Knowledge and Communication in Decentralized Discrete-Event Control

by

S. L. Ricker

A thesis submitted to the
Department of Computing and Information Science
in conformity with the requirements for
the degree of Doctor of Philosophy

Queen's University
Kingston, Ontario, Canada
December 1999

Copyright © S. L. Ricker, 1999

Contents

List of Tables

List of Figures

Abstract

A formal method for reasoning about knowledge in distributed systems is applied to the analysis of decentralized discrete-event control problems. Solutions to this class of control problems require that controllers achieve their control objectives without communication. A necessary and sufficient condition is given (equivalent to one from existing discrete-event control theory) to describe when decentralized controllers have enough knowledge to find a control solution.

When controllers do not have sufficient knowledge, a solution where controllers may communicate is presented. The relationship between communication and control is difficult. Control decisions may be affected by information a supervisor received from another supervisor. 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. These procedures yield a control solution while ensuring that supervisors communicate in a consistent manner.

Chapter 1

Introduction

We live in a world where technology plays an increasingly crucial role in solving complex problems. It seems somewhat irregular, though, to consider that a machine or any other inanimate object has knowledge. We use machines as a tool to accomplish a task. Yet despite the fact that a machine lacks the ability for introspection and self-awareness during the problem-solving process, it is still possible to describe what that inanimate object knows. For instance, a robot arm that is controlled to perform a task along a car assembly line "knows" when to reject a component. Further, as part of the distributed assembly process, the information that a component has been rejected may be of interest to another sector of the automated process. Having knowledge about a task does not imply any anthropomorphic assumptions of the system. In this dissertation, we are interested in understanding what it means for decentralized—and quite inanimate—agents to "know enough" to solve control problems. In particular, we are interested in a class of problems where communication is necessary to successfully achieve a control solution.

A discrete-event system (DES) is a set of sequences of events that describes the behaviour of a physical process. A change in the system state of the process is not time-driven, but rather, is precipitated by the occurrence of an action or event. A discrete-event control problem arises when we want to restrict the system to performing a specified subset of the overall behaviour of the system. A control solution exists if we can construct an overseer, or *supervisor*, to achieve the set of desired behaviour by either preventing some events from taking place (*disabling* an event) or allowing—but not forcing—others to occur (*enabling* an event).

Decentralized discrete-event problems originate when more than one supervisor is required to ensure that the system avoids undesirable behaviour. For this class of problems no one supervisor has a complete view of the system behaviour. The supervisors must coordinate—without communication—the disabling and enabling of events to realize the desirable or *legal* behaviour. In other words, each supervisor must know enough of what the system is doing to make correct decisions to turn events off or on.

The framework for decentralized discrete-event control that we adopt for this thesis is based on the theory of formal languages [28]. A discrete-event system is viewed as a generator of a formal language and establishing control for the system amounts to determining which sequences in the language should be recognized by each supervisor. Intrinsic to the study of these processes is the informal argument that as long as at least one supervisor *knows* the correct control action to take in preventing illegal behaviour of the system, an overall control strategy may be synthesized.

At present, decentralized control decisions are based solely on what each supervisor observes. A control solution cannot be constructed if after observing some sequence of events there is no supervisor that "knows enough" to disable a particular event. When such a stalemate is reached it means that in isolation a supervisor lacks appropriate information to make the correct control decision. However, if supervisors could access their collective knowledge about the situation—thereby eliminating some of the uncertainty in making the correct control decision—it may be the case that a control strategy can be formulated.

In this thesis, the decentralized discrete-event control framework is recast into knowledge theory [16] where we formally reason about what supervisors need to know to solve control problems. We use this knowledge formalism to characterize the nature of knowledge in a discrete-event control system. Of equal importance is understanding how and when the dissemination of knowledge among supervisors leads to control

solutions for a class of control problems that decentralized supervisory control theory does not at present address. For our purposes, we consider communication between supervisors as a means of improving the knowledge each supervisor possesses.

As noted above, informal reasoning about knowledge is already an integral part of analyzing decentralized supervisory control problems. Thus it seems natural to consider formally what it means for each supervisor to "know enough" to solve a control problem. We use the model of knowledge (based on modal logic) formulated by Halpern and Moses [16] to analyze distributed systems. The model is based on the concept of *possible worlds*. The idea is that an *agent* (equivalent to the notion of an overseer) has only a partial view of the distributed system and may be unable to distinguish different system states from the true state of the system. In the knowledge logic setting, the basic variables correspond to answers for questions like "is fact p true?" or "does an agent know that fact p is true?". An agent's knowledge of the system depends only on its local view of the system behaviour. As an agent acquires knowledge about the system, it considers fewer "worlds" to be possible and the agent is closer to determining the true state of the world. When an agent has insufficient knowledge, the knowledge model allows us to consider the knowledge of *groups* of agents. We consider this aspect of the knowledge model when we want to determine if combining the knowledge of two DES agents will produce "enough" information to reach a control solution.

Decentralized discrete-event control theory lends itself to the analysis of control theoretic behaviour for many distributed systems applications, including manufacturing systems and telecommunication (network) problems. At the outset of this research it was only possible to discuss deriving a control solution for these applications in the absence of communication between supervisors. In the past year, several strategies, including some of the work in this thesis, have been proposed for incorporating communication into decentralized discrete-event control. We review these

models in section 1.1.

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 the correct control decisions are made. In addition, since a supervisor has only a partial view of the system, if it communicates at a particular place in the system, it must also communicate at every place it cannot distinguish from that place. This property of supervisor behaviour is called *consistency* [31]. 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) [31].

The contribution of this thesis is the introduction of a novel framework for reasoning about control and communication in decentralized control problems. In the course of developing knowledge models for DES, we provide necessary and sufficient conditions for the existence of control solutions in the new knowledge model (which correspond to analogous properties in decentralized DES theory). When the condition for finding a control solution fails, we identify particular places where, subject to certain conditions, supervisors could communicate information to other supervisors and achieve the initial objective of reaching a control solution. We also present procedures to incorporate communication into the DES. We show that our procedures

meet the objectives of solving the control problem with communication while satisfying consistency. An algorithm for generating a set of minimal communications with respect to the set generated by our previous procedures is also provided. Finally, we propose an extension to our original knowledge model where places to communicate are determined solely on whether or not an agent knows the other agent needs information to make a control decision.

1.1 Related Work

Nearly twenty years ago, Ramadge and Wonham [28] introduced control-theoretic strategies for discrete-event systems using automata and formal language theory. The control objective was as follows: given a supervisor that sees the full behaviour of the system and given a subset of behaviour (called the *legal* behaviour) that is deemed desirable, produce a control solution that prevents the system from performing anything but the legal behaviour. At present, solutions to discrete-event control problems can be described under conditions of full and partial observability (when a supervisor cannot see all system behaviour) and expressed across a range of system architectures (e.g., hierarchical, modular, decentralized). In addition, the theory has been extended to address control issues for nondeterministic systems. It is also possible to construct control solutions that allow the exact legal behaviour to occur or a subset of behaviour that lies within some tolerance of the legal behaviour. The control problem can also be specified in alternate frameworks such as Petri nets [18] or as vector DESs [21, 22]. In particular, we are interested in decentralized discrete-event control problems expressed as finite-state machines under conditions of partial observation. References for decentralized control without communication include [9, 23, 24, 32, 33, 36, 40]. This collection of work is representative of the research that led to the identification of necessary and sufficient conditions for solving the class of DES problems where no one agent has a complete view of system behaviour. Chapter 2 contains more specific

details about decentralized DES theory.

Recent work has explored the relationship between control and communication in distributed discrete-event systems [3, 30, 31, 35, 39, 41]. As noted previously, the basic idea of this class of problems is that no single agent in the multi-agent system has a complete view of the system behaviour and a given co-operative task cannot be completed without communication among the agents. (The models noted here—with the exception of [39] where three agents are considered—assume the existence of only two decentralized agents.) 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 places where agents could communicate; (ii) establishing who should communicate; and (iii) determining what should be communicated to realize a control solution.

The model proposed in [41] identifies a necessary and sufficient condition for the existence of a solution which is similar to the notion of distributed observability we independently formulated in the context of knowledge theory [30]. In [41] a control solution exists if and only if, after each supervisor—based on its partial view of the system—discloses sequences it considers the plant could have generated, the intersection of these sets does not contain both an illegal and a legal sequence.

A similar condition was introduced in [35] to understand how to perform failure diagnosis in a distributed system. A decentralized diagnostic solution exists if and only if whenever the local observations of a diagnoser at a remote site i is insufficient to make the correct diagnosis, there exists another diagnoser at site j that observes and communicates the information diagnoser i requires.

While the work mentioned so far provided insight into the role that communication might play in decentralized control, a more formal specification of a communication protocol for decentralized supervisors was required. More recent work [3, 39] introduced models for communication that detailed a more specific role for communicating

agents. Interestingly, both approaches utilize information structures from stochastic control to express the control solution.

For the models of [39] and [3] controllers (i) communicate every time an event is observed; (ii) broadcast and exchange their observations; and (iii) communicate the set of sequences they consider could have been generated by the system. The model of [3] proposes an alternate formulation for communication where instead of exchanging sequences observed, controllers exchange state estimates. Of more interest in this approach is the motivation for communication: the notion of a *conflict state*. A conflict state is a place in the system that leads to both a legal and an illegal sequence and further, neither controller is able to determine the correct control decision from its partial view of the system. The communication protocol is thus to communicate to eliminate conflict states. That is, when a controller receives the observations of all the other controllers, it is no longer confounded by the presence of the conflict state and can make the correct control decision. Also discussed is the notion of "optimal" communication, where optimal corresponds to "communicate as late as possible". Because there may be more than one conflict state for a given pair of illegal/legal sequences, there exists a tolerance within which communication could occur. An optimal solution picks the last possible place communication needs to occur to solve the control problem.

The most recent model for communication [31] focuses on agents in a distributed DES performing a monitoring or control task. In this scenario, agents need to know where they are at every step of the system evolution and not which events to enable or disable. This approach differs from the others because the original communication set is refined to eliminate unnecessary communication. A communication is removed from the initial set if the absence of that information prevents an agent from either completing its task (i.e., it no longer knows exactly what state it is in) or violates consistency.

Our work on decentralized control and communication uses the concept of minimality from [31] and our motivation for communication is similar to the idea of avoiding the conflict states in [3]. One significant difference in our approach is that our agents do not exchange observations: a two-way broadcast would occur only if neither agent 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.

Our use of a formal logic to analyze control problems is not novel to DES. Temporal logic has been applied to the study of supervisory control problems [2, 26, 38] and modal logic has been used as the basis for a computer language that simulates discrete-event processes [27]; however, a formal model of knowledge using modal logic has yet to be incorporated into the study of discrete-event control problems. Recently, modal μ-calculus has been introduced into the analysis of hybrid systems [11].

Reasoning about knowledge has been part of the analysis of a variety of applications in the areas of economics [1, 25], computer security [6, 13], distributed database systems [14], robotics [4] and communication protocols [17].

1.2 Outline of the Thesis

The remainder of the thesis is organized as follows. In chapter 2 we review the relevant definitions and results from supervisory control theory and knowledge theory. In chapter 3 we present two knowledge models for analyzing decentralized discrete-event control problems. We provide a necessary and sufficient condition for solving the control problem within this new framework. (This result is analogous to co-observability, a condition needed to solve decentralized DES problems.) We also speculate on the role knowledge and communication might play when this condition is not satisfied. Chapter 4 contains our model for communication and control in decentralized DES.

We provide a description of places where agents could communicate to solve the decentralized control problem, subject to certain assumptions. We also prove that our strategy for incorporating communication into the decentralized framework will both solve the control problem and produce a set of consistent communications. Chapter 4 also contains a greedy algorithm for producing a set of minimal communications. We summarize our results in chapter 5 and describe areas for further research.

Chapter 2

Background and Notation

In this chapter we describe the notation that is necessary to discuss discrete-event control problems and concepts from modal logic. One of the difficulties in bringing together the notation from two established fields is addressing the overlap of symbols used to represent distinctly different concepts. Wherever possible, we have tried to accommodate the more serious notational discrepancies, but also include references that could be consulted for further clarification.

In addition, in sections 2.1.2 and 2.1.3 we describe variations on some standard operators and structures from supervisory control. These new structures will be useful when we want to describe how to avoid generating infinite-state structures for reasoning about knowledge in discrete-event systems.

2.1 Discrete-Event Systems

This dissertation adopts the framework for discrete-event systems as developed by Ramadge and Wonham [28]. A brief review of essential notation is provided in this section. More comprehensive introductions to discrete-event control theory include [7, 28, 29, 37].

2.1.1 Review

In the discrete-event control theory of Ramadge and Wonham [28], the system requiring control (the *plant*) is described as a generator of a formal language (i.e.,

an automaton). The behaviour 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 behaviour of the plant to that of a pre-specified sublanguage (the legal language). The supervisor averts undesirable behaviour 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 behaviour* 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 $L(G) \subseteq \Sigma^*$. A *marked* language of G defines behaviour of the system that corresponds to completed tasks. We do not consider marked languages in this dissertation.

For any strings $t, u, v \in \Sigma^*$, we say that u is a *prefix* of t if $t{=}uv$. 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: $\overline{L} := \{u \in \Sigma^* : u \text{ is a prefix of } t\}$. 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 behaviour 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 [8] and [20]. 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 (see figure 2.1), 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. The initial state is marked with a small entry arrow. Illegal transitions are indicated with a dashed line. That is, the legal automaton E is the collection of solid-line transitions.

Informally, a supervisor is an agent that has the ability to control some events based on a (partial) view of the plant's behaviour. 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.

Formally, a supervisor $\mathcal{S}$ is a pair (T, ψ) in which T is an automaton $T = (X, \Sigma, \xi, x_0)$, where X is a set of states for the supervisor; Σ is the alphabet used

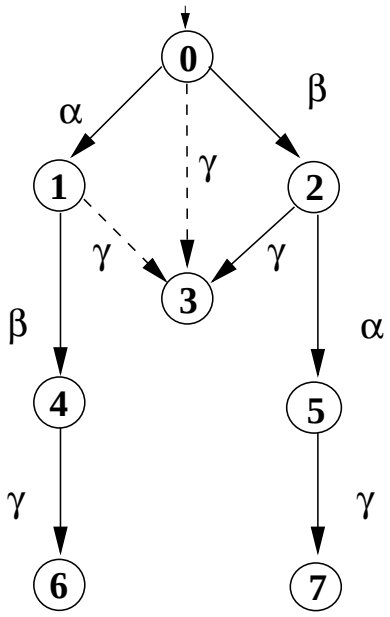

Figure 2.1: A plant G and its legal automaton E.

by G; ξ is the transition function, a partial function $\xi : \Sigma \times X \to X$; x_0 is the initial state for the supervisor; and ψ, called a *feedback map*, is given by $\psi : \Sigma \times X \to \{0, 1\}$ satisfying $\psi(\sigma, x) = 1$ if $\sigma \in \Sigma_{uc}, x \in X$, and $\psi(\sigma, x) \in \{0, 1\}$ if $\sigma \in \Sigma_c, x \in X$. The number 0 is interpreted as the command "disable" and the number 1 as "enable". That is, ψ is interpreted as a rule for disablement such that uncontrollable events are never disabled. The automaton T monitors the behaviour of G and changes state according to the events generated by G. The control rule $\psi(\sigma, x)$ indicates whether σ should be enabled or disabled at the corresponding state in G. The behaviour of G when it is constrained by $\mathcal{S}$ is described by the automaton $\mathcal{S}/G$, called a *supervised discrete-event system*:

$$\mathcal{S}/G = (Q \times X, \Sigma, (\delta \times \xi)^{\psi}, (q_0, x_0)).$$

The behaviour of $\mathcal{S}/G$ is described by $L(\mathcal{S}/G)$. The modified transition function

$(\delta \times \xi)^{\psi}$ is defined as a mapping $\Sigma \times Q \times X \to Q \times X$:

$$(\delta \times \xi)^{\psi}(\sigma,(q,x)) := \begin{cases} (\delta(\sigma,q),\xi(\sigma,x)) \text{ if } \delta(\sigma,q)!, \\ \quad \xi(\sigma,x)!, \text{and } \psi(\sigma,x)=1; \\ \textit{undefined} \text{ otherwise.} \end{cases}$$

The centralized control problem introduced by Ramadge and Wonham[28] is as follows:

> *Given a plant G over an alphabet Σ (with controllable events Σ_c) and given some non-empty language $L(E)$ where $L(E) \subseteq L(G)$ find a supervisor $\mathcal{S}$ such that*
>
> $$L(\mathcal{S}/G) = L(E).$$

This formalism captures problems where we are given some process that can be described as an automaton (in this case, G), and some set of desirable (or "legal") sequences (in this case, $L(E)$), and a controller is sought to inhibit process behaviour so that all and only the desirable sequences are generated.

A variation on the centralized control problem has a supervisor that no longer sees every event in Σ. Instead a supervisor observes events in some subset $\Sigma_o \subset \Sigma$. Supervisory control under partial observation was initiated by Lin and Wonham [23]. To describe a supervisor's view of sequences we use the *canonical projection* P, where P 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:

$$\begin{aligned} P(\varepsilon) &= \varepsilon \\ P(\sigma) &= \varepsilon, \quad \sigma \in \Sigma \setminus \Sigma_o \\ P(\sigma) &= \sigma, \quad \sigma \in \Sigma_o \\ P(t\sigma) &= P(t)P(\sigma), \quad t \in \Sigma^*, \sigma \in \Sigma. \end{aligned} \tag{2.1}$$

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* with respect to G, P if

$$(\forall t, t' \in \Sigma^*)(\forall \sigma \in \Sigma)$$
$$P(t) = P(t') \Rightarrow (t'\sigma \in L \wedge t \in L \wedge t\sigma \in L(G) \Rightarrow t\sigma \in L). \tag{2.2}$$

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. This problem was first studied by Cieslak, et al., [9] and Rudie and Wonham [33]. 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 projection operator of (2.1) is updated as follows: P_i is defined for each supervisor and is a mapping from Σ^* to $\Sigma_{i,o}^*$, for $i = 1, 2$:

$$\begin{aligned} P_i(\varepsilon) &= \varepsilon \\ P_i(\sigma) &= \varepsilon, \quad \sigma \in \Sigma \setminus \Sigma_{i,o} \\ P_i(\sigma) &= \sigma, \quad \sigma \in \Sigma_{i,o} \\ P_i(t\sigma) &= P_i(t)P_i(\sigma), \quad t \in \Sigma^*, \sigma \in \Sigma. \end{aligned} \tag{2.3}$$

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.

Let the two local supervisors acting on G be

$$\mathcal{S}_1 = (T_1, \phi) \text{ and } \mathcal{S}_2 = (T_2, \psi),$$

where $T_1 = (X, \Sigma, \xi, x_0)$ and $T_2 = (Y, \Sigma, \eta, y_0)$. The *conjunction* of $\mathcal{S}_1$ and $\mathcal{S}_2$ is the supervisor

$$\mathcal{S}_1 \wedge \mathcal{S}_2 := (T_1 \times T_2, \phi * \psi),$$

where

$$T_1 \times T_2 := (X \times Y, \Sigma, \xi \times \eta, (x_0, y_0))$$

and $\sigma \in \Sigma$, $x \in X$, $y \in Y \Rightarrow$

$$(\xi \times \eta)(\sigma, x, y) := \begin{cases} (\xi(\sigma, x), \eta(\sigma, y)) & \text{if } \xi(\sigma, x)! \wedge \eta(\sigma, y)! \\ \textit{undefined} & \text{otherwise} \end{cases}$$

$$(\phi * \psi)(\sigma, x, y) := \begin{cases} \textit{disable} & \text{if either } \phi(\sigma, x) = \textit{disable} \text{ or } \psi(\sigma, y) = \textit{disable} \\ \textit{enable} & \text{otherwise.} \end{cases}$$

That is, the composite supervisor $\mathcal{S}_1 \wedge \mathcal{S}_2$ disables an event if either $\mathcal{S}_1$ or $\mathcal{S}_2$ issues a disablement command. When a supervisor $\mathcal{S}$ is the result of a conjunction of two supervisors $\mathcal{S}_1$ and $\mathcal{S}_2$, we write $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$.

It is often convenient in the case of partial observability, to define a supervisor $\mathcal{S}_i$ only in terms of events in $\Sigma_{i,c}$ and $\Sigma_{i,o}$. In this case $\mathcal{S}_i$ can be extended to a supervisor $\widetilde{\mathcal{S}}_i$. The *local* supervisor $\mathcal{S}_i$ acts only on events in $\Sigma_{i,c} \subseteq \Sigma$ and observes events in $\Sigma_{i,o} \subseteq \Sigma$ while $\widetilde{\mathcal{S}}_i$ takes the same control action as $\mathcal{S}_i$ on $\Sigma_{i,c}$, enables all events in $\Sigma \setminus \Sigma_{i,c}$, makes the same transitions as $\mathcal{S}_i$ on $\Sigma_{i,o}$ and stays at the same state for events in $\Sigma \setminus \Sigma_{i,o}$. A supervisor $\widetilde{\mathcal{S}}_i$ that acts on all of Σ and mirrors the control actions of a supervisor $\mathcal{S}_i$ that observes and controls only a subset of Σ is called the *global extension* of $\mathcal{S}_i$.

The decentralized problem we consider is described in [33]:

> *Given a plant G over an alphabet Σ (with controllable events $\Sigma_{1,c}, \Sigma_{2,c} \subseteq \Sigma$ and observable events $\Sigma_{1,o}, \Sigma_{2,o} \subseteq \Sigma$), and an automaton E, where*

$L(E)$ represents legal sequences, $L(E) \subseteq L(G)$ and $L(E) \neq \emptyset$, find local supervisors $\mathcal{S}_1$ and $\mathcal{S}_2$ such that $\widetilde{\mathcal{S}}_1 \wedge \widetilde{\mathcal{S}}_2$ is a supervisor for G and such that

$$L(\widetilde{\mathcal{S}}_1 \wedge \widetilde{\mathcal{S}}_2/G) = L(E). \tag{2.4}$$

Here, for $i = 1, 2$, local supervisor $\mathcal{S}_i$ can observe only events in $\Sigma_{i,o}$ and can control only events in $\Sigma_{i,c}$ and $\widetilde{\mathcal{S}}_i$ is the global extension of $\mathcal{S}_i$. The set of uncontrollable events, Σ_{uc}, is understood to be $\Sigma \setminus (\Sigma_{1,c} \bigcup \Sigma_{2,c})$.

To describe a solution to the above problem, it is convenient to use the notion of *controllability* [28]. Given G over an alphabet Σ, for a language $K \subseteq L(G)$, K is *controllable* with respect to G if

$$\overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K} \tag{2.5}$$

where $\overline{K}\Sigma_{uc} := \{t\sigma \mid t \in \overline{K} \text{ and } \sigma \in \Sigma_{uc}\}$. If we think of K as a set of "legal" sequences, then we want to know when it will be impossible to stop an illegal sequence from happening. It must be that the introduction of an uncontrollable event into a legal sequence results in another legal sequence. Therefore, to solve (2.4), it is necessary that $L(E)$ be controllable. If $L(E)$ is not controllable, a largest (or supremal) controllable sublanguage of $L(E)$ (possibly $\emptyset$), denoted $\sup \underline{C}(L(E), G)$, can always be found [28]. The standard solution to the centralized control problem with full observation produces a supervisor that acts on G to generate $\sup \underline{C}(L(E), G)$. The important point to note is that such a solution is said to be "minimally restrictive" in that the supervisor disables events in G only when absolutely necessary to prevent an illegal sequence from occurring. That is, the largest possible subset of legal sequences is generated.

A necessary and sufficient condition for the solution to the above decentralized problem can be found using the notion of *co-observability*. Given G over an alphabet

Σ, sets $\Sigma_{1,c}, \Sigma_{2,c}, \Sigma_{1,o}, \Sigma_{2,o} \subseteq \Sigma$, projections $P_1 : \Sigma^* \to \Sigma_{1,o}^*$, $P_2 : \Sigma^* \to \Sigma_{2,o}^*$, a prefix-closed language $K \subseteq L(G)$ is *co-observable* with respect to G, P_1, P_2 if

$$\begin{aligned}
&\forall t, t', t'' \in \Sigma^*, P_1(t) = P_1(t'), P_2(t) = P_2(t'') \Rightarrow \\
&\qquad (\forall \sigma \in \Sigma_{1,c} \cap \Sigma_{2,c}) t \in \overline{K} \wedge s\sigma \in L(G) \wedge t'\sigma, t''\sigma \in \overline{K} \Rightarrow t\sigma \in \overline{K} \quad \textbf{conjunct 1} \\
&\wedge \quad (\forall \sigma \in \Sigma_{1,c} \setminus \Sigma_{2,c}) t \in \overline{K} \wedge t\sigma \in L(G) \wedge t'\sigma \in \overline{K} \Rightarrow t\sigma \in \overline{K} \qquad \textbf{conjunct 2} \\
&\wedge \quad (\forall \sigma \in \Sigma_{2,c} \setminus \Sigma_{1,c}) t \in \overline{K} \wedge t\sigma \in L(G) \wedge t''\sigma \in \overline{K} \Rightarrow t\sigma \in \overline{K} \qquad \textbf{conjunct 3}
\end{aligned}$$

Just as with observability for the centralized case, we would like a decentralized supervisor's view of a string to be enough for it to take the correct control action. If both supervisors can control the event in question (i.e., conjunct 1), then we just need one of the supervisors to be able to have an unambiguous view of the strings t, t', t'' to make the correct control decision regarding σ; however, when an event is controlled by only one supervisor (i.e., conjuncts 2 and 3), then that supervisor's view of t, t', t'' must be sufficient to decide on the control action for σ.

It is now possible to discuss the existence of a solution to the decentralized problem. The following theorem (along with its proof) appears as Theorem 4.1 in [33]:

THEOREM **2.1** *There exist supervisors $\widetilde{\mathcal{S}}_1$ and $\widetilde{\mathcal{S}}_2$ that solve the above decentralized supervisory control problem if and only if $L(E)$ is controllable with respect to G and co-observable with respect to G, P_1, P_2.*

Thus we can find decentralized controllers that synthesize $L(E)$ provided that the legal language satisfies the properties of controllability and co-observability. While it is possible to find the supremal controllable sublanguage of $L(E)$, if $L(E)$ is not co-observable there is no unique supremal co-observable sublanguage of $L(E)$.

2.1.2 A Projection Automaton

The projection operator in (2.3) 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) a sequence. An example: 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 a *projection automaton* [34], based on an algorithm in [19] to translate a nondeterministic finite-state automaton into a deterministic finite-state automaton: $P^{G_i} = (Q^{P^{G_i}}, \Sigma_{i,o}, \delta^{P^{G_i}}, q_0^{P^{G_i}})$ where $Q^{P^{G_i}} = 2^{Q^G}$ is the set of states, $\Sigma_{i,o} \subseteq \Sigma$ is the set of events *observable* to agent i (and the set of events unobservable to agent i is $\Sigma_{i,uo}$). The transition function $\delta^{P^{G_i}}$ and initial state $q_0^{P^{G_i}}$ are defined as follows:

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

The initial state $q_0^{P^{G_i}}$ 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.

Figure 2.2(a) contains a plant where $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$, $\Sigma_{1,o} = \Sigma_{1,c} = \{a_1, c\}$ and $\Sigma_{2,o} = \Sigma_{2,c} = \{b_2, c\}$. The projection automata of the plant for each supervisor

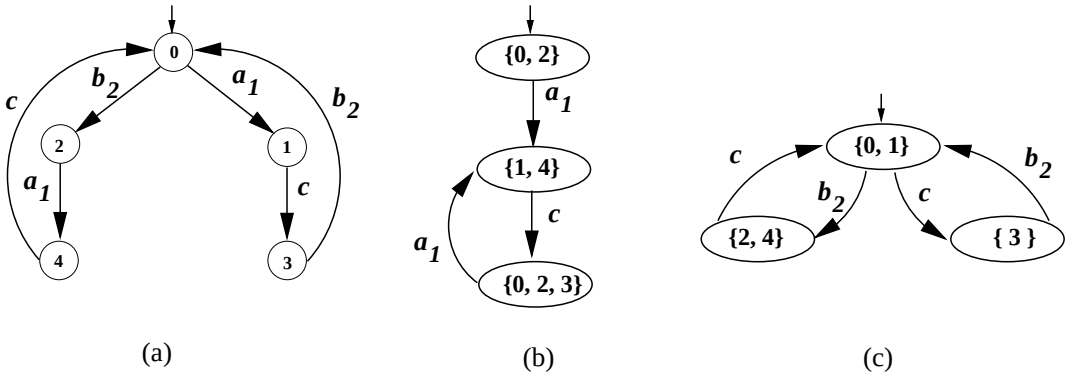

Figure 2.2: The projection automata of a DES plant: (a) the plant G; (b) P^{G_1}; (c) P^{G_2}.

are shown in figure 2.2(b) and (c). For example, because $\mathcal{S}_1$ does not see event b_2 happening, if it initially sees "nothing", then it does not know if the plant is in state 0 or state 2. If no events have yet occurred, then the plant will indeed be in state 0; however, if b_2 happens, $\mathcal{S}_1$ still sees nothing—despite the fact that the plant is really in state 2. Once $\mathcal{S}_1$ observes its first occurrence of event a_1, it still does not know the true plant state. Assume the plant was previously in state 0 and now a_1 happens. The plant is currently at state 1. But $\mathcal{S}_1$ considers it possible that the plant could have been at state 2 before a_1 happened. This uncertainty is reflected in the transition from state $\{0, 2\}$ to state $\{1, 4\}$ via event a_1 in figure 2.2(b).

2.1.3 A Monitoring Automaton

We will also find it necessary to be able 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 [32]. The monitoring automaton is formally defined as follows: $A = (Q^A, \Sigma, \delta^A, q_0^A)$, where $Q^A \subseteq Q^G \times Q^{P^{G_1}} \times Q^{P^{G_2}}$ (Q^A will be fully defined below), the initial state is $q_0^A =$

$(q_0^G, q_0^{P^{G_1}}, q_0^{P^{G_2}})$, 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:

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

$$\delta^A(\sigma, (q^G, q^{P^{G_1}}, q^{P^{G_2}})) = (\delta^G(\sigma, q^G), q^{P^{G_1}}, q^{P^{G_2}}),$$

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

$$\delta^A(\sigma, (q^G, q^{P^{G_1}}, q^{P^{G_2}})) = (\delta^G(\sigma, q^G), \delta^{P^{G_1}}(\sigma, q^{P^{G_1}}), q^{P^{G_2}}),$$

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

$$\delta^A(\sigma, (q^G, q^{P^{G_1}}, q^{P^{G_2}})) = (\delta^G(\sigma, q^G), q^{P^{G_1}}, \delta^{P^{G_2}}(\sigma, q^{P^{G_2}})),$$

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

$$\delta^A(\sigma, (q^G, q^{P^{G_1}}, q^{P^{G_2}})) = (\delta^G(\sigma, q^G), \delta^{P^{G_1}}(\sigma, q^{P^{G_1}}), \delta^{P^{G_2}}(\sigma, q^{P^{G_2}})),$$

where $q^G \in Q^G, q^{P^{G_1}} \in Q^{P^{G_1}}, q^{P^{G_2}} \in Q^{P^{G_2}}$. When $\delta^G(\sigma, q^G)$ is not defined, $\delta^A(\sigma, (q^G, q^{P^{G_1}}, q^{P^{G_2}}))$ is also not defined. The set of states Q^A is the set of states in $Q^G \times Q^{P^{G_1}} \times Q^{P^{G_2}}$ reachable from the initial state via the δ^A defined above.

For example, if the plant is at state x and an event occurs that only $\mathcal{S}_2$ observes, taking the plant to state y, the next state in the monitoring automaton reflects the fact that the plant state changed, $\mathcal{S}_1$ has seen nothing and its estimate of where it is in the plant has therefore not changed, and $\mathcal{S}_2$ has updated its estimate to its view of the plant at state y.

Note that a state $(q^G, q^{P^{G_1}}, q^{P^{G_2}}) \in Q^A$ is only reachable if there exists $t \in L(G)$ such that $\delta^G(t, q_0^G) = q^G$. Thus, $L(A) = L(G)$ by construction.

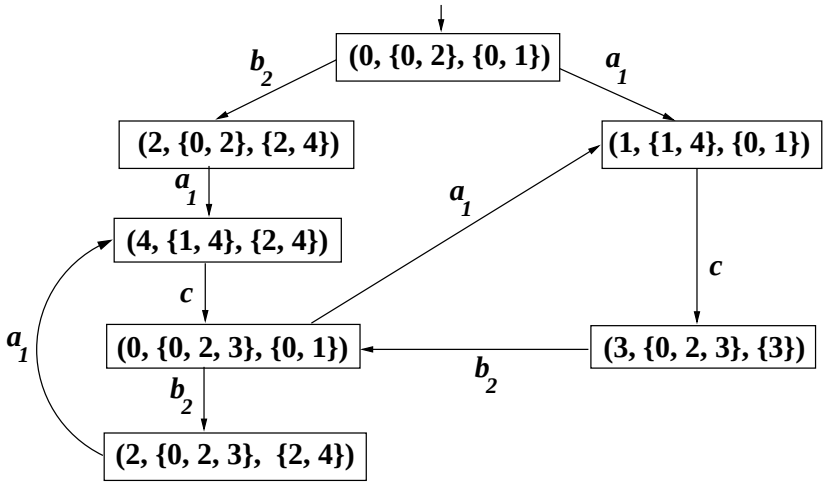

Figure 2.3: The monitoring automaton A for G, P^{G_1}, P^{G_2} of figure 2.2.

