{21}$, respectively. Let $\mathcal{XV}_3 = \mathcal{XV} \setminus \mathcal{XV}_2$. A row in $\hat{D}$ corresponds to a compatible communication pair $(x, v) \in \mathcal{X}V_3$. A non-empty entry in row (x, v) of $\hat{D}$ means that there is a possible communication event that occurs along sequence v but before the system reaches state x. Suppose that (x, v) was compatible with control communication pair (q, t) before considering the existence of earlier communication events along s. If row (x, v) of $\hat{D}$ has the same pattern of empty and non-empty entries as row (q, t) in D, (x, v) may still be compatible with (q, t). To verify that (x, v) is compatible with (q, t) (where $(q, t) \in \mathcal{C}_{ij}$) we must make certain that any communication events that occur before v and s occur in the same order and that the sequences still have the same projection for supervisor i. The communication events corresponding to the pairs (x, v) that survive this culling process are added to G^{com}.

This procedure is used to determine which sequences—other than the communication sequences—will contain additional communication events. As was the case for Procedure 2, Procedure 3 terminates when all sequences v in the set of compatible communication pairs $\mathcal{XV}_3$ have been examined, i.e., when all these compatible communication pairs are "marked" (Steps 3 and 4(b)). Note that in Step 3, a sequences v does not contain any additional communication events in G^{com} if, after the initialization of $\hat{D}$ in Step 2, the corresponding row in $\hat{D}$ contains all $\emptyset$ entries *and* the corresponding row in D also contains all $\emptyset$ entries. That is, if the communication sequence contains no communication events, neither does any sequence v from which it is indistinguishable to a given supervisor. The same communication event (e.g., com_{ij}:q if $(q, t) \in \mathcal{C}_{ij}$) will be added to G^{com} after s occurs and after v occurs but no other communication events appear along s or v. Such sequences do not need to be examined any further and are therefore labeled "marked".

Procedure 3

1. Let $\mathcal{XV}_3 = \mathcal{XV} \setminus \mathcal{XV}_2$. Denote all elements of $\mathcal{XV}_3$ to be "unmarked".

2. Initialize $\hat{D}$, for all $(x, v) \in \mathcal{XV}_3$, as follows:

$$\hat{D}[(x, v), (q', t')] = \begin{cases} (x', v') & \text{if } \exists (q', t') \in \mathcal{C}_{12} \cup \mathcal{C}_{21} \text{ such that } (x, v) \text{ depends on } (q', t') \text{ and} \\ & (x', v') \text{ is the associated compatible communication pair for } (q', t'); \\ \emptyset & \text{otherwise.} \end{cases}$$

3. For (x, v) that is a compatible communication pair for control communication pair (q, t), if

all entries for row (x, v) in $\hat{D}$ are $\emptyset$ and all entries for row (q, t) in D are $\emptyset$, then $\mathcal{C}_{ij}^{compat} = \mathcal{C}_{ij}^{compat} \cup \{(x, v)\}$. Denote (x, v) "marked".

4. While there remain "unmarked" compatible communication pairs, choose "unmarked" (x, v) such that all the non-empty entries in the corresponding row of $\hat{D}$ are "marked":

   (a) For each non-empty entry of row (x, v), if $\hat{D}[(x, v), (q, t)] = (x', v')$ and $(x', v') \notin \mathcal{C}_{12}^{compat} \cup \mathcal{C}_{21}^{compat}$:
   $$\hat{D}[(x, v), (q, t)] = \emptyset.$$

   (b) Denote (x, v) "marked".

5. For each $(x, v) \in \mathcal{XV}_3$ and $(x, v) \notin \mathcal{C}_{ij}^{compat}$, where (x, v) is a compatible communication pair for $(q, t) \in \mathcal{C}_{ij}$ ($i, j \in \{1, 2\}$ and $i \neq j$): if the pattern of empty and non-empty entries for row (x, v) in $\hat{D}$ coincide with row (q, t) in D *and* $P_i^{\mathbf{c}}(\widetilde{v^{\mathbf{c}}}) = P_i^{\mathbf{c}}(\widetilde{s^{\mathbf{c}}})$ then
   $$\mathcal{C}_{ij}^{compat} = \mathcal{C}_{ij}^{compat} \cup \{(x, v)\}.$$

□ Procedure 3

Procedure 3 identifies compatible communication pairs (x, v), for each control communication pair (q, t) found in Procedure 1, that occur *along* any other sequences in the plant—with the exception of the communication sequences. As before, once more compatible communication pairs are identified, additional communication events must be incorporated using function $ConsistentG^{com}$ with the now-completed sets $\mathcal{C}_{12}^{compat}$ and $\mathcal{C}_{21}^{compat}$.

As was the case for Procedure 2, when a compatible communication pair (x, v) is identified for a control communication pair (q, t), with which we associate the communication event com_{ij}:q, the communication event that is added to G^{com} at state x is com_{ij}:q. This will be an important feature of the construction of G^{com} which ensures that the communication protocols of the supervisors are well-defined.

The time complexity of Procedure 3, like its predecessors, is also $O(n^3)$. This is because using $\mathcal{XV}_3$ in step 1 means that we have to calculate $\mathcal{XV} \setminus \mathcal{XV}_2$. The complexity of finding $\mathcal{XV}_2$ is $O(n^3)$. The other steps of the procedure involve checking matrices and can be accomplished in $O(n^2)$ time.

We want to describe a sequence in $L(G)$ as it appears after following Procedure 3 and $ConsistentG^{com}$. At this point $L(G^{com})$ is the language generated by the G^{com} that results from the completion of Procedure 3 and $ConsistentG^{com}$. We write $\widetilde{t^{\mathbf{c}}}$ for the sequence in $L(G^{com})$ such that
$$\delta^G(t, q_0^G) = \delta^{G^{com}}(\widetilde{t^{\mathbf{c}}}, q_0^{G^{com}})$$
and
$$\hat{P}(\widetilde{t^{\mathbf{c}}}) = t.$$
We abuse terminology and also refer to $\widetilde{t^{\mathbf{c}}}$ as a communication-equivalent sequence for t.

In Lemma 2 we noted that when supervisor i could not distinguish between control sequence t and its control twin t', incorporating a communication event according to Theorem 1 renders the respective communication-equivalent sequences distinguishable. In Lemma 3 we showed that adding additional communications identified by Procedure 2 preserved this distinguishability. Now we show that adding additional communication identified by Procedure 3 still preserves the distinguishability of sequences.

Lemma 4 *If $P_i(t) = P_i(t')$ (for control sequence t and its control twin t') and we follow Procedure 3 and ConsistentG^{com} then $P_i^{\mathbf{c}}(\widetilde{t^{\mathbf{c}}}) \neq P_i^{\mathbf{c}}(\widetilde{t'^{\mathbf{c}}})$.*

Proof. (By contradiction.) Let $P_i(t) = P_i(t')$ but $P_i^{\mathbf{c}}(\widetilde{t^{\mathbf{c}}}) = P_i^{\mathbf{c}}(\widetilde{t'^{\mathbf{c}}})$.

Case 1: t' is a control sequence

In Procedure 3, communication events are added only to those sequences that are not control sequences. Since t and t' are control sequences, $\widehat{t^{\mathbf{c}}} = \widetilde{t^{\mathbf{c}}}$ and $\widehat{t'^{\mathbf{c}}} = \widetilde{t'^{\mathbf{c}}}$. By Lemma 3, $P_i^{\mathbf{c}}(\widehat{t^{\mathbf{c}}}) \neq P_i^{\mathbf{c}}(\widehat{t'^{\mathbf{c}}})$. Therefore, $P_i^{\mathbf{c}}(\widetilde{t^{\mathbf{c}}}) \neq P_i^{\mathbf{c}}(\widetilde{t'^{\mathbf{c}}})$.

Case 2: t' is not a control sequence

Procedures 1 and 2 only add communication events to control sequences and thus $\widehat{t'^{\mathbf{c}}} = t'$ (i.e., the communication-equivalent sequence for t' contains no communication events). Because t is a control sequence, $\widetilde{t^{\mathbf{c}}}$ contains at least one communication event. Suppose that the first such communication event is $com_{ji}{:}q$ (i.e., $\widetilde{t^{\mathbf{c}}} = \widetilde{u^{\mathbf{c}}}\sigma_j\ com_{ji}{:}q\ \widetilde{v^{\mathbf{c}}}$, where $\delta^G(u\sigma_j, q_0^G) = q$).

By step 5 of Procedure 3, $com_{ji}{:}q$ is added to $\widetilde{t'^{\mathbf{c}}}$ if $\exists b \in \overline{t'}$ such that $P_j(b) = P_j(s)$, where s is the communication sequence for some control communication pair (q, t'') (where t'' is some sequence that passes through q). The rest of the proof proceeds in a similar way to that of Lemma 3, where the appropriate substitution of $\tilde{}$ is made for $\hat{}$.

□ Lemma 4

6 A Well-defined Communication Protocol

Until now, we have been somewhat vague about what we mean for G^{com} to generate well-defined communication protocols. Here we provide a formal definition:

Definition 17 *A communication protocol G^{com_i} for supervisor i is said to be well-defined if*

$(\forall \sigma \in \Sigma_{ij}^{com})(\forall q^{G^{com_i}} \in Q^{G^{com_i}})$

$\delta^{G^{com_i}}(\sigma, q^{G^{com_i}})$ *is defined* $\Longrightarrow \nexists \sigma' \in ((\Sigma \cup \Sigma^{com}) \setminus \{\sigma\})$ *such that* $\delta^{G^{com_i}}(\sigma', q^{G^{com_i}})$ *is defined,*

where G^{com_i} is the observer automaton of G^{com} for supervisor i.

The communication protocol is determined by calculating the observer automaton of G^{com} with respect to each supervisor. A communication protocol is well-defined when a communication event in G^{com_i} indicates that supervisor i must communicate to supervisor j such that there is no ambiguity in what supervisor i does. Note that it can be shown that if G^{com} is consistent then the protocols generated with G^{com} are well-defined. When a communication event for supervisor i (i.e., $\sigma \in \Sigma_{ij}^{com}$) is defined at one of its local states $q^{G^{com_i}}$, that particular communication event is the only event defined at $q^{G^{com_i}}$.

The way in which we add communication events to G^{com} ensures that our communication protocols are well-defined (because consistency is satisfied). If a system is consistent then if a communication event for supervisor i appears in G^{com} at some state it also appears after all states that are indistinguishable to supervisor i (i.e., all states have the same local view). Then in the observer automaton of G^{com} for supervisor i, the only event defined at this particular local state will be the aforementioned communication event. This is exactly the notion of well-defined communication protocols. In particular, after Procedure 1, for all $(q, t) \in \mathcal{C}_{ij}$, the communication sequence s is followed only by the communication event $com_{ij}{:}q$ (i.e., $\delta^{G^{com}}(com_{ij}{:}q, q^{\mathbf{c}}) = q$). Similarly, after Procedures 2 and 3, if (x, v) is a compatible communication pair for (q, t) that is

added to $\mathcal{C}_{ij}^{compat}$, it is because $P_i^{\mathbf{c}}(\widetilde{v^{\mathbf{c}}}) = P_i^{\mathbf{c}}(\widetilde{s^{\mathbf{c}}})$, where $\widetilde{v^{\mathbf{c}}}, \widetilde{s^{\mathbf{c}}} \in L(G^{com})$. The only event defined at $q^{\mathbf{c}}$ and any state that supervisor i finds indistinguishable from $q^{\mathbf{c}}$ is com_{ij}:q. When supervisor i reaches the state in G^{com_i} that represents its local view of $q^{\mathbf{c}}$, the only event that can occur is one in which it must communicate its local view to supervisor j, thereby satisfying our definition of what it means to construct a well-defined communication protocol.

7 Return to the Knowledge Domain

We use the plant we have augmented with communication events G^{com} along with the updated legal automaton E^{com} to build a new interpreted system $\mathcal{I}^{cDES}$. The transition function $\delta^{E^{com}}$ of E^{com} is characterized by the transitions in G^{com} that lead to states in Q^E. The automaton E^{com} thus constructed is a sub-automaton of G^{com}. Note that the legal language $L(E^{com})$ contains the communication-equivalent sequences for all the sequences in $L(E)$:

$$L(E^{com}) := \{\widetilde{t^{\mathbf{c}}} \mid (\exists t \in L(E))\ \widetilde{t^{\mathbf{c}}} \text{ is the communication-consistent sequence for } t\}.$$

As with $\mathcal{I}^{DES}$, the set of worlds in $\mathcal{I}^{cDES}$ are defined by the state-based evolution of the sequences in $L(G^{com})$. The updated set of primitive propositions Φ^{cDES} includes the propositions from $\mathcal{I}^{DES}$, and propositions corresponding to the communication events in Σ^{com}:

$$\Phi^{cDES} = \Phi \cup \Phi^{com},$$

where Φ^{com} are the propositions that represent the communication events in Σ^{com}. Following the construction of Φ^{DES} in [15], to form Φ^{cDES} we associate with each $\sigma \in \Sigma^{G^{com}}$ two distinct propositions: one to represent the fact that at a particular state in the plant the event is defined (i.e., is possible), and the other to represent the fact that at the corresponding state in the legal automaton state the event is defined. If $\Sigma^{G^{com}}$ is finite, it can be written as $\Sigma^{G^{com}} = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$. We let $\Phi^{cDES} = \{\sigma_i^G, \sigma_i^E | i = 1, \ldots, n\}$. As before, we partition Φ^{cDES} into two disjoint sets: $\Phi_G^c = \{\sigma_i^G | i = 1, \ldots, n\}$ and $\Phi_E^c = \{\sigma_i^E | i = 1, \ldots, n\}$ where Φ_G^c and Φ_E^c are sets containing $|\Sigma^{G^{com}}|$ symbols. To associate σ_j^G with its counterpart σ_j^E, we extend the relation R_Σ from section 2.1 and define a relation $\mathsf{R}_{\Sigma^{G^{com}}}$ such that $\mathsf{R}_{\Sigma^{G^{com}}} \subseteq \Phi_G^{\mathbf{c}} \times \Phi_E^{\mathbf{c}}$ and $\mathsf{R}_{\Sigma^{G^{com}}} := \{(\sigma_G, \sigma_E) | \exists \sigma_i \in \Sigma^{G^{com}}$ where $\sigma_G = \sigma_i^G, \sigma_E = \sigma_i^E\}$.

The interpretation for the propositions in Φ^{cDES} is defined for all $\sigma \in \Sigma^{G^{com}}$:

$$\pi^{cDES}(w^{\mathbf{c}})(\sigma_G) := \begin{cases} \textbf{true} & \text{if } \delta^{G^{com}}(\sigma, w_e)!, \\ \textbf{false} & \text{otherwise.} \end{cases} \tag{20}$$

$$\pi^{cDES}(w^{\mathbf{c}})(\sigma_E) := \begin{cases} \textbf{true} & \text{if } \delta^{E^{com}}(\sigma, w_e)!, \\ \textbf{false} & \text{otherwise.} \end{cases} \tag{21}$$

Because of the way in which events in Σ^{com} are added to G^{com}, either a communication event occurs and is legal or it is undefined. Thus at any global state of $\mathcal{I}^{cDES}$ it will be the case $\forall \sigma \in \Sigma^{com}$:

$$\pi^{cDES}(w)(\sigma_G) = \textbf{true} \text{ and } \pi^{cDES}(w)(\sigma_E) = \textbf{true} \tag{22}$$

or

$$\pi^{cDES}(w)(\sigma_G) = \textbf{false} \text{ and } \pi^{cDES}(w)(\sigma_E) = \textbf{false}. \tag{23}$$

7.1 Kripke-observability after communication is incorporated

We will show that if the Kripke structure based on the plant G is not Kripke-observable (but G, E are both observable with respect to P), after constructing G^{com}, the resulting Kripke structure for G^{com} and E^{com} is Kripke-observable.