We illustrate the construction of A by revisiting the example of figure 2.2. Recall that the two supervisors of this plant see the following events: $\Sigma_{1,o} = \{a_1, c\}$ and $\Sigma_{2,o} = \{b_2, c\}$. At each state of the monitoring automaton we want to determine the current plant state and, based on the partial view each supervisor has of the plant, the set of states each supervisor considers to be the current plant state. To build the states in Q^A, we begin by examining transitions from the initial state in the automaton $q_0^A = (0, \{0, 2\}, \{0, 1\})$. Subsequent states are constructed in a breadth-first manner by following the transitions of the plant and simultaneously determining how the supervisors advance through the projections of the plant, P^{G_i}, $i = 1, 2$. For instance, at state 0 in the plant, transition a_1 may occur and would lead to state 1. At plant state 0, $\mathcal{S}_1$ is at state $\{0, 2\}$ in P^{G_1} and a transition of a_1 takes $\mathcal{S}_1$ to state $\{1, 4\}$ in P^{G_1}. In P^{G_2}, though, $\mathcal{S}_2$ makes no such state change since it does not see event a_1. The resulting state is thus $(1, \{1, 4\}, \{0, 1\})$. We continue in this fashion until the calculation of δ^A yields no more new states. The complete monitoring automaton A for this example system is shown in figure 2.3.

There may be two distinct states q, q' in Q^A that have the same state as the first entry of the state label. This reflects the fact that there could be more than one way

of reaching a plant state x. For instance, if $\delta^G(t, q_0^G) = \delta^G(t', q_0^G) = x$, $t \neq t'$ and $P_i(t) \neq P_i(t')$, $i = 1, 2$, there will be two distinct states q, q' in Q^A with state x as the first entry of the state label. The second and third entry in the state label for state q record the state-based view $\mathcal{S}_1$ and $\mathcal{S}_2$ have of sequence t. Similarly, the state label of state q' stores the state-based view $\mathcal{S}_1$ and $\mathcal{S}_2$ have of sequence t'. In figure 2.3, states $(2, \{0, 2\}, \{2, 4\})$ and $(2, \{0, 2, 3\}, \{2, 4\})$ are two such states.

The creation of a monitoring automaton yields a finite structure that can be used to track the progress in the plant and the projection automata of the plant. No matter how the plant arrives at a particular state q—even when the corresponding sequence observed by a supervisor is arbitrarily long—a supervisor will make the same control decision every time it arrives at state q. This is a characteristic of the structure we will exploit when we discuss communication in decentralized control problems. For example, if the plant in figure 2.2 is in state 4, and if transition c out of state 4 is illegal then we do not care if the current sequence is b_2a_1, $a_1cb_2a_1cb_2b_2a_1$ or $a_1cb_2b_2a_1$ and so on. We are only concerned with the fact that every time the plant reaches state 4 event c must be disabled.

2.2 A Model for Knowledge

The framework for modeling knowledge that we use is based on a knowledge logic for distributed systems [16], 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 [12]. 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 behaviour. We assume that there is a set of agents $\mathsf{G}= \{1, \ldots, n\}$ to which we ascribe knowledge about the system.

The system behaviour 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. When we introduce our knowledge model for DES in chapter 3, we 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 M 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., $\pi(w)(p) = \mathbf{false}$), 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,

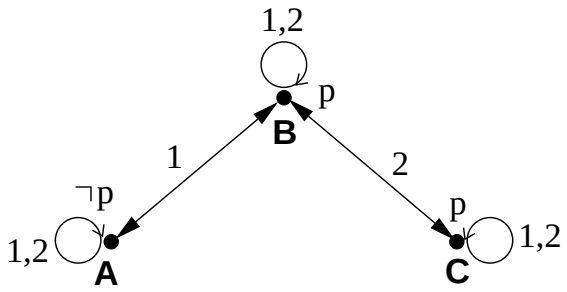

Figure 2.4: A simple Kripke structure.

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**.

The following example illustrates a simple Kripke structure and is adapted from [12]. The graphical representation of this system is shown in figure 2.4. Suppose $\Phi = \{p\}$ and $n = 2$. Let the set of worlds be $\{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ and the interpretation function defined be such that proposition p is true at states B and C but false at state A (i.e., $\pi(\mathsf{A})(p) =$ **false**, $\pi(\mathsf{B})(p) = \pi(\mathsf{C})(p) =$ **true**). The possibility relations for the agents are defined as follows: agent 1 cannot tell the difference between A and B while states B and C look alike to agent 2. These relations are captured in figure 2.4 by the edge label of "1" joining states A and B and the edge label "2" joining states B and C. The self-loops at all three states with edge label "1,2" indicate that a given state cannot be distinguished from itself. For example, in addition to state B looking like A to agent 1, state A also looks like state A. Note that because we assume that the possibility relation is an equivalence relation, reflexivity and symmetry always hold. Therefore, from now on, self-loops and arrows will be omitted from diagrams of Kripke structures.

We now have all the components we need to reason about knowledge: a set of worlds describing the behaviour 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:

$$w \sim_i w' \text{ if } w_i = w'_i. \tag{2.6}$$

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. For example, at state B in figure 2.4, we can say p is true because $\pi(\mathsf{B})(p) = \mathbf{true}$. More formally:

$$(\mathcal{I}, w) \models p \text{ (for } p \in \Phi) \text{ iff } \pi(w)(p) = \mathbf{true}.$$

The clause for negation indicates that $\neg p$ is true at w exactly if p is not true:

$$(\mathcal{I}, w) \models \neg p \text{ iff } (\mathcal{I}, w) \not\models p.$$

In figure 2.4, at state A, we can say $\neg p$ holds because p is not true at A.

We can consider more than one fact holding at a world:

$$(\mathcal{I}, w) \models p_1 \wedge p_2 \text{ iff } (\mathcal{I}, w) \models p_1 \text{ and } (\mathcal{I}, w) \models p_2.$$

Thus, the conjunction of two propositions holds at w if it is the case that each proposition is true at w.

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:

$$(\mathcal{I}, w) \models K_i p \text{ iff } (\mathcal{I}, w') \models p \text{ for all } w' \text{ such that } w \sim_i w'. \tag{2.7}$$

Referring to figure 2.4 again, we can now describe the knowledge of agents at any world in the system: e.g., at world B the formula $\neg K_1 p \wedge K_2 p$ holds. That is, at B agent 1 does not know whether p is true, while at B agent 2 knows that p is true. At world B, agent 1 considers the existence of two possible worlds: A and B. It considers both p and $\neg p$ to be possible because p is false at world A while p is true at world B. Agent 2 also considers the existence of two possible worlds: B and C. However, since p holds at both those worlds, at B agent 2 knows that p is true.

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

$$(\mathcal{I}, w) \models K_i p \text{ iff } (\mathcal{I}, w') \models K_i p \tag{2.8}$$

for all w' such that $w \sim_i w'$. For instance, agent 2 considers that worlds B and C "look alike". Since p is true at both these states, we can say that at B, agent 2 knows p. Similarly, we can say that at C agent 2 also knows p .

Finally, we note a property, called the *Knowledge Axiom*, that states that if an agent knows a fact, then the fact is true:

$$(\mathcal{I}, w) \models K_i p \Rightarrow (\mathcal{I}, w) \models p. \tag{2.9}$$

Note that if $K_i p$ holds at some world w, we know by (2.8) that $K_i p$ holds at all worlds that agent i cannot distinguish from w. Since p is true in all worlds that agent i considers possible, in particular, p is true at w.

Chapter 3

Using Knowledge for Control

In this chapter we describe how to recast decentralized supervisory control problems as interpreted systems. We do not claim that the reformulation of this problem provides a more efficient solution but, rather, suggest that knowledge theory provides a more natural way of thinking about discrete-event control problems.

We introduce two knowledge models: control decisions in the first model are made by an agent on the basis of a partial view of a sequence of events generated by the plant, while control decisions in the second model are made based on the set of plant states an agent considers possible. The latter model allows us to avoid generating an infinite-state Kripke structure, whereas the first model is more faithful to the DES theory of [29]. Most of the material in this chapter first appeared in a joint paper with K. Rudie[30].

3.1 Sequence-Based Knowledge Model

We denote our "sequence-based" interpreted system as $\mathcal{I}^{DES}(G, E)$, where G is the plant and E is the legal automaton. Like local supervisors in the decentralized DES formulation of [9, 33], the agents in this interpreted system make control decisions based on their partial view of the *event sequences* generated by the DES plant G. The *environment* of $\mathcal{I}^{DES}$ is the prefix-closed language generated by the plant and the *agents* play a role equivalent to that of decentralized DES supervisors.

A global state for n agents in $\mathcal{I}^{DES}(G, E)$, which records the environment state, w_e, and the local states of each agent, w_i, captures a "snapshot" of a sequence from

the plant language $L(G)$. The set of states for the environment is the set of sequences in the plant language $L(G)$, while the set of local states for the agents is the set of sequences each agent observes according to the projection operation of (2.3). More formally, a global state for n agents is defined as $w = (w_e, w_1, \ldots, w_n) = (t, P_1(t), \ldots, P_n(t))$ for $t \in L(G)$. We assume that $n = 2$ so that the group of agents, denoted by G, is $\{1, 2\}$.

In addition to consisting of a set of global states, $\mathcal{I}^{DES}(G, E)$ is also associated with an interpretation function. The interpretation π^{DES} captures the notion of whether or not an event in Σ is permissible as sequences evolve in the plant. To form Φ, the set of primitive propositions for $\mathcal{I}^{DES}(G, E)$, we want to associate with each $\sigma \in \Sigma$ 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 another to represent the fact that at the corresponding state in the legal automaton state the event is defined. The propositions are defined in terms of events because we want to reason about the knowledge an agent has of the occurrence of an event, instead of, for instance, a certain sequence. If Σ is finite (i.e., $|\Sigma| = N$), it can be written as $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_N\}$. We let $\Phi = \{\sigma_i^G, \sigma_i^E | i = 1, \ldots, N\}$. We also partition Φ into two disjoint sets: $\Phi_G = \{\sigma_i^G | i = 1, \ldots, N\}$ and $\Phi_E = \{\sigma_i^E | i = 1, \ldots, N\}$ where Φ_G and Φ_E are sets containing N symbols. Because we will frequently want to associate σ_i^G with its counterpart σ_i^E, we define the relation R_Σ such that $\mathsf{R}_\Sigma \subseteq \Phi_G \times \Phi_E$ and $\mathsf{R}_\Sigma := \{(\sigma_G, \sigma_E) | \exists \sigma_i \in \Sigma \text{ where } \sigma_G = \sigma_i^G, \sigma_E = \sigma_i^E\}$. For convenience, we will use the notation σ_G (respectively, σ_E) without explicit reference to R_Σ when we mean σ_i^G (respectively, σ_i^E). The proposition σ_G is "event σ can occur" and σ_E is "event σ is legal". For convenience, we will refer to Φ_{uc} when we need to identify those propositions in Φ which represent events in Σ_{uc} (as defined in section 2.1).

The interpretation for the propositions in Φ is defined for all $\sigma \in \Sigma$:

$$\pi^{DES}(w)(\sigma_G) := \begin{cases} \textbf{true} & \text{if } \delta^G(w_e\sigma, q_0^G)!, \\ \textbf{false} & \text{otherwise.} \end{cases} \tag{3.1}$$

$$\pi^{DES}(w)(\sigma_E) := \begin{cases} \textbf{true} & \text{if } \delta^E(w_e\sigma, q_0^E)!, \\ \textbf{false} & \text{otherwise.} \end{cases}$$

In other words, a proposition σ_G is true at a global state w if the event σ can happen directly following the event sequence described by w_e. A proposition σ_G is false (denoted $\neg\sigma_G$) at a global state w if the event σ is not defined directly following the event sequence described by w_e. Similarly, a proposition σ_E is true at a global state w if the event σ can happen directly following the event sequence described by w_e and $w_e\sigma$ is part of the legal behaviour of the plant. A proposition σ_E is false (denoted $\neg\sigma_E$) at a global state w if either $\sigma_G = \textbf{false}$ or if the sequence $w_e\sigma$ is part of the illegal behaviour of the plant.

Because the set of global states in $\mathcal{I}^{DES}(G, E)$ and the truth values assigned by π^{DES} to the primitive propositions in Φ are derived from the legal automaton E and the plant G, we can consider that $\mathcal{I}^{DES}$ has implicit parameters G and E. Thus, for ease of notation, we drop these arguments for the remainder of the chapter since G and E are understood.

We illustrate the sequence-based knowledge model by constructing a Kripke structure for the plant and legal automaton in figure 2.1. In this example, suppose that agent 1 sees and controls events α and γ while agent 2 sees event β and controls events β and γ.

The complete Kripke structure contains ten states, corresponding to the ten sequences in $L(G)$. We show a representative portion of the structure in figure 3.1. The set of propositions is $\Phi = \{\alpha_G, \alpha_E, \beta_G, \beta_E, \gamma_G, \gamma_E\}$ and the truth assignments for π^{DES} are made according to (3.1). For example, at state $(\alpha, \alpha, \varepsilon)$, proposition γ_G is

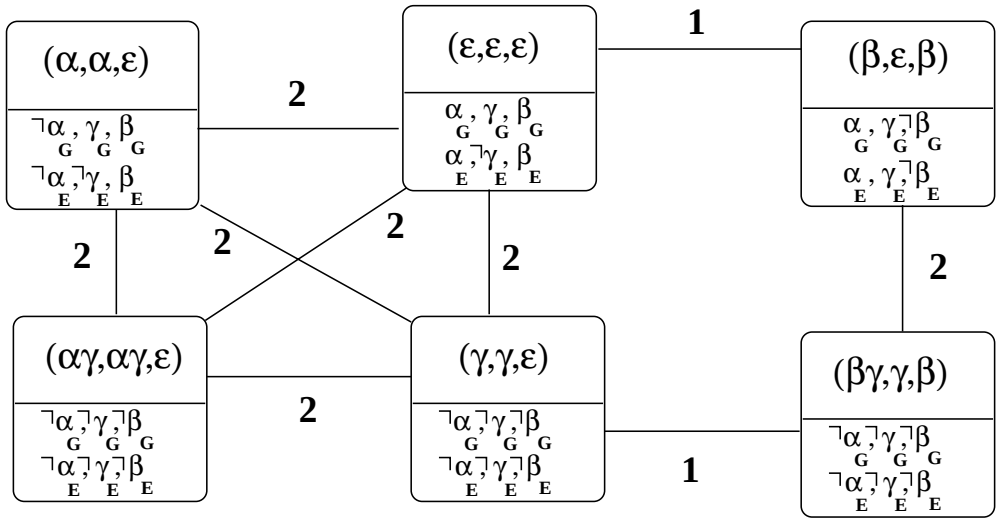

Figure 3.1: A portion of the Kripke structure for G in figure 2.1.

assigned a value of **true** because there is a transition of γ from sequence α in the plant; however, γ_E has a truth assignment of **false** because this same transition is not defined in the legal automaton. The possibility relations describe how agents view the world. In this model, the possibility relation for each agent is defined using the indistinguishability relation $\sim_i$ of (2.6). That is, two global states look alike to agent i if the global states have the same local state according to agent i. For instance, the possibility relation for agent 1 would contain the pair of states $((\gamma, \gamma, \varepsilon), (\beta\gamma, \gamma, \beta))$ because these states have the same local state according to agent 1, namely γ.

The top half of a node in figure 3.1 contains one of the global states in the system and the bottom half of the node shows the truth values for the primitive propositions at that global state. The diagram exactly describes the states of the plant where we want to impose control (as constrained by the legal language) and what each agent believes are the possible worlds of this system. For example, in figure 2.1 after α occurs in the plant, event γ is not legal and therefore must be disabled. This information is captured by the node labelled $(\alpha, \alpha, \varepsilon)$ in figure 3.1. At this state the primitive proposition γ_G is true, meaning that $\alpha\gamma$ is part of the plant language; however, γ_E is

false, which means that $\alpha\gamma$ is not part of the legal language. By following the edges connecting the nodes, we can also determine what each agent believes are its possible worlds. For instance, the node labelled $(\beta, \varepsilon, \beta)$ looks the same to agent 2 as the node labelled $(\beta\gamma, \gamma, \beta)$ because the local state of agent 2 is β in both the global states. This is indicated by an edge labelled "2" joining these nodes.

We can describe the knowledge of each agent at a particular state in the interpreted system. For example, at $w = (\varepsilon, \varepsilon, \varepsilon)$:

$$(\mathcal{I}^{DES}, w) \models K_1(\alpha_G \wedge \alpha_E \wedge \gamma_G) \wedge K_2 \neg \gamma_E.$$

At state w the set of worlds that agent 1 considers possible is $\{(\varepsilon, \varepsilon, \varepsilon),\ (\beta, \varepsilon, \beta)\}$. At both these states the truth values of α_G, α_E and γ_G are true. Therefore we say that "Agent 1 knows α can happen and is legal at $(\varepsilon, \varepsilon, \varepsilon)$ and that γ can happen at $(\varepsilon, \varepsilon, \varepsilon)$". Similarly, we say that "Agent 2 knows that it is not the case that γ is legal at $(\varepsilon, \varepsilon, \varepsilon)$" because at all states indistinguishable from $(\varepsilon, \varepsilon, \varepsilon)$ to agent 2—namely $(\varepsilon, \varepsilon, \varepsilon)$, $(\alpha, \alpha, \varepsilon)$, $(\alpha\gamma, \alpha\gamma, \varepsilon)$ and $(\gamma, \gamma, \varepsilon)$—the formula $\neg\gamma_E$ is true. We can also make note of the lack of knowledge an agent has: at w, agent 1 does not know whether γ is legal because at $(\varepsilon, \varepsilon, \varepsilon)$ the formula $\neg\gamma_E$ is true while at $(\beta, \varepsilon, \beta)$ the formula γ_E is true. This is denoted by $(\mathcal{I}^{DES}, w) \models \neg K_1 \gamma_E$ and is read as "Agent 1 does not know whether γ is legal at $(\varepsilon, \varepsilon, \varepsilon)$".

3.2 State-Based Knowledge Model

The previous knowledge model 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 Kripke structure for a system with such a language 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 set of plant states Q^G, while the agents are a slight variation of the DES local supervisors $\mathcal{S}_1$ and $\mathcal{S}_2$. Each supervisor still has only a partial view of the complete system behaviour, but these views are based on the projection automaton of section 2.1.2, rather than the projection operator of (2.3).

The worlds in the system are no longer composed of sequences from the plant language $L(G)$, but rather the worlds 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 stategy for generating states in Q^A of the monitoring automaton A as described in section 2.1.3. Consequently, a global state w has the form (w_e, w_1, w_2) where $w_e \in Q^G$, $w_1 \in Q^{PG_1}$ and $w_2 \in Q^{PG_2}$.

The interpretation $\pi^{DES'}$ changes slightly for the state-based knowledge model. Instead of determining if a sequence $t \in L(G)$ precedes event σ, we want to check to see if σ is defined at the current plant state (as recorded in w_e):

$$\pi^{DES'}(w)(\sigma_G) := \begin{cases} \textbf{true} & \text{if } \delta^G(\sigma, w_e)!, \\ \textbf{false} & \text{otherwise.} \end{cases} \tag{3.2}$$

$$\pi^{DES'}(w)(\sigma_E) := \begin{cases} \textbf{true} & \text{if } \delta^E(\sigma, w_e)!, \\ \textbf{false} & \text{otherwise.} \end{cases} \tag{3.3}$$

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 (3.3) is a consistent reference to the same state in both automata.

3.3 Knowledge-Based Protocols and Kripke-observability

The interpreted systems $\mathcal{I}^{DES}$ and $\mathcal{I}^{DES'}$ describe the knowledge that each agent has concerning the validity of a particular sequence. We need to associate actions with

an agent's knowledge—for instance, if an agent knows that a particular proposition is possible but not legal for a set of possible worlds, then we want it to disable the corresponding event. A knowledge-based protocol [15] is a strategy that links actions and knowledge for agents (and the environment). We believe that it is natural to think of a supervisor basing its control actions on what the supervisor "knows" about the present state of the system. Even though we depart from some of the specifics of the formalism for knowledge-based protocols in [15], we incorporate the idea that there ought to be a connection between action and knowledge.

We examine knowledge-based protocols where actions to disable or enable an event are based only on local state information and where an uncontrollable event cannot be disabled. We describe protocols for agents but not for the environment, which we view as incapable of taking control actions. For the decentralized DES we consider, when we say that a supervisor $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ solves a problem, we mean that when G is under the control of $\mathcal{S}$, the resultant language generated, namely $L((\mathcal{S}_1 \wedge \mathcal{S}_2)/G)$, equals the legal language: $L((\mathcal{S}_1 \wedge \mathcal{S}_2)/G) = L(E)$. Solving the decentralized problem with a knowledge-based protocol amounts to constructing a protocol that will ensure that only legal sequences and all legal sequences are generated.

3.3.1 Knowledge-based protocols for decentralized control

A local supervisor in a decentralized DES system disables controllable event σ if the supervisor is able to determine that the occurrence of σ will lead to illegal behaviour; otherwise event σ is enabled. An event σ is disabled if at least one local agent takes a "disable σ" action at w.

The actions that drive the global state changes of the system are performed according to a selection rule or *protocol*. A *knowledge-based* protocol is a protocol where actions are taken on the basis of the local knowledge of an agent. We define a knowledge-based protocol as a mapping that characterizes which events are disabled

$\mathcal{KP}_i : \mathcal{L}_i \times \Sigma \to \{enable, disable\}$, where $\mathcal{L}_i$ is the set of local states for agent i. Since the knowledge-based protocol is defined on the local view of an agent, the actions of agent i are applied at all w' that are indistinguishable to agent i at w. Just as a local decentralized DES supervisor makes control decisions based on its partial view of a sequence, we want an agent to use knowledge and its local states to determine if a given event should be disabled. A *joint* knowledge-based protocol is the collection of the knowledge-based protocols for all agents in G.

We identify the group of agents that can control a given event as $\mathsf{G}_\sigma := \{i : \sigma \in \Sigma_{i,c}\}$, i.e., for all $\sigma \in \Sigma_{1,c} \cap \Sigma_{2,c}$, $\mathsf{G}_\sigma = \{1, 2\}$; for all $\sigma \in \Sigma_{1,c} \setminus \Sigma_{2,c}$, $\mathsf{G}_\sigma = \{1\}$; for all $\sigma \in \Sigma_{2,c} \setminus \Sigma_{1,c}$, $\mathsf{G}_\sigma = \{2\}$; and for all $\sigma \in \Sigma_{uc}, \mathsf{G}_\sigma = \emptyset$.

A joint knowledge-based protocol $\mathcal{KP} = (\mathcal{KP}_1, \mathcal{KP}_2)$ solves the decentralized problem if for all $w \in \mathcal{I}^{DES}$ and all $(\sigma_G, \sigma_E) \in \mathsf{R}_\Sigma$:

$$\text{(i)} \quad (\mathcal{I}^{DES}, w) \models \sigma_G \wedge \neg\sigma_E \Rightarrow (\exists i \in \mathsf{G}_\sigma)\mathcal{KP}_i(w_i, \sigma) = \text{ disable};$$

$$\text{(ii)} \quad (\mathcal{I}^{DES}, w) \models \sigma_G \wedge \sigma_E \Rightarrow (\nexists i \in \mathsf{G}_\sigma)\mathcal{KP}_i(w_i, \sigma) = \text{ disable}.$$

That is, solving the problem amounts to allowing only legal sequences and all legal sequences to occur. Note that since we assume that E is a subautomaton of G, it will never be the case that $(\mathcal{I}^{DES}, w) \models \neg\sigma_G \wedge \sigma_E$.

To solve the decentralized control problem using knowledge-based protocols we must formalize what it means for agents to "know enough". We describe several conditions that $\mathcal{I}^{DES}$ (equivalently $\mathcal{I}^{DES'}$) must satisfy before a solution can be achieved. In particular, we present a necessary and sufficient condition so that our knowledge-based protocol admits only (and all) the legal sequences in $L(E)$.

We define a property equivalent to co-observability [33] and controllability [29] that characterizes the nature of knowledge an interpreted system requires to yield a decentralized solution to the DES control problem.

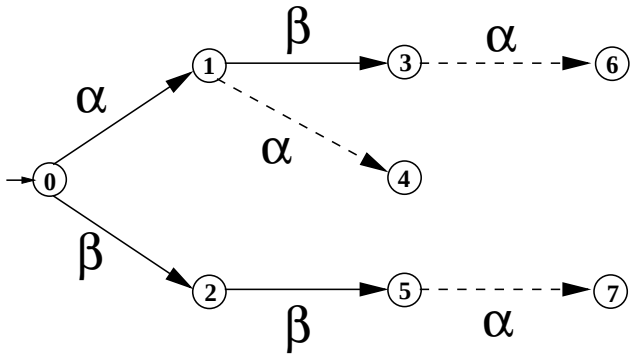

Figure 3.2: A plant G and legal automaton E.

DEFINITION **3.1** *An interpreted system $\mathcal{I}^{DES}$ (respectively, $\mathcal{I}^{DES'}$) is Kripke-observable if:*

$$\begin{aligned} &\forall w \in \mathcal{I}^{DES}, \forall (\sigma_G, \sigma_E) \in \mathsf{R}_\Sigma, \\ &\qquad (\mathcal{I}^{DES}, w) \models \neg\sigma_G \vee \sigma_E \\ &\vee \quad (\exists i \in \mathsf{G}_\sigma) \textit{ such that } (\mathcal{I}^{DES}, w) \models K_i \neg\sigma_E. \end{aligned} \tag{3.4}$$

That is, if an illegal event σ is about to occur, at least one agent that can control σ knows that it should be disabled. We note here that co-observability is a condition on set membership and set containment for sets of sequences while Kripke-observability involves logic tests on propositions.

The condition we were initially trying to capture in the definition of Kripke-observability was that for every event that *can* occur, at least one agent knows whether or not that event is legal. Our intuition led us to the following logic formulation:

$$\begin{aligned} &\forall w \in \mathcal{I}^{DES}, \forall (\sigma_G, \sigma_E) \in \mathsf{R}_\Sigma, \\ &\qquad (\exists i \in \mathsf{G}_\sigma) \text{ such that } (\mathcal{I}^{DES}, w) \models K_i \sigma_E \vee K_i \neg\sigma_E. \end{aligned} \tag{3.5}$$

However, this is actually too strong a condition as the following example will illustrate.

The plant and legal automaton in figure 3.2 represents a co-observable language (with supervisors $\mathcal{S}_1$ and $\mathcal{S}_2$) if $\Sigma_{1,o} = \{\alpha\} = \Sigma_{1,c}$, $\Sigma_{2,o} = \{\beta\}$, and $\Sigma_{2,c} = \{\alpha, \beta\}$.

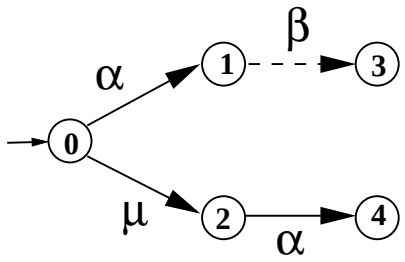

Figure 3.3: A plant G and legal automaton E.

Thus a control strategy for this decentralized problem is as follows: $\mathcal{S}_1$ disables α after seeing α and $\mathcal{S}_2$ disables α after seeing $\beta\beta$. If we were to use the condition in (3.5), then the condition would fail at $w = (\varepsilon, \varepsilon, \varepsilon)$. At this state, the possible worlds of agent 1 are $(\varepsilon, \varepsilon, \varepsilon)$, $(\beta, \varepsilon, \beta)$, and $(\beta\beta, \varepsilon, \beta\beta)$. Both disjuncts of (3.5) fail for event α at w since $(\mathcal{I}^{DES}, w) \models \alpha_E$, whereas at w' and w'', when $w' = (\beta, \varepsilon, \beta)$ and $w'' = (\beta\beta, \varepsilon, \beta\beta)$, we have $(\mathcal{I}^{DES}, w') \models \neg\alpha_E$ and $(\mathcal{I}^{DES}, w'') \models \neg\alpha_E$. The possible worlds for agent 2 at $w = (\varepsilon, \varepsilon, \varepsilon)$ are states $w' = (\alpha, \alpha, \varepsilon)$, $w'' = (\alpha\alpha, \alpha\alpha, \varepsilon)$ and w itself. As it did for agent 1, both disjuncts fail on event α at all states indistinguishable from w since $(\mathcal{I}^{DES}, w) \models \alpha_E$ while $(\mathcal{I}^{DES}, w') \models \neg\alpha_E$ and $(\mathcal{I}^{DES}, w'') \models \neg\alpha_E$. In fact, the failure to meet the condition of (3.5) was because we required an agent to know when an event should be enabled.

This observation led to a revised definition:

$$\begin{aligned} \forall w \in \mathcal{I}^{DES}, \forall (\sigma_G, \sigma_E) \in \mathsf{R}_\Sigma, & \\ (\mathcal{I}^{DES}, w) \models \sigma_E & \\ \vee \quad (\exists i \in \mathsf{G}_\sigma) \text{ such that } (\mathcal{I}^{DES}, w) \models K_i \neg\sigma_E. & \end{aligned} \tag{3.6}$$

Now that the condition in (3.6) does not insist on knowledge if the event is legal, this updated condition is satisfied for event α at all states in figure 3.2. However, figure 3.3 shows another co-observable system where (3.6) also fails. Let $\Sigma_{1,o} = \{\alpha\}$, $\Sigma_{1,c} = \{\alpha, \beta\}$, $\Sigma_{2,o} = \{\mu\}$, $\Sigma_{2,c} = \{\mu\}$. There is a problem with event μ, even though

this event never needs to be disabled. At state $w = (\alpha, \alpha, \varepsilon)$, agent 2 neither knows that μ should be disabled nor that it should not be disabled since $(\mathcal{I}^{DES}, w) \models \neg\mu_E$, while at $w' = (\varepsilon, \varepsilon, \varepsilon)$ it is the case that $(\mathcal{I}^{DES}, w') \models \mu_E$ and $w \sim_2 w'$. Therefore, at w neither disjunct of condition (3.6) is satisfied. Since μ cannot actually happen after α occurs, it is too strong to require than an agent possess any knowledge about μ at w.

If we use the definition of (3.4), the system of figure 3.3 is Kripke-observable. In particular, at state $w = (\alpha, \alpha, \varepsilon)$, the first disjunct for event μ is now satisfied: $(\mathcal{I}^{DES}, w) \models \neg\mu_G$.

Note that even though the definitions in (3.5) and (3.6) fail on this example system, there is still a solution to the problem. This is because the default action is to enable an event when no agent knows whether or not to disable that event. The requirement in the first disjunct of (3.5) to *know* that the event should be enabled is too strong. Similarly (3.6) fails as we neglected to notice that we can omit knowledge tests on "do not care" states, that is, states where events are not even defined in the corresponding DES plant states. Hence this knowledge condition can be relaxed so that a test for knowledge is only performed when an event σ is possible but is not legal (i.e., when the disjunct $\neg\sigma_G \vee \sigma_E$ does not hold).

THEOREM **3.1** *Given G, E, there exists a joint knowledge-based protocol $\mathcal{KP} = (\mathcal{KP}_1, \mathcal{KP}_2)$ that solves the decentralized problem iff $\mathcal{I}^{DES}(G, E)$ is Kripke–observable.*

Proof: ($\Leftarrow$) Suppose $\mathcal{I}^{DES}$ is Kripke-observable. Define the following knowledge-based protocol:

$(\forall\sigma \in \Sigma)(\forall\ell \in \mathcal{L}_i)$

$$(\forall i \in \mathsf{G}_\sigma) \qquad \mathcal{KP}_i(\ell, \sigma) = \begin{cases} \text{disable}, & \text{if } \exists w \text{ such that } \ell = w_i \\ & \wedge\, (\mathcal{I}^{DES}, w) \models K_i\neg\sigma_E, \\ \text{enable}, & \text{otherwise.} \end{cases} \tag{3.7}$$

$$(\forall j \notin \mathsf{G}_\sigma) \qquad \mathcal{KP}_j(\ell, \sigma) = \text{enable}. \tag{3.8}$$

If an agent knows than an event is illegal, it will disable the event. Therefore, unless an agent knows that an event is illegal, the event will be enabled. Note that for an event controllable by agent i, the definition of $\mathcal{KP}_i$ in (3.7) is robust to the choice of w in (3.7) (i.e., if a different w' were chosen such that $\ell = w'_i$, then by (2.8), $(\mathcal{I}^{DES}, w) \models K_i \neg\sigma_E$ iff $(\mathcal{I}^{DES}, w') \models K_i \neg\sigma_E$).

We want to show that $\mathcal{KP} = (\mathcal{KP}_1, \mathcal{KP}_2)$ solves the decentralized problem.

(i) Suppose $(\mathcal{I}^{DES}, w) \models (\sigma_G \wedge \neg\sigma_E)$. We want to show that this implies $(\exists i \in \mathsf{G}_\sigma)$ $\mathcal{KP}_i(w_i, \sigma) =$ disable.

Since $(\mathcal{I}^{DES}, w) \models \sigma_G \wedge \neg\sigma_E$, we have $(\mathcal{I}^{DES}, w) \not\models \neg\sigma_G \vee \sigma_E$. Thus, since Kripke-observability holds, it must be the case that $\exists i \in \mathsf{G}_\sigma$ such that $(\mathcal{I}^{DES}, w) \models K_i \neg\sigma_E$. Therefore either $\mathcal{KP}_1(w_i, \sigma) =$ disable or $\mathcal{KP}_2(w_i, \sigma) =$ disable.