Theorem 2 *Given $\mathcal{I}^{DES}(G, E)$ that is not Kripke-observable. If we follow Procedures 1, 2 and 3 and functions BuildG^{com} and ConsistentG^{com} to construct G^{com} and E^{com} and subsequently construct $\mathcal{I}^{cDES}(G^{com}, E^{com})$, then $\mathcal{I}^{cDES}(G^{com}, E^{com})$ is Kripke-observable.*

Proof. (By contradiction)

Recall the definition of Kripke-observability: for all $w \in \mathcal{I}^{cDES}$, for all $(\sigma_G, \sigma_E) \in \mathsf{R}_{\Sigma^{G^{com}}}$ it must be the case that either $(\mathcal{I}^{cDES}, w) \models \neg\sigma_G \vee \sigma_E$ or there exists $i \in \mathsf{G}_\sigma$ such that $(\mathcal{I}^{cDES}, w) \models K_i \neg\sigma_E$.

Suppose that $\mathcal{I}^{cDES}(G^{com}, E^{com})$ is not Kripke-observable.

There must exist $w \in \mathcal{I}^{cDES}$ and $(\sigma_G, \sigma_E) \in \mathsf{R}_{\Sigma^{G^{com}}}$ where $(\mathcal{I}^{cDES}, w) \not\models \neg\sigma_G \vee \sigma_E$ and $(\forall i \in \mathsf{G}_\sigma)(\mathcal{I}^{cDES}, w) \not\models K_i \neg\sigma_E$. That is,

$$(\mathcal{I}^{cDES}, w) \models (\sigma_G \wedge \neg\sigma_E) \tag{24}$$

and for all $i \in \mathsf{G}_\sigma$ there exists $w' \in \mathcal{I}^{cDES}$ such that $w \sim_i w'$ and

$$(\mathcal{I}^{cDES}, w') \models (\sigma_G \wedge \sigma_E). \tag{25}$$

Note that by (22) and (23), it is not possible for σ_G, σ_E to correspond to $\sigma \in \Sigma^{com}$. Therefore, for $i \in \mathsf{G}_\sigma$, $\sigma \in \Sigma \cap \Sigma_{i,c}$.

By definition, if $w \sim_i w'$ then w and w' have the same local state according to agent i. This means that we can find a path in G^{com} that leads to w_e and a path in G^{com} that leads to w'_e such that agent i cannot distinguish between these paths. In particular, let a path that leads to w_e be reconstructed as the sequence $\widetilde{t^{\mathbf{c}}}$ (i.e., $\delta^{G^{com}}(\widetilde{t^{\mathbf{c}}}, q_0^{G^{com}}) = w_e$) while a path that leads to w'_e be the sequence $\widetilde{t'^{\mathbf{c}}}$ (i.e., $\delta^{G^{com}}(\widetilde{t'^{\mathbf{c}}}, q_0^{G^{com}}) = w'_e$). Since $w \sim_i w'$,

$$P_i^{\mathbf{c}}(\widetilde{t^{\mathbf{c}}}) = P_i^{\mathbf{c}}(\widetilde{t'^{\mathbf{c}}}). \tag{26}$$

Let $\widetilde{t^{\mathbf{c}}}$ be the communication-equivalent sequence for $t \in L(G)$:

$$\hat{P}(\widetilde{t^{\mathbf{c}}}) = t \tag{27}$$

and let $\widetilde{t'^{\mathbf{c}}}$ be the communication-equivalent sequence for $t' \in L(G)$:

$$\hat{P}(\widetilde{t'^{\mathbf{c}}}) = t'. \tag{28}$$

Case 1. $P_i(t) \neq P_i(t')$

By (27), it must also be the case that if all communication events are erased from supervisor i's observation of $\widetilde{t^{\mathbf{c}}}$ that this is exactly supervisor i's view of t:

$$\hat{P}(P_i^{\mathbf{c}}(\widetilde{t^{\mathbf{c}}})) = P_i(t). \tag{29}$$

Similarly, by (28)

$$\hat{P}(P_i^{\mathbf{c}}(\widetilde{t'^{\mathbf{c}}})) = P_i(t'). \tag{30}$$

If we apply the $\hat{P}$ operator to both sides of (26) we get (from (29) and (30)) $P_i(t) = P_i(t')$, which contradicts the assumption.

Case 2. $P_i(t) = P_i(t')$

By definition, there must exist $v, v' \in \mathcal{I}^{DES}$ such that $v \sim_i v'$, t leads to state v and t' leads to state v' where v, v' are states in the monitoring automaton A. Since A generates the same language as G, we also know that $\delta^G(t, q_0^G)$ is defined and that $\delta^G(t', q_0^G)$ is defined. In particular, $\delta^G(t, q_0^G) = v_e$ and $\delta^G(t', q_0^G) = v'_e$ where $v_e, v'_e \in Q^G$.

From (24) we know that $\delta^{G^{com}}(\sigma, w_e)$ is defined but that $\delta^{E^{com}}(\sigma, w_e)$ is not defined.

By Observation 1 in section 4.3, we note that if $\delta^{G^{com}}(\widetilde{t^{\mathbf{c}}}, q_0^{G^{com}}) = w_e$ then it is the case that $\delta^G(\hat{P}(\widetilde{t^{\mathbf{c}}}), q_0^G) = w_e$. Thus, using (27), we have $\delta^G(t, q_0^G) = w_e$. Since, from above, $\delta^G(t, q_0^G) = v_e$, we have $v_e = w_e$.

By the construction of G^{com}, since $\delta^{G^{com}}(\sigma, w_e)$ is defined and since $\sigma \in \Sigma$, it must also be the case that $\delta^G(\sigma, w_e)$ is defined. Similarly, since $\delta^{E^{com}}(\sigma, w_e)$ is not defined it must be that $\delta^E(\sigma, w_e)$ is not defined. Further, it must be the case (since $v_e = w_e$) that

$$(\mathcal{I}^{DES}, v) \models \sigma_G \wedge \neg\sigma_E. \tag{31}$$

From (25) we know that $\delta^{G^{com}}(\sigma, w'_e)$ is defined and that $\delta^{E^{com}}(\sigma, w'_e)$ is also defined.

Again, we can use Observation 1 to conclude that if $\delta^{G^{com}}(\widetilde{t'^{\mathbf{c}}}, q_0^{G^{com}}) = w'_e$ then $\delta^G(\hat{P}(\widetilde{t'^{\mathbf{c}}}), q_0^G) = w'_e$. Therefore it is also the case that $\delta^G(t', q_0^G) = w'_e$. And since $\delta^G(t', q_0^G) = v'_e$, we have $v'_e = w'_e$.

By the construction of G^{com}, since $\delta^{G^{com}}(\sigma, w'_e)$ and $\delta^{E^{com}}(\sigma, w'_e)$ are defined and since $\sigma \in \Sigma$, it is also the case that $\delta^G(\sigma, w'_e)$ and $\delta^E(\sigma, w'_e)$ are defined. Thus it must be the case that

$$(\mathcal{I}^{DES}, v') \models \sigma_G \wedge \sigma_E. \tag{32}$$

By (31) and (32) and the fact that $v \sim_i v'$, we know that $(\mathcal{I}^{DES}, v) \models \neg K_i \neg\sigma_E$. Since the above reasoning works for all $i \in \mathsf{G}_\sigma$, Kripke-observability fails for $\mathcal{I}^{DES}$ at v—in particular, corresponding to sequences t and t'.