(ii) Suppose $(\mathcal{I}^{DES}, w) \models \sigma_G \wedge \sigma_E$. We want to show that this implies $(\nexists\, i \in \mathsf{G}_\sigma)$ $\mathcal{KP}_i(w_i, \sigma) =$ disable.

The requirement for $\mathcal{KP}_i(w_i, \sigma) =$ disable for some $i \in \mathsf{G}_\sigma$ is that $(\mathcal{I}^{DES}, w) \models K_i \neg\sigma_E$. However, since $(\mathcal{I}^{DES}, w) \models \sigma_E$, we cannot have $(\mathcal{I}^{DES}, w) \models K_i \neg\sigma_E$ for any i (by (2.9)). Therefore $\mathcal{KP}_i(w_i, \sigma) =$ enable for all $i \in \mathsf{G}_\sigma$. By (3.8), for all $i \notin \mathsf{G}_\sigma$, $\mathcal{KP}_i(w_i, \sigma) =$ enable.

($\Rightarrow$) We need to show that if $\mathcal{I}^{DES}$ is not Kripke-observable then there is no joint knowledge-based protocol $\mathcal{KP}$ that solves the decentralized problem.

Suppose that some $\mathcal{KP}$ solves the decentralized problem. Since $\mathcal{I}^{DES}$ is not Kripke-observable, there must exist $w \in \mathcal{I}^{DES}$ such that $\exists(\sigma_G, \sigma_E) \in \mathsf{R}_\Sigma$ where $(\forall i \in \mathsf{G}_\sigma)$ $(\mathcal{I}^{DES}, w) \not\models \neg\sigma_G \vee \sigma_E \vee K_i \neg\sigma_E$. That is,

$$(\mathcal{I}^{DES}, w) \not\models \neg\sigma_G \vee \sigma_E; \tag{3.9}$$

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

$$(\mathcal{I}^{DES}, w) \not\models K_i \neg\sigma_E. \tag{3.10}$$

The expression in (3.9) implies that $(\mathcal{I}^{DES}, w) \models (\sigma_G \wedge \neg\sigma_E)$. Since $\mathcal{KP}$ solves the decentralized problem, and since $(\mathcal{I}^{DES}, w) \models (\sigma_G \wedge \neg\sigma_E)$, it must be the case that $\exists j$ such that $\mathcal{KP}_j(w_j, \sigma) =$ disable. Note that (3.10) holds for agent j and implies that $\exists w'$ such that $(\mathcal{I}^{DES}, w') \models \sigma_E$ and $w \sim_j w'$. Since $\mathcal{KP}$ solves the decentralized problem, $\mathcal{KP}_j(w'_j, \sigma) \neq$ disable. However, since $\mathcal{KP}_j(w_j, \sigma) =$ disable, this means that $\mathcal{KP}_j(w'_j, \sigma) =$ disable (since $w \sim_j w'$ means that $w_j = w'_j$), which leads to a contradiction.

□ THEOREM 3.1

Note that this result also holds for $\mathcal{I}^{DES'}(G, E)$. We are checking the knowledge that an agent has about events at its possible worlds. What is important is that the possible worlds capture the knowledge of an agent for a given system. Therefore, as long as the knowledge of the agents is described accurately, it does not matter whether the possible worlds be described as sequences of the plant language or as plant states. Note also that Kripke-observability is equivalent to controllability and co-observability taken together.

We want to ensure that the actions taken by agents as a result of executing the knowledge protocol exactly permit the legal language of the DES plant. In what follows, we are referring only to the sequence-based model $\mathcal{I}^{DES}$, not the state-based model $\mathcal{I}^{DES'}$. Therefore, we make precise the set of sequences that are generated by the supervised interpreted system $\mathcal{I}^{DES}$.

DEFINITION **3.2** *The language that contains all sequences determined by the actions of the two knowledge-based protocols is $\mathcal{KL}(\mathcal{KP}, G)$, defined as follows:*

$$\begin{aligned} \varepsilon &\in \mathcal{KL}, \\ w_e\sigma &\in \mathcal{KL} \text{ if } w_e \in \mathcal{KL} \wedge w_e\sigma \in L(G) \wedge \mathcal{KP}_i(w_i, \sigma) = \text{ enable }, (i = 1, 2). \end{aligned}$$

Thus a sequence is in $\mathcal{KL}$ if its prefix is already in $\mathcal{KL}$, the sequence is actually generated by the plant, and the control action at the corresponding global state is

to allow σ to happen. If the knowledge-based protocol takes a "disable" action with resepct to σ at global state w, it is because at least one agent knows that $w_e\sigma$ is not part of the legal language and is therefore, by the above definition, not included in $\mathcal{KL}$.

3.3.2 Example: a Kripke–observable system for $\mathcal{I}^{DES}$

We return to the plant and legal automaton of figure 2.1 where agent 1 sees and controls events α and γ while agent 2 sees β but controls both β and γ. Part of the Kripke structure associated with the plant is shown in figure 3.1.

We want to show that $\mathcal{I}^{DES}$ is Kripke-observable. Let $w = (\varepsilon, \varepsilon, \varepsilon)$. We want to ascertain that at this state either agent 1 or agent 2 knows to disable γ:

$$(\mathcal{I}^{DES}, w) \not\models \neg\gamma_G \vee \gamma_E. \tag{3.11}$$

Therefore, for Kripke-observability, we must find an agent i such that $(\mathcal{I}^{DES}, w) \models K_i\neg\gamma_E$.

We first check to see if agent 1 has the appropriate knowledge about event γ. At $w = (\varepsilon, \varepsilon, \varepsilon)$, agent 1 considers one other world to be possible: $(\beta, \varepsilon, \beta)$.

Recall that knowledge of a fact at w requires that the fact hold at all states indistinguishable from w. Thus if agent 1 has knowledge that event γ is not legal at this part of the plant, it must be the case that $\neg\gamma_E$ holds in the two worlds noted above. Agent 1 fails to have the required knowledge at state $w' = (\beta, \varepsilon, \beta)$, since $(\mathcal{I}^{DES}, w') \models \gamma_E$. Thus, $(\mathcal{I}^{DES}, w) \models \neg K_1\neg\gamma_E$ and we must check to see if Kripke-observability is satisfied by agent 2's knowledge at this state.

There are three other possible worlds agent 2 cannot distinguish from $(\varepsilon, \varepsilon, \varepsilon)$: $(\alpha, \alpha, \varepsilon)$, $(\gamma, \gamma, \varepsilon)$ and $(\alpha\gamma, \alpha\gamma, \varepsilon)$. As was the case for agent 1, we need to determine that agent 2 knows $\neg\gamma_E$ holds at w. This means that $\neg\gamma_E$ must hold in all worlds

that look like $(\varepsilon, \varepsilon, \varepsilon)$ to agent 2. Note that because $\neg\gamma_E$ holds at all four possible worlds, $(\mathcal{I}^{DES}, w) \models K_2 \neg\gamma_E$.

A similar check can be performed at every other state in $\mathcal{I}^{DES}$ to show that this system is Kripke-observable (summarized in Table 3.1).

Table 3.1: Checking Kripke-observability.

w	Disjunct of Kripke-observability satisfied		
$(\varepsilon, \varepsilon, \varepsilon)$	$(\mathcal{I}^{DES}, w) \models K_2 \neg\gamma_E$	$(\mathcal{I}^{DES}, w) \models \alpha_E$	$(\mathcal{I}^{DES}, w) \models \beta_E$
$(\beta, \varepsilon, \beta)$	$(\mathcal{I}^{DES}, w) \models \gamma_E$	$(\mathcal{I}^{DES}, w) \models \alpha_E$	$(\mathcal{I}^{DES}, w) \models \neg\beta_G$
$(\gamma, \gamma, \varepsilon)$	$(\mathcal{I}^{DES}, w) \models K_2 \neg\gamma_E$	$(\mathcal{I}^{DES}, w) \models \neg\alpha_G$	$(\mathcal{I}^{DES}, w) \models \neg\beta_G$
$(\alpha, \alpha, \varepsilon)$	$(\mathcal{I}^{DES}, w) \models K_2 \neg\gamma_E$	$(\mathcal{I}^{DES}, w) \models \neg\alpha_G$	$(\mathcal{I}^{DES}, w) \models \beta_E$
$(\beta\gamma, \gamma, \beta)$	$(\mathcal{I}^{DES}, w) \models \neg\gamma_G$	$(\mathcal{I}^{DES}, w) \models \neg\alpha_G$	$(\mathcal{I}^{DES}, w) \models \neg\beta_G$
$(\alpha\gamma, \alpha\gamma, \varepsilon)$	$(\mathcal{I}^{DES}, w) \models K_2 \neg\gamma_E$	$(\mathcal{I}^{DES}, w) \models \neg\alpha_G$	$(\mathcal{I}^{DES}, w) \models \neg\beta_G$
$(\alpha\beta, \alpha, \beta)$	$(\mathcal{I}^{DES}, w) \models \gamma_E$	$(\mathcal{I}^{DES}, w) \models \neg\alpha_G$	$(\mathcal{I}^{DES}, w) \models \neg\beta_G$
$(\beta\alpha, \alpha, \beta)$	$(\mathcal{I}^{DES}, w) \models \gamma_E$	$(\mathcal{I}^{DES}, w) \models \neg\alpha_G$	$(\mathcal{I}^{DES}, w) \models \neg\beta_G$
$(\alpha\beta\gamma, \alpha\gamma, \beta)$	$(\mathcal{I}^{DES}, w) \models \neg\gamma_G$	$(\mathcal{I}^{DES}, w) \models \neg\alpha_G$	$(\mathcal{I}^{DES}, w) \models \neg\beta_G$
$(\beta\alpha\gamma, \alpha\gamma, \beta)$	$(\mathcal{I}^{DES}, w) \models \neg\gamma_G$	$(\mathcal{I}^{DES}, w) \models \neg\alpha_G$	$(\mathcal{I}^{DES}, w) \models \neg\beta_G$

Table 3.2: A joint knowledge-based protocol for G and E and event γ in figure 2.1.

w	$\mathcal{KP}_1(w_1, \gamma)$	$\mathcal{KP}_2(w_2, \gamma)$
$(\varepsilon, \varepsilon, \varepsilon)$	enable	disable
$(\beta, \varepsilon, \beta)$	enable	enable
$(\gamma, \varepsilon, \varepsilon)$	enable	disable
$(\alpha, \alpha, \varepsilon)$	enable	disable
$(\beta\gamma, \varepsilon, \beta)$	enable	enable
$(\alpha\gamma, \alpha, \varepsilon)$	enable	disable
$(\alpha\beta, \alpha, \beta)$	enable	enable
$(\beta\alpha, \alpha, \beta)$	enable	enable
$(\alpha\beta\gamma, \alpha, \beta)$	enable	enable
$(\beta\alpha\gamma, \alpha, \beta)$	enable	enable

The joint knowledge-based protocol for events α and β in this system is straightforward: both events are enabled by both agents at every state in $\mathcal{I}^{DES}$. To realize

the legal language, however, the control decisions for γ must ensure that γ is disabled before either agent sees any event occur and that γ is disabled after α is generated by the plant. At $w = (\varepsilon, \varepsilon, \varepsilon)$ agent 2 does know to disable γ and thus $\mathcal{KP}_2(\varepsilon, \gamma) =$ disable. This action occurs at all global states where $w_2 = \varepsilon$, namely $(\alpha, \alpha, \varepsilon)$—which makes certain that $\alpha\gamma$ will not occur—and at $(\gamma, \varepsilon, \varepsilon)$ and $(\alpha\gamma, \alpha, \varepsilon)$. The "disable γ" action at the latter two states is irrelevant since the previous disablement actions will guarantee that we never reach these states. The complete set of control actions for event γ is summarized in Table 3.2. Note that as long as agent i takes a "disable" action for some event at w_i, this action takes precedence over any other agent's "enable" action for the same event at any global states $w' \sim_i w$, thereby ensuring that the event is disabled in all possible worlds of agent i at w.

3.3.3 Example: a Kripke-observable system for $\mathcal{I}^{DES'}$

The definitions of Kripke-observability, a joint knowledge-based protocol and Theorem 3.1 also hold for $\mathcal{I}^{DES'}$—simply replace $\mathcal{I}^{DES}$ with $\mathcal{I}^{DES'}$ in all relevant places.

Figure 3.4 (a) shows a plant G and a legal automaton E, where $\Sigma_{1,o} = \Sigma_{1,c} = \{\alpha, \gamma\}$ and $\Sigma_{2,o} = \Sigma_{2,c} = \{\beta, \gamma\}$. The projection automata of the plant are shown in (b) and (c) of figure 3.4. Thus when agent 2 initially sees "nothing", i.e., the empty string ε, it cannot determine if the plant has generated α or if no event has yet occurred. Thus agent 2 considers the plant could be in plant states 0 or 1. Similarly, when agent 1 sees $\alpha\gamma$, it cannot determine if the plant has generated $\alpha\beta\gamma$, $\alpha\beta\gamma\beta$ or $\alpha\gamma$. Thus it considers the plant could be in plant states 1, 2 or 3.

To determine if $\mathcal{I}^{DES'}$ satisfies Kripke-observability, we still defer to Definition 3.1 and check the truth values of the primitive propositions at each state of the system. The Kripke-structure of $\mathcal{I}^{DES'}$ is shown in Figure 3.5. Let $w = (3, \{1, 3\}, \{3\})$. Since the only event defined at this state in the plant is γ, we have $\pi^{DES'}(w)(\alpha_G) =$

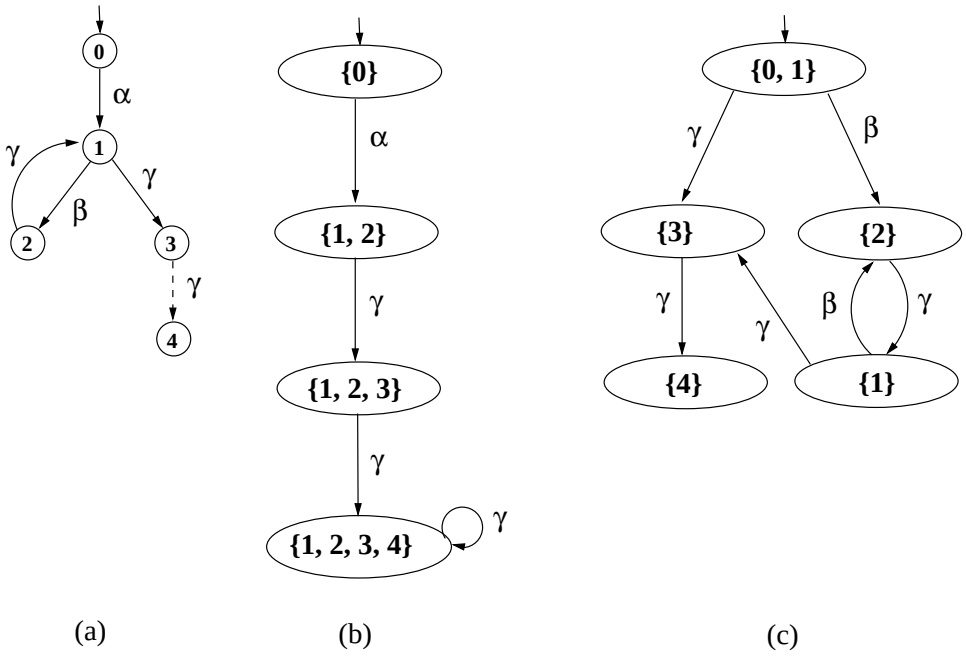

Figure 3.4: The automata for a state-based DES: (a) G and E; (b) P^{G_1}; (c) P^{G_2}.

$\pi^{DES'}(w)(\beta_G) =$ **false**. Thus the truth values of the propositions for events α and β satisfy the first disjunct of Definition 3.1. While γ is defined in the plant at state 3, it is not part of the legal automaton and therefore $\pi^{DES'}(w)(\gamma_G) =$ **true** and $\pi^{DES'}(w)(\gamma_E)$ = **false**. If the interpreted system of G and E satisfies Kripke-observability, one or the other or both agents will know to disable γ at w.

In figure 3.5, agent 1 considers two possible worlds at $w = (3, \{1, 3\}, \{3\})$, namely w and $w' = (1, \{1, 3\}, \{0, 1\})$, and hence the two states are joined by an edge with the label "1". It is not the case that agent 1 knows that γ is not legal at w since $\pi^{DES'}(w)(\gamma_E) =$ **false** but $\pi^{DES'}(w')(\gamma_E) =$ **true**.

Agent 2, on the other hand, considers that w looks like global states $(3, \{1, 2, 3, 4\}, \{3\})$ and $(3, \{1, 2, 3\}, \{3\})$. Note that agent 2's local state contains only one plant state, and therefore the only state agent 2 considers the plant to be in is plant state 3. The multiple global states with the same state of the environment reflects the different paths the plant could have taken to reach state 3. Thus the multiple possible

worlds for agent 2 at w simply indicate that agent 2 does not differentiate among the paths to plant state 3 and regardless of the path, it only ever considers it possible that the plant is in state 3. Since $\pi^{DES'}(w)(\gamma_E) = \textbf{false}$, then by definition of $\pi^{DES'}$, the truth value for γ_E is clearly **false** at $(3, \{1,2,3,4\}, \{3\})$ and $(3, \{1,2,3\}, \{3\})$. Therefore we have $(\mathcal{I}, w) \models K_2 \neg \gamma_E$.

Checking the rest of the points in the system reveals that the system is indeed Kripke-observable. Table 3.3 contains a summary of the test for Kripke-observability for the entire system.

Table 3.3: Checking Kripke-observability for G and E in figure 3.4.

w	Disjuncts of Kripke-observability satisfied		
$(0, \{0\}, \{0,1\})$	$(\mathcal{I}^{DES'}, w) \models \alpha_E$	$(\mathcal{I}^{DES'}, w) \models \neg\beta_G$	$(\mathcal{I}^{DES'}, w) \models \neg\gamma_G$
$(1, \{1,3\}, \{0,1\})$	$(\mathcal{I}^{DES'}, w) \models \neg\alpha_G$	$(\mathcal{I}^{DES'}, w) \models \beta_E$	$(\mathcal{I}^{DES'}, w) \models \gamma_E$
$(1, \{1,2,3,4\}, \{1\})$	$(\mathcal{I}^{DES'}, w) \models \neg\alpha_G$	$(\mathcal{I}^{DES'}, w) \models \beta_E$	$(\mathcal{I}^{DES'}, w) \models \gamma_E$
$(1, \{1,2,3\}, \{1\})$	$(\mathcal{I}^{DES'}, w) \models \neg\alpha_G$	$(\mathcal{I}^{DES'}, w) \models \beta_E$	$(\mathcal{I}^{DES'}, w) \models \gamma_E$
$(2, \{1,2,3\}, \{2\})$	$(\mathcal{I}^{DES'}, w) \models \neg\alpha_G$	$(\mathcal{I}^{DES'}, w) \models \neg\beta_G$	$(\mathcal{I}^{DES'}, w) \models \gamma_E$
$(2, \{1,2,3,4\}, \{2\})$	$(\mathcal{I}^{DES'}, w) \models \neg\alpha_G$	$(\mathcal{I}^{DES'}, w) \models \neg\beta_G$	$(\mathcal{I}^{DES'}, w) \models \gamma_E$
$(3, \{1,3\}, \{3\})$	$(\mathcal{I}^{DES'}, w) \models \neg\alpha_G$	$(\mathcal{I}^{DES'}, w) \models \neg\beta_G$	$(\mathcal{I}^{DES'}, w) \models K_2\neg\gamma_E$
$(3, \{1,2,3\}, \{3\})$	$(\mathcal{I}^{DES'}, w) \models \neg\alpha_G$	$(\mathcal{I}^{DES'}, w) \models \neg\beta_G$	$(\mathcal{I}^{DES'}, w) \models K_2\neg\gamma_E$
$(3, \{1,2,3,4\}, \{3\})$	$(\mathcal{I}^{DES'}, w) \models \neg\alpha_G$	$(\mathcal{I}^{DES'}, w) \models \neg\beta_G$	$(\mathcal{I}^{DES'}, w) \models K_2\neg\gamma_E$
$(4, \{1,2,3,4\}, \{4\})$	$(\mathcal{I}^{DES'}, w) \models \neg\alpha_G$	$(\mathcal{I}^{DES'}, w) \models \neg\beta_G$	$(\mathcal{I}^{DES'}, w) \models \neg\gamma_G$

The joint knowledge-based protocol for this system, as with the previous example, is straightforward for events α and β: both agents enable these events at all points in $\mathcal{I}^{DES'}$. The control actions for γ must have at least one agent disabling γ at plant state 3. That is, for each point in the interpreted system where $w_e = 3$, at least one agent must take the control action "disable γ". In fact, agent 2 is never uncertain about the plant being in state 3 since, whenever $w_e = 3$, it is always the case that $w_2 = \{3\}$. For instance, at $w = (3, \{1,3\}, \{3\})$, agent 2 knows $\neg\gamma$ and thus $\mathcal{KP}_2(w_2, \gamma) = \text{disable}$. Since every point where $w_2 = \{3\}$ is also every point where $w_e = 3$, the joint knowledge-based protocol prevents γ from occurring at plant state 3. The complete set of control actions taken by each agent for γ is shown in Table 3.4.

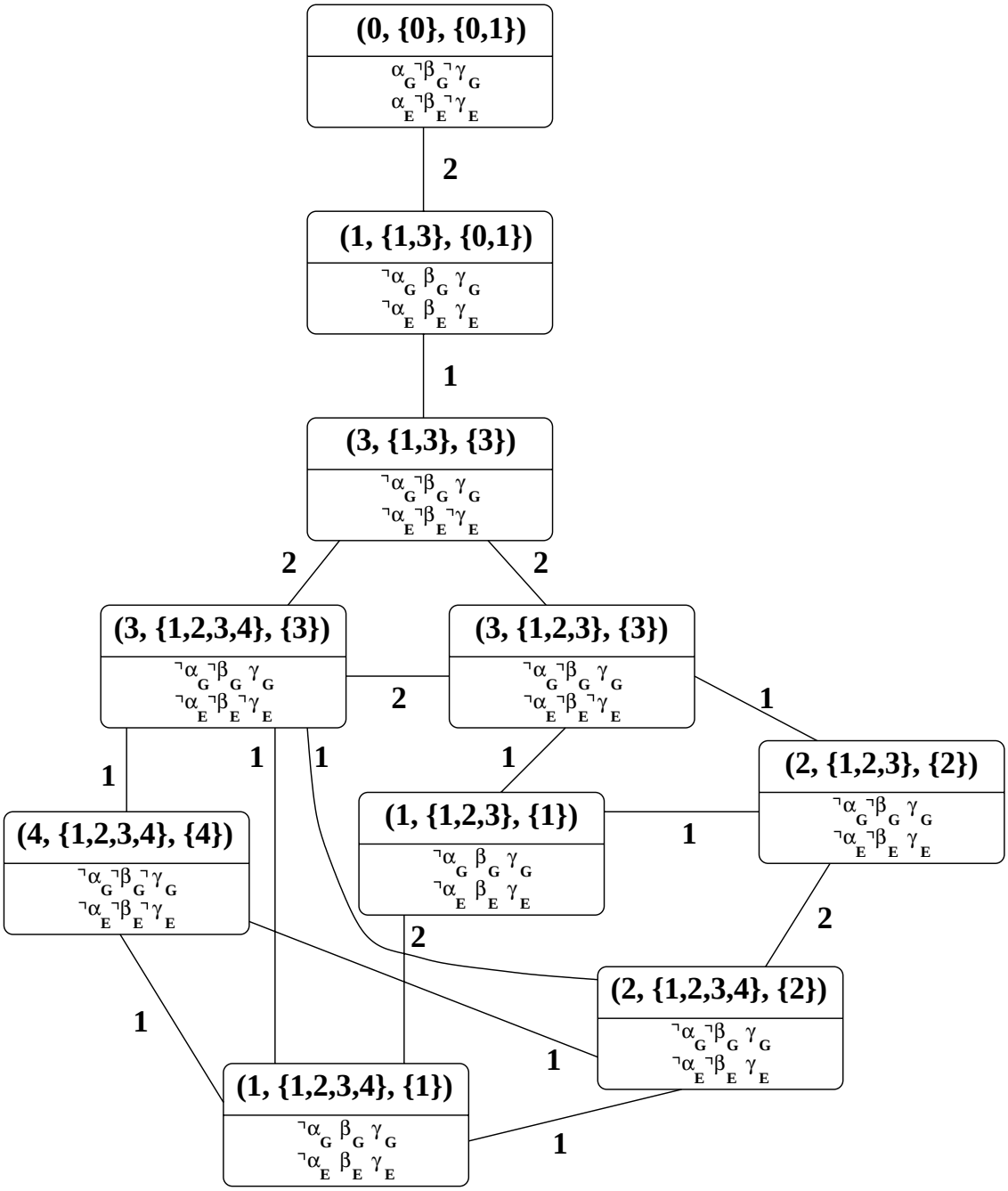

Figure 3.5: The Kripke structure for G in figure 3.4.

Table 3.4: A joint knowledge-based protocol for G and E and event γ in figure 3.4.

w	$\mathcal{KP}_1(w_1, \gamma)$	$\mathcal{KP}_2(w_2, \gamma)$
$(0, \{0\}, \{0, 1\})$	enable	enable
$(1, \{1, 3\}, \{0, 1\})$	enable	enable
$(1, \{1, 2, 3, 4\}, \{1\})$	enable	enable
$(1, \{1, 2, 3\}, \{1\})$	enable	enable
$(2, \{1, 2, 3\}, \{2\})$	enable	enable
$(2, \{1, 2, 3, 4\}, \{2\})$	enable	enable
$(3, \{1, 3\}, \{3\})$	enable	disable
$(3, \{1, 2, 3\}, \{3\})$	enable	disable
$(3, \{1, 2, 3, 4\}, \{3\})$	enable	disable
$(4, \{1, 2, 3, 4\}, \{4\})$	enable	enable

3.4 Distributed Observability

Previously we considered what it means for an agent to know a fact; however, what does it mean for a group of agents to know a fact? To find a joint knowledge-based protocol that solves the decentralized control problem, we require that the interpreted system be Kripke-observable. But even if the system is not Kripke-observable, it may be the case that the group of agents has the *combined* knowledge to generate the correct control strategy. We call this notion of successfully pooling information to generate a control decision *distributed observability*.

Distributed observability is based on the concept of *distributed knowledge* (taken from [12]). Distributed knowledge is the weakest form of group knowledge: in essence, a group has distributed knowledge of p if after combining all the knowledge of the group, p holds. This amounts to taking the intersection of all sets of worlds each agent in the group considers possible at a given state in the system.

DEFINITION **3.3** *A group* G *of agents has distributed knowledge of* $p \in \Phi$ *at state* w, *denoted* $(\mathcal{I}, w) \models D_{\mathsf{G}}\ p$, *iff* $(\mathcal{I}, w') \models p$ *for all* w' *where, for all agents* i *in a group* G, $w_i = w'_i$.

The modal operator D_{G} means "it is distributed knowledge among the agents in G" [16]. It could be the case that no individual agent knows p, but after combining their possible worlds (i.e., take the intersection of the possible worlds for the agents) the group of agents knows p—only if p holds in all the remaining possible worlds of the 'intersection'.

Stronger assertions about group knowledge include "everyone in the group knows p" and common knowledge, where "everyone in the group knows p, everyone in the group knows that everyone in the group knows p" etc. We do not consider these states of knowledge here, but merely note that there exists a hierarchy of states of group knowledge for distributed systems.

Distributed knowledge is the key to a concept we introduce, called *distributed observability*:

DEFINITION **3.4** *An interpreted system $\mathcal{I}^{DES}$ has distributed observability with respect to a group of agents G if*

$$\forall w \in \mathcal{I}^{DES}, \forall (\sigma_G, \sigma_E) \in \mathsf{R}_\Sigma$$
$$(\mathcal{I}^{DES}, w) \models \neg\sigma_G \vee \sigma_E \vee D_{\mathsf{G}} \neg\sigma_E .$$

That is, at all states in the interpreted system where an event would need to be disabled, there is distributed knowledge about whether to disable that event. Note that if $\mathcal{I}^{DES}$ is Kripke-observable then by definition $\mathcal{I}^{DES}$ has distributed observability since at every state, for each event in Σ_c, at least one agent (even before pooling knowledge) will know the correct control decision to make.

Intuitively, to solve a decentralized problem, even with communication, it would have to be the case that what one agent lacks in knowledge or information, the other can supply. Consider the case of sequences t and t' which look alike to both agents where t is legal and t' is illegal. If agent 1 were to communicate to agent 2 that it (agent 1) knows that one of t or t' has occurred, or if agent 2 were to communicate

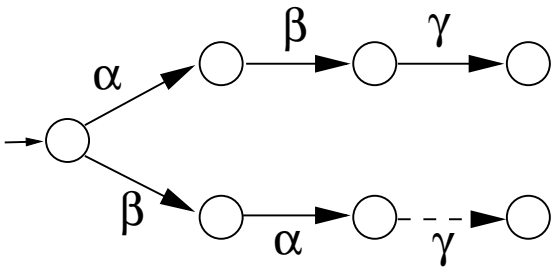

Figure 3.6: The combined DES plant G and its legal automaton E.

similar information to agent 1, communication will not help the agents make a control decision.

Using the knowledge framework we can exploit the possible-worlds model to identify system states that are indistinguishable to both agents (and where, therefore, information pooling would be of no help). Further, we can identify states where an agent's ability to make control decisions would be improved by communication.

We present two examples where the two agents have partial observation of a system: one where the pooling of possible worlds is not enough to achieve control and one where we believe that combined knowledge can achieve control.

3.4.1 Example: when pooling knowledge is not enough

In figure 3.6 the language generated by the legal automaton is not Kripke-observable[1] if $\Sigma_{1,o} = \{\alpha\}$, $\Sigma_{1,c} = \{\alpha, \gamma\}$, $\Sigma_{2,o} = \{\beta\}$, and $\Sigma_{2,c} = \{\beta\}$. When agent 1 sees α (equivalently, agent 2 sees β), it does not know whether or not $\alpha\beta$ or $\beta\alpha$ has occurred. Thus a control decision about γ cannot be reached. The Kripke structure in figure 3.7 shows that $\mathcal{I}^{DES}$ is not Kripke-observable. To see this, suppose the system were Kripke-observable. Then when $w = (\beta\alpha, \alpha, \beta)$, it must be that agent 1 knows $\neg\gamma_E$ in its set of possible worlds at w—since $(\mathcal{I}^{DES}, w) \not\models \neg\gamma_G \vee \gamma_E$. In fact, this is

[1]This example arose from discussions K. Rudie had with S. Lafortune, F. Lin, A. Overkamp and D. Teneketzis.

not the case because $(\mathcal{I}^{DES}, (\alpha\beta, \alpha, \beta)) \models \gamma_E$.

If both agents were to pool their knowledge at a state where agent 1 sees α and agent 2 sees β, so that the resulting possible worlds are $(\alpha\beta, \alpha, \beta)$, $(\beta\alpha, \alpha, \beta)$, and $(\alpha\beta\gamma, \alpha, \beta)$, the Kripke structure indicates that still there is no distributed knowledge about γ_E for the same reasons that the system is not Kripke-observable: the conflicting truth value of γ_E at states $(\alpha\beta, \alpha, \beta)$ and $(\beta\alpha, \alpha, \beta)$. That is, distributed observability is not satisfied.

When distributed observability is not satisfied this tells us that at some place, pooling information does not help. In this example, by the time agent 1 sees α and agent 2 sees β, the relative ordering of α and β has been lost, i.e., even if at that point agent 2 were to tell agent 1 that it has seen β, that would not convey to agent 1 whether $\alpha\beta$ or $\beta\alpha$ had occurred. This would suggest that the agents must communicate prior to agent 1 seeing α and agent 2 seeing β.

One possible communication protocol could assume that an agent sends a query as soon as it is uncertain about whether to disable an event. Unfortunately, such a strategy is highly sensitive to small communication delays. In this example, because agent 2 sees and controls only β, there is never a situation when agent 2 is confused about its control decisions. So it never sends a query to agent 1. Agent 1 would need to submit a query when it sees α. If only α has happened and agent 1 sends a query to agent 2, then it would appear that a decision about γ can be made since the pooled information would indicate that the only possible world is $(\alpha, \alpha, \varepsilon)$. However, if β had taken place before agent 2 receives the query, agent 1 would not know whether or not to disable γ since it would not know if β had occurred just before α or just after α happened. That is, the usefulness of pooled information depends on whether β can occur after α has occurred but before agent 2 receives the query. In other words, even if a query results in a response, the solution is sensitive to the precise moment the query is received.