Also by (31) we have $t\sigma \in L(G)$ but $t\sigma \notin L(E)$. Similarly, by (32) we have $t'\sigma \in L(G)$ and $t'\sigma \in L(E)$ so t, t' and σ satisfy the hypothesis of Theorem 1. Therefore we apply Procedures 1 through to 3. By Lemma 4, $P_i^{\mathbf{c}}(\widetilde{t^{\mathbf{c}}}) \neq P_i^{\mathbf{c}}(\widetilde{t'^{\mathbf{c}}})$, which contradicts (26). Therefore there exists $i \in \mathsf{G}_\sigma$ such that $(\mathcal{I}^{cDES}, w') \models K_i \neg\sigma_E$.

□ Theorem 2

8 An example system requiring communication

Figure 9 shows a plant G and a legal automaton E where a control solution cannot be reached unless there is communication between the decentralized supervisors. Events that are observable to supervisor i are subscripted with an i and events with no subscripts are unobservable to both supervisors. When we translate this plant into the knowledge domain, we detect three worlds (in the Kripke structure, which is not shown due to space limitations) where Kripke-observability fails:

1. Supervisor 1 does not know whether to disable c_1 since it cannot distinguish bad state $(14, \{3, 8, 14, 17\}, \{14, 15, 18, 20, 21, 22, 24, 28\})$ from good state $(17, \{3, 8, 14, 17\}, \{11, 17, 23\})$.

2. Supervisor 1 does not know whether to disable c_1 since it cannot distinguish bad state $(24, \{24, 25\}, \{14, 15, 18, 20, 21, 22, 24, 28\})$ from good state $(25, \{24, 25\}, \{1, 6, 7, 8, 9, 12, 13, 16, 19, 25, 29\})$.

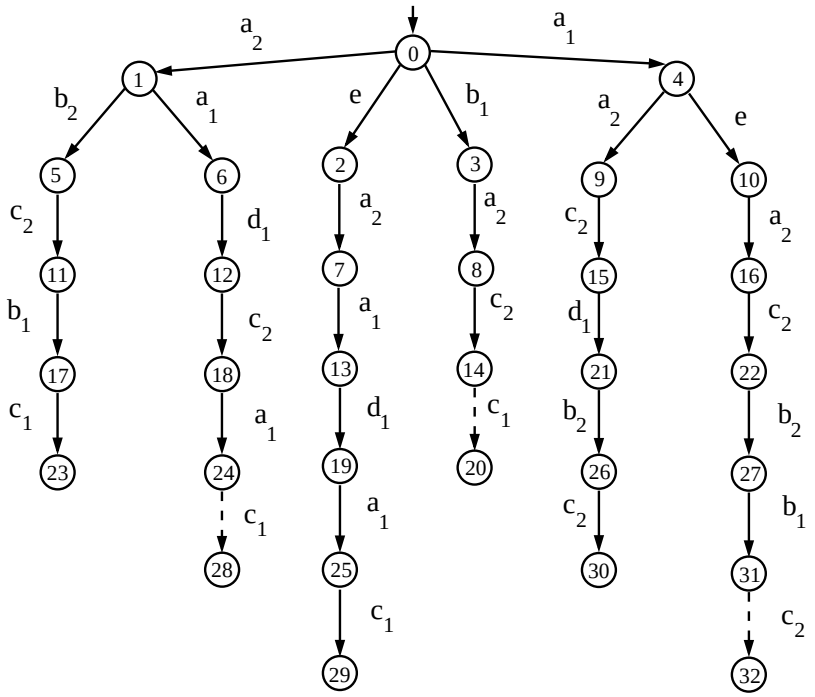

Figure 9: An example requiring communication

3. Supervisor 2 does not know whether to disable c_2 since it cannot distinguish bad state $(31, \{31, 32\}, \{26, 27, 31\})$ from good state $(26, \{12, 18, 19, 21, 26, 30\}, \{26, 27, 31\})$.

Using the monitoring automaton (not shown), we can reconstruct the sequences which a supervisor cannot distinguish without further information. There are three pairs of sequences that lead to $\mathcal{I}^{DES}$ not satisfying Kripke-observability:

$$P_1(b_1a_2c_2) = P_1(a_2b_2c_2b_1); \tag{33}$$

$$P_1(a_2a_1d_1c_2a_1) = P_1(ea_2a_1d_1a_1); \tag{34}$$

$$P_2(a_1ea_2c_2b_2b_1) = P_2(a_1a_2c_2d_1b_2); \tag{35}$$

We use Procedure 1 to identify the control communication pairs for each supervisor. For example, supervisor 1 cannot distinguish the sequences in (34) since $P_1(a_2a_1d_1c_2a_1) = P_1(ea_2a_1d_1a_1) = a_1d_1a_1$. The only maximal-P pair for these sequences is $(a_2a_1d_1, ea_2a_1d_1)$. Supervisor 2 communicates following one of the entries of the maximal-P pair that is immediately followed by an event observable to supervisor 2. Since $a_2a_1d_1$ is followed by c_2 and $ea_2a_1d_1$ is followed by a_1, supervisor 2 communicates at state 18 when it sees a_2c_2 (i.e., after $a_2a_1d_1c_2$ occurs). The control communication pair is therefore $(18, a_2a_1d_1c_2a_1)$, the communication sequence is $a_2a_1d_1c_2$ and the associated control twin is $ea_2a_1d_1a_1$. The complete set of control communication pairs for this example is

$$\begin{aligned} \mathcal{C}_{12} &= \{(21, a_1a_2c_2d_1b_2)\}, \\ \mathcal{C}_{21} &= \{(1, a_2b_2c_2b_1), (18, a_2a_1d_1c_2a_1)\}. \end{aligned} \tag{36}$$

The dependency matrix D has the following row/column assignments:

- row/column 1 corresponds to control communication pair $(21, a_1a_2c_2d_1b_2)$;

- row/column 2 corresponds to control communication pair $(1, a_2b_2c_2b_1)$;
- row/column 3 corresponds to control communication pair $(18, a_2a_1d_1c_2a_1)$.

In Procedure 2, the dependency matrix D is initialized as

$$D = \begin{bmatrix} \emptyset & (9, a_1a_2) & (15, a_1a_2c_2) \\ \emptyset & \emptyset & \emptyset \\ (12, a_2a_1d_1) & \emptyset & \emptyset \end{bmatrix}$$

For example, $D[1,2] = (9, a_1a_2)$ because the communication sequence for the control communication pair $(21, a_1a_2c_2d_1b_2)$ contains the prefix a_1a_2 which has the same projection (according to supervisor 2) as the communication sequence for control communication pair $(1, a_2b_2c_2b_1)$. Note that there is a cycle of length two present between $D[1,3]$ and $D[3,1]$. That is, in the dependency graph, there is an edge from the node representing $(21, a_1a_2c_2d_1b_2)$ to the node representing $(18, a_2a_1d_1c_2a_1)$ and vice versa. Both these rows have the same number of non-empty entries so we arbitrarily choose one of these entries to break the cycle (Step 1(b) of function *DetectandBreakCycles*): $(18, a_2a_1d_1c_2a_1)$. Thus we set $D[3,1] = \emptyset$ (by Step 1(c) of the same function) and observe that there are no more cycles present in D. After breaking the cycle and completing Procedure 2, the dependency matrix becomes

$$D = \begin{bmatrix} \emptyset & (9, a_1a_2) & (15, a_1a_2c_2) \\ \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset \end{bmatrix}$$

Additionally, by Step 7 of Procedure 2, we identify the following entries of D as compatible communication pairs:

- $\mathcal{C}_{12}^{compat} = \emptyset$;
- $\mathcal{C}_{21}^{compat} = \{(9, a_1a_2), (15, a_1a_2c_2)\}$.