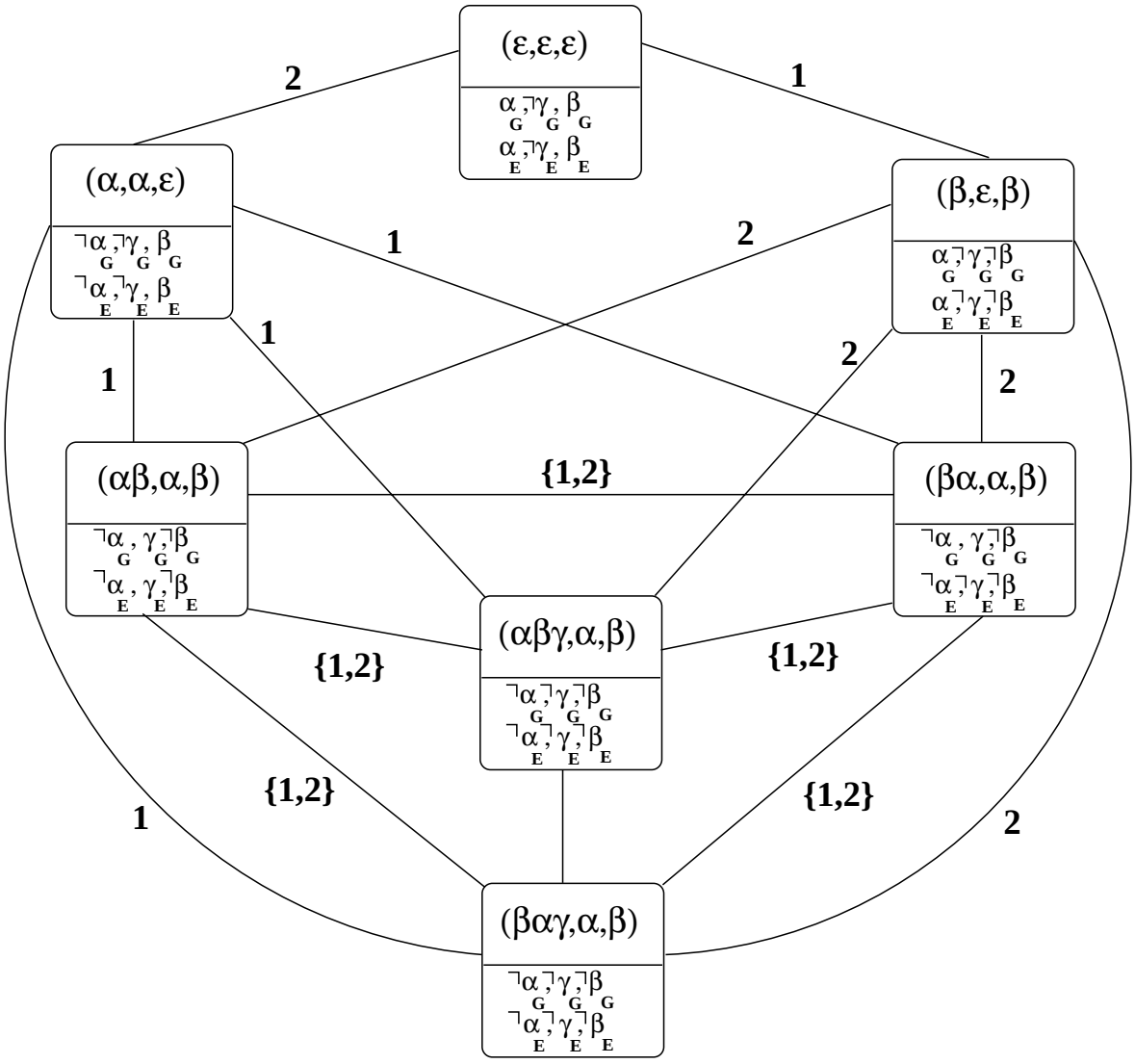

Figure 3.7: The Kripke structure for the plant in figure 3.6.

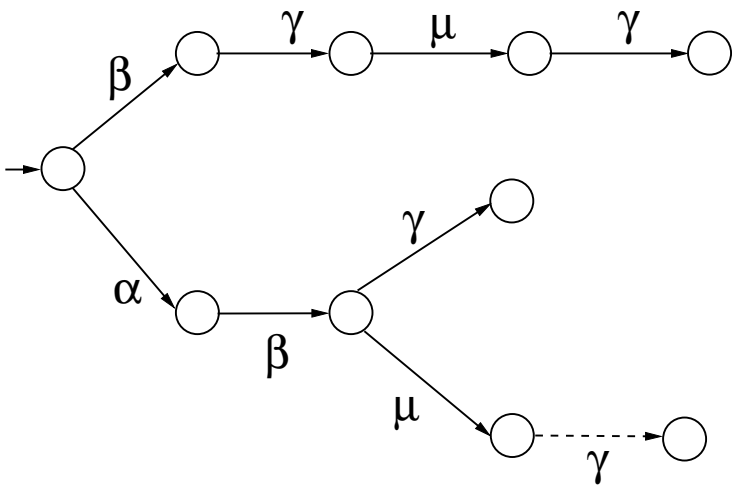

Figure 3.8: A combined DES plant G and legal automaton E.

3.4.2 Example: when pooling knowledge is enough

The legal language corresponding to the legal automaton illustrated in figure 3.8 is not Kripke-observable. Let $\Sigma_{1,o} = \{\alpha\}$, $\Sigma_{1,c} = \{\alpha, \gamma\}$, $\Sigma_{2,o} = \{\beta, \mu\}$, and $\Sigma_{2,c} = \{\beta, \mu, \gamma\}$. Upon observing α, agent 1 (which sees only α) does not know if $\alpha\beta$ or $\alpha\beta\mu$ has occurred and hence does not know whether or not to disable γ. Similarly by observing $\beta\mu$, agent 2 would not know whether $\alpha\beta\mu$ or $\beta\gamma\mu$ had occurred and could make no decision about disabling γ. However, we can check the Kripke structure and see that distributed observability is satisfied. That is, when an agent is unable to make a control decision, pooling information will help. In fact, an agent has more flexibility: to submit a query every time it is stuck is possibly unnecessary. If each agent has available to it a record of the history of its queries then it may be possible to deduce further information based on queries it has not received from the other agent. We illustrate this below. In addition, there remains the issue of when information should be pooled. There may be several states where pooling is beneficial and it may be possible to ascertain whether communication should be delayed to the last possible moment or should occur as early as possible.

In this example, as soon as agent 1 sees α it does not know whether or not to disable γ. At what state should it communicate or query agent 2 so that they can pool their knowledge? We can assume here that communication between agents occurs instantaneously so that as soon as one agent cannot continue, the other agent receives a query to pool knowledge. In this case, agent 2 can continue making control decisions until it sees $\beta\mu$ at which point it must submit a query to pool knowledge. On the other hand, if agent 2 sees $\beta\mu$ but has already received a query from agent 1 (after agent 1 sees α), then agent 2 no longer needs to query as it knows that previously agent 1 did not know what to do about γ. Therefore agent 2 can deduce that α must have occurred.

Chapter 4

Communication and Decentralized DES

In chapter 3 we used knowledge models to analyze decentralized discrete-event problems that satisfied specific conditions to generate a control solution. The solution precluded the possibility of agents pooling or communicating information regarding their partial view of the system. The notion of distributed observability for interpreted systems provides a starting point from which we begin our understanding of how agents might communicate to solve a particular class of decentralized control problems.

This chapter 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?

4.1 Knowledge, communication and control

In this section we present a broad overview of the motivation explaining why we want to introduce communication into decentralized DES. As well, we indicate the underlying assumptions we make regarding the nature of communication in decentralized systems.

4.1.1 Why communicate?

When the correct control decision cannot be reached in the absence of communication, as was the case for the example in figure 3.8, sharing information with other agents could lead to a control solution. Thus an agent may communicate to allow another agent to reach the correct control decision. Because an agent has a partial view of the system, if a communicating agent shares information with another agent at state x, it is also communicating at every state that it finds indistinguishable from x, a notion termed *consistency* [31]. Therefore communication occurs for two reasons: to solve a control problem, and to satisfy consistency.

The strategy for communication we present in this chapter does, by definition, satisfy consistency. In the construction of the communication protocol we instead insist that our procedures (1) render the protocol *well-defined* in the sense that there is no ambiguity when an agent must communicate (this is described in more detail in section 4.4); and (2) when taking into account any prior information that an agent could receive, the agent's view of the world—in light of this new information—is correctly refined. As an example of this latter point, suppose that our strategy determines that agent i communicates at one of its local states $w_i = \{x, y, z\}$, where $x, y, z \in Q^G$ and $w_i \in Q^{P^{G_i}}$ for some plant G. That is, with its partial view of the world, agent i cannot distinguish between plant states x, y and z. Further suppose that a prior communication from another agent allows agent i to distinguish x from y (i.e., w_i should really be $\{x, z\}$). We want our protocol to ensure that agent i communicates its updated view of the world as refined by this additional information.

4.1.2 Who communicates?

In the previous chapter we described how to ascribe knowledge to agents in a decentralized discrete-event control problem. As long as at least one agent has enough

knowledge to make the correct control decision and takes the correct control action, a solution to the control problem is achieved.

When an agent does not have sufficient knowledge to make the correct control decision—and there is no other agent capable of making the decision—we assume that another agent in the system communicates information to facilitate the correct control decision.

4.1.3 What to communicate?

At any point in the interpreted system an agent has access to two pieces of information: its local state (i.e., the set of states it considers the plant could be at) and its knowledge of the system at that local state. We assume that the communicating agent sends its local state to the agent that lacks knowledge. An agent receiving communication then updates its own local state by intersecting its local state with the communicated local state.

For example, suppose an agent must make a control decision to disable event σ at plant state q, but σ is allowed to happen at plant state q'. If the agent considers it possible that the plant could be in either state q or q', then it cannot make the correct control decision to disable σ if the plant is actually at state q. Suppose another agent considers it possible that the plant is either at state q or state q'' and σ is not defined at state q''. Therefore, if the agent lacking knowledge about σ is sent the information $\{q, q''\}$, it updates its own local state to $\{q, q'\} \cap \{q, q''\} = \{q\}$. Now the correct control decision (i.e., to disable σ) can be made since after communication the states q and q' can be distinguished.

4.1.4 Where to communicate

An agent is confused if it must make a control decision and cannot distinguish between a state leading to a legal sequence and a state leading to an illegal sequence. In

section 4.2 we describe our strategy for identifying places (e.g., states) where communication to achieve a control solution occurs. These are places in the system where the information an agent receives leads to a situation where the agent makes the correct control decision. As noted earlier, we also must formulate a communication protocol that takes into account the effects of prior communication from another agent. A procedure for ensuring that an agent's view of the world is refined appropriately is presented in section 4.3.

4.1.5 When to communicate: a communication protocol

We represent the action of agent i communicating with agent j at some state q in the plant (for purposes of solving the control problem) by the event com_{ij}:q. Therefore, when a place where agents "communicate for control" has been identified (e.g., state q), we insert a communication event into an updated version of the plant at that state. Subsequently, communication events must be incorporated at all states that an agent finds indistinguishable from the places it communicates for control (e.g., states that look like q). Once all the communication events have been incorporated into the plant, the communication protocol for each agent is derived by calculating its projection automaton of the augmented plant. To ensure that the projection automata reflect the effects of communication on an agent's view of the world, we must ascertain that the communication events have been added to the augmented plant at the appropriate states. The updated plant can then be translated into our knowledge model where we can determine whether or not the addition of communication now renders the system Kripke-observable.

4.2 Communication for Control

In section 3.3, 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 place in the interpreted system that contributes to the system not being Kripke-observable. We then speculated in section 3.4 about the role distributed observability might play in providing agents with more information to make the correct control decisions. The notion of distributed observability suggests that pooling takes place just before a control decision needs to be made. However, it is possible to come up with examples where a control solution exists but where pooling possible worlds under the conditions of distributed observability does not lead to an agent having the knowledge to make the correct control decision. Therefore, in our strategy for communication, we identify places where pooling information at that place *is* helpful.

We have established that an agent requires extra information, for example, communicated from another agent, when it cannot distinguish an illegal sequence from a legal sequence and it must make the correct control decision. We have yet to establish how to identify specific places where communication will give agents the knowledge to solve the control problem. In this section we claim that, subject to certain conditions, we can always find a place for agents to communicate so that a control solution will eventually be reached. At such a place in the interpreted system, the communicating agent i can provide agent j with information that allows j to distinguish 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 places where communication occurs to solve the control problem.

DEFINITION **4.1** *A* **communication state** *is a state $q \in Q^G$ where agent i communicates to agent j (for $i, j \in \{1, 2\}$ and $i \neq j$) so that agent 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 agent is imparted to the agent responsible for making a control decision. The communicated information allows the latter agent to enable or disable the appropriate event at a subsequent point in the system.

DEFINITION **4.2** *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 (2.1) 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 places in the interpreted system where an illegal and a legal sequence in $L(G)$ look alike using canonical projection.

DEFINITION **4.3** *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/agent* 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 agent 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 agent i at state q will contain q and q'.

DEFINITION **4.4** *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 **4.5** *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 an agent 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 and corollary that follow, we use P^A to identify a canonical projection operator from Σ^* to A^*, where A is a subset of Σ.

LEMMA **4.1** *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 $P^B(t) = P^B(t')$.*

Informal Proof. Let $A' = \{\sigma \mid \sigma \in \Sigma \text{ and } \sigma \notin A\}$ and $B' = \{\sigma \mid \sigma \in \Sigma \text{ and } \sigma \notin B\}$ (i.e., A' is the complement of A, B' is the complement of B). Since $B \subseteq A$, therefore $A' \subseteq B'$. Note that P^A will replace events in A' by ε. Then P^B will replace events in the larger set B' by ε. Thus $P^A(t) = P^A(t')$ implies that $P^B(t) = P^B(t')$. The result can be proved more formally using induction on the length of strings.

□ LEMMA 4.1

Thus sequences that are indistinguishable to an agent are also indistinguishable to other agents that observe fewer events.

We prove here that under a certain condition we can find places where a communicating agent can eliminate the confusion of the agent incapable of making the correct control decision. The confused agent 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 agents see)

could solve the control problem. Otherwise one agent lacks observations that could not necessarily be supplied by the other agent.

THEOREM **4.1** *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 4.1 (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:

$$\begin{aligned} P_i(t') &= P_i(t) \\ &= P_i(u\sigma v) \\ &= P_i(u)P_i(\sigma)P_i(v) \end{aligned} \tag{4.1}$$

Since (u, t') is a maximal-P pair, $P(u) = P(t')$. Since $\Sigma_{i,o} \cup \Sigma_{j,o} \subseteq \Sigma_o$, by Lemma 4.1 $P_i(u) = P_i(t')$. Therefore (4.1) holds only if $P_i(\sigma)P_i(v) = \varepsilon$. This leads to a

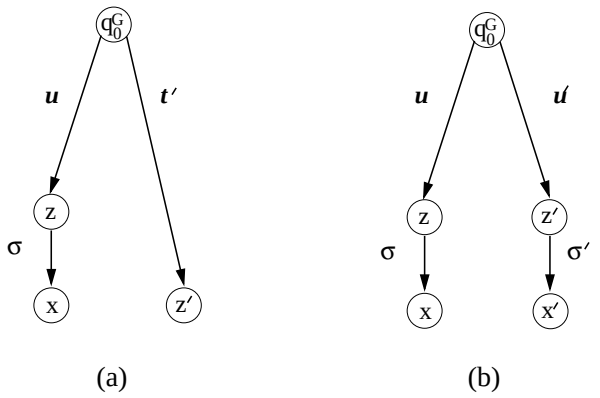

Figure 4.1: Identifying a communication state.

contradiction as $P_i(\sigma) \neq \varepsilon$.

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

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 $t' = w'v'$:

$$\begin{aligned} P_j(w') &= P_j(u\sigma) \\ &= P_j(u)P_j(\sigma) \\ &= P_j(t')P_j(\sigma) \quad (\text{since } \Sigma_{j,o} \subseteq \Sigma_{i,o} \cup \Sigma_{j,o} \text{ and } P(u) = P(t'), \\ &\qquad\qquad \text{by Lemma 4.1, } P_j(u) = P_j(t')) \\ &= P_j(w')P_j(v')P_j(\sigma) \end{aligned} \tag{4.2}$$

For (4.2) to hold, $P_j(v')P_j(\sigma) = \varepsilon$ which leads to a contradiction as $P_j(\sigma) \neq \varepsilon$ (since

$\sigma \in \Sigma_{j,o}$).

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 4.1 (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

$$\begin{aligned} P_i(uw) &= P_i(u'\sigma') \\ P_i(u)P_i(w) &= P_i(u')P_i(\sigma') \\ &= P_i(u)P_i(\sigma') \quad (\text{ since } \Sigma_{i,o} \subseteq \Sigma_{i,o} \cup \Sigma_{j,o}, \\ &\qquad \text{by Lemma 4.1, } P_i(u) = P_i(u')) \end{aligned} \tag{4.3}$$

For (4.3) 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

$$
\begin{aligned}
P_j(\hat{w}) &= P_j(u'\sigma') && (4.4)\\
&= P_j(u')P_j(\sigma') \\
&= P_j(u)P_j(\sigma') \quad (\text{since } \Sigma_{j,o} \subseteq \Sigma_{i,o} \cup \Sigma_{j,o}, \text{ by Lemma 4.1, } P_j(u) = P_j(u')) \\
&= P_j(\hat{w}\hat{v})P_j(\sigma') \\
&= P_j(\hat{w})P_j(\hat{v})P_j(\sigma') && (4.5)
\end{aligned}
$$

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

(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':

$$
\begin{aligned}
P_i(u\sigma v) &= P_i(u'\sigma' v') \\
P_i(u)P_i(\sigma)P_i(v) &= P_i(u')P_i(\sigma')P_i(v') \\
&= P_i(u)P_i(\sigma')P_i(v') \quad (\text{since } \Sigma_{i,o} \subseteq \Sigma_{i,o} \cup \Sigma_{j,o}, \\
&\qquad\qquad \text{by Lemma 4.1, } P_i(u) = P_i(u'))
\end{aligned}
$$

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

$$\begin{aligned} P_j(uv) &= P_j(u'\sigma') \\ P_j(u)P_j(v) &= P_j(u')P_j(\sigma') \\ &= P_j(u)P_j(\sigma') \quad (\text{ since } \Sigma_{j,o} \subseteq \Sigma_{i,o} \cup \Sigma_{j,o}, \\ &\qquad \text{by Lemma 4.1, } P_j(u) = P_j(u')) \end{aligned} \tag{4.6}$$

Since $v = \sigma w$, for (4.6) 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 procedure presented in (4.4).

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 $P_j(\hat{w}) = P_j(u\sigma)$.

□ THEOREM 4.1

The idea of Theorem 4.1 is that when agent 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 place—somewhere along either t or t'—where agent j can distinguish between t and t'. At this place or communication state, agent

j sends its local state or local view to agent i. Prior to receiving the communication, agent i does not know whether or not the current state of the system leads to an illegal sequence or a legal sequence. When agent i updates its own local state with the communicated information, agent i can tell the difference between the legal and the illegal sequence.

4.2.1 Avoiding unintentional communication

We make an assumption regarding the structure of the automaton G. In particular, we want to avoid situations where the identification of a communication state results in unintentional communication. This corresponds to the case in Theorem 4.1 where the place we want to communicate is the state reached by both $u\sigma$ and $u'\sigma$ (where (u, u') is the maximal-P pair for a pair of sequences we want to distinguish). When an intention to communicate is thwarted by the structure of the plant, we assume that we can "split" that state in the sense described below.

We will use Theorem 4.1 to identify states where communication will be inserted, as follows. For every t, t' satisfying the hypotheses of Theorem 4.1 for agent i, we find a maximal-P pair (u, u'). From the construction in the proof of the theorem, we know that after either u or u' (or after both), there is an event in $\Sigma_{j,o}$, called σ_j, leading to state x along t (or σ_j' leading to state x' along t'). That state x is a state where the intersection of the agents' views (i.e., $x_i \cap x_j$) does not contain states along t *and* states along t'. That is, it definitively indicates to agent i that the system has progressed along t and not along t' (respectively, along t' and not along t).

Actually, so far we have hidden a subtle possibility. If after both u and u', there are events σ and σ' (as in figure 4.1(b)), leading to *the same state* (i.e., $x = x'$ in figure 4.1(b)) or if after u there is an event σ leading to the same state that t' leads to (i.e., $x = z'$ in Figure 4.1(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 agent j to agent i that

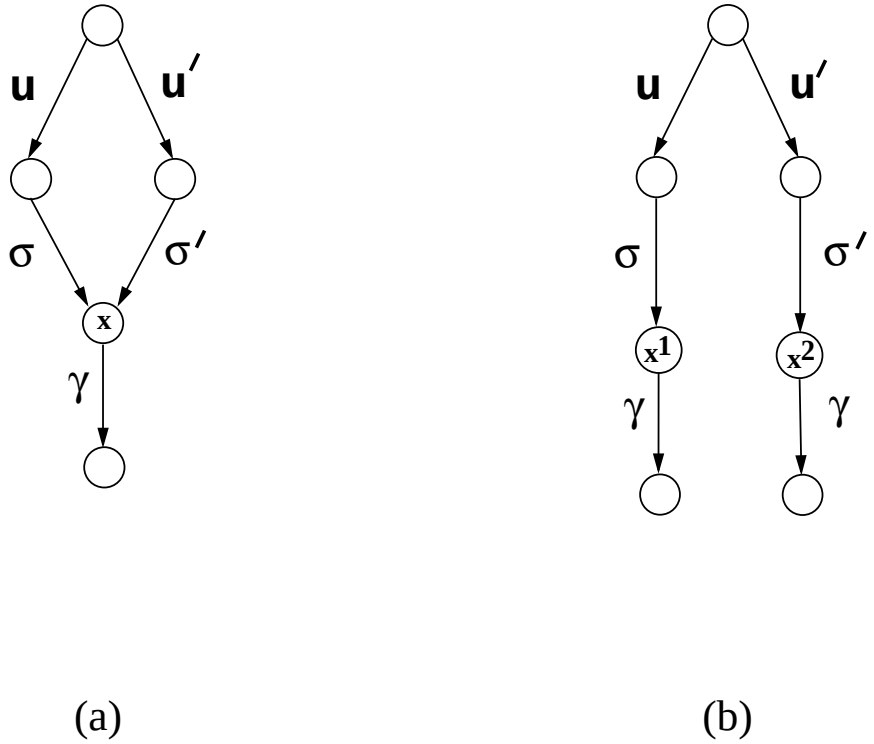

Figure 4.2: Splitting G: (a) intention is for communication to occur at state x after $u\sigma$; (b) rewrite G and split state x to find a definitive communication state x^1.

it is at state x would not yield any helpful information for agent 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 4.2. In figure 4.2(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 agent j—after seeing $P_j(u\sigma)$—communicates its local view of state x, with the intention of allowing agent i to distinguish between $u\sigma$ and $u'\sigma'$. But communication occurs whenever agent j believes the plant to be at state x. This happens not only when agent j sees $P_j(u\sigma)$ but also when it sees $P_j(u'\sigma')$. Yet we only want agent j to communicate after $P_j(u\sigma)$ or $P_j(u'\sigma')$ and not after both. In figure 4.2(b) we rewrite G and split state x to find a definitive communication state x^1.

Now, either state x^1 or state x^2 in figure 4.2 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, 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 plant G has been rewritten to accommodate all occurrences of the above scenario.

4.2.2 Finding a place to communicate: picking control communication pairs

In the Kripke structure, a global state where Kripke-observability is not satisfied corresponds to a world w where for all $i \in \mathsf{G}_\sigma$ the following holds: $\mathcal{I}^{DES'} \models \sigma_G \wedge \neg\sigma_E \wedge$

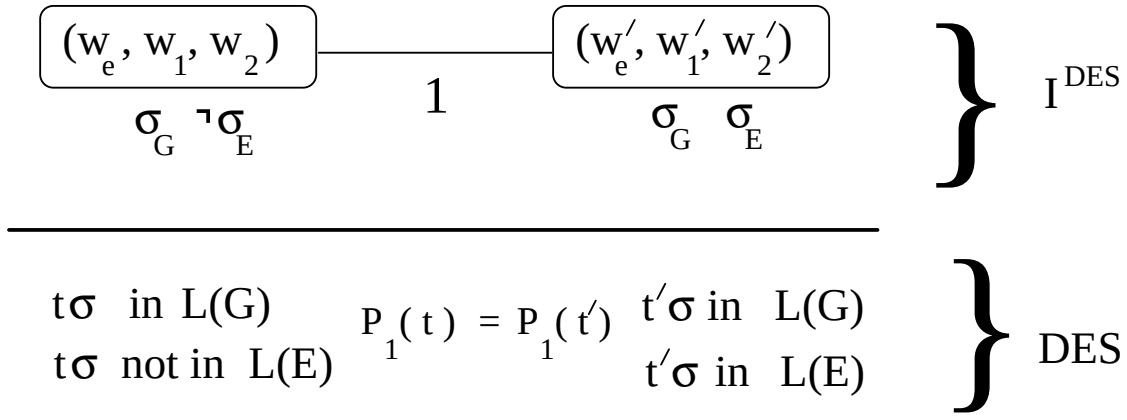

Figure 4.3: Reasoning about knowledge in the Kripke structure associated with $\mathcal{I}^{DES}$ allows us to identify *where* agents do not have enough information to solve the control problem. The diagrams above (in the knowledge theory framework) and below (in the DES environment) the line are equivalent statements about what it means to not solve the control problem.

$\neg K_i \neg \sigma_E$. That is, agent i does not have the knowledge to disable event σ. Therefore there exists a state w' that is indistinguishable from w to agent i where σ is allowed to happen (i.e., σ_G and σ_E hold at w'). Figure 4.3 shows equivalent notions of what we mean in the knowledge world (top of figure) for agent 1 to not have the knowledge to disable σ at global state w and its equivalent translation into DES theory (bottom of figure). To find a place to communicate, we will want to find the sequences t and t' as noted in the figure. A communication state will be identified as a state that occurs somewhere along the path from q_0^G to $\delta^G(t\sigma, q_0^G) = w_e$ or $\delta^G(t'\sigma', q_0^G) = w_e$.

Thus, using the Kripke structure we can identify a state, say q, in the plant where without communication a decentralized agent might not be able to make the correct control decision. If communication from agent j to i occurs somewhere along the paths to state q, agent j could give agent i the knowledge to disable σ at state q of the plant. Thus we need to identify those paths along which communication could occur.

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 to distinguish every sequence that leads to y from every sequence that leads to y'. Since there may be infinitely many sequences leading to y (due to cycles in the plant), it appears on the face of it that comparing all pairs t, t' that lead to states y, y', respectively, is an intractable task. However, we can exploit the finite-state representation of a Kripke structure by making the following observation. We conjecture that when there are infinitely many sequences leading to state y, we need only reconstruct those paths that satisfy the following (i)a path from the initial state to y that contains no cycles; (ii)a path from the initial state to y that contains one iteration of cycles that has embedded in it one of the paths identified in (i); (iii)those paths that contain just one instance of any self-loops or cycles that extend from y and return to y. The identification of such paths (including those with cycles) in a directed graph can be performed using a dynamic-programming algorithm in $O(n^3)$ time, where n is the number of nodes in the graph[10].

We describe our intuition via the example in figures 4.4 and 4.5. Suppose that agent 1 sees and controls a and b while agent 2 sees b and c. The states of the Kripke structure for the plant shown in figure 4.4 are simply the states of the monitoring automaton for the same plant. The monitoring automaton of interest is shown in figure 4.5. In the associated Kripke structure (not illustrated here) Kripke-observability fails at state $(6, \{1, 4, 6\}, \{2, 5, 6\})$ because agent 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 $b_G =$ **true** and $b_E =$ **false**.

There are two other states—$(1, \{1,4,6\}, \{0,1,3,4\})$ and $(4, \{1,4,6\}, \{0,1,3,4\})$—that agent 1 cannot distinguish from $(6, \{1,4,6\}, \{2,5,6\})$. At both these states ($(1, \{1,4,6\}, \{0,1,3,4\})$ 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 agent 1 not knowing $\neg b_E$ at state $(6, \{1,4,6\}, \{2,5,6\})$.

We use the monitoring automaton (shown in figure 4.5) to reconstruct the t and t' sequences, such that $\hat{\sigma} = b$, that will satisfy the hypothesis of Theorem 4.1. Thus we want to find paths to state $(6, \{1,4,6\}, \{2,5,6\})$ that correspond to some $t\hat{\sigma} \notin L(E)$. In addition, we want to find paths to states $(1, \{1,4,6\}, \{0,1,3,4\})$ and $(4, \{1,4,6\}, \{0,1,3,4\})$ that correspond to some $t'\hat{\sigma} \in L(E)$ where $P_1(t) = P_1(t')$.

We begin by looking at state $(6, \{1,4,6\}, \{2,5,6\})$. Our conjecture says we first look at the paths to this state that contain no cycles: thus $t = daba$ or $t = abcdaba$ or $t = abcabcdaba$. There is a path that contains one iteration of a cycle: $t = abcabcabcdaba$. There are no paths extending from and returning to $(6, \{1,4,6\}, \{2,5,6\})$ so we are done.

Similarly we examine the paths to state $(1, \{1,4,6\}, \{0,1,3,4\})$. There is one path with no cycles: $t' = abca$. Additionally, there is a cycle that extends from $(1, \{1,4,6\}, \{0,1,3,4\})$ and returns to $(1, \{1,4,6\}, \{0,1,3,4\})$. We need only reconstruct one iteration of the cycle: $t' = abcabca$.

Finally, the paths to state $(4, \{1,4,6\}, \{0,1,3,4\})$ yield two possibilities that do not contain cycles: $t' = abcda$ or $t' = abcabcda$. There is another path that does contain one iteration of a cycle: $t' = abcabcabcda$.

In this example we have five possibilities for t' and only four possibilities for t; however, we need only consider five t, t' pairs, namely those pairs that have the same projection according to agent 1. For example, one pair of sequences would be $t = daba$ and $t' = abca$ because $P_1(t) = P_1(t') = aba$. Our claim is that the identification of these five pairs of t and t' sequences is sufficient to determine where communication

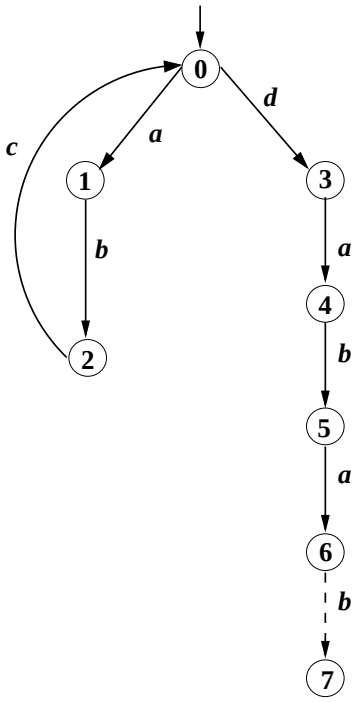

Figure 4.4: Finding places to communicate in the presence of cycles.

should be added for purposes of solving the control problem.

The proof of Theorem 4.1 yields the following update to our definition of a communication state:

DEFINITION **4.6** *Given t, t' satisfying the hypotheses of Theorem 4.1, 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 on p. 62);*

(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 on p. 63);*

(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 on p. 65).*

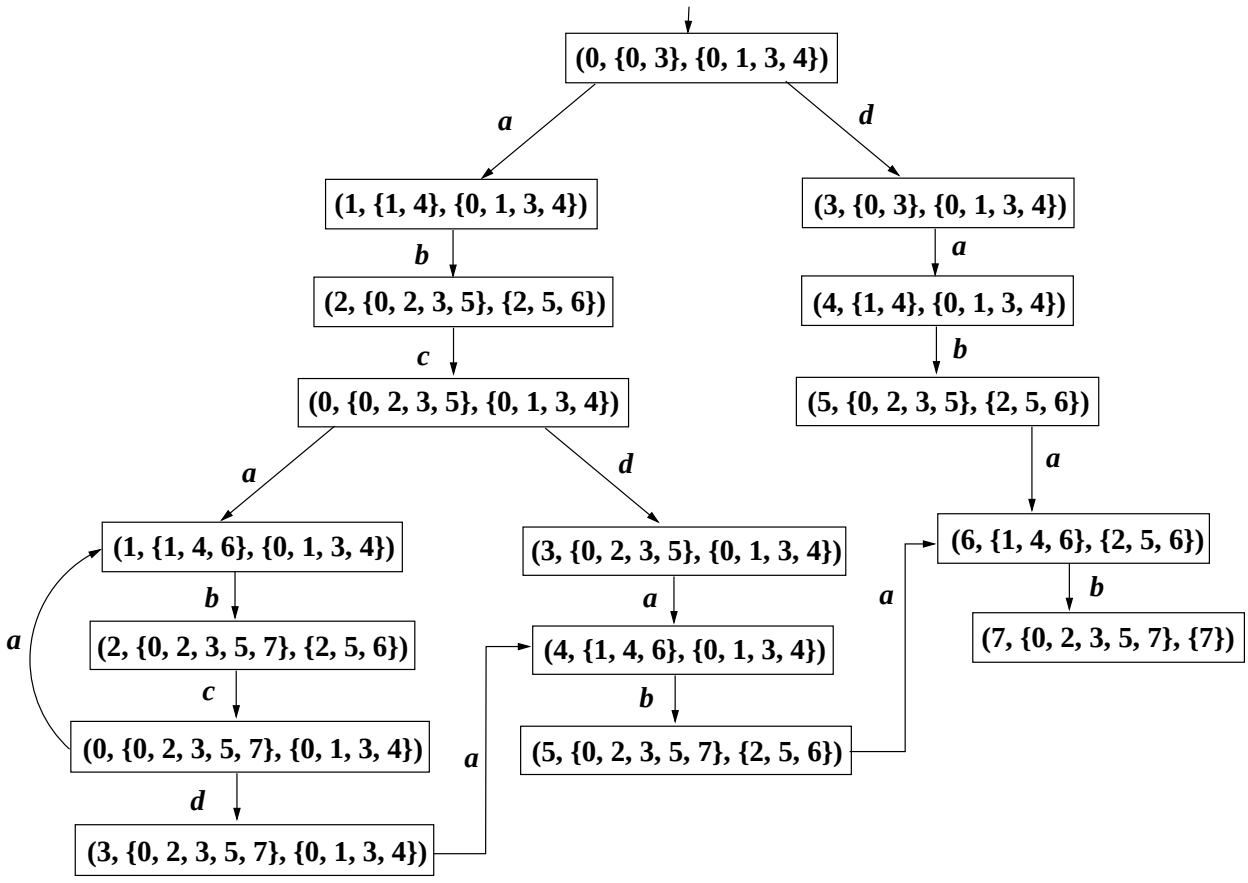

Figure 4.5: The monitoring automaton for the plant in figure 4.4.

In the definitions that follow, the sequences t and t' are those satisfying the hypothesis of Theorem 4.1.

DEFINITION **4.7** *A* **control sequence for communication state q** *is the finite sequence along which a communication state has been identified. If t is the control sequence for q then t' is a* **control twin** *for t. (Equivalently, if t' is a control sequence for q then t is the control twin for t'.)*

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

DEFINITION **4.8** *A* **control communication pair** *for agent i is a pair (q, t) and consists of a communication state q and a control sequence t.*

Communication from agent i that allows agent j to make the correct control decision about an event σ after sequence t occurs, happens along sequence t at state q.

DEFINITION **4.9** *A* **communication sequence s for a control communication pair (q, t)** *is a prefix of t if q is a bad state with respect to $t\sigma$ (or s is a prefix of t' if q is a good state with respect to $t'\sigma$) that leads to q (i.e., $\delta^G(s, q_0^G) = q$).*

We can now uniquely identify when and where agents communicate to solve the control problem: communication from one agent to another occurs at a communication state q, after the communication sequence s is observed by the communicating agent, say agent i. The idea is that agent i communicates its local view q_i to agent j when the plant is at state q. The sets $\mathcal{C}_{12}$ and $\mathcal{C}_{21}$ store the control communication pairs for agents 1 and 2, respectively.

DEFINITION **4.10** *The* **communication event** *associated with a control communication pair $(q, t) \in \mathcal{C}_{ij}$ is com_{ij}:q.*

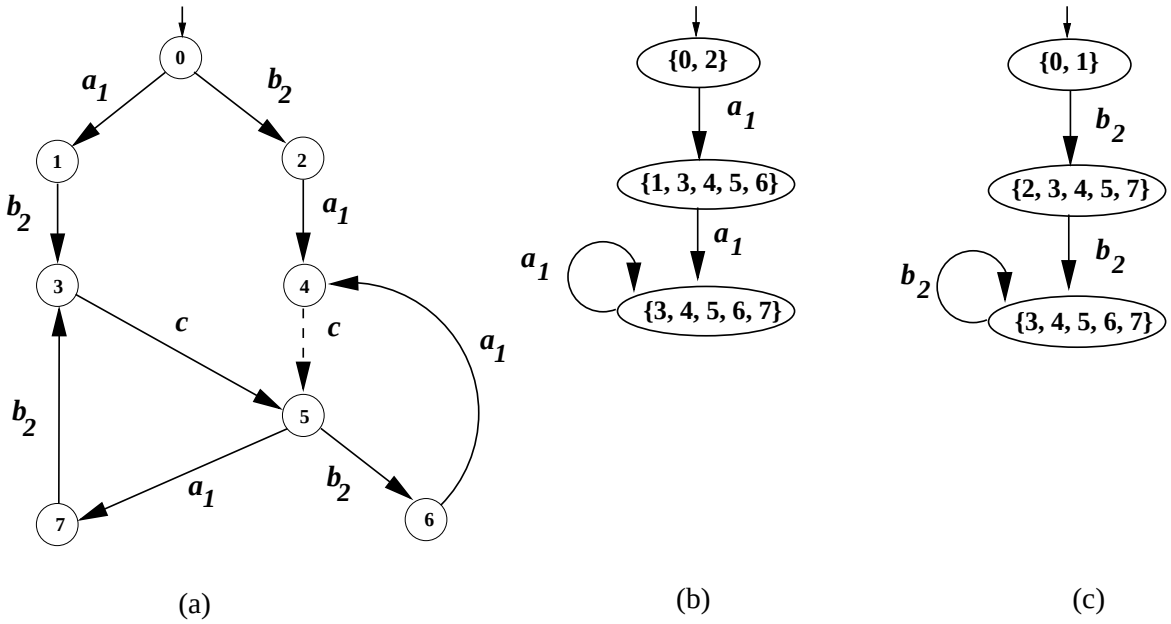

Figure 4.6: The projection automata of a DES plant G: (a) the plant G; (b) P^{G_1}; (c) P^{G_2}.

This notation represents the action of agent i communicating its local state to agent 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 agents i and j but is controllable only by agent i.

We illustrate our strategy for choosing communication states with the plant shown in figure 4.6 (a). In this example, agent 1 sees a_1 and controls events a_1 and c while agent 2 sees and controls b_2. The interpreted system constructed from this plant and legal automaton is not Kripke-observable because agent 1 does not know that event c should be disabled at state 4. In particular, agent 1 cannot distinguish among the following global states (which correspond to the sequences in which agent 1 sees a_1a_1):

- $(3, \{3, 4, 5, 6, 7\}, \{3, 4, 5, 6, 7\})$ where c_G and c_E hold;
- $(4, \{3, 4, 5, 6, 7\}, \{3, 4, 5, 6, 7\})$ where c_G and $\neg c_E$ hold;

- $(5, \{3,4,5,6,7\}, \{3,4,5,6,7\})$ where $\neg c_G$ and $\neg c_E$ hold;
- $(6, \{3,4,5,6,7\}, \{3,4,5,6,7\})$ where $\neg c_G$ and $\neg c_E$ hold; and
- $(7, \{3,4,5,6,7\}, \{3,4,5,6,7\})$ where $\neg c_G$ and $\neg c_E$ hold.

Because of the conflicting truth values for c_E at the first two global states, agent 1 does not have enough knowledge to make the correct control decision about c.

Back at the plant in figure 4.6(a), agent 1's lack of knowledge in $\mathcal{I}^{DES}(G,E)$ corresponds to the existence of sequences t, t', event $\hat{\sigma} \in \Sigma_{1,c}$ and $P_1(t) = P_1(t')$ where $t'\hat{\sigma}$ is legal but $t\hat{\sigma}$ is illegal: $t = b_2a_1$, $t' = a_1b_2$ and $\hat{\sigma} = c$. Therefore, by Theorem 4.1 we can find a communication state q where agent 2 can communicate q_2 to agent 1, allowing the latter agent to distinguish between $t'\hat{\sigma}$ and $t\hat{\sigma}$.

The good states with respect to $t'\hat{\sigma}$ are 0, 1 and 3, while the bad states with respect to $t\hat{\sigma}$ are 0, 2 and 4. A maximal-P pair (in fact, the only maximal-P pair in this case) for t and t' is $(u, u') = (\varepsilon, \varepsilon)$. The communication state is determined by the nature of the events that directly follow u and u'.

The next event after u is b_2 and the next event after u' is a_1. This corresponds to the second category of states described in definition 4.6. Therefore, agent 2 communicates along the illegal sequence t where the communication sequence has the form $u\sigma_2$ and the communication state is $\delta^G(u\sigma_2, q_0^G)$. Since $u = \varepsilon$ and $\sigma_2 = b_2$, the control communication pair for agent 2 is $(2, b_2a_1)$, where the communication sequence is b_2.

We use the same procedure and find another $t = a_1b_2cb_2a_1$, $t' = a_1b_2ca_1b_2$ and $\hat{\sigma} = c$. This leads to another control communication pair for agent 2: $(6, a_1b_2cb_2a_1)$, where the communication sequence is $a_1b_2cb_2$.

4.2.3 How to incorporate communication into G^{com}

We represent the action of communication from one agent 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

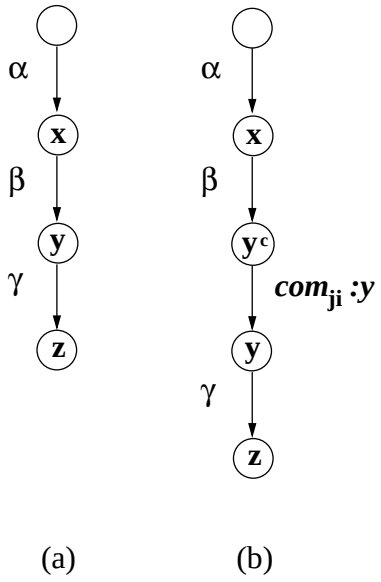

Figure 4.7: Adding a communication event to an automaton: (a) before communication; (b) after communication.

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 agent i communicates to agent 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 agent i communicates to agent 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 $\delta^{G^{com}} = \delta^G$.

Figure 4.7 illustrates how a communication event is added to G^{com}. Let $(y, \alpha\beta\gamma) \in \mathcal{C}_{ji}$ be a control communication pair for the sequence in figure 4.7(a). That is, state y is a place where agent j communicates to agent i. For the pair $(y, \alpha\beta\gamma)$, we create a new state $y^{\mathbf{c}}$ and a communication event $com_{ji}{:}y$. The communication event is a transition from state $y^{\mathbf{c}}$ to y (shown in figure 4.7(b)).

To accommodate the new communication event, the transition function for G^{com} must be updated and extended. Update the transition function $\delta^{G^{com}}$ by removing $\delta^{G^{com}}(\beta, x) = y$. Add the following transitions:

$$\begin{aligned} \delta^{G^{com}}(\beta, x) &= y^{\mathbf{c}}, \\ \delta^{G^{com}}(com_{ji}{:}y, y^{\mathbf{c}}) &= y. \end{aligned} \tag{4.7}$$

In addition, we update the communication alphabet $\Sigma^{com} = \Sigma^{com} \cup \{com_{ji}{:}y\}$ and update the state set $Q^{com} = Q^{com} \cup \{y^{\mathbf{c}}\}$.

OBSERVATION **4.1** *Suppose 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 can now describe what sequences agents would see after communication events are added to the plant in figure 4.6(a). For the first t and t', without communication agent 1 sees a_1. With the addition of the communication event at state 2, either agent 1 sees $com_{21}{:}2a_1$ and knows the plant is along a path to an illegal sequence, or it sees a_1 (with no communication event) and knows that plant is along a path to a legal sequence.

For the second t and t', without communication agent 1 sees a_1a_1. With the addition of the communication event at state 6, either agent 1 sees $a_1com_{21}{:}6a_1$ and it knows that plant is along a path to an illegal sequence, or it sees a_1a_1 and knows that the plant is along a path to a legal sequence.

4.2.4 Formally adding control communication pairs to G^{com}

We present the first of three main procedures that transform G into G^{com}. Procedure 4.1 describes how to incorporate the control communication pairs into G^{com}. The second and third procedures, presented in section 4.3, render the communication protocol derived from G^{com} well-defined.

Procedure 4.1 : 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 4.1, identify control communication pairs (q, t) and their corresponding control twins t' for agent 1 and for agent 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 4.1

Procedure 4.1 identifies the control communication pairs (q, t) that indicate where an agent discloses its local state to another agent. This information must now be translated into places where we add communication events to the augmented plant G^{com}. The following procedure elucidates a strategy for 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}.

Procedure 4.1a : Steps to Building G^{com} from G

For each $(q, t) \in \mathcal{C}_{ij}$, for $i, j \in \{1, 2\}$ and $i \neq j$:

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

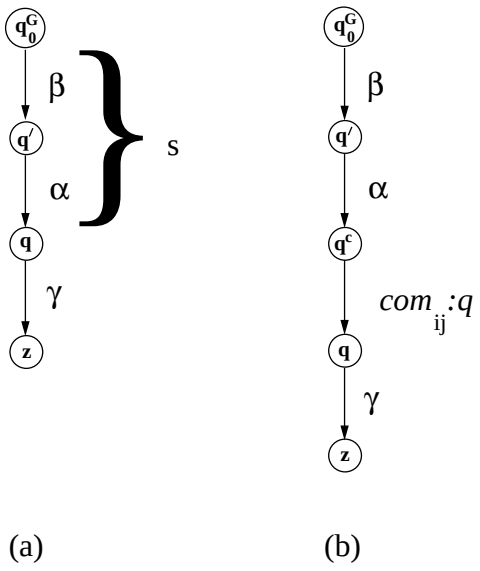

Figure 4.8: Adding a communication event to G^{com} to state q. (a) Before communication for communication sequence $s = \beta\alpha$ for agent i. (b) After communication added to state q.

- Create a new event called $com_{ij}{:}q$ which represents the action of agent i communicating local view q_i to agent j. If $com_{ij}{:}q \notin \Sigma^{com}$, update the alphabet: $\Sigma^{com} = \Sigma^{com} \cup \{com_{ij}{:}q\}$.

- 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 $\delta^{G^{com}}$ (see figure 4.8 for an example):

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

  It could be the case that a communication event representing communication from agent i to agent j has already been added to state q in G^{com}. That is, more

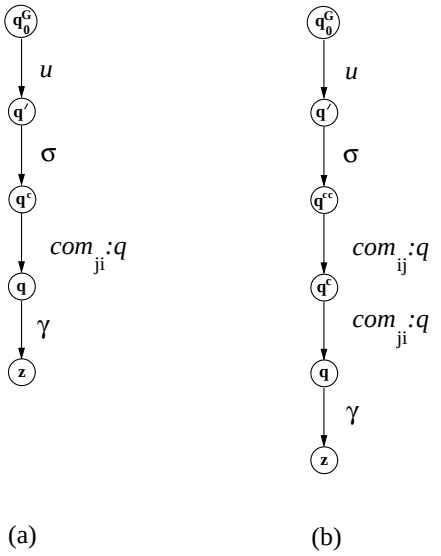

Figure 4.9: Adding an additional communication event to G^{com} at state $q^{\mathbf{c}}$. (a) Another communication event has previously been added at state q. (b) After adding a second communication event.

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 agent j to agent i has already been added to state q in G^{com}. This scenario is shown in figure 4.9(a). If a communication event from agent j to agent 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 agent i communicates to agent j occurs only after an event that agent i sees. Then we remove the following transition from G^{com}:

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

Add the following transitions to $\delta^{G^{com}}$ (see figure 4.9(b) for a graphical representation):

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

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

□ **Procedure 4.1a**

We interpret the appearance of two consecutive communication events in G^{com} as a two-way broadcast between agents i and j. That is, each agent 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 $\mathcal{C}_{ji}$. We elaborate on the effect this has on the construction of a well-defined communication protocol in section 4.4.

The time complexity of Procedures 4.1 and 4.1a is dominated by step 3 of Procedure 4.1: 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. As noted previously, a dynamic-programming algorithm to find the paths of these sequences takes $O(n^3)$ time, where n is the number of states in the monitoring automaton.

4.2.5 Communication that solves the control problem

We must formally show that when agent i finds a control sequence indistinguishable from its corresponding control twin, the addition of a communication event along the communication sequence allows agent 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 in (3.3) as follows. Let $\hat{P}$ be a mapping from $(\Sigma \cup \Sigma^{com})^*$ to Σ^* and

therefore $(\Sigma^{com})^* \to \varepsilon$. Similarly, $\hat{P}_i$ becomes a mapping 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 $\Sigma_o \setminus \Sigma_{i,o}$ but also those events in Σ^{com} from a string 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 4.1.

DEFINITION **4.11** *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 4.1 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 4.1.

We define a mapping $P_i^{\mathbf{c}}$ (for $i = 1, 2$) to be a canonical projection from $(\Sigma \cup \Sigma^{com})^*$ to $(\Sigma_{i,o} \cup \Sigma^{com})^*$. We want to use this mapping to show that if we add a communication event along a control sequence t and not along its control twin t' (according to Theorem 4.1) the two sequences will no longer look alike to the agent making the control decision at t or t'.

LEMMA **4.2** *For a control sequence t and its control twin t' defined with respect to agent i (i.e., $P_i(t) = P_i(t')$), after following Procedure 4.1, $P_i^{\mathbf{c}}(t^{\mathbf{c}}) \neq P_i^{\mathbf{c}}(t'^{\mathbf{c}})$.*

Proof. Since $P_i(t) = P_i(t')$ we know that agent j will be communicating at least once to agent i. Let com_{ji}:q be such a communication event added to t. Since the

plant has been rewritten such that t and t' do not share communication states, the state q does not appear along t'. Therefore, after Procedure 4.1 is complete, $com_{ji}{:}q$ will not be added along t'. Therefore $P_i^{\mathbf{c}}(t^{\mathbf{c}}) \neq P_i^{\mathbf{c}}(t'^{\mathbf{c}})$.

□ LEMMA 4.2

It remains to be shown that after adding the remaining communication events to the rest of the plant (i.e., for consistency), the communication-equivalent sequence for control sequence t remains distinguishable from the updated communication-equivalent sequence for the control twin t'.

4.3 Communication for Consistency

Our communication goal is two-fold: (i) to have agents 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 agent 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 to the plant for each control sequence t that appears in $(q, t) \in \mathcal{C}_{ij}$. 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 [31]. We will return to this observation shortly.

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

DEFINITION **4.12** *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 agent i's local views of states y and $q^{\mathbf{c}}$, respectively.*

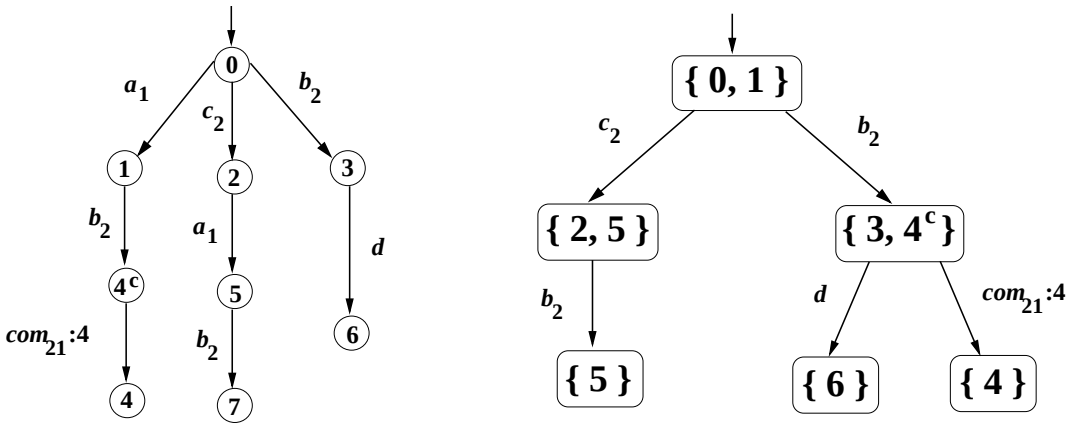

Figure 4.10: A G^{com} that does not satisfy consistency.

That is, whenever we identify a communication state q from a control communication pair (q, t) for an agent, 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 agent i's local view of y is equal to agent i's local view of $q^{\mathbf{c}}$.

Figure 4.10 illustrates a scenario we must preclude. Suppose that agent 1 sees and controls events a_1 and d while agent 2 sees and controls events b_2, c_2 and d. The left hand side of figure 4.10 contains a G^{com}. The right hand side of figure 4.10 contains the projection automaton of G^{com} with respect to agent 2. This particular G^{com} is not consistent. Note that the local view of communication state $4^{\mathbf{c}}$ for agent 2 is $\{3, 4^{\mathbf{c}}\}$. Similarly, the local view of state 3 for agent 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 (the projection automaton on the right hand side of figure 4.10) is because the projection automaton will form the basis of an agent's communication protocol. The idea is that if a communication event occurs at a particular state, an agent must communicate. If more than one event is defined at that state, an agent would not have a clear directive

as to when communication should happen. For example, when agent 2 is at state $\{3, 4^{\mathbf{c}}\}$ it is not straightforward when communication should occur. We clarify this notion, which we refer to as a well-defined communication protocol, in section 4.4.

One option for adding additional communication events to G^{com} would be to identify all the states in G that are indistinguishable from state $\delta^G(s, q_0^G)$ (for all s corresponding to the control communication pairs in $\mathcal{C}_{ij} \cup \mathcal{C}_{ji}$). Then add communication events to the corresponding states in G^{com}. This is the correct strategy if none of the communication sequences contain communication events. However, the following scenario could unfold: suppose that agent j must communicate for control to agent i at state x and suppose that its local view of x is $x_j = \{x, y, z\}$. Thus, in G, agent j is unable to distinguish plant states x, y and z. Further suppose that because of some prior communication from agent i, agent 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 agent with respect to the original plant G by taking into account the effects of prior communication along a communication sequence. We identify the relationships between the control communication pairs by building a dependency graph for the elements of $\mathcal{C}_{ij} \cup \mathcal{C}_{ji}$. A dependency graph of an object graphically illustrates all of its relations to other objects. 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 agent or that is another communication sequence. We use the dependency graph to identify which communication events should be added to the communication sequences before any new events are added to G^{com}. We describe this approach in section 4.3.1. Our strategy concludes by considering the rest of the sequences in the plant (i.e., all the sequences that are not communication sequences). For all sequences $v \in L(G)$ (such that v

is not a communication sequence) that are indistinguishable from a communication sequence s (according to communicating agent i), we will add a communication event to the G^{com} at state $\delta^G(v, q_0^G)$ only if v has identical dependencies on the control communication pairs to s.

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

DEFINITION **4.13** *A pair* $(\boldsymbol{x}, \boldsymbol{v})$ *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* $(\boldsymbol{q}, \boldsymbol{t}) \in \mathcal{C}_{ij}$, *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 agent i from communication sequence s. Note that by not permitting $v = s$, we eliminate (q, s) from being compatible with (q, t).

Let $\mathcal{X}(q, t) = \{(x, v) \mid (x, v)$ is compatible with $(q, t) \in \mathcal{C}_{ij}\}$. We want to be able to identify places in the plant where we add communication events to produce well-defined communication protocols: sequences that are indistinguishable to the agent sending a communication for control after it observes s. Moreover, we want to narrow down our set of such places agents communicate and omit any pairs (x, v) such that v ends in a sequence unobservable to the communicating agent. We remove these pairs because we assume that an agent communicates the instant it observes the communication sequence.

DEFINITION **4.14** *A pair* $(\boldsymbol{x}, \boldsymbol{v}) \in \mathcal{X}(q, t)$ *is called a* **compatible communication pair for** $(\boldsymbol{q}, \boldsymbol{t})$ *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 state our assumption regarding where we place communication events along sequences that are indistinguishable from a communication sequence to a communicating agent.

ASSUMPTION 4.1 *If the system is at a communication state, we assume that communication from one agent 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 agent.*

We want to apply this assumption to any sequences that look like the communication sequence. This means that if two pairs (x, v) and (x', v') are both compatible with a control communication pair $(q, t) \in \mathcal{C}_{ij}$ and $v' = vw$, where w is a sequence that is unobservable to agent i, then we want to communicate after v occurs and not after v' occurs. In fact, as will become apparent, when the appropriate communication event is added to state x in G^{com}, sequence v' (previously indistinguishable from both v and s) will no longer look like either v or s to the communicating agent. Subsequently, (x', v') will no longer be compatible with (q, t).

If (x, v) is a compatible communication pair for $(q, t) \in \mathcal{C}_{ij}$, then (x, v) is added to a set $\mathcal{C}_{ij}^{compat}$ (for $i, j \in \{1, 2\}$ and $i \neq j$). The communication event com_{ij}:q is added to G^{com} at state x according to step 3 in Procedure 4.1 (substituting x for q and v for s).

In the following subsections, we formally describe the two stages involved in adding communication events to G^{com} so that the effects of prior communication can be incorporated into the final communication protocol.

4.3.1 Refining local views of control communication pairs

When we say G^{com} satisfies consistency, we want to make sure that the appropriate communication event for agent i is added at states in G^{com} that agent i cannot

distinguish. We want to make sure that an agent's local view of a state in G^{com} is correct. That is, there are some situations where we may be required to update the local view an agent has of a particular state from G to G^{com}. If there are any prior communications from another agent that occur along the communication sequence, then an agent's local view of state q could change. We want to look at sequences identified by Procedure 4.1 as requiring communication and see if there are earlier communications that occur along those sequences. We first will see if the communication-equivalent sequence $s^{\mathbf{c}}$ could contain any communication events. This is because an earlier communication may have altered an agent's view of a sequence.

There are two situations that give rise to a communication sequence containing more than one communication event. For instance, after Procedure 4.1, if a control sequence $s' \in L(G)$ is a prefix of another control sequence $s \in L(G)$, then $s^{\mathbf{c}} \in L(G^{com})$ will contain the communication event associated with the identification of s'. Therefore we do not want to add communication events everywhere a communicating agent i sees $P_i^{\mathbf{c}}(s)$. Rather, we want to add communication events to G^{com} when agent i sees $P_i^{\mathbf{c}}(s^{\mathbf{c}})$.

The other circumstance where a communication sequence could contain more than one communication event is shown in figure 4.11. Suppose that we have two communication sequences s and s' corresponding to communication states q and q', where $(q,t) \in \mathcal{C}_{ij}$ and $(q',t') \in \mathcal{C}_{ji}$. Assume that no communication events were added along s or s' during Procedure 4.1 (see figure 4.11(a)). The communication event associated with (q',t'), namely com_{ji}:q' is added to state q' as described in Procedure 4.1. Similarly, com_{ij}:q is added to state q (see figure 4.11(b)). Further suppose that there exists a prefix v of communication sequence s (i.e., $v \in \overline{s}$) such that $P_j(v) = P_j(s')$. That is, in the original plant, agent j cannot distinguish states q' and x. Therefore, a communication event com_{ji}:q' should be added to the plant after v occurs (see figure 4.11(b)). Similarly, when adding communication events after those sequences

that to agent j look like $s \in L(G)$, we really mean that we append communication events only to sequences that agent j cannot distinguish from the updated version of s, namely $s^{\mathbf{c}}$.

This example illustrates some of the subtle issues involved in incorporating communication into the analysis of decentralized control problems. The situation in figure 4.11 would be more complicated if $s'^{\mathbf{c}}$ contained an additional communication event. Would agent j still find v indistinguishable from the updated s'? If not, then we would have to update $\delta^{G^{com}}$ by removing transition $\delta^{G^{com}}(com_{ji}{:}q', x^{\mathbf{c}})$ and removing state $x^{\mathbf{c}}$ from Q^{com}.

We can resolve some of these issues if we establish the relationships that the control communication pairs have with respect to each other. Does the communication sequence for a control communication pair (q,t) contain a prefix that is indistinguishable from the communication sequence for the pair (q',t')? How do we communicate for control at "every state that looks like q'" and "every state that looks like q" if, at the same time, the communication sequence for (q',t') also contains a prefix that is indistinguishable from the communication sequence for (q,t)?

We introduce some terminology to identify when a communication event is preceded by another communication event:

DEFINITION **4.15** *We say that a control communication pair* $\boldsymbol{(q,t)}$ **depends on** *control communication pair* $\boldsymbol{(q',t')}$ *if 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 control communication sequence for* (q,t).

That is, a communication sequence for (q,t) potentially contains another communication event, namely the event associated with the control communication pair (q',t'). In Procedure 4.2 we restrict our attention to whether a control communication pair depends on any other control communication pair.

To detect some of the potential "dependencies" between control communication

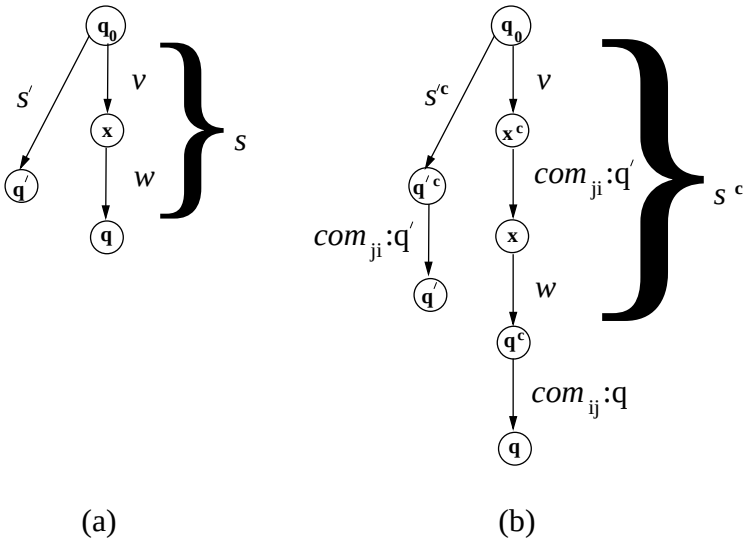

Figure 4.11: Communication sequences can contain communication events: (a) Suppose that $s, s' \in L(G)$ are communication sequences such that $s = vw$ and $P_j(v) = P_j(s')$ where agent i communicates after s, agent j communicates after s'; (b) The updated version of the communication sequences $s'^{\mathbf{c}}, s^{\mathbf{c}} \in L(G^{com})$.

pairs, we build a *dependency graph* D. 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 graph consists of all the 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 control communication sequence for (q,t)), $v \in \overline{s}$ and agent 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 $v, v' \in \overline{t}$, such that $\delta^G(v, q_0^G) = x$ and $\delta^G(v', q_0^G) = x'$, then $x = x'$ and $v = v'$ (since neither v nor v' can end in events that are unobservable to agent i).

For the remainder of this discussion, we represent D as a 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. 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 4.14.

The dependency graph could contain cycles. We say an *undesirable* cycle occurs in D when there are control communication pairs $(q,t) \in \mathcal{C}_{12}$, $(q',t') \in \mathcal{C}_{21}$ and $(\hat{q},\hat{t}),(q'',t'') \in \mathcal{C}_{12} \cup \mathcal{C}_{21}$ such that $D[(q,t),(q',t')] \neq \emptyset$ and $D[(q',t'),(q,t)] \neq \emptyset$ (a cycle involving just two states) or a longer cycle such as

- $D[(q,t),(\hat{q},\hat{t})] \neq \emptyset$,
- $D[(\hat{q},\hat{t}),(q',t')] \neq \emptyset$,
- $D[(q',t'),(q'',t'')] \neq \emptyset$ and
- $D[(q'',t''),(q,t)] \neq \emptyset$.

The form of the cycle is important: there must be an alternation of a pair in $\mathcal{C}_{12}$ and a pair in $\mathcal{C}_{21}$ or a pair in $\mathcal{C}_{21}$ and a pair in $\mathcal{C}_{12}$ to constitute an undesirable cycle. We use the dependency graph to determine if any other communication events could occur along a communication sequence (as identified at the completion of Procedure 4.1). Figure 4.12 shows a scenario that would give rise to a cycle of length 2 in the dependency graph. This represents a situation where somewhere prior to reaching state q along sequence s there is another sequence v (i.e., $v \in \overline{s}$) that agent 1 cannot distinguish from s' (i.e., $P_1(v) = P_1(s')$). Therefore communication event com_{21}:q' could be added at state x prior to com_{12}:q. At the same time, there is another sequence $v' \in \overline{s'}$ such that agent 2 cannot distinguish v from s (i.e., $P_2(v') = P_2(s)$). Similarly, com_{12}:q could be added at state x'.

We break cycles in D as follows: resolve the mutual dependency by choosing a pair $(q,t) \in \mathcal{C}_{ij}$ in the cycle that depends on the fewest number of control communication pairs in the other set of control communication pairs (i.e., the fewest non-empty

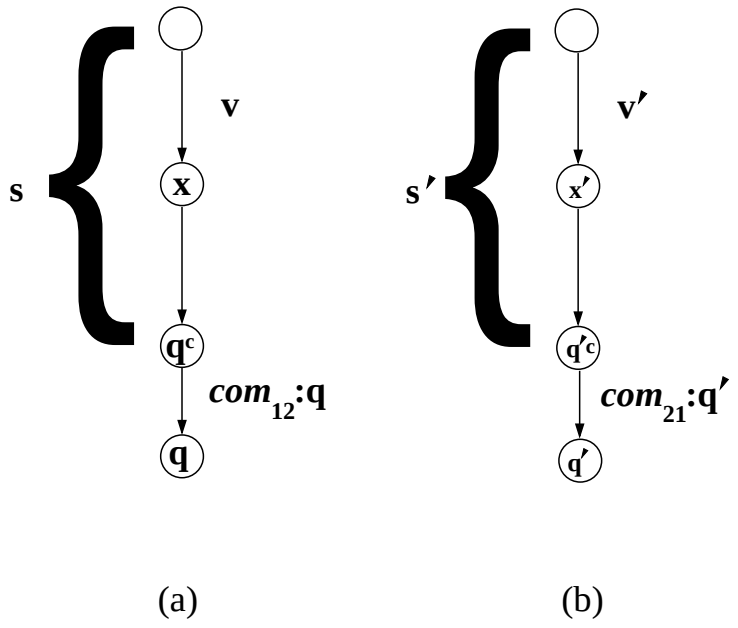

Figure 4.12: A scenario that results in a cycle in D. Let $P_1(v') = P_1(s)$ and $P_2(v) = P_2(s')$: (a) event com_{21}:q' could occur before event com_{12}:q; (b) event com_{12}:q could occur before event com_{21}:q'.

column entries for pairs in $\mathcal{C}_{ji}$). Note that we can break the cycle by randomly choosing any of the control communication pairs involved in the cycle. We choose here to break a cycle by fixing a communication event for an agent at a control communication pair that has the fewest number of potential communications from another agent preceding its own occurrence. This corresponds to picking the pair that has the fewest non-empty entries in its row of the dependency graph. By selecting (q,t) to be the place where the cycle is broken, we indicate that communication with respect to the communication event for (q',t') will not occur along s and "fix" communication by setting $D[(q,t),(q',t')] = \emptyset$. We consider that the non-empty entry at $D[(q',t'),(q,t)]$ is an entry that cannot be changed. It is now the case that (q,t) no longer depends on (q',t') but (q',t') still depends on (q,t). An example of how to resolve cycles in D is presented after Procedure 4.2.