To construct $\hat{D}$, it is necessary to first identify the compatible communication pairs for the control communication pairs in $\mathcal{C}_{12}$ and $\mathcal{C}_{21}$ that do not lie along a control sequence. For example, $(7, ea_2)$ is a compatible communication pair for $(1, a_2b_2c_2b_1)$ because $P_2(ea_2) = P_2(a_2)$, where a_2 is the communication sequence for $(1, a_2b_2c_2b_1)$. Each such compatible communication pair is represented by a row in $\hat{D}$ (where the columns correspond to the column entries of D). At the outset of Procedure 3, the rows of $\hat{D}$ are as follows:

- (row 1) $(7, ea_2)$ is compatible with $(1, a_2b_2c_2b_1)$;
- (row 2) $(8, b_1a_2)$ is compatible with $(1, a_2b_2c_2b_1)$;
- (row 3) $(16, a_1ea_2)$ is compatible with $(1, a_2b_2c_2b_1)$;
- (row 4) $(14, b_1a_2c_2)$ is compatible with $(12, a_2a_1c_2a_1)$;
- (row 5) $(22, a_1ea_2c_2)$ is compatible with $(12, a_2a_1c_2a_1)$;
- (row 6) $(19, ea_2a_1d_1)$ is compatible with $(21, a_1a_2c_2d_1b_2)$.

The dependency matrix $\hat{D}$ is initially

$$\hat{D} = \begin{bmatrix} \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset \\ \emptyset & (8, b_1a_2) & \emptyset \\ \emptyset & (16, a_1ea_2) & \emptyset \\ \emptyset & (7, ea_2) & \emptyset \end{bmatrix}$$

The entry $\hat{D}[5,2] = (16, a_1ea_2)$ because the compatible communication pair $(22, a_1ea_2c_2)$ (row 5) contains a prefix a_1ea_2 such that $P_2(a_1ea_2) = P_2(a_2)$, where a_2 is the communication sequence for $(1, a_2b_2c_2b_1)$ (column 2). The first three rows of $\hat{D}$ represent compatible communication pairs for the control communication pair $(1, a_2b_2c_2b_1)$. Note that the corresponding row in D for $(1, a_2b_2c_2b_1)$—row 2—contains all $\emptyset$ entries. Since these first three rows of $\hat{D}$ also have all $\emptyset$ entries, we add each of these compatible communication pairs to $\mathcal{C}_{21}^{compat}$ (Step 3 of Procedure 3). To see if $(14, b_1a_2c_2)$ should be added to $\mathcal{C}_{21}^{compat}$, we first note that its only non-empty entry $(8, b_1a_2)$ has already been examined and that $(8, b_1a_2) \in \mathcal{C}_{21}^{compat}$. Similar arguments hold for $(22, a_1ea_2c_2)$ of row 5. In this example, both rows 4 and 5 of $\hat{D}$ have the same pattern of empty and non-empty entries as row 3 of D. Additionally, when communication events as indicated by the non-empty entries of $\hat{D}$ are added to these two sequences, the projection of the sequences with respect to supervisor 2 is equal to that of the updated communication sequence, namely, a_2 com_{21}:1 a_1c_2. For the last row of $\hat{D}$, we keep the entry $\hat{D}[7,2]$ since it is already in $\mathcal{C}_{21}^{compat}$; however, when we compare the empty and non-empty entries of row 6 to the corresponding row in D (row 1), the pattern of entries is dissimilar. Thus $(19, ea_2a_1d_1)$ is not added to $\mathcal{C}_{12}^{compat}$.

The final set of compatible communication pairs for this example is

- $\mathcal{C}_{12}^{compat} = \emptyset$;
- $\mathcal{C}_{21}^{compat} = \{(9, a_1a_2), (15, a_1a_2c_2), (7, ea_2), (8, b_1a_2), (16, a_1ea_2), (14, b_1a_2c_2), (22, a_1ea_2c_2)\}$.

Note that the sets of control communication pairs and the sets of compatible communication pairs do not form a minimal communication set. For instance, if supervisor 2 communicated when the plant is at state 18, the compatible communication at state 14 would be enough to allow supervisor 1 to distinguish between the sequence leading to state 17 and the sequence leading to state 14.

9 Minimal Communication

In section 4 we discussed our strategy for constructing a set of control communication pairs where one supervisor communicates to another supervisor to solve the control problem. Although communication at every $(q,t) \in \mathcal{C}_{ij}$ will lead to supervisor j making all its correct control decisions, it may be that we can eliminate extraneous communication. That is, some subset of control communication pairs (i.e., $\tilde{\mathcal{C}}_{ij} \subset \mathcal{C}_{ij}$) may lead to a control solution.

What do we mean when we say that an element of $\mathcal{C}_{ij}$ represents extraneous communication? One of our communication goals is to communicate enough information to allow each supervisor to distinguish a bad state from an indistinguishable good state(s). We choose each $(q,t) \in \mathcal{C}_{ij}$ so that bad states can be distinguished from good states. It could be the case that the communication for (q,t) is necessary to allow supervisor j to distinguish a bad state from several sets of look-alike good

states. Or it is possible that a compatible communication pair $(x, v) \in \mathcal{C}_{ij}^{compat}$ occurring prior to (q, t) (where (x, v) is not a compatible pair for (q, t)) provides supervisor j with enough information to make the correct control decision. We want to find a set of communications that does not contain extraneous communication pairs. In addition, we want the the set of communications to satisfy consistency for G^{com}. We seek a set of *minimal* communications.

The notion of minimality for a set of communications was introduced in [16] where a set of communications is minimal when it is the case that if any one event occurrence is not communicated from one supervisor to another, the supervisors will not be able to achieve their objectives. We adopt this notion of minimality for examining communication in decentralized discrete-event control problems. The objectives of supervisors in our system are to solve the control problem and satisfy consistency.

Definition 18 *Let $\mathcal{C} := \mathcal{C}_{12} \cup \mathcal{C}_{21}$ and $\mathcal{C}^{compat} := \mathcal{C}_{12}^{compat} \cup \mathcal{C}_{21}^{compat}$. A set of communication pairs $\mathcal{C} \cup \mathcal{C}^{compat}$ for a consistent system G^{com} that solves the control problem is said to be* **minimal** *if $\nexists\ (a, b) \in \mathcal{C} \cup \mathcal{C}^{compat}$ such that $(\mathcal{C} \cup \mathcal{C}^{compat}) \setminus \{(a, b)\}$ also solves the control problem and renders G^{com} consistent.*

Were such an extraneous (a, b) to exist, it would mean that either by the time the plant reached state a one of the supervisors already had enough information to solve the control problem or (a, b), which had been added because it was compatible with a control communication pair (q, t), is no longer indistinguishable from (q, t) after other communication pairs were added to the system.

9.1 An algorithm for minimal communication

Our algorithm for minimal communication uses a "greedy" strategy to optimize our original set of control communication pairs by removing those we deem extraneous. Optimizing this set amounts to removing communication that is not necessary to solve the control problem (i.e., remove (q, t) from $\mathcal{C}$). We then must ensure that the final set of communications also contains all communication pairs that are compatible with the optimized set of control communication pairs.

Greedy algorithms are used as a technique for solving optimization problems [2]. A greedy algorithm proceeds by choosing, at every step, a particular entry in a set of *candidates* that will maximize the user-defined criteria for selection. At each step of a greedy algorithm a "best" or maximum candidate is selected and is never exchanged. Thus we must ensure that our selection function chooses the control communication pair that will optimize our solution at that step. If this selected candidate produces a feasible solution (i.e., can we eventually reach a solution if we choose this value now?) then add the candidate to a final set and continue until a solution has been reached, or all the candidates have been examined and no solution was achieved.

The goal of solving our decentralized control problem is to have supervisors distinguish between certain "good" states and "bad" states hence making all the correct control decisions while satisfying consistency. After following Procedures 1, 2 and 3 we have a set of communication pairs that, when incorporated into G^{com}, will allow supervisors to solve the control problem. (As noted previously, G^{com} is already consistent.) However, it may be the case that the presence of one of the control communication pairs along with its compatible communication pairs allows a supervisor to distinguish between additional "good" and "bad" states and makes the inclusion of another control communication pair redundant. The framework of our greedy algorithm is based on an algorithm presented in chapter 3 of [2].

Under what circumstances could a (q, t) allow a supervisor to distinguish more than one set of good and bad states in our state-based system? Let $(q, t) \in \mathcal{C}_{ij}$ be a control communication

pair chosen to allow supervisor j to distinguish states along sequence t from those along its control twin t'. Further, let $(q'', t'') \in \mathcal{C}_{ij}$ be a communication that distinguishes the states along t'' from those along its control twin t'''. There are three scenarios where (q, t) could allow supervisor j to distinguish more than just the states along t and t'.

1. Suppose the communication sequence s for (q, t) is a prefix of the communication sequence for (q'', t''). In addition, let communication at q (after s occurs) be sufficient to allow supervisor j to distinguish not only t from t', but also distinguish t'' from t'''. Then communication at (q'', t'') would be unnecessary.

2. Suppose (q'', t'') depends on (q, t). That is, there exists some (x, v) that is compatible with (q, t) such that $v \in \overline{s''}$, where s'' is the communication sequence for (q'', t''). If communication at (x, v) allows supervisor j to distinguish states along t from those along t' and t'' from t''', then additional communication at (q'', t'') would be unnecessary.

3. Suppose (x, v), a compatible communication pair for (q, t), is such that $v \in \overline{t'''}$. Again, if communication at (x, v) allows supervisor j to distinguish t'' from t''', then additional communication at (q'', t'') would be unnecessary.

We introduce a set New of the form $\{(q_1, t_1), (q_2, t_2), \ldots, (q_n, t_n)\}$, i.e., the elements of New are control communication pairs. If an element (q, t) is in the set New, this indicates that sequence t needs to be distinguished from its control twin t'. Initially this set is precisely $\mathcal{C}$. The set $FinalCom$ is the set of communication pairs that constitute the optimized output from the greedy algorithm. Initially this set is $\emptyset$.

Before discussing our greedy algorithm, we first describe what characteristics of our candidate set we want to use to select an optimal subset.

The intuition behind our selection strategy, presented in Algorithm 1, is that we want to count the number of "good" and "bad" pairs of sequences that can be distinguished by communication at a given $(q, t) \in \mathcal{C}$ and at all (x, v) compatible with (q, t). In particular, we want to know how many control sequences, as represented by elements in New, can be distinguished from their control twins by communication associated with (q, t) and its corresponding set of compatible communication pairs. After all the candidates are examined, we will choose the control communication pair that allows a given supervisor to distinguish the largest number of control communication sequences from their control twins (as represented by the elements of New). The control communication pair (q, t) and any of its compatible communication pairs that are necessary to solve part of the control problem are stored in the set *control_com*. We always include (q, t) in *control_com* even if (q, t) is not in New. If $(q, t) \notin New$ this means that some previously-chosen element of $FinalCom$ or $ControlCom$ also distinguishes t from its control twin. In Algorithm 1 we put (q, t) in *control_com* because we want to keep track of the control communication pair associated with the compatible communication pairs that might also be in *control_com*.

Algorithm 1 is performed for each $(q, t) \in \mathcal{C}$. We want to find out which control sequences and their respective control twins would be distinguished if communication events were added along these sequences at (q, t) and all its compatible communication pairs (x, v).

Algorithm 1 *Selection Strategy*

Input. A control communication pair $(q, t) \in \mathcal{C}$ and New, the set of control communication pairs representing control sequences that either agent i or j cannot distinguish from their control twins with the current set of communication pairs in $FinalCom$.

Output. A set of control communication pairs that agent i or j can distinguish if communication events are added to G^{com} at q and at states x for all compatible communication pairs (x, v) for (q, t)—denoted *distinguish*—and the set, denoted *control_com*, that contains (q, t) and those of its compatible communication pairs (x, v) pairs that allow agent i or j to distinguish the control sequences associated with pairs in *distinguish* from their control twins.

begin
1. **if** $(q, t) \in New$ **then**
   $distinguish \leftarrow \{(q, t)\}$
   else
   $distinguish \leftarrow \emptyset$
2. $control\_com \leftarrow \{(q, t)\}$
3. **for all** $(a, b) \in New$
4. $b' \leftarrow$ control twin for b
5. $\mathcal{X}^{\mathbf{c}}(a, b) \leftarrow \{(x, v) \mid (x, v)$ is compatible with (q, t) and $(v \in \overline{b}$ or $v \in \overline{b'})\}$
6. **for all** $(c, d) \in \mathcal{X}^{\mathbf{c}}(a, b)$
7. **if** $\nexists y, y' \in (c_i \cap c_j)$ (for $y \neq y'$) where if c is a good (resp., bad) state with respect to $b\sigma$ then y is a good (resp., bad) state with respect to $b\sigma$ and y' is a bad (resp., good) state with respect to $b'\sigma$ **then**
8. $distinguish = distinguish \cup \{(a, b)\}$
9. $control\_com = control\_com \cup \{(c, d)\}$
10. **return** *distinguish*, *control_com*

end

At step 5 of the algorithm, we collect all the compatible communication pairs for (q, t) that lie along either a control communication sequence in New or its associated control twin.

Step 7 examines each of the compatible communication pairs of (q, t) in $\mathcal{X}^{\mathbf{c}}(a, b)$. If communication of a supervisor's local state at (c, d) means that b and b' can be distinguished by the appropriate supervisor (i.e., the intersection of the local views of c do not contain both a good and bad state with respect to b, b'), then (a, b) is added to the set of communication pairs that (q, t) distinguishes. Since communication at c allows a supervisor to make the correct control decision about b and b', in step 9 (c, d) is added to the set of communication pairs necessary to solve the overall control problem.

Our greedy strategy for decentralized supervisors is described in Algorithm 2. The set of candidates for this algorithm is the set of control communication pairs $\mathcal{C}$. The algorithm selects a subset of the candidate set that allows the appropriate supervisor to distinguish "good" from "bad" states. Once a candidate is selected and examined, the candidate is removed from $\mathcal{C}$ (step 9). If there are still sequences that remain indistinguishable and the selected candidate (i.e., the one that maximizes Algorithm 1) distinguishes no sequences, then we cannot reach a solution (step 20). If, though, the selected candidate would allow the appropriate supervisor to distinguish some sequences represented by the elements in New, then the candidate is added to the final set of communication pairs and the sequences the candidate distinguishes are removed from consideration. The algorithm continues until either all the control communication pairs have been considered or until there are no more sequences for the supervisor to distinguish.