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.

Procedure 4.2

1. Let $\mathcal{XV}$ be the set of all compatible communication pairs (x,v) for all control communication pairs in $\mathcal{C}_{ij} \cup \mathcal{C}_{ji}$. We only want to consider a particular subset of $\mathcal{XV}$ at this point, namely the subset that contains the (x,v)'s such that v occurs along a communication sequence s. Let $\mathcal{XV}_{4.2} = \{(x,v) \mid (x,v)$ satisfies Definition 4.15 and gives rise to a dependency of (q,t) on (q',t'), for $(q,t),(q',t') \in \mathcal{C}_{ij} \cup \mathcal{C}_{ji}\}$. Initialize all entries of D to $\emptyset$.

2. We indicate possible dependencies as follows. For $(q,t),(q',t') \in \mathcal{C}_{12} \cup \mathcal{C}_{21}$:

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

if (q,t) depends on (q',t') and (x,v) is a compatible communication pair for (q',t') of the form in Definition 4.15.

3. Detect and resolve cycles in D. A cycle of length two, for instance, occurs when $\exists(q,t) \in \mathcal{C}_{ij}$ and $\exists(q',t') \in \mathcal{C}_{ji}$ such that $D[(q,t),(q',t')] \neq \emptyset$ and $D[(q',t'),(q,t)] \neq \emptyset$. Algorithms to detect cycles of length greater than 2 exist [5]. We will choose to break the cycle with the control communication pair that is part of the cycle such that if the pair is in $\mathcal{C}_{ij}$ (respectively, $\mathcal{C}_{ji}$) it depends on the fewest number of other control communication pairs in $\mathcal{C}_{ji}$ (respectively, $\mathcal{C}_{ij}$). For instance, if we choose $(q,t) \in \mathcal{C}_{ij}$ we tally the number of non-empty entries along row (q,t) in the columns corresponding to all $(q',t') \in \mathcal{C}_{ji}$. Do this for all pairs involved in the cycle and pick the pair with the fewest number of dependencies. Set $D[(q,t),(q',t')] = \emptyset$ and consider that the communication represented by $D[(q',t'),(q,t)] \neq \emptyset$ is communication that must occur. If more than one cycle is detected, then after each cycle is resolved check the updated D to see if the previously-detected cycles still exist.

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

5. If a row of D contains all $\emptyset$ entries, mark the corresponding control communication pair "compatible".

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

   - Choose an "incompatible" control communication pair that depends on only "compatible" pairs. That is, pick a row of D corresponding to an "incompatible" control communication pair where all the non-empty column entries correspond to pairs already marked "compatible". Let $(q,t) \in \mathcal{C}_{12}$ be the chosen row of D. Find the control communication pair $(\tilde{q},\tilde{t})$ associated with the communication event that occurs just prior to the occurrence of com_{12}:q along t (i.e., $D[(q,t),(\tilde{q},\tilde{t})] \neq \emptyset$). Using D, we want to find out if it is possible that s would contain the same communication events (in

the same order) as $\tilde{s}$. Because $(\tilde{q}, \tilde{t})$ is marked "compatible", the number of communication events that occur along its communication sequence $\tilde{s}$ has already been determined. To see if (q, t) still depends on $(\tilde{q}, \tilde{t})$, compare the entries in the corresponding rows of D. Because we are not interested in the dependency that a control communication pair has with itself (by definition $D[(q,t),(q,t)] = \emptyset$), we block out the column entries for (q, t). In addition, we block out the column entries for $(\tilde{q}, \tilde{t})$ because we are trying to ascertain whether or not this dependency is still valid. That is, if all the other entries in the two rows corresponding to (q, t) and $(\tilde{q}, \tilde{t})$ coincide, we assume that (q, t) still depends on $(\tilde{q}, \tilde{t})$ and therefore $D[(q,t),(\tilde{q},\tilde{t})] \neq \emptyset$.

(a) If the remaining entries in the two rows do have the same pattern of empty and non-empty entries, then we are done examining this communication control pair. We have found a valid dependency.

(b) If the remaining entries in the two rows do not have the same pattern of empty and non-empty entries and if row $(\tilde{q}, \tilde{t})$ contains non-empty entries that row (q, t) does not, then set $D[(q,t),(\tilde{q},\tilde{t})] = \emptyset$. This means that control sequence $\tilde{t}$ contains communication events that t does not—and will not since we do not add entries to D at this point.

(c) If the remaining entries in the two rows do not have the same pattern of empty and non-empty entries and if row (q, t) contains non-empty entries that row $(\tilde{q}, \tilde{t})$ does not, then check the other dependencies for (q, t) before deciding that $D[(q,t),(\tilde{q},\tilde{t})] = \emptyset$. this means that control sequence $\tilde{t}$ contains fewer communication events than t, but until we check the rest of the dependencies for (q, t), we do not know whether or not t still depends on $\tilde{t}$.

(d) Repeat from (a) until all dependencies for (q, t) have been checked

or until a valid dependency is found. Note that if (q,t) has an additional dependency, for example $(\hat{q},\hat{t})$, then rows (q,t) and $(\hat{q},\hat{t})$ of D are compared as described above except that the row entries for previously-checked dependencies, such as $(\tilde{q},\tilde{t})$, of (q,t) are ignored. We do this because we are checking the potential placement of communication events along t in the reverse order of appearance. That is, since the row for (q,t) in D indicates that the communication event associated with $(\tilde{q},\tilde{t})$ would occur after the communication event associated with $(\hat{q},\hat{t})$, we can disregard the entry for $(\tilde{q},\tilde{t})$ when comparing rows (q,t) and $(\hat{q},\hat{t})$. Mark (q,t) "compatible".

7. Initialize $\mathcal{C}_{12}^{compat} = \mathcal{C}_{21}^{compat} = \emptyset$. These are sets that store the compatible communication pairs (x,v) as identified by the non-empty entries in the final version of D. For $(q,t) \in \mathcal{C}_{12} \cup \mathcal{C}_{21}$ and $(q',t') \in \mathcal{C}_{ij}$, where s' is the communication sequence for (q',t'), if $D[(q,t),(q',t')] = (x,v)$ and if $P_i^{\mathbf{c}}(v^{\mathbf{c}}) = P_i^{\mathbf{c}}(s'^{\mathbf{c}})$ then:

$$\mathcal{C}_{ij}^{compat} = \mathcal{C}_{ij}^{compat} \cup \{(x,v)\}.$$

□ **Procedure 4.2**

Procedure 4.2 identifies compatible communication pairs (x,v), for each control communication pair (q,t) found in Procedure 4.1, that occur *along* control communication sequences. The purpose of this procedure is to refine—if necessary—an agent's local view of communication states in light of any communication it receives from another agent 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 4.2, we have the sets of compatible

communication pairs that give rise to dependencies between control communication pairs. We must next translate the entries in $\mathcal{C}_{12}^{compat} \cup \mathcal{C}_{21}^{compat}$ into communication events that are added to G^{com}.

Procedure 4.2a : Steps to Building G^{com} from G, Part 2

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

   - Create a new state $x^{\mathbf{c}}$. If $x^{\mathbf{c}} \notin Q^{com}$, update the state set: $Q^{com} = Q^{com} \cup \{x^{\mathbf{c}}\}$.
   - Update the transition function $\delta^{G^{com}}$. Suppose that v has the form $v = v'\sigma$ where $\delta^G(v', q_0^G) = x'$ 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 $\delta^{G^{com}}$

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

   As in Procedure 4.1a, if there is already a transition of $com_{ij}{:}q$ at state $x^{\mathbf{c}}$, we do not add the same event more than once as a transition out of $x^{\mathbf{c}}$. If the communication event $com_{ji}{:}q$ has already been added to G^{com} at state $x^{\mathbf{c}}$, then create a new state $x^{\mathbf{cc}}$ and update Q^{com}:

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

   Update $\delta^{G^{com}}$ as follows. Remove

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

   and add the transitions

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

The two consecutive communication events are added to state x if $(x, v) \in \mathcal{C}_{ij}^{compat} \cap \mathcal{C}_{ji}^{compat}$, for $i, j \in \{1, 2\}$ and $i \neq j$.

□ Procedure 4.2a

Note that 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 also com_{ij}:q.

We make a similar assumption regarding the structure of G as described in section 4.2.1. We want to add a communication event com_{ij}:q at state x in G^{com} corresponding to an (x, v) identified in Procedure 4.2. 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 rewritten to accommodate all occurrences of the above scenario. The same comments about time complexity for this procedure made in section 4.2.1 apply here.

The time complexity for Procedure 4.2 is dominated, as was Procedure 4.1, 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 final step, as given further on in Procedures 4.3 and 4.3a, is to find the remaining compatible communication pairs for the updated version of the control communication pairs.

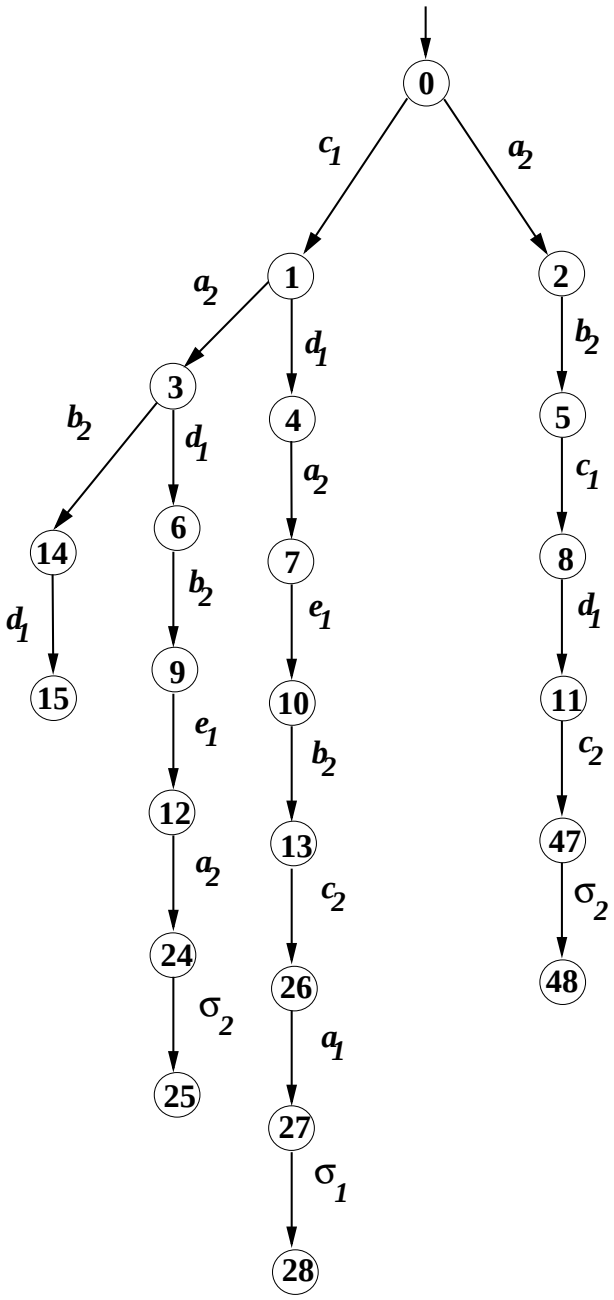

Figure 4.13: A portion of a plant that would give rise to cycles in a dependency graph.

We first consider an example of resolving cycles in the dependency graph. Figure 4.13 contains part of a plant G, where the control communication pairs have been *a priori* established as: $\mathcal{C}_{12} = \{(11, a_2b_2c_1d_1c_2), (12, c_1a_2d_1b_2e_1a_2)\}$ and $\mathcal{C}_{21} = \{(13, c_1d_1a_2e_1b_2c_2a_1)\}$. For this example, D is a 3×3 matrix (i.e., $n_1 = 2$ and $n_2 = 1$). The rows and columns of D correspond to the following control communication pairs:

- row (and column) 1: $(11, a_2b_2c_1d_1c_2)$;
- row (and column) 2: $(12, c_1a_2d_1b_2e_1a_2)$; and
- row (and column) 3: $(13, c_1d_1a_2e_1b_2c_2a_1)$.

From the plant in figure 4.13 we identify the compatible communication pairs for the control communication pairs noted above:

- $(4, c_1d_1)$ and $(6, c_1a_2d_1)$ are compatible with $(11, a_2b_2c_1d_1c_2)$, where its communication sequence is $s = a_2b_2c_1d_1$, since $P_1(c_1d_1) = P_1(c_1a_2d_1) = P_1(s)$;
- $(10, c_1d_1a_2e_1)$ is compatible with $(12, c_1a_2d_1b_2e_1a_2)$, where $s = c_1a_2d_1b_2e_1$, since $P_1(c_1d_1a_2e_1) = P_1(s)$; and
- $(5, a_2b_2)$ and $(9, c_1a_2d_1b_2)$ are compatible with $(13, c_1d_1a_2e_1b_2c_2a_1)$, where $s = c_1d_1a_2e_1b_2$, since $P_2(a_2b_2) = P_2(c_1a_2d_1b_2) = P_2(s)$.

The following dependencies between control communication pairs exist:

- $(11, a_2b_2c_1d_1c_2)$ depends on $(13, c_1d_1a_2e_1b_2c_2a_1)$ because there is a compatible communication pair—$(5, a_2b_2)$—for $(13, c_1d_1a_2e_1b_2c_2a_1)$ that is a prefix of the communication sequence for $(11, a_2b_2c_1d_1c_2)$;
- $(12, c_1a_2d_1b_2e_1a_2)$ depends on $(11, a_2b_2c_1d_1c_2)$ and $(13, c_1d_1a_2e_1b_2c_2a_1)$ because of the compatible communication pairs $(6, c_1a_2d_1)$ and $(9, c_1a_2d_1b_2)$, respectively; and

- $(13, c_1d_1a_2e_1b_2c_2a_1)$ depends on $(11, a_2b_2c_1d_1c_2)$ and $(12, c_1a_2d_1b_2e_1a_2)$ because of the compatible communication pairs $(4, c_1d_1)$ and $(10, c_1d_1a_2e_1)$, respectively.

The dependency graph for $\mathcal{C}_{12} \cup \mathcal{C}_{21}$ contains several cycles:

$$D = \begin{bmatrix} \emptyset & \emptyset & (5, a_2b_2) \\ (6, c_1a_2d_1) & \emptyset & (9, c_1a_2d_1b_2) \\ (4, c_1d_1) & (10, c_1d_1a_2e_1) & \emptyset \end{bmatrix}$$

In particular, there is a cycle involving communication states 12 and 13:

$$D[(12, c_1a_2d_1b_2e_1a_2), (13, c_1d_1a_2e_1b_2c_2a_1)] = (9, c_1a_2d_1b_2)$$

and

$$D[(13, c_1d_1a_2e_1b_2c_2a_1), (12, c_1a_2d_1b_2e_1a_2)] = (10, c_1d_1a_2e_1).$$

That is, the communication event com_{12}:12 that occurs for communication pair $(12, c_1a_2d_1b_2e_1a_2)$ could be preceded by com_{21}:13 at state 9 because agent 2 cannot distinguish state 9 from 13. At the same time, agent 1 cannot distinguish state 10 from state 12, and com_{12}:12 could occur at state 10—just before com_{21}:13 happens in accordance with the control communication pair $(13, c_1d_1a_2e_1b_2c_2a_1)$.

If both compatible communication pairs $(9, c_1a_2d_1b_2)$ and $(10, c_1d_1a_2e_1)$ are added to the set of compatible communication pairs $\mathcal{C}_{21}^{compat}$ and $\mathcal{C}_{12}^{compat}$, one of the communication pairs constitutes unnecessary communication. Figure 4.14 shows what $(12, c_1a_2d_1b_2e_1a_2)$ and $(13, c_1d_1a_2e_1b_2c_2a_1)$ would look like if the two compatible communication pairs mentioned above were added to G^{com} at states 9 and 10. The communication sequences become $c_1a_2d_1b_2com_{21}$:$13c_1$ and $c_1d_1a_2e_1com_{12}$:$12b_2$. If we then go back and check the compatible communication pairs, it is the case that neither $(9, c_1a_2d_1b_2)$ nor $(10, c_1d_1a_2e_1)$ are compatible with the new version of their respective control communication pairs. For instance, $(9, c_1a_2d_1b_2)$ is no longer compatible with $(13, c_1d_1a_2e_1b_2c_2a_1)$ because after adding the communication event com_{12}:12

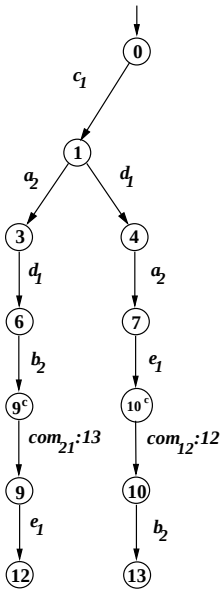

Figure 4.14: Adding unnecessary communication.

to state 10, the communication sequence for this control communication pair becomes $c_1d_1a_2e_1com_{12}{:}12b_2c_2a_1$ and since the communication event $com_{12}{:}12$ is not added to $c_1a_2d_1b_2$, it is the case that $P_2(c_1a_2d_1b_2) \neq P_2(c_1d_1a_2e_1com_{12}{:}12b_2)$. If the no-longer-compatible communication pair $(9, c_1a_2d_1b_2)$ is removed from $\mathcal{C}_{21}^{compat}$, the control communication pair at state 12 is once again $(12, c_1a_2d_1b_2e_1a_2)$. At this point, $(10, c_1d_1a_2e_1)$ is also once again a compatible communication state for $(12, c_1a_2d_1b_2e_1a_2)$. Note that if $(9, c_1a_2d_1b_2)$ is not removed from $\mathcal{C}_{21}$ and is added to G^{com}, all that happens is that agent 2 sends its local state to agent 1 even though agent 1 does not need this additional information (i.e., to satisfy consistency or to solve the control problem). An analogous situation occurs if we instead remove $(10, c_1d_1a_2e_1)$ from $\mathcal{C}_{12}^{compat}$ and update the communication pairs accordingly.

To break the cycle involving states 12 and 13, we will choose one of these states to communicate at and update the dependency graph. We choose state 12 because, according to the row for $(12, c_1a_2d_1b_2e_1a_2)$ in D, it depends on only one pair in $\mathcal{C}_{21}$ (one non-empty entry in the last n_2 entries of the row) while the corresponding calculation

for state 13 reveals that state 13 depends on two pairs in $\mathcal{C}_{12}$. We update D as follows:

$$D[(12, c_1a_2d_1b_2e_1a_2), (13, c_1d_1a_2e_1b_2c_2a_1)] = \emptyset,$$

and therefore communication at state 13 must depend on state 12 and we do not allow the corresponding entry to change from

$$D[(13, c_1d_1a_2e_1b_2c_2a_1), (12, c_1a_2d_1b_2e_1a_2)] = (10, c_1d_1a_2e_1).$$

The updated dependency graph is

$$D = \begin{bmatrix} \emptyset & \emptyset & (5, a_2b_2) \\ (6, c_1a_2d_1) & \emptyset & \emptyset \\ (4, c_1d_1) & (10, c_1d_1a_2e_1) & \emptyset \end{bmatrix}$$

We cannot mark control communication pair $(12, c_1a_2d_1b_2e_1a_2)$ "compatible" because this pair still depends on an "incompatible" pair $(11, a_2b_2c_1d_1c_2)$.

Of the remaining incompatible control communication pairs, a cycle exists between communication states 11 and 13. By breaking the cycle between states 12 and 13, state 12 now does not depend on state 13 but state 13 does depend on state 12. That is, there will be a communication event com_{12}:12 along the path to state 13. We need to see if this change to the dependency graph in any way affects the cycle we initially detected between states 11 and 13 (since the path to state 13 will contain a communication event not originally considered when we first listed the dependencies that formed D). Since state 13 depends on state 12, for state 11 to depend on state 13 it must be the case that state 11 also depends on state 12 (i.e., the event com_{12}:12 would have to appear at the same place with respect to projection). Since $D[(11, a_2b_2c_1d_1c_2), (12, c_1a_2d_1b_2e_1a_2)] = \emptyset$, state 11 does not depend on state 12 and hence it now does not depend on state 13. We remove the potential dependency of $(11, a_2b_2c_1d_1c_2)$ on $(13, c_1d_1a_2e_1b_2c_2a_1)$:

$$D[(11, a_2b_2c_1d_1c_2), (13, c_1d_1a_2e_1b_2c_2a_1)] = \emptyset.$$

The dependency graph becomes

$$D = \begin{bmatrix} \emptyset & \emptyset & \emptyset \\ (6, c_1 a_2 d_1) & \emptyset & \emptyset \\ (4, c_1 d_1) & (10, c_1 d_1 a_2 e_1) & \emptyset \end{bmatrix}$$

The row $(11, a_2 b_2 c_1 d_1 c_2)$ now contains all $\emptyset$ entries and we mark this pair "compatible".

We now go back and revisit the row for pair $(12, c_1 a_2 d_1 b_2 e_1 a_2)$ the only non-empty column entry is for the "compatible" pair $(11, a_2 b_2 c_1 d_1 c_2)$:

$$D[(12, c_1 a_2 d_1 b_2 e_1 a_2), (11, a_2 b_2 c_1 d_1 c_2)] = (6, c_1 a_2 d_1).$$

To see if this dependency is still valid, we block out the column representing $(11, a_2 b_2 c_1 d_1 c_2)$ and compare the rows of these two control communication pairs (i.e., the first and second rows minus the first column). Both rows contain $\emptyset$ in the second column, and $\emptyset$ in the third column. Therefore, com_{12}:11 does occur in conjunction with communication pair $(6, c_1 a_2 d_1)$. We mark $(12, c_1 a_2 d_1 b_2 e_1 a_2)$ "compatible".

We check the only remaining "incompatible" pair $(13, c_1 d_1 a_2 e_1 b_2 c_2 a_1)$ again. The two non-empty entries in the third row represents a dependency on a "compatible" pair $(11, a_2 b_2 c_1 d_1 c_2)$ and on the fixed communication for breaking the cycle with $(12, c_1 a_2 d_1 b_2 e_1 a_2)$. The closest communication event is associated with the "fixed" communication dependency associated with $(12, c_1 a_2 d_1 b_2 e_1 a_2)$ so we check it first. Note that when the column representing the pair $(12, c_1 a_2 d_1 b_2 e_1 a_2)$ is blocked out, the rows corresponding to the control communication pairs involving states 12 and 13 are equivalent: column 1 in both rows is non-empty and column 3 in both rows contains $\emptyset$. Therefore, the dependency is still valid. The pair $(13, c_1 d_1 a_2 e_1 b_2 c_2 a_1)$ is marked "compatible". The final dependency graph for the control communication pairs is

$$D = \begin{bmatrix} \emptyset & \emptyset & \emptyset \\ (6, c_1 a_2 d_1) & \emptyset & \emptyset \\ (4, c_1 d_1) & (10, c_1 d_1 a_2 e_1) & \emptyset \end{bmatrix} \tag{4.8}$$

We examine each non-empty entry (x, v) of D to see if incorporating communication events into v means that the updated v is still indistinguishable from the updated communication sequence associated with the dependency. The first possible compatible communication pair $(x, v) = (6, c_1a_2d_1)$ represents a dependency of $(12, c_1a_2d_1b_2e_1a_2)$ on $(11, a_2b_2c_1d_1c_2)$, via the latter's communication sequence $s = a_2b_2c_1d_1$, because $P_1(v) = P_1(s)$. When these sequences are updated with communication events—adding the appropriate event after each v and each s—we have $v^{\mathbf{c}} = c_1a_2d_1com_{12}$:11 and $s^{\mathbf{c}} = a_2b_2c_1d_1com_{12}$:11. In this case $P_1^{\mathbf{c}}(s^{\mathbf{c}}) = P_1^{\mathbf{c}}(v^{\mathbf{c}}) = c_1d_1com_{12}$:11. We go through a similar procedure for $D[(13, c_1d_1a_2e_1b_2c_2a_1), (11, a_2b_2c_1d_1c_2)] = (4, c_1d_1)$ for the same communication sequence s and find that the projections of these updated sequences are also the same. When $D[(13, c_1d_1a_2e_1b_2c_2a_1), (12, c_1a_2d_1b_2e_1a_2)] = (10, c_1d_1a_2e_1)$ note that communication will also be added to v after c_1d_1 because of the presence of the compatible communication pair $(4, c_1d_1)$. For this pair, $v^{\mathbf{c}} = c_1d_1com_{12}$:11$a_2e_1com_{12}$:12 and $s^{\mathbf{c}} = c_1a_2d_1com_{12}$:11$b_2e_1com_{12}$:12. Since $P_1(v^{\mathbf{c}}) = P_1(s^{\mathbf{c}}) = c_1d_1com_{12}$:11$e_1com_{12}$:12, we also add this (x, v) to our set of compatible communication pairs. Therefore $\mathcal{C}_{12}^{compat} = \{(4, c_1d_1), (6, c_1a_2d_1), (10, c_1d_1a_2e_1)\}$, $\mathcal{C}_{21}^{compat} = \emptyset$.

Note that there is more than one way to break a cycle. We choose to fix communication for the control communication pair that depends on the fewest number of other pairs. 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.

We want to describe a sequence in $L(G)$ as it appears after following Procedure 4.2. At this point $L(G^{com})$ is the language generated by the G^{com} that results from

the completion of Procedure 4.2. We write $\widehat{t^{\mathbf{c}}}$ for the sequence in $L(G^{com})$ such that

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

and

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

We abuse terminology and also refer to $\widehat{t^{\mathbf{c}}}$ as a communication-equivalent sequence for t. In Procedure 4.1 communication events are added to control sequences. In Procedure 4.2 new communication events are added only to control sequences.

In Lemma 4.2 we showed that any communication event added to G^{com} after following Procedure 4.1 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 4.2 is completed.

LEMMA **4.3** *For a control sequence t and its control twin t' defined with respect to agent i (i.e., $P_i(t) = P_i(t')$), after following Procedure 4.2, $P_i^{\mathbf{c}}(\widehat{t^{\mathbf{c}}}) \neq P_i^{\mathbf{c}}(\widehat{t'^{\mathbf{c}}})$.*

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 4.2 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 4.1.

Suppose that we added $com_{ji}{:}q$ along $t'^{\mathbf{c}}$ according to step 7 of Procedure 4.2. Note that an event $com_{ji}{:}q$ could only get added in one place along $t'^{\mathbf{c}}$ according to Procedure 4.2. 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 4.14. By Procedure 4.1, 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 4.6 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 $b \notin \overline{u'}$, $b = u'v''\sigma_j v'''$ for some $v'', v''' \in (\Sigma \setminus \Sigma_{j,o})^*$.

(Since $b \in \overline{t}$ and $P_j(b) = P_j(u\sigma_j)$
$= P_j(u)\sigma_j$
$= P_j(u')\sigma_j$.)

For b to satisfy definition 4.14, 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

$$t' = u'v''\sigma_j b''. \tag{4.9}$$

Previously we assumed that

$$t' = u'\sigma_i v'. \tag{4.10}$$

Equating (4.9) and (4.10) 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 4.14 so that $b = u'v''\sigma_j$.

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

$$t' = u'v''\sigma_j b''. \tag{4.11}$$

Previously we assumed that

$$t' = u'\hat{\sigma}_j v'. \tag{4.12}$$

Equating (4.11) and (4.12) 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 agent 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 4.1 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 $P_i^{\mathbf{c}}(\widehat{u^{\mathbf{c}}}) = P_i^{\mathbf{c}}(\widehat{u'^{\mathbf{c}}}\sigma_i\widehat{v''''^{\mathbf{c}}})$. It should be the case, by Lemma 4.1, that these sequences look the same with the communication events "erased":

$$\begin{aligned}
\hat{P}_i(\widehat{u^{\mathbf{c}}}) &= \hat{P}_i(\widehat{u'^{\mathbf{c}}}\sigma_i\widehat{v''''^{\mathbf{c}}}) \\
P_i(u) &= P_i(u'\sigma_i v'''') \qquad \text{(by definition of } \hat{P}_i) \\
P_i(u) &= P_i(u')P_i(\sigma_i)P_i(v'''') \\
&= P_i(u)P_i(\sigma_i)P_i(v'''') \quad \text{(since } (u, u') \text{ is a maximal-P pair)}
\end{aligned}$$

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

□ Lemma 4.3

4.3.2 Refining local views of compatible communication pairs

If an agent'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 agent'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 4.2) affects the agent'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 **4.16** *A compatible communication pair* $\boldsymbol{(x, v)}$ *for control communication pair* (q, t) **depends on** *control communication pair* $\boldsymbol{(q', t')}$ *if we can find a compatible communication pair* (x', v') *for* (q', t') *such that, for* $w \in \Sigma^*$, $v = v'w$ *and* $\delta^G(w, x') = x$.

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 a dependency graph $\hat{D}$ and refer to it only in its 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 $\mathcal{XV} \setminus \mathcal{XV}_{4.2}$ and n_1 and n_2 are still the number of control communication pairs in $\mathcal{C}_{12}$ and $\mathcal{C}_{21}$, respectively. Let $\mathcal{XV}_{4.3} = \mathcal{XV} \setminus \mathcal{XV}_{4.2}$. A row in $\hat{D}$ corresponds to a compatible communication pair $(x, v) \in \mathcal{XV}_{4.3}$. 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 agent i. The communication events corresponding to the pairs (x, v) that survive this culling process are added to G^{com}.

We want to describe a sequence in $L(G)$ as it appears after following Procedure 4.3. At this point $L(G^{com})$ is the language generated by the G^{com} that results from the completion of Procedure 4.3. We write $\tilde{t^{\mathbf{c}}}$ for the sequence in $L(G^{com})$ such that

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

and

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

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

Procedure 4.3

1. Initialize all entries of $\hat{D}$ to $\emptyset$. Initialize all elements of $\mathcal{XV}_{4.3}$ to be "unresolved".

2. Indicate potential dependencies of elements $(x, v) \in \mathcal{XV}_{4.3}$, where (x, v) is a compatible communication pair for $(q, t) \in \mathcal{C}_{ij}$ (for $i, j \in \{1, 2\}$ but $i \neq j$), on control communication pairs $(q', t') \in \mathcal{C}$ as follows:

   $$\hat{D}[(x, v), (q', t')] = (x', v')$$

   if (x, v) depends on (q', t') and (x', v') is compatible with (q', t') as described in Definition 4.16.

3. If all entries for row (x, v) in $\hat{D}$ are $\emptyset$ and all entries for row (q, t) in D are $\emptyset$, then $\mathcal{C}_{ij}^{compat} = \mathcal{C}_{ij}^{compat} \cup \{(x, v)\}$. Mark (x, v) "resolved". This represents a situation where no prior communication has occurred before x. Thus if the communication sequence s and the sequence v both contain no prior communication events, they still look alike and a communication event associated with s will be added after v.

4. While there remain "unresolved" compatible communication pairs $(x, v) \in \mathcal{XV}_{4.3}$, where (x, v) is a compatible communication pair for $(q, t) \in \mathcal{C}_{ij}$ (for $i, j \in \{1, 2\}, i \neq j$):

   (a) If all the non-empty entries are "resolved" compatible communication pairs, compare row (x, v) to row (q, t) in D.

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

     $$\hat{D}[(x, v), (q, t)] = \emptyset.$$

     Mark (x, v) "resolved". After checking all the non-empty column entries for row (x, v), compare row (x, v) in $\hat{D}$ to row (q, t) in D. This represents a situation where a communication event would have been added along v if v still looked like v'. Since (x', v') is marked "resolved", it has already tested for membership in $\mathcal{C}^{compat}$. Therefore, prior to checking (x, v), it was determined that (x', v') was no longer a compatible communication pair for (q, t). Therefore v no longer looks like v'.

   (b) If the pattern of empty and non-empty entries is the same for row (x, v) of $\hat{D}$ and row (q, t) of D, check to make sure that when the communication events are added to v and s (where s is the communication sequence for

$(q,t))$ (x,v) is still compatible with (q,t). That is, for updated versions of each communication sequence it is still the case that $P_i^{\mathbf{c}}(\widetilde{v^{\mathbf{c}}}) = P_i^{\mathbf{c}}(\widetilde{s^{\mathbf{c}}})$. If this is the case, then $\mathcal{C}_{ij}^{compat} = \mathcal{C}_{ij}^{compat} \cup \{(x,v)\}$.

□ Procedure 4.3

Procedure 4.3 identifies compatible communication pairs (x,v), for each control communication pair (q,t) found in Procedure 4.1, that occur *along* any other sequences in the plant but the communication sequences. As before, once more compatible communication pairs are identified, they must be incorporated into G^{com} via Procedure 4.3a.