Algorithm 2 *Greedy Communication*

Input. A set of control communication pairs for each agent: $\mathcal{C}_{12}, \mathcal{C}_{21}$.
Output. Sets of communication pairs $FinalCom_{12}, FinalCom_{21} \subseteq \mathcal{C}$, $\mathcal{X}_{12}^{compat}, \mathcal{X}_{21}^{compat} \subseteq \mathcal{XV}$ that, when incorporated into G^{com} will allow decentralized supervisors to make the correct control decisions.

begin
1. $\mathcal{C} \leftarrow \mathcal{C}_{12} \cup \mathcal{C}_{21}$; $New \leftarrow \mathcal{C}$; $ControlCom_{12} \leftarrow \emptyset$; $ControlCom_{21} \leftarrow \emptyset$;
   $FinalCom_{12} \leftarrow \emptyset$; $FinalCom_{21} \leftarrow \emptyset$
2. **while** $(New \neq \emptyset)$ **and** $(\mathcal{C} \neq \emptyset)$ **do**
3. $(q,t) \leftarrow$ an element of $\mathcal{C}$ maximizing the cardinality of *distinguish* from Algorithm 1
4. **if** $\nexists$ a maximum (q,t) **then**
5. randomly choose one of the maximal (q,t)'s
6. $distinguish_{(q,t)} \leftarrow distinguish$, where *distinguish* is associated with (q,t) selected from step 3 or 5
7. **if** $distinguish_{(q,t)} \neq \emptyset$ and $(q,t) \in \mathcal{C}_{ij}$ (where $i,j \in \{1,2\}$ and $i \neq j$) **then**
8. $control\_com_{(q,t)} \leftarrow control\_com$, where *control_com* is associated with (q,t) selected from step 3 or 5
9. $\mathcal{C} \leftarrow \mathcal{C} \setminus \{(q,t)\}$
10. $New \leftarrow New \setminus distinguish_{(q,t)}$
11. $ControlCom_{ij} \leftarrow ControlCom_{ij} \cup (control\_com_{(q,t)} \setminus \{(q,t)\})$
    $FinalCom_{ij} \leftarrow FinalCom_{ij} \cup \{(q,t)\}$
12. **else return** *"there are no solutions"*.
13. **if** $New = \emptyset$ **then**
    $\mathcal{C} \leftarrow \mathcal{C}_{12} \cup \mathcal{C}_{21}$
14. **for all** $(q,t) \in FinalCom_{ij}$ (where $i,j \in \{1,2\}$ and $i \neq j$)
    Let $distinguish_{(q,t)}^{\mathcal{C}}$ be the output of Algorithm 1 using input of (q,t) and $\mathcal{C}$
    Let $Distinguish_{(q',t')}^{\mathcal{C}} \leftarrow \bigcup_{(q',t') \in FinalCom_{ij} \setminus \{(q,t)\}} distinguish_{(q',t')}^{\mathcal{C}}$
15. **if** $distinguish_{(q,t)}^{\mathcal{C}} \setminus Distinguish_{(q',t')}^{\mathcal{C}} = \emptyset$ **then**
16. $FinalCom_{ij} \leftarrow FinalCom_{ij} \setminus \{(q,t)\}$
17. $ControlCom_{ij} \leftarrow ControlCom_{ij} \setminus \{(x,v) \mid (x,v) \in ControlCom_{ij}$ and (x,v) is a compatible communication pair for $(q,t)\}$
18. $FinalCom_{ij} \leftarrow FinalCom_{ij} \cup ControlCom_{ij}$
19. $\mathcal{X}_{ij}^{compat} \leftarrow \{(x,v) \mid (x,v)$ is compatible with an element of $FinalCom_{ij}\}$
20. **return** $FinalCom_{12}, FinalCom_{21}, \mathcal{X}_{12}^{compat}, \mathcal{X}_{21}^{compat}$
21. **else return** *"there are no solutions"*.
end

The success of the greedy algorithm depends on how we describe the selection of a candidate (q,t) in step 3. A control communication pair (q,t) that maximizes Algorithm 1 is a communication that distinguishes the largest number of control communication pairs in the set New from their respective control twins. By the way that we define the control communication pairs, each (q,t) distinguishes at least t from its control twin. (Although note that this is only relevant if (q,t) is also in the set New.)

It is possible that after step 3 instead of one maximum candidate, we could have several maximal candidate communication pairs (i.e., of the ones that distinguish the largest number of elements). In this case, at steps 4 and 5, we randomly select one of the maximal candidates.

After step 11, $FinalCom_{ij}$ contains all the (q,t) where the communication of a supervisor's local view of q would lead to the other supervisor making the correct control decision. We want to make sure that each element of $FinalCom_{ij}$ (and any of its compatible communication pairs that might be in $ControlCom$) distinguishes at least one control sequence from its control twin that the other elements in $FinalCom_{ij}$ do not. If this is not the case, then we could remove (q,t) from $FinalCom_{ij}$ and still find a control solution. Thus at step 15 we determine which control sequences (as represented by the control communication pairs in the original set $\mathcal{C}_{12} \cup \mathcal{C}_{21}$) would be distinguished from their control twins by the occurrence of the communication event associated with control communication pair (q,t). The set of control communication pairs that correspond to these control sequences is denoted here as $distinguish^{\mathcal{C}}_{(q,t)}$. Note that $distinguish^{\mathcal{C}}_{(q,t)}$ is also the result of passing (q,t) as a parameter to Algorithm 1 during the selection of the candidate that maximizes Algorithm 1 during the first iteration of Algorithm 2. We calculate this set for all the other elements (q',t') in $FinalCom_{ij}$. The union of all these sets is denoted $Distinguish^{\mathcal{C}}_{(q',t')}$. If we remove from $distinguish^{\mathcal{C}}_{(q,t)}$ all those elements that occur in both $Distinguish^{\mathcal{C}}_{(q',t')}$ and $distinguish^{\mathcal{C}}_{(q,t)}$, we are left with the control communication pairs that correspond to the control sequences that can only be distinguished from their control twins when a supervisor communicates its local view of q. If the result is the empty set, then anything that (q,t) distinguishes can already be distinguished by other elements of $FinalCom_{ij}$. Thus (q,t) is removed from $FinalCom_{ij}$ in step 16. In addition, any of its compatible communication pairs are removed from $ControlCom$ (step 17).

Note that step 19 may not have to be calculated since this set may already have been calculated by prior utilization of Procedures 2 and 3. Thus $\mathcal{X}^{compat}_{ij}$ might simply be a subset of $\mathcal{C}^{compat}_{ij}$.

The output of the greedy algorithm, $FinalCom_{ij}$ and $\mathcal{X}^{compat}_{ij}$, is used to create G^{com} as follows. First, set $\mathcal{C}_{ij} = FinalCom_{ij}$ and call function $BuildG^{com}$. Next, set $\mathcal{C}^{compat}_{ij} = \mathcal{X}^{compat}_{ij}$ and call function $ConsistentG^{com}$. The communication protocol for each supervisor is then generated by calculating the observer automata of G^{com}.

In the following we denote $FinalCom = FinalCom_{12} \cup FinalCom_{21}$ and $\mathcal{X}^{compat} = \mathcal{X}^{compat}_{12} \cup \mathcal{X}^{compat}_{21}$.

Theorem 3 *The set of communication pairs $FinalCom \cup \mathcal{X}^{compat}$ obtained from executing Algorithm 2 is a set of minimal communication pairs.*

Proof.(By contradiction)

Suppose that $FinalCom \cup \mathcal{X}^{compat}$ is not a minimal set. Then $\exists (a,b) \in FinalCom \cup \mathcal{X}^{compat}$ such that the control problem can still be solved with $(FinalCom \cup \mathcal{X}^{compat}) \setminus \{(a,b)\}$ and adding communication events to G using $BuildG^{com}$ and $ConsistentG^{com}$ (as described above) with respect to the elements of $(FinalCom \cup \mathcal{X}^{compat}) \setminus \{(a,b)\}$ yields a consistent G^{com}.

Case 1. Remove (a,b) from $FinalCom$.

We must argue that there exists some t, t' that communication at state a distinguishes that no other element in $FinalCom$ or $\mathcal{X}^{compat}$ distinguishes. By step 14 of Algorithm 2, (a,b) represents either (i) a communication event that uniquely distinguishes some t from t' that no other element of $FinalCom$ or $\mathcal{X}^{compat}$ does or (ii) (a,b) looks like another element in $FinalCom$ that *does* uniquely distinguish t, t'. Note that if there is no t, t' that communication at state a uniquely distinguishes, (a,b) would be removed from $FinalCom$ in steps 15 and 16 of Algorithm 2. Thus, if (a,b) satisfies (i)

and is removed from $FinalCom$, the control problem cannot be solved, leading to a contradiction. Similarly, if (a, b) satisfies (ii), that is, (a, b) is a compatible communication pair for an element of $FinalCom$, say (d, e), then the removal of (a, b) means that no communication event will be added to G^{com} for state a in Procedure 3. That is, $\delta^{G^{com}}(d^{\mathbf{c}}, com_{ij}{:}d) = d$ and there will be a state y in $Q^{G^{com}}$ (either $y = a^{\mathbf{c}}$ or $y = a^{\mathbf{cc}}$) that has the same local view as $d^{\mathbf{c}}$ but $\delta^{G^{com}}(y, com_{ij}{:}d)$ is not defined even though $y_i = d_i^{\mathbf{c}}$. This violates our notion of consistency from Definition 12. Therefore the system is no longer consistent, leading to a contradiction.

Case 2. Remove (a, b) from $\mathcal{X}^{compat}$.
By the definition of $\mathcal{X}^{compat}$ in step 18 of Algorithm 2, removing (a, b) means that a communication event will not be added to state a in Procedure 3. Using similar reasoning to that of Case 1, the removal of (a, b) means the system is no longer consistent. This contradicts our assumption.

□ THEOREM 3

Note that if some algorithm other than Procedures 1, 2 and 3 was used to generate a communication solution to the decentralized control problem, then Algorithm 2 could still be used to pare the solution down to a minimal communication set.

We return to the example of figure 9. Our input to the greedy algorithm is the set of control communication pairs in (36). The first control communication pair to maximize *distinguish* of Algorithm 1 is $(18, a_2a_1d_1c_2a_1)$. It is the pair corresponding to a communication that would allow a supervisor to distinguish the largest number of control sequences from their control twins: (33) and (34) are both distinguished by $(18, a_2a_1d_1c_2a_1)$ and its compatible communication pair $(14, b_1a_2c_2)$.

We remove $(18, a_2a_1d_1c_2a_1)$ and $(1, a_2b_2c_2b_1)$ from New and remove $(18, a_2a_1d_1c_2a_1)$ from $\mathcal{C}$. The set $FinalCom_{21}$ now contains $(18, a_2a_1d_1c_2a_1)$ and $ControlCom_{21}$ contains $(14, b_1a_2c_2)$.

The set New now contains only $(21, a_1a_2c_2d_1b_2)$ and it is also the only element in $\mathcal{C}_{12}$ that distinguishes the control sequence associated with itself from its control twin. We now have $FinalCom_{12} = \{(21, a_1a_2c_2d_1b_2)\}$, $FinalCom_{21} = \{(18, a_2a_1d_1c_2a_1)\}$, $ControlCom_{12} = \emptyset$ and $ControlCom_{21} = \{(14, b_1a_2c_2)\}$.

At this point, $New = \emptyset$. Therefore $FinalCom_{12} = \{(21, a_1a_2c_2d_1b_2)\}$ and $FinalCom_{21} = \{(18, a_2a_1d_1c_2a_1), (14, b_1a_2c_2)\}$. It can be shown that each element in $FinalCom$ represents a communication that allows a supervisor to uniquely distinguish at least one control sequence t from its control twin t'.

Finally, we calculate the compatible communication pairs $\mathcal{X}_{12}^{compat} = \emptyset$ and $\mathcal{X}_{21}^{compat} = \{(15, a_1a_2c_2), (22, a_1ea_2c_2)\}$. Note that this solution constitutes a minimal set of communications: if we remove any control communication pair from $FinalCom$ we would not be able to solve the control problem and if we remove any elements from $\mathcal{X}^{compat}$ the system is no longer consistent.

10 Conclusions

Previously in [15] we used our knowledge model to identify when there is insufficient knowledge to reach the correct control solution. Understanding what it means for a supervisor to have sufficient knowledge to solve the control problem allowed us to determine a strategy for communication whereby a supervisor has enough information to make correct control decisions. We used the underlying structure of our knowledge model to identify locations for agents to communicate. More communication injects knowledge into the system, which allows supervisors to solve decentralized control problems that could not be solved without communication.

We are, subject to certain assumptions, able to identify locations in the knowledge model where one supervisor provides the other with enough information to solve the control problem. We find

these locations (from Theorem 1) based on an understanding of the underlying structure of the plant language. We are currently developing equivalent tests to find communication locations based solely on the propositions in modal logic. That is, we would like to be able to describe when a supervisor either knows it should communicate or knows that it requires communication. For instance, we would like to be able to reason about whether or not communication at a particular possible world will lead to a supervisor making the correct control decision. Such deductions will require the addition of new primitive propositions that describe whether or not a world will lead to a good or bad state with respect to a given state of the environment. Finally, work in progress includes a proof of our earlier conjecture that only a finite number of t and t' pairs needs to be considered when calculating control communication pairs.

References

[1] G. Barrett and S. Lafortune. On the synthesis of communicating controllers with decentralized information structures for discrete-event systems. In *Proceedings of IEEE Conference on Decision and Control*, pages 3281–3286, 1998.

[2] G. Brassard and P. Bratley. *ALGORITHMICS Theory and Practice*. Prentice Hall, Englewood Cliffs, 1988.

[3] C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer, Boston, 1999.

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

[5] R. Cieslak, C. Desclaux, A. S. Fawaz, and P. Varaiya. Supervisory control of discrete-event processes with partial observations. *IEEE Transactions on Automatic Control*, 33(3):249–260, 1988.

[6] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, 1990.

[7] R. Debouk, S. Lafortune, and D. Teneketzis. Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discrete Event Dynamic Systems: Theory and Applications*, 10(1/2):33–86, 2000.

[8] R. Fagin, J.Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. MIT Press, Cambridge, 1995.

[9] J.Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990.

[10] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, 1979.

[11] S. Lafortune and E. Chen. The infimal closed controllable superlanguage and its application in supervisory control. *IEEE Transactions on Automatic Control*, 35(4):398–405, 1990.

[12] F. Lin and W.M. Wonham. On observability of discrete-event systems. *Information Sciences*, 44:173–198, 1988.

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

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

[15] S.L. Ricker and K. Rudie. Know means no: Incorporating knowledge into discrete-event control systems. *IEEE Transactions on Automatic Control*. To appear.

[16] K. Rudie, S. Lafortune, and F. Lin. Minimal communication in a distributed discrete-event control system. In *Proceedings of the American Control Conference*, pages 1965–1970, 1999.

[17] K. Rudie and J. C. Willems. The computational complexity of decentralized discrete-event control problems. *IEEE Transactions on Automatic Control*, 40(7):1313–1319, 1995.

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

[19] R. Sengupta. Diagnosis and communication in distributed systems. In *Proceedings of the International Workshop on Discrete Event Systems*, pages 144–151, 1998.

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

[21] J.H. van Schuppen. Decentralized supervisory control with information structures. In *Proceedings of the International Workshop on Discrete Event Systems*, pages 36–41, 1998.

[22] K.C. Wong and J.H. van Schuppen. Decentralized supervisory control of discrete-event systems with communication. In *Proceedings of the International Workshop on Discrete Event Systems*, pages 284–289, 1996.