Procedure 4.3a : Steps to Building G^{com} from G, Part Three

1. For each $(x,v) \in \mathcal{C}_{ij}^{compat} \cap \mathcal{XV}_{4.3}$, for $i,j \in \{1,2\}$, $i \neq j$ and (x,v) is a compatible communication pair for $(q,t) \in \mathcal{C}_{ij}$:

   - Create a new state $x^{\mathbf{c}}$ and update the state set: $Q^{com} = Q^{com} \cup \{x^{\mathbf{c}}\}$.
   - Update the transition function $\delta^{G^{com}}$. Suppose that v has the form $v = v'\sigma$ where $\delta^G(v', q_0^G) = x'$ 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 $\delta^{G^{com}}$

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

     As was the case in Procedures 4.1a and 4.2a, if $com_{ij}{:}q$ has already been added at $x^{\mathbf{c}}$, it is not added again. If communication from agent j to agent i has been added to state x already (i.e., $\delta^{G^{com}}(\sigma, x') \neq x$), then create a new state $x^{\mathbf{cc}}$ and update Q^{com}:

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

Remove the following transition from $\delta^{G^{com}}$:

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

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

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

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

□ **Procedure 4.3a**

As was the case for Procedure 4.2a, 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} that ensures the communication protocols of the agents are well-defined.

We reiterate our comments from section 4.2.1 and those that appeared just after Procedure 4.2a regarding splitting states to avoid ambiguous communication. We want to add a communication event $com_{ij}{:}q$ at state x in G^{com} corresponding to an (x, v) identified in Procedure 4.3. 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 x^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 $\delta^G(v, q_0^G) = x^2$. As before, we assume that the plant G has been rewritten to accommodate all occurrences of the above scenario. Again, the comments made in section 4.2.1 regarding time complexity hold here as well.

The time complexity of Procedure 4.3, like its predecessors, is also $O(n^3)$. This is because using $\mathcal{XV}_{4.3}$ in step 1 means that we have to calculate $\mathcal{XV} \setminus \mathcal{XV}_{4.2}$. The complexity of finding $\mathcal{XV}_{4.2}$ is $O(n^3)$. The other steps of the procedure involve checking matrices and can be accomplished in $O(n^2)$ time.

To illustrate some of the salient parts of Procedure 4.3, we focus on the rows of $\hat{D}$ that arise from the partial plant shown in figure 4.13. Previously, we constructed D for the three control communication pairs: $(11, a_2b_2c_1d_1c_2), (12, c_1a_2d_1b_2e_1a_2)$ and $(13, c_1d_1a_2e_1b_2c_2a_1)$. These three pairs form the columns of $\hat{D}$ (the same order as D). While there could be more "unresolved" compatible communication pairs for the whole plant, we can identify two pairs from figure 4.13 :

- $(14, c_1a_2b_2)$ is compatible with $(13, c_1d_1a_2e_1b_2c_2a_1)$, where $s = c_1d_1a_2e_1b_2$, since $P_2(c_1a_2b_2) = P_2(s)$; and
- $(15, c_1a_2b_2d_1)$ is compatible with $(11, a_2b_2c_1d_1c_2)$, where $s = a_2b_2c_1d_1$, since $P_1(c_1a_2b_2d_1) = P_1(s)$.

We focus on the contents of these rows of $\hat{D}$. Row $(14, c_1a_2b_2)$ of $\hat{D}$ is initially

$$\hat{D} = \begin{bmatrix} \cdots & \cdots & \cdots \\ \emptyset & \emptyset & \emptyset \\ \cdots & \cdots & \cdots \end{bmatrix}$$

because $(14, c_1a_2b_2)$ does not depend on any control communication pair. That is, there is no proper prefix of $c_1a_2b_2$ that looks—to agent 1—like the sequence leading to state 11 or the sequence leading to state 12 and there is no proper prefix of $c_1a_2b_2$ that looks—to agent 2—like the sequence leading to state 13. Since all the entries in row $(14, c_1a_2b_2)$ are $\emptyset$, we mark this pair "resolved". We now compare it to row $(13, c_1d_1a_2e_1b_2c_2a_1)$ in D (displayed in (4.8) on p. 100) because $(14, c_1a_2b_2)$ is compatible with $(13, c_1d_1a_2e_1b_2c_2a_1)$. We do not add $(14, c_1a_2b_2)$ to $\mathcal{C}_{21}^{compat}$ because row $(13, c_1d_1a_2e_1b_2c_2a_1)$ in D does not contain all $\emptyset$ entries.

The corresponding row for $(15, c_1a_2b_2d_1)$ is

$$\hat{D} = \begin{bmatrix} \cdots & \cdots & \cdots \\ \emptyset & \emptyset & (14, c_1a_2b_2) \\ \cdots & \cdots & \cdots \end{bmatrix}$$

because $(15, c_1a_2b_2d_1)$ depends on control communication pair $(13, c_1d_1a_2e_1b_2c_2a_1)$ in the form of $(14, c_1a_2b_2)$. That is, $(14, c_1a_2b_2)$ is a compatible communication pair for $(13, c_1d_1a_2e_1b_2c_2a_1)$ and $c_1a_2b_2$ is a prefix of $c_1a_2b_2d_1$.

The only dependency for $(15, c_1a_2b_2d_1)$ involves a "resolved" pair. But because $(14, c_1a_2b_2)$ was not added to $\mathcal{C}_{21}^{compat}$, after considering the effect of communication events, it must no longer be compatible with the control communication pair $(13, c_1d_1a_2e_1b_2c_2a_1))$. Thus the communication event com_{21}:13 is not added to state 14. Since $(14, c_1a_2b_2)$ was the compatible communication pair that caused $(15, c_1a_2b_2d_1)$ to depend on $(13, c_1d_1a_2e_1b_2c_2a_1)$, we consider that this dependency no longer exists. As a result we set $\hat{D}[(15, c_1a_2b_2d_1), (13, c_1d_1a_2e_1b_2c_2a_1)] = \emptyset$, leaving row $(15, c_1a_2b_2d_1)$ with all its entries $\emptyset$. It is the case that row $(11, a_2b_2c_1d_1c_2)$ of D is also a row of $\emptyset$ entries. Therefore, we add $(15, c_1a_2b_2d_1)$ to $\mathcal{C}_{12}^{compat}$. Figure 4.15 shows the part of G^{com} (constructed after following Procedures 4.1, 4.2 and 4.3) that corresponds to the part of G in figure 4.13.

Notice that prior to adding communication events the sequence $c_1a_2d_1b_2$ leading to state 9 appears the same to agent 2 as the sequence $c_1d_1a_2e_1b_2$ leading to state 13 (i.e., both sequences appear as a_2b_2). For control purposes agent 2 must communicate at state 13. Therefore, prior to the pruning of Procedure 4.2 it appears that agent 2 might also have to communicate the same event (i.e., com_{21}:13) at state 9. However, we can see from figure 4.15 that after various other communications are included, the sequence leading to state 13^c is c_1d_1 com_{12}:11 a_2e_1 com_{12}:12 b_2. This sequence can be distinguished from the sequence $c_1a_2d_1$ com_{12}:11 b_2 that now leads to state 9. (To agent 2 the former sequence appears as com_{12}:11 a_2 com_{12}:12 b_2 whereas the latter appears as a_2 com_{12}:11 b_2.)

In a similar vein, prior to adding communication events, the sequence $c_1d_1a_2e_1$ leading to state 10 appears the same to agent 1 as the sequence $c_1a_2d_1b_2e_1$ leading to state 12. In this case, however, even after other communication events are

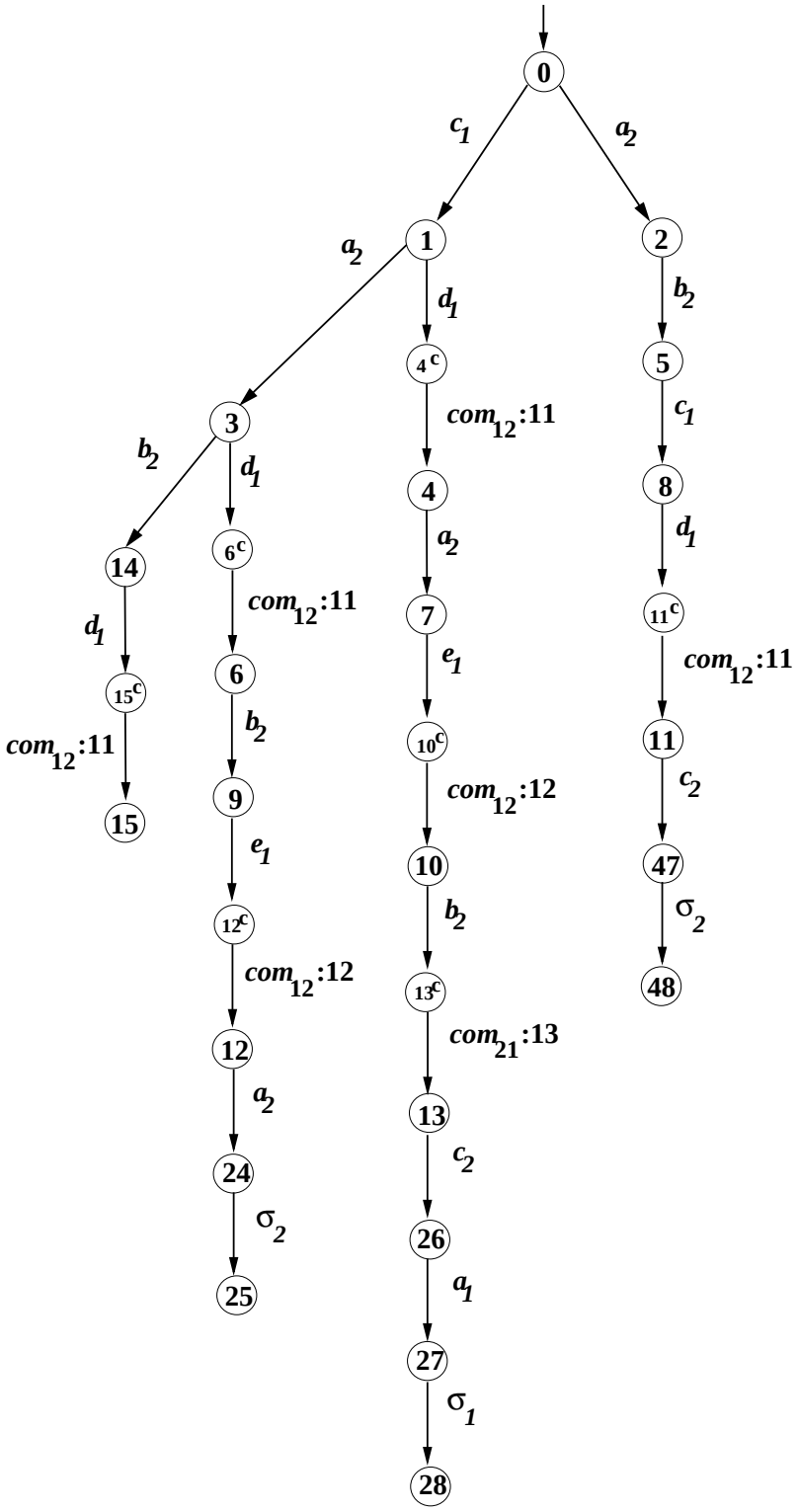

Figure 4.15: The portion of G^{com} for the plant in figure 4.13.

inserted (such as com_{12}:11 at states $6^{\mathbf{c}}$ and $4^{\mathbf{c}}$) the event com_{12}:12 at state $10^{\mathbf{c}}$ is included because the sequences $c_1 d_1\ com_{12}$:11 $a_2 e_1$ and $c_1 a_2 d_1\ com_{12}$:11 $b_2 e_1$ are still indistinguishable (i.e., both appear to agent 1 as $c_1 d_1\ com_{12}$:11 e_1).

In Lemma 4.2 we noted that when agent i could not distinguish between control sequence t and its control twin t', incorporating a communication event according to Theorem 4.1 renders the respective communication-equivalent sequences distinguishable. In Lemma 4.3 we showed that adding additional communications in Procedures 4.2 and 4.2a preserved this distinguishability. Now we show that adding additional communication in Procedures 4.3 and 4.3a still preserves the distinguishability of sequences.

LEMMA **4.4** *If $P_i(t) = P_i(t')$ (for control sequence t and its control twin t') and we follow Procedures 4.3 and 4.3a 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 4.3, communication events are added only to those sequences that are not control sequences. Therefore, since t and t' are control sequences then $\widehat{t^{\mathbf{c}}} = \widetilde{t^{\mathbf{c}}}$ and $\widehat{t'^{\mathbf{c}}} = \widetilde{t'^{\mathbf{c}}}$. By Lemma 4.3, $P_i^{\mathbf{c}}(\widehat{t^{\mathbf{c}}}) \neq P_i^{\mathbf{c}}(\widehat{t'^{\mathbf{c}}})$. 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 4.1 and 4.2 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 steps 3 and 4(b) of Procedure 4.3, 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). As in Case 1 of Lemma 4.3

we can find a prefix of t' of the form $b = u'\sigma_i v''''\sigma_j$ where $\delta^G(b, q_0^G) = x$. Suppose that this is the case and $\widetilde{t'^{\mathbf{c}}} = \widetilde{u'^{\mathbf{c}}}\sigma_i\widetilde{v''''^{\mathbf{c}}}\sigma_j\ com_{ji}{:}q\ \widetilde{b''^{\mathbf{c}}}$.

We initially assumed that $P_i^{\mathbf{c}}(\tilde{t}^{\mathbf{c}}) = P_i^{\mathbf{c}}(\widetilde{t'^{\mathbf{c}}})$:

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

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

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

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

(since $\sigma_j \notin \Sigma_{i,o}$)

(4.13)

As with Lemma 4.3, if $P_i^{\mathbf{c}}(\widetilde{u^{\mathbf{c}}}) = P_i^{\mathbf{c}}(\widetilde{u'^{\mathbf{c}}})$ then we have a contradiction because $P_i^{\mathbf{c}}(\sigma_i) \neq \varepsilon$. However, if $P_i^{\mathbf{c}}(\widetilde{u^{\mathbf{c}}}) \neq P_i^{\mathbf{c}}(\widetilde{u'^{\mathbf{c}}})$ then we must show it is not possible for $P_i^{\mathbf{c}}(\widetilde{u^{\mathbf{c}}}) = P_i^{\mathbf{c}}(\widetilde{u'^{\mathbf{c}}})\sigma_i P_i^{\mathbf{c}}(\widetilde{v''''^{\mathbf{c}}})$. Suppose that $P_i^{\mathbf{c}}(\widehat{u^{\mathbf{c}}}) = P_i^{\mathbf{c}}(\widehat{u'^{\mathbf{c}}}\sigma_i\widehat{v''''^{\mathbf{c}}})$. By Lemma 4.1, these sequences must look the same with the communication events "erased":

$$\begin{aligned}
\hat{P}_i(\widetilde{u^{\mathbf{c}}}) &= \hat{P}_i(\widetilde{u'^{\mathbf{c}}}\sigma_i\widehat{v''''^{\mathbf{c}}}) \\
P_i(u) &= P_i(u'\sigma_i v'''') \qquad \text{(by definition of } \hat{P}_i) \\
P_i(u) &= P_i(u')P_i(\sigma_i)P_i(v'''') \\
&= P_i(u)P_i(\sigma_i)P_i(v'''') \quad \text{(since } (u, u') \text{ is a maximal-P pair)}
\end{aligned}$$

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

□ LEMMA 4.4

4.4 A Well-defined Communication Protocol for G^{com}

Up 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 **4.17** *A communication protocol $P^{G_i^{com}}$ for agent i is said to be well-defined if*

$$(\forall \sigma \in \Sigma_{ij}^{com})(\forall q^{P^{G_i^{com}}} \in Q^{P^{G_i^{com}}})$$

$$\delta^{P^{G_i^{com}}}(\sigma, q^{P^{G_i^{com}}})! \Longrightarrow \nexists \sigma' \in ((\Sigma \cup \Sigma^{com}) \setminus \{\sigma\}) \text{ such that } \delta^{P^{G_i^{com}}}(\sigma', q^{P^{G_i^{com}}})!$$

That is, when the communication protocol for agent i indicates that agent i must communicate to agent j there is no ambiguity in what agent i does. Note that if G^{com} is consistent then the protocols generated with G^{com} are well-defined. When a communication event for agent i (i.e., $\sigma \in \Sigma_{ij}^{com}$) is defined at one of its local states $q^{P^{G_i^{com}}}$, that particular communication event is the only event defined at $q^{P^{G_i^{com}}}$.

Note that the way in which we add communication events to G^{com} ensures that our communication protocols are well-defined (because consistency is satisfied). In particular, we add communication events in such a way that unintentional communication is avoided. Recall our assumption about the structure of G: whenever more than one sequence leads to a communication state for agent i or a state that agent i finds indistinguishable from the communication state, this state is split. Thus, after Procedure 4.1a, 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 4.2 and 4.3, if (x,v) is a compatible communication pair for (q,t) that is added to $\mathcal{X}_{ij}^{compat}$, it is because $P_i^{\mathbf{c}}(\widetilde{v^{\mathbf{c}}}) = P_i^{\mathbf{c}}(\widetilde{s^{\mathbf{c}}})$, where $\widetilde{v^{\mathbf{c}}}, \widetilde{s^{\mathbf{c}}} \in L(G^{com})$. Again, when adding the communication event $com_{ij}{:}q$ to G^{com} with respect to the compatible communication pairs for (q,t) that appear in $\mathcal{X}_{ij}^{compat}$, we split any states that lead to unintentional communication. That is, without taking into account the nuances of two-way communication that could arise, $\delta^{G^{com}}(com_{ij}{:}q, x^{\mathbf{c}}) = x$. Thus, the only event defined at $q^{\mathbf{c}}$ and any state that agent i finds indistinguishable from $q^{\mathbf{c}}$ is $com_{ij}{:}q$. When agent i reaches the state in $P^{G_i^{com}}$ 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 agent j, thereby satisfying our definition of what it means to construct a well-defined communication protocol.

We illustrate the effects that the previously-defined procedures have on the construction of G^{com} from G and therefore on the generation of communication protocols for the decentralized agents. Figure 4.16(a) shows a portion of a plant where agent 1 sees and controls events a and b while agent 2 sees and controls events b and d; neither agent sees c. After passing the entire G (not shown here) through Procedure 4.1, there exists a $(q,t) \in \mathcal{C}_{12}$ with a communication sequence $s = dab$ as well as a $(q',t') \in \mathcal{C}_{21}$ with a communication sequence $s' = cb$. Note that this portion of G does not include a complete specification of t, t' or their respective control twins. This section briefly explains the mechanics of constructing a communication protocol: no control solution is described.

After Procedures 4.1 and 4.1a are complete, the portion of G^{com} relevant to G in figure 4.16(a) is shown in figure 4.16(b). Note that communication for control is added at states 6, corresponding to $s' = cb$, and 9, corresponding to $s = dab$. For this particular portion of G, no other communication events are added along the communication sequences. Therefore, after Procedures 4.2 and 4.2a, G^{com} is not altered.

There are several compatible communication pairs for (q,t) and (q',t') that can be identified from figure 4.16(a). In particular, $(5, ab)$ is a compatible communication pair for (q',t') and $(5, ab)$ is also a compatible communication pair for (q,t). When Procedure 4.3a is complete, this leads to two-way communication occurring at state $5^{\mathbf{cc}}$. We interpret the presence of two-way communication in a special way. For each agent to generate a well-defined protocol there must be no ambiguity as to when an agent sends its local state to another agent. Note that if we calculated the projection automaton of G^{com} in figure 4.16(c) for agent 1, after ab occurs agent

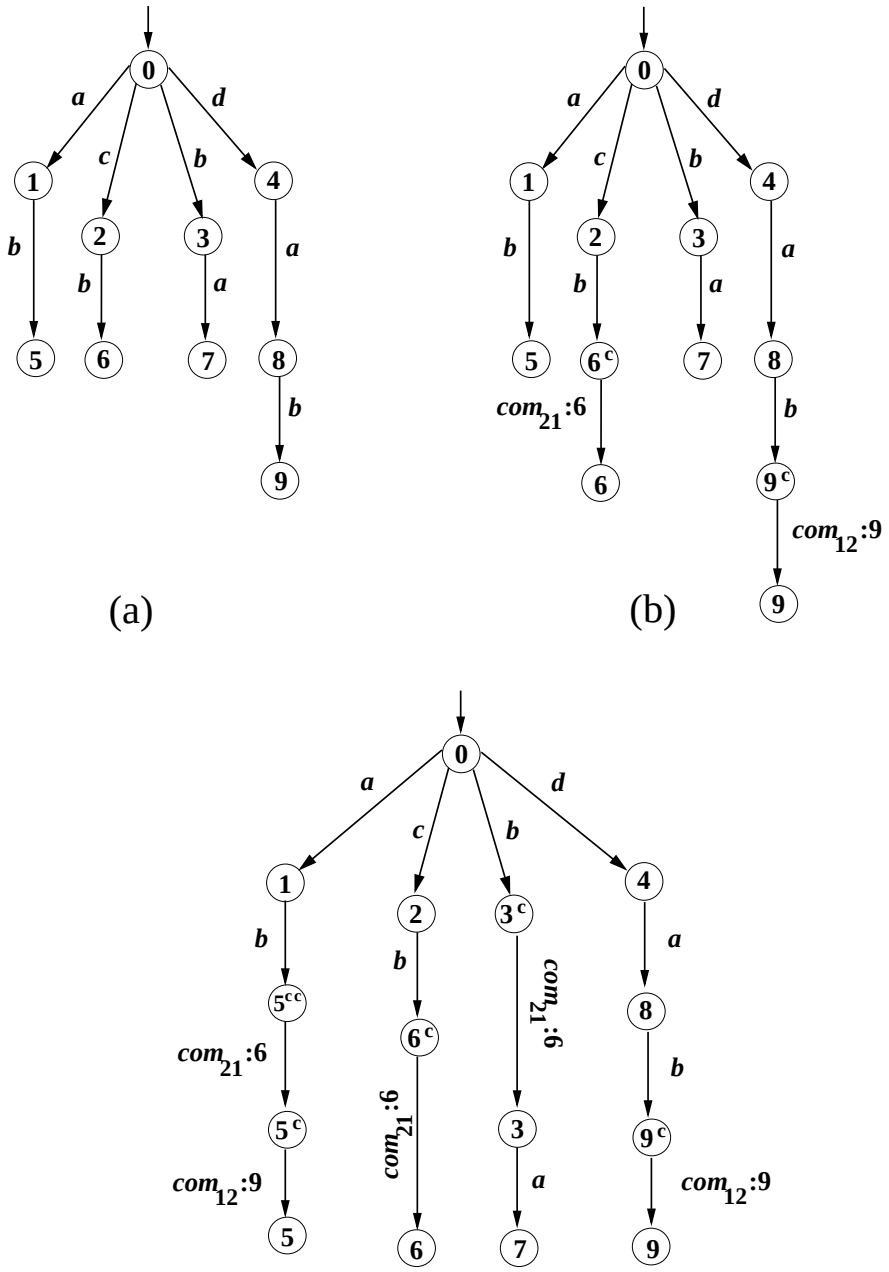

Figure 4.16: Developing well-defined communication protocols: (a) portion of plant G; (b) G^{com} from G after Procedures 4.1/4.1a; (c) G^{com} from G after Procedures 4.3/4.3a.

1 would be at a local state of $\{5^{\mathbf{cc}}, 9^{\mathbf{c}}\}$. At this local state, two events would be defined: com_{12}:6 and com_{21}:9. This does not constitute a well-defined communication protocol because com_{12}:6 is not the only event defined at state $\{5^{\mathbf{cc}}, 9^{\mathbf{c}}\}$. Our intent is that after seeing ab agent 1 must communicate its local state. If our plant is to accurately reflect our communication intention we must make the following sleight-of-hand: whenever a two-way communication event occurs in the plant, each agent "sees" its own communication event first and then observes the event representing the reception of communication from the other agent. Thus the "view" agent 1 has of the plant in figure 4.16(c) reverses the order of the two communication events defined at states $5^{\mathbf{cc}}$ and $5^{\mathbf{c}}$. Figure 4.16(c) also reflects the addition of $(3, b)$, a compatible communication pair for (q', t') and therefore the communication event com_{21}:6 is added after the new state $3^{\mathbf{c}}$. The final version of G^{com} (for the portion of G in figure 4.16(a)) after Procedures 4.3/4.3a are finished is shown in figure 4.16(c).

How do the agents determine their communication protocol from G^{com}? The communication protocol is determined by calculating the projection automaton (described in section 2.1.2) of G^{com} with respect to each agent. The projection automata of the portion of G^{com} from figure 4.16(c) are shown in figure 4.17. For instance, the communication protocol for agent 1, illustrated in figure 4.17(a), indicates that after seeing b, agent 1 can expect a communication from agent 2. Agent 2 communicates its local view of state 6. The states agent 1 considers possible after receiving the communication is the result of intersecting agent 2's local view of state 6 with agent 1's current local state. In our example, this has the result that agent 1 believes that plant could be in either state 3 or state 6. Since we are not describing the complete control solution for this example, the local views for each agent may be incomplete since they have been determined using only the states illustrated in figure 4.16.

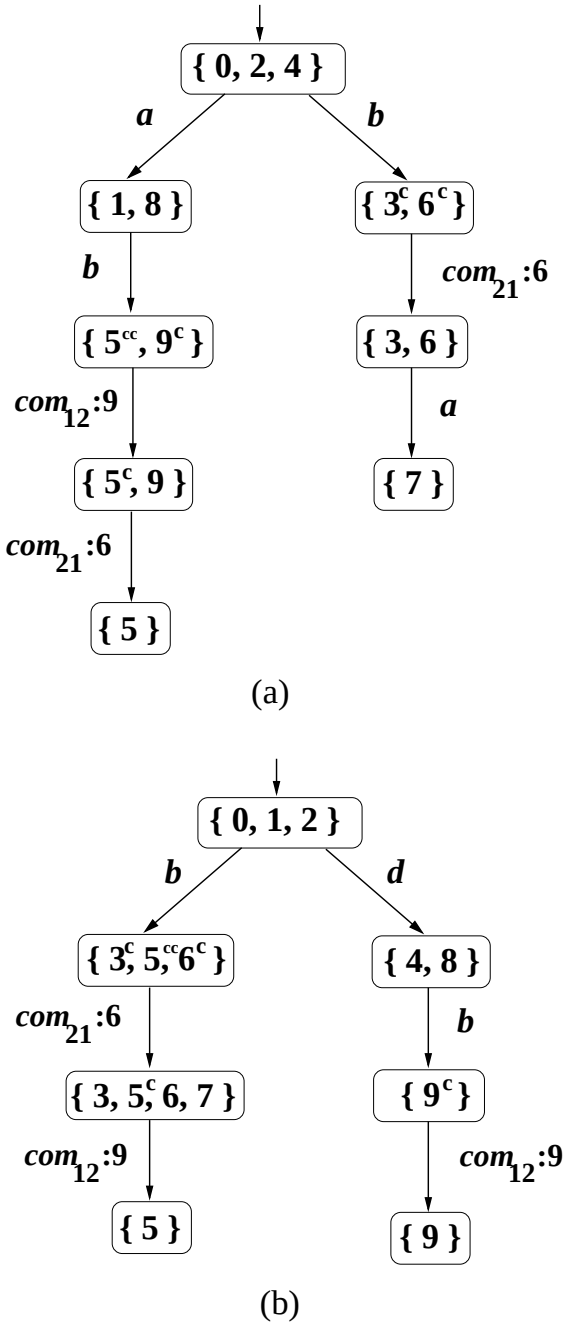

Figure 4.17: The communication protocols for each agent generated from G^{com} in figure 4.16 (a) $P^{G_1^{com}}$ for agent 1; (b) $P^{G_2^{com}}$ for agent 2.

4.5 Constructing $\mathcal{I}^{cDES'}$ from G^{com} and E^{com}

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. 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 $L(E)$:

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

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 $\Phi^{cDES'}$ 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}. As we did in section 3.1, to form $\Phi^{cDES'}$ we want to 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 $\Phi^{cDES'}$ 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 3.1 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 $\Phi^{cDES'}$ is defined for all $\sigma \in \Sigma^{G^{com}}$:

$$\pi^{cDES}(w^{\mathbf{c}})(\sigma_G) := \begin{cases} \textbf{true} & \text{if } \delta^{G^{com}}(\sigma, w_e)!, \\ \textbf{false} & \text{otherwise.} \end{cases} \tag{4.14}$$

$$\pi^{cDES'}(w^{\mathbf{c}})(\sigma_E) := \begin{cases} \textbf{true} & \text{if } \delta^{E^{com}}(\sigma, w_e)!, \\ \textbf{false} & \text{otherwise.} \end{cases} \tag{4.15}$$

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 $\forall \sigma \in \Sigma^{com}$:

$$\pi^{cDES'}(w)(\sigma_G) = \textbf{true} \text{ and } \pi^{cDES'}(w)(\sigma_E) = \textbf{true} \tag{4.16}$$

or

$$\pi^{cDES'}(w)(\sigma_G) = \textbf{false} \text{ and } \pi^{cDES'}(w)(\sigma_E) = \textbf{false}. \tag{4.17}$$

4.6 Is $\mathcal{I}^{cDES'}$ Kripke-observable?

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 **4.2** *Given $\mathcal{I}^{DES'}(G, E)$ that is not Kripke-observable. If we follow Procedures 4.1, 4.2 and 4.3 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,

$$(\mathcal{I}^{cDES'}, w) \models (\sigma_G \wedge \neg\sigma_E) \tag{4.18}$$

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

$$(\mathcal{I}^{cDES'}, w') \models (\sigma_G \wedge \sigma_E). \tag{4.19}$$

Note that by (4.16) and (4.17), it is not possible for σ_G, σ_E to correspond to $\sigma \in \Sigma^{com}$. Therefore, $\sigma \in \Sigma$ and, since $i \in \mathsf{G}_\sigma$, more specifically $\sigma \in \Sigma \cap \Sigma_{i,c}$.

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 $\widetilde{t^{\mathbf{c}}}$ while a path that leads to w'_e be the sequence $\widetilde{t'^{\mathbf{c}}}$. Since $w \sim_i w'$,

$$P_i^{\mathbf{c}}(\widetilde{t^{\mathbf{c}}}) = P_i^{\mathbf{c}}(\widetilde{t'^{\mathbf{c}}}). \tag{4.20}$$

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

$$\hat{P}(\widetilde{t^{\mathbf{c}}}) = t \tag{4.21}$$

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

$$\hat{P}(\widetilde{t'^{\mathbf{c}}}) = t'. \tag{4.22}$$

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

By (4.21), it must also be the case that if all communication events are erased from agent i's observation of $\widetilde{t^{\mathbf{c}}}$ that this is exactly agent i's view of t:

$$\hat{P}(P_i^{\mathbf{c}}(\widetilde{t^{\mathbf{c}}})) = P_i(t). \tag{4.23}$$

Similarly, by (4.22)

$$\hat{P}(P_i^{\mathbf{c}}(\widetilde{t'^{\mathbf{c}}})) = P_i(t'). \tag{4.24}$$

If we apply the $\hat{P}$ operator to both sides of (4.20) we get (from (4.23) and (4.24)) $P_i(t) = P_i(t')$, 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 (4.18) we know that $\delta^{G^{com}}(\sigma, w_e)$ is defined but that $\delta^{E^{com}}(\sigma, w_e)$ is not defined.

By Observation 4.1 in section 4.2.3, 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 (4.21), 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

$$(\mathcal{I}^{DES'}, v) \models \sigma_G \wedge \neg\sigma_E. \tag{4.25}$$

From (4.19) 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 4.1 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

$$(\mathcal{I}^{DES'}, v') \models \sigma_G \wedge \sigma_E. \tag{4.26}$$

By (4.25) and (4.26) 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 (4.25) we have $t\sigma \in L(G)$ but $t\sigma \notin L(E)$. Similarly, by (4.26) we have $t'\sigma \in L(G)$ and $t'\sigma \in L(E)$ so t, t' and σ satisfy the hypothesis of Theorem 4.1. Therefore we apply Procedures 4.1 through to 4.3a. By Lemma 4.4, $P_i^{\mathbf{c}}(\tilde{t^{\mathbf{c}}}) \neq P_i^{\mathbf{c}}(\widetilde{t'^{\mathbf{c}}})$, which contradicts (4.20). Therefore there exists $i \in \mathsf{G}_\sigma$ such that $(\mathcal{I}^{cDES'}, w') \models K_i \neg \sigma_E$.

□ Theorem 4.2

4.7 Minimal Communication

In section 4.2 we discussed our strategy for constructing a set of control communication pairs where one agent communicates to another agent to solve the control problem. Although communication at every $(q, t) \in \mathcal{C}_{ij}$ will lead to agent 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., $\tilde{\mathcal{C}}_{ij} \subset \mathcal{C}_{ij}$) will also lead to a control solution.

What do we mean by saying an element of $\mathcal{C}_{ij}$ represents extraneous communication? One of our communication goals is to communicate enough information to allow each agent 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 agent j

to distinguish a set of bad state from a several sets of look-alike good states. Or it is possible that a compatible communication pair $(x, v) \in \mathcal{C}_{ij}^{compat}$ (where (x, v) is not a compatible pair for (q, t)) occurring prior to (q, t) provides agent 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 [31] where a set of communications is minimal when it is the case that if "any one event occurrence is not communicated from an agent to the other, the agents 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 agents in our system is to solve the control problem and satisfy consistency.

DEFINITION **4.18** *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 plant* G^{com} *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 agents 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.

4.7.1 An example system requiring communication

This section revisits the portion of a plant introduced in figure 4.13. We will see that the control and communication solution generated by following Procedures 4.1,

4.2 and 4.3 contains extraneous communication. Therefore the procedures applied so far do not yield a minimal communication set.

Figure 4.18 shows a plant G and E where communication between the two supervisors/agents is necessary to find a control solution. As is clear from the figure, the automaton in figure 4.13 is a subautomaton of the plant in figure 4.18. In this example we let $\Sigma_{1,o} = \Sigma_{1,c} = \{a_1, c_1, d_1, e_1, f_1, m_1, \sigma_1\}$ and $\Sigma_{2,o} = \Sigma_{2,c} = \{a_2, b_2, c_2, f_2, g_2, h_2, \sigma_2\}$. There is one event that neither agent observes: $\Sigma_{uo} = \{\gamma\}$. The projection automaton for each agent is shown in figure 4.19. When we translate these structures into an interpreted system $\mathcal{I}^{DES}$, we detect five global states where Kripke-observability fails:

1. Agent 2 does not know whether to disable σ_2 since it cannot distinguish bad state 56 from good state 47:

$$(56, \{1, 3, 8, 14, 29, 30, 55, 56, 57\}, \{26, 27, 28, 47, 56\});$$

2. Agent 2 does not know whether to disable σ_2 since it cannot distinguish bad state 38 from good state 24:

$$(38, \{10, 12, 13, 24, 25, 26, 35, 37, 38, 40\}, \{24, 36, 38\});$$

3. Agent 1 does not know whether to disable σ_1 since it cannot distinguish bad state 39 from good state 27:

$$(39, \{27, 39\}, \{2, 3, 6, 7, 10, 30, 32, 33, 35, 37, 39, 41, 53\});$$

4. Agent 1 does not know whether to disable σ_1 since it cannot distinguish bad state 51 from good state 58:

$$(51, \{49, 50, 51, 58\}, \{21, 22, 23, 51, 52\});$$

and

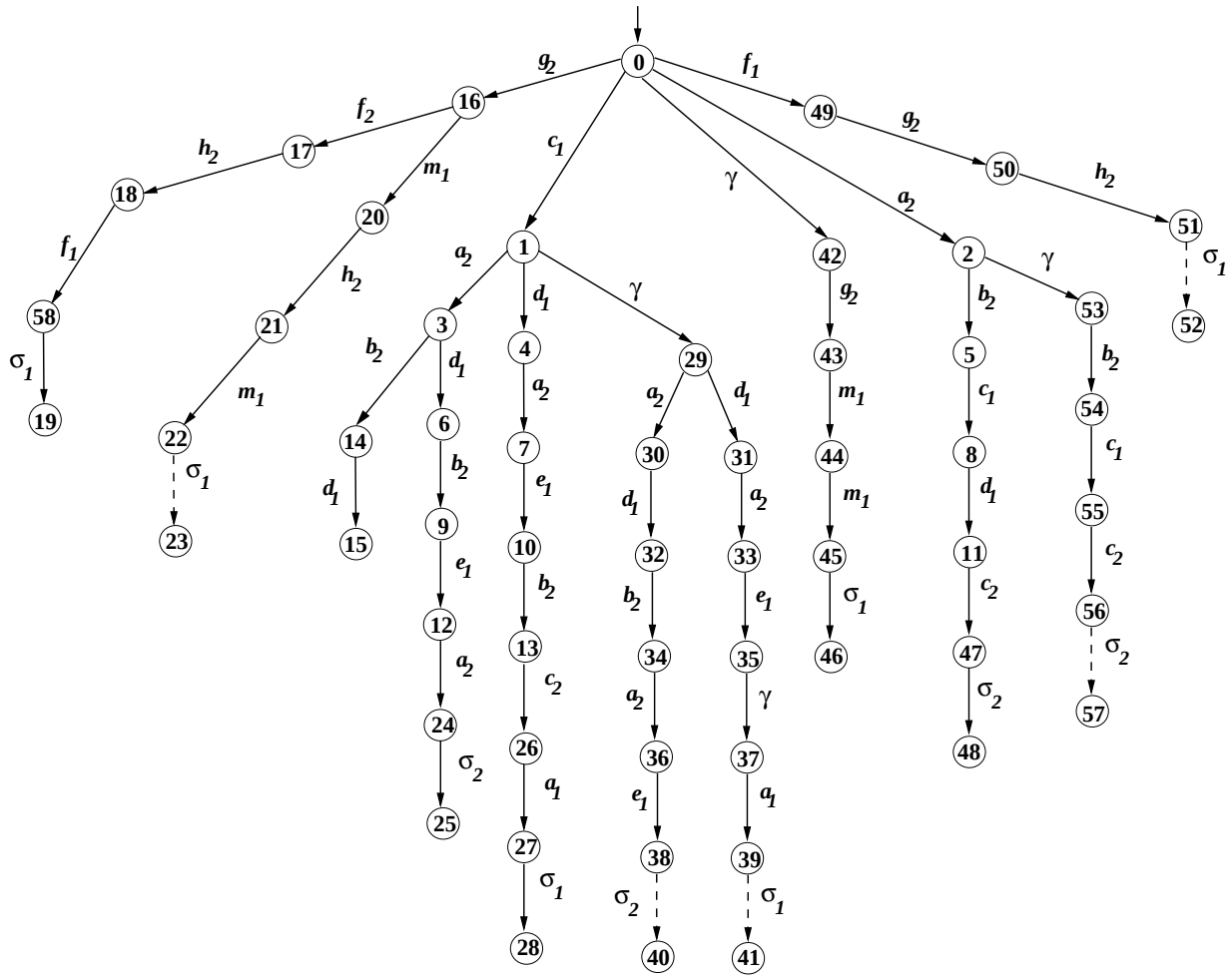

Figure 4.18: A plant requiring communication. Illegal transitions are marked by a dashed line.

5. Agent 1 does not know whether to disable σ_1 since it cannot distinguish bad state 22 from good state 45 :

$$(22, \{22, 45\}, \{21, 22, 23, 51, 52\}).$$

For the remainder of the discussion of this example, we will refer to the global states using local state labels as noted in figure 4.19. For example, global state $(44, \{20, 21, 44\}, \{16, 20, 43, 44, 45, 46, 50\})$ becomes $(44, 3, 2)$ since the automaton state $\{20, 21, 44\}$ in P^{G_1} (at the top of figure 4.19) has a label of "3". Similarly, the

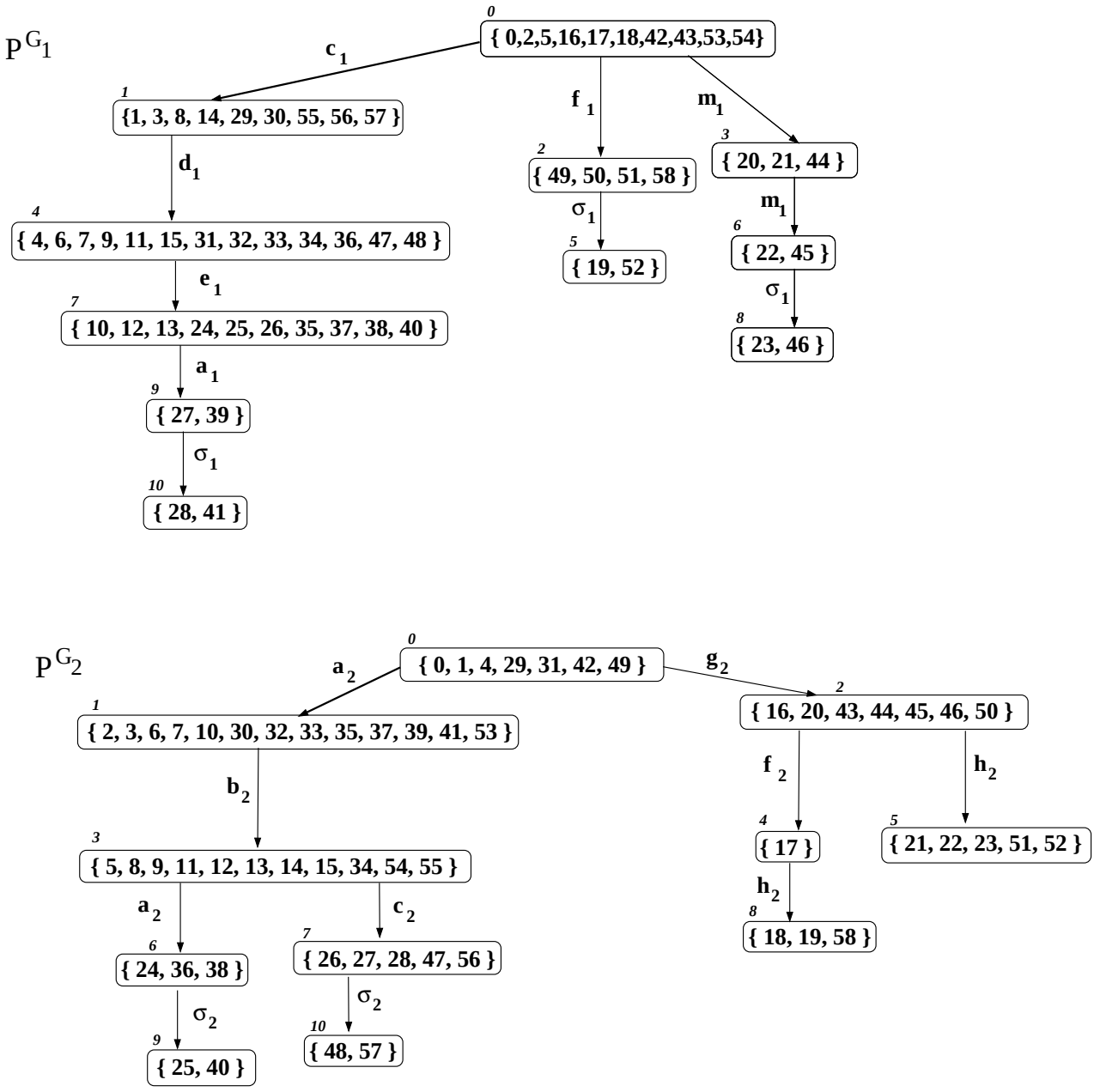

Figure 4.19: The projection automaton for agent 1 (the top of the figure) and agent 2 (bottom of the figure) for G in figure 4.18. Italicized numbers in top left of the corners of each state denote a label we use to refer to the state.

state $\{16, 20, 43, 44, 45, 46, 50\}$ of P^{G_2} (the bottom half of figure 4.19) has a label of "2".

Why does agent 1 not know whether to disable σ_1 at global state $r(m) = (51, 2, 5)$? We focus on the truth assignments to this global state, and in particular the propositions σ_{1_G} and σ_{1_E} in Φ^{DES}:

$$\sigma_{1_G} = \textbf{true},$$

and

$$\sigma_{1_E} = \textbf{false}.$$

These values correspond to the fact that at state 51 in the plant (see figure 4.18), the only event that is defined at that plant state is σ_1 (therefore σ_{1_G} is **true**) but the dashed line for the transition means that σ_1 is not defined in the legal automaton (and σ_{1_E} is **false**). Agent 1 does not know whether to disable σ_1 because its local state $r_1(m)$ contains both plant states 51 and 58, which means these states look alike to agent 1. The truth value of σ_{1_E} is **false** at $(51, 2, 5)$ but the truth value of σ_{1_E} at the corresponding global state of plant state 58 (which happens to be $(58, 2, 8)$) is **true** since σ_1 is defined in the plant and the legal automaton according to figure 4.18. The conflicting truth values for σ_{1_E} at states that are indistinguishable to agent 1 means that agent 1 does not know whether to disable σ_1 at this point in the system.

We have identified the places where Kripke-observability fails, and therefore, using the monitoring automaton, we can reconstruct the sequences which an agent cannot distinguish without further information. There are five pairs of sequences that lead to $\mathcal{I}^{DES'}$ not satisfying Kripke-observability:

$$P_1(c_1 d_1 a_2 e_1 b_2 c_2 a_1) = P_1(c_1 \gamma d_1 a_2 e_1 \gamma a_1); \tag{4.27}$$

$$P_1(f_1 g_2 h_2) = P_1(g_2 f_2 h_2 f_1); \tag{4.28}$$

$$P_1(g_2 m_1 h_2 m_1) = P_1(\gamma g_2 m_1 m_1); \tag{4.29}$$

$$P_2(a_2 b_2 c_1 d_1 c_2) = P_2(a_2 \gamma b_2 c_1 c_2 \sigma_2); \tag{4.30}$$

$$P_2(c_1a_2d_1b_2e_1a_2) = P_2(c_1\gamma a_2d_1b_2a_2e_1). \tag{4.31}$$

We use Procedure 4.1 to identify the control communication pairs for each agent. For example, agent 1 cannot distinguish the sequences in (4.29) since $P_1(g_2m_1h_2m_1) = P_1(\gamma g_2m_1m_1) = m_1m_1$. The maximal-P pair for these sequences is $(g_2m_1, \gamma g_2m_1)$. Agent 2 communicates following one of the entries of the maximal-P pair that is immediately followed by an event observable to agent 2. Since g_2m_1 is followed by h_2 and γg_2m_1 is followed by m_1, agent 2 communicates at state 21 when it sees $g_2m_1h_2$. The control communication pair is therefore $(21, g_2m_1h_2m_1)$, the communication sequence is $g_2m_1h_2$ and the associated control twin is $\gamma g_2m_1m_1$. The complete set of control communication pairs for this example is

$$\begin{aligned} \mathcal{C}_{12} &= \{(11, a_2b_2c_1d_1c_2), (12, c_1a_2d_1b_2e_1a_2)\}, \\ \mathcal{C}_{21} &= \{(13, c_1d_1a_2e_1b_2c_2a_1), (16, g_2f_2h_2f_1), (21, g_2m_1h_2m_1)\}. \end{aligned} \tag{4.32}$$

The communication sequence s for each communication control pair is as follows:

- Agent 1 communicates after seeing $s = a_2b_2c_1d_1$;
- Agent 1 communicates after seeing $s = c_1a_2d_1b_2e_1$
- Agent 2 communicates after seeing $s = c_1d_1a_2e_1b_2$
- Agent 2 communicates after seeing $s = g_2$
- Agent 2 communicates after seeing $s = g_2m_1h_2$

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

- row/column 1 corresponds to control communication pair $(11, a_2b_2c_1d_1c_2)$;
- row/column 2 corresponds to control communication pair $(12, c_1a_2d_1b_2e_1a_2)$;
- row/column 3 corresponds to control communication pair $(13, c_1d_1a_2e_1b_2c_2a_1)$;

- row/column 4 corresponds to control communication pair $(16, g_2 f_2 h_2 f_1)$;
- row/column 5 corresponds to control communication pair $(21, g_2 m_1 h_2 m_1)$.

After Procedure 4.2, the dependency matrix D looks like

$$D = \begin{bmatrix} \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ (6, c_1 a_2 d_1) & \emptyset & \emptyset & \emptyset & \emptyset \\ (4, c_1 d_1) & (10, c_1 d_1 a_2 e_1) & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & (16, g_2) & \emptyset \end{bmatrix}$$

Additionally, after Procedure 4.2, we identify the following compatible communication pairs (as shown in section 4.3.1, p. 100):

- $\mathcal{C}_{12}^{compat} = \{(4, c_1 d_1), (6, c_1 a_2 d_1), (10, c_1 d_1 a_2 e_1)\}$;
- $\mathcal{C}_{21}^{compat} = \emptyset$.

Note that $(16, g_2)$ is not added to $\mathcal{C}_{21}^{compat}$ because g_2 corresponds to the communication sequence for the control communication pair $(16, g_2 f_2 h_2 f_1)$.

At the initiation of Procedure 4.3, the set $\mathcal{X}_{4.3}$ contains the following entries (the corresponding row in $\hat{D}$ is also indicated):

- $(15, c_1 a_2 b_2 d_1)$ is compatible with $(11, a_2 b_2 c_1 d_1 c_2)$ (row 1);
- $(31, c_1 \gamma d_1)$ is also compatible with $(11, a_2 b_2 c_1 d_1 c_2)$ (row 2);
- $(32, c_1 \gamma a_2 d_1)$ is also compatible with $(11, a_2 b_2 c_1 d_1 c_2)$ (row 3);
- $(35, c_1 \gamma d_1 a_2 e_1)$ is compatible with $(12, c_1 a_2 d_1 b_2 e_1 a_2)$ (row 4);
- $(38, c_1 \gamma a_2 d_1 b_2 a_2 e_1)$ is compatible with $(12, c_1 a_2 d_1 b_2 e_1 a_2)$ (row 5);
- $(14, c_1 a_2 b_2)$ is compatible with $(13, c_1 d_1 a_2 e_1 b_2 c_2 a_1)$ (row 6);

- $(34, c_1\gamma a_2 b_2)$ is also compatible with $(13, c_1 d_1 a_2 e_1 b_2 c_2 a_1)$ (row 7);
- $(54, a_2\gamma b_2)$ is also compatible with $(13, c_1 d_1 a_2 e_1 b_2 c_2 a_1)$ (row 8);
- $(43, \gamma g_2)$ is compatible with $(16, g_2 f_2 h_2 f_1)$ (row 9);
- $(50, f_1 g_2)$ is also compatible with $(16, g_2 f_2 h_2 f_1)$ (row 10);
- $(51, f_1 g_2 h_2)$ is compatible with $(21, g_2 m_1 h_2 m_1)$ (row 11).

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

$$\hat{D} = \begin{bmatrix} \emptyset & \emptyset & (14, c_1 a_2 b_2) & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ (31, c_1\gamma d_1) & \emptyset & \emptyset & \emptyset & \emptyset \\ (32, c_1\gamma a_2 d_1) & \emptyset & (34, c_1\gamma a_2 b_2) & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & (50, f_1 g_2) & \emptyset \end{bmatrix}$$

In section 4.3.2 (p. 108) we determined the membership of $(15, c_1 a_2 b_2 d_1)$ (row 1 of $\hat{D}$) and $(14, c_1 a_2 b_2)$ (row 4 of $\hat{D}$) in $\mathcal{C}_{12}$ (i.e., the former is in the set, the latter is not in the set). We continue here by examining the remaining rows.

Rows 2, 3, 7, 8, 9 and 10 of $\hat{D}$ contain all $\emptyset$ entries. To determine if the corresponding compatible communication pair is added to $\mathcal{C}^{compat}$, we simply see if the appropriate row of D also contains all $\emptyset$ entries. For instance, rows 2 and 3 in $\hat{D}$ correspond to $(31, c_1\gamma d_1)$ and $(32, c_1\gamma a_2 d_1)$ and both are compatible with $(11, a_2 b_2 c_1 d_1 c_2)$—row 1 in D. Since all three of these rows contain all $\emptyset$ entries, we add $(31, c_1\gamma d_1)$ and $(32, c_1\gamma a_2 d_1)$ to $\mathcal{C}_{12}^{compat}$ and mark these two compatible communication pairs "resolved". Similar analysis of $(43, \gamma g_2)$ and $(50, f_1 g_2)$—checking to see if the entries

in row 4 of D (corresponding to the control communication pair both are compatible with, namely $(16, g_2 f_2 h_2 f_1)$)—means that $(43, \gamma g_2)$ and $(50, f_1 g_2)$ are added to $\mathcal{C}_{21}^{compat}$. The two pairs $(43, \gamma g_2)$ and $(50, f_1 g_2)$ are marked "resolved". Finally we compare rows 7 and 8 in $\hat{D}$ to row 3 in D. Since row 3 of D does not contain all $\emptyset$ entries, we do not add $(34, c_1 \gamma a_2 b_2)$ and $(54, a_2 \gamma b_2)$ to $\mathcal{C}_{21}^{compat}$. This means that after communication events are added along the communication sequence $c_1 d_1 a_2 e_1 b_2$ and along the sequences $c_1 \gamma a_2 b_2$ and $a_2 \gamma b_2$, the latter two sequences no longer have the the same projection according to agent 2 in $L(G^{com})$ as the communication sequence.

We can now check $(51, f_1 g_2 h_2)$ since all its non-empty elements (i.e., $(50, f_1 g_2)$) are marked "resolved". Since $(50, f_1 g_2)$ is in $\mathcal{C}_{21}^{compat}$, this is a communication event that will be added to G^{com} so it will be inserted along sequence $f_1 g_2 h_2$. Thus we do take its presence into consideration when determining whether or not $(51, f_1 g_2 h_2)$ is still compatible with $(21, g_2 m_1 h_2 m_1)$. We compare row 11 of $\hat{D}$ to row 5 of D and note that both these rows have the same pattern of empty and non-empty entries. That is, the fourth column in each row is non-empty: $(16, g_2)$ in D and $(50, f_1 g_2)$ in $\hat{D}$. We must next make certain that the projections of these sequences (once communication is added) would be the same: $P_2(g_2) = P_2(f_1 g_2)$. We thus add $(51, f_1 g_2 h_2)$ to $\mathcal{C}_{21}^{compat}$. Similarly we check $(35, c_1 \gamma d_1 a_2 e_1)$ and can show that it too is added to $\mathcal{C}_{12}^{compat}$.

The final set of compatible communication pairs for this example is

- $\mathcal{C}_{12}^{compat} = \{(4, c_1 d_1), (6, c_1 a_2 d_1), (10, c_1 d_1 a_2 e_1), (15, c_1 a_2 b_2 d_1),$
  $(31, c_1 \gamma d_1), (32, c_1 \gamma a_2 d_1), (35, c_1 \gamma d_1 a_2 e_1), (38, c_1 \gamma a_2 d_1 b_2 a_2 d_1)\}$;
- $\mathcal{C}_{21}^{compat} = \{(43, \gamma g_2), (50, f_1 g_2), (51, f_1 g_2 h_2)\}$.

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 agent 2 communicated when the plant is at state 21, the compatible communication

at state 51 would be enough to allow agent 1 to distinguish between the sequence leading to state 51 and the sequence leading to state 58.

4.7.2 A minimal algorithm for 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 (see [5] for an excellent summary). 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 agents distinguish between certain "good" states and "bad" states hence making all the correct control decisions while satisfying consistency. After following Procedures 4.1, 4.2 and 4.3 we have a set of communication pairs that, when incorporated into G^{com}, will allow agents 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 an agent 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 [5].

Under what circumstances could a (q, t) allow an agent 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 agent j to distinguish states along sequence t from those along its control twin t'. Further, let $(\hat{q}, \hat{t}) \in \mathcal{C}_{ij}$ be a communication that distinguishes the states along $\hat{t}$ from those along its control twin $\widehat{t'}$. There are three scenarios where (q, t) could allow agent 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 $(\hat{q}, \hat{t})$. In addition, let communication at q (after s occurs) be sufficient to allow agent j to distinguish not only t from t', but also distinguish $\hat{t}$ from $\widehat{t'}$. Then communication at $(\hat{q}, \hat{t})$ would be unnecessary.

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

3. Suppose (x, v), a compatible communication pair for (q, t), is such that $v \in \overline{\widehat{t'}}$. Again, if communication at (x, v) allows agent j to distinguish $\hat{t}$ from $\widehat{t'}$, then additional communication at $(\hat{q}, \hat{t})$ 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 represents the fact 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 4.1, 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 agent to distinguish the most 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 4.1 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 4.1 is performed for each $(q,t) \in \mathcal{C}$. We want to find out how many 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 **4.1** *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 an agent's local state at (c, d) means that b and b' can be distinguished by the appropriate agent (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 an agent 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 agents is described in Algorithm 4.2. 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 agent 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 4.1) distinguishes no sequences, then we cannot reach a solution (step 20). If, though, the selected candidate would allow the appropriate agent 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 agent to distinguish.

ALGORITHM **4.2** *Greedy Communication*

Input. A set of control communication pairs for each agent: $\mathcal{C}_{12}$, $\mathcal{C}_{21}$.

Output. A set of communication pairs $(FinalCom \cup \mathcal{X}^{compat}) \subseteq \mathcal{C} \cup \mathcal{XV}$ that, when incorporated into G^{com} will allow agent j to make the correct control decisions.

begin
1. $\mathcal{C} \leftarrow \mathcal{C}_{12} \cup \mathcal{C}_{21}$; $New \leftarrow \mathcal{C}$; $ControlCom \leftarrow \emptyset$; $FinalCom \leftarrow \emptyset$
2. **while** $(New \neq \emptyset)$ **and** $(\mathcal{C} \neq \emptyset)$ **do**
3. $\quad (q, t) \leftarrow$ an element of $\mathcal{C}$ maximizing the cardinality of *distinguish* from Algorithm 4.1
4. $\quad$ **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** $distinguish_{(q,t)} \neq \emptyset$ **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. $ControlCom \leftarrow ControlCom \cup (control\_com_{(q,t)} \setminus \{(q,t)\})$
$FinalCom \leftarrow FinalCom \cup \{(q,t)\}$
12. **if** $New = \emptyset$ **then**
$\mathcal{C} \leftarrow \mathcal{C}_{12} \cup \mathcal{C}_{21}$
13. **for all** $(q,t) \in FinalCom$
Let $distinguish^{\mathcal{C}}_{(q,t)}$ be the output of Algorithm 4.1 using input of (q,t) and $\mathcal{C}$
Let $Distinguish^{\mathcal{C}}_{(q',t')} \leftarrow \bigcup_{(q',t') \in FinalCom \setminus \{(q,t)\}} distinguish^{\mathcal{C}}_{(q',t')}$
14. **if** $distinguish^{\mathcal{C}}_{(q,t)} \setminus Distinguish^{\mathcal{C}}_{(q',t')} = \emptyset$ **then**
15. $FinalCom \leftarrow FinalCom \setminus \{(q,t)\}$
16. $ControlCom \leftarrow ControlCom \setminus \{(x,v) \mid (x,v) \in ControlCom$ and (x,v) is a compatible communication pair for $(q,t)\}$
17. $FinalCom \leftarrow FinalCom \cup ControlCom$
18. $\mathcal{X}^{compat} \leftarrow \{(x,v) \mid (x,v)$ is compatible with an element of $FinalCom\}$
19. **return** $FinalCom \cup \mathcal{X}^{compat}$
20. **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 4.1 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 (q,t)'s, 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 most number of elements). In this case, at steps 4 and 5, we randomly select one of the maximal candidates.

After step 11, $FinalCom$ contains all the (q,t) where the communication of an agent's local view of q would lead to the other agent making the correct control decision. We want to make sure that each element of $FinalCom$ (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 $FinalCom$ do not. If this is not the case, then we could remove (q,t) from $FinalCom$ and still find a control solution. Thus at step 14 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 the 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 4.1 during the selection of the candidate that maximizes Algorithm 4.1 during the first iteration of Algorithm 4.2. We calculate this set for all the other elements (q',t') in $FinalCom$. 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 an agent 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 $FinalCom$. Thus (q,t) is removed from $FinalCom$ in step 15. In addition, any of its compatible communication pairs are removed from $ControlCom$ (step 16).

Note that step 18 may not have to be calculated since this set may already have been calculated by prior utilisation of Procedures 4.2 and 4.3. Thus $\mathcal{X}^{compat}$ might simply be a subset of $\mathcal{C}^{compat}$.

The output of the greedy algorithm, $FinalCom \cup \mathcal{X}^{compat}$, is used to create G^{com}

in the manner described by Procedures 4.1a, 4.2a and 4.3a. The communication protocol for each agent is then generated by calculating the projection automata of G^{com}.

THEOREM **4.3** *The set of communication pairs $FinalCom \cup \mathcal{X}^{compat}$ obtained from executing Algorithm 4.2 is a set of minimal communication pairs.*

Proof. (By contradiction)

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

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 4.2, (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 4.2. 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 4.3a. 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 the our notion of consistency from Definition 4.12. 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 4.2, removing (a, b) means that a communication event will not be added to state a in Procedure 4.3a. Using the same reasoning as for Case 1, the removal of (a, b) means the system is no longer consistent. This contradicts our assumption.

□ THEOREM 4.3

Note that if some algorithm other than Procedures 4.1, 4.2, and 4.3 was used to generate a communication solution to the decentralized control problem, then Algorithm 4.2 could still be used to pare the solution down to a minimal communication set.

We return to the example of figure 4.18. Our input to the greedy algorithm is the set of control communication pairs in (4.32). The first control communication pair to maximize *distinguish* of Algorithm 4.1 is $(21, g_2m_1h_2m_1)$. It is the pair corresponding to a communication that would allow an agent to distinguish the largest number of control sequences from their control twins: (4.28) and (4.29) are both distinguished by $(21, g_2m_1h_2m_1)$ and its compatible communication pair $(51, f_1g_2h_2)$.

We remove $(21, g_2m_1h_2m_1)$ and $(16, g_2f2h_2f_1)$ from New and remove $(21, g_2m_1h_2m_1)$ from $\mathcal{C}$. The set $FinalCom$ now contains $(21, g_2m_1h_2m_1)$ and $ControlCom$ contains $(51, f_1g_2h_2)$.

At the next iteration we choose the control communication pair in $\mathcal{C}$ that distinguishes the most number of the remaining elements in New.

In this case, we have a three-way tie for a candidate that maximizes *distinguish* of Algorithm 4.1: each of $(11, a_2b_2c_1d_1c_2)$, $(12, c_1a_2d_1b_2e_1a_2)$ and $(13, c_1d_1a_2e_1b_2c_2a_1)$ distinguishes one of the control sequence/control twin pairs associated with an element of New. We randomly choose $(12, c_1a_2d_1b_2e_1a_2)$ and remove it from $\mathcal{C}$ and remove the element of New whose control sequence/control twin it distinguishes (namely itself). $FinalCom$ is now $\{(21, g_2m_1h_2m_1), (12, c_1a_2d_1b_2e_1a_2)\}$ while $ControlCom$ is

still $\{(51, f_1 g_2 h_2)\}$.

The next iteration of the algorithm sees a two-way tie for a maximal element: $(13, c_1 d_1 a_2 e_1 b_2 c_2 a_1)$ and $(11, a_2 b_2 c_1 d_1 c_2)$ distinguish one element each in New. The selection of $(11, a_2 b_2 c_1 d_1 c_2)$ is random, once again, and we remove it from both New (since it distinguishes the control sequence associated with it from its control twin) and $\mathcal{C}$. $FinalCom$ is $\{(21, g_2 m_1 h_2 m_1), (12, c_1 a_2 d_1 b_2 e_1 a_2), (11, a_2 b_2 c_1 d_1 c_2)\}$ and $ControlCom$ is $\{(51, f_1 g_2 h_2)\}$.

The element maximizing the selection algorithm in the last iteration of the greedy algorithm is $(13, c_1 d_1 a_2 e_1 b_2 c_2 a_1)$, and like the previous two iterations, it distinguishes the control sequence associated with itself from its control twin. $FinalCom$ for this example is $\{(21, g_2 m_1 h_2 m_1), (12, c_1 a_2 d_1 b_2 e_1 a_2), (11, a_2 b_2 c_1 d_1 c_2), (13, c_1 d_1 a_2 e_1 b_2 c_2 a_1)\}$. $ControlCom$ is $\{(51, f_1 g_2 h_2)\}$.

At this point, $New = \emptyset$. We want to make sure that each element in $FinalCom$ represents a communication that allows an agent to uniquely distinguish at least one control sequence t from its control twin t'. Using the original set of control communication pairs $\mathcal{C}_{12} \cup \mathcal{C}_{21}$, we calculate the following for the elements of $FinalCom$ using Algorithm 4.1:

$$distinguish_{(21, g_2 m_1 h_2 m_1)} = \{(21, g_2 m_1 h_2 m_1), (16, g_2 f 2 h_2 f_1)\}, \tag{4.33}$$

$$distinguish_{(12, c_1 a_2 d_1 b_2 e_1 a_2)} = \{(12, c_1 a_2 d_1 b_2 e_1 a_2)\}, \tag{4.34}$$

$$distinguish_{(11, a_2 b_2 c_1 d_1 c_2)} = \{(11, a_2 b_2 c_1 d_1 c_2)\}, \tag{4.35}$$

$$distinguish_{(13, c_1 d_1 a_2 e_1 b_2 c_2 a_1)} = \{(13, c_1 d_1 a_2 e_1 b_2 c_2 a_1)\}. \tag{4.36}$$

Note that we do not remove any of the elements of $FinalCom$. For example, to see if communication with respect to control communication pair $(12, c_1 a_2 d_1 b_2 e_1 a_2)$ distinguishes anything unique, we take the union of sets marked (4.33), (4.35) and (4.36) and subtract this from (4.34). The result is not empty, therefore we leave

$(12, c_1a_2d_1b_2e_1a_2)$ in $FinalCom$. We repeat this exercise for each of the other elements in $FinalCom$. By step 17 of Algorithm 4.2, $FinalCom = \{(21, g_2m_1h_2m_1)$, $(12, c_1a_2d_1b_2e_1a_2)$, $(11, a_2b_2c_1d_1c_2)$, $(13, c_1d_1a_2e_1b_2c_2a_1)$, $(51, f_1g_2h_2)\}$.

The compatible communication pairs were previously calculated at the beginning of this section. The only ones not included are the compatible communication pairs for $(16, g_2f_2h_2f_1)$. Thus $\mathcal{X}^{compat} = \{(4, c_1d_1)$, $(6, c_1a_2d_1)$, $(10, c_1d_1a_2e_1)$, $(15, c_1a_2b_2d_1)$, $(31, c_1\gamma d_1)$, $(32, c_1\gamma a_2d_1)$, $(35, c_1\gamma d_1a_2e_1)$, $(38, c_1\gamma a_2d_1b_2a_2d_1)\}$. If we remove any control communication pair from $FinalCom$ we would not be able to solve the control problem.

The final version of G^{com}, after adding the communication events associated with the elements of $FinalCom$ and $\mathcal{X}^{compat}$, is shown in figure 4.20. The communication protocol for each agent is illustrated in figure 4.21.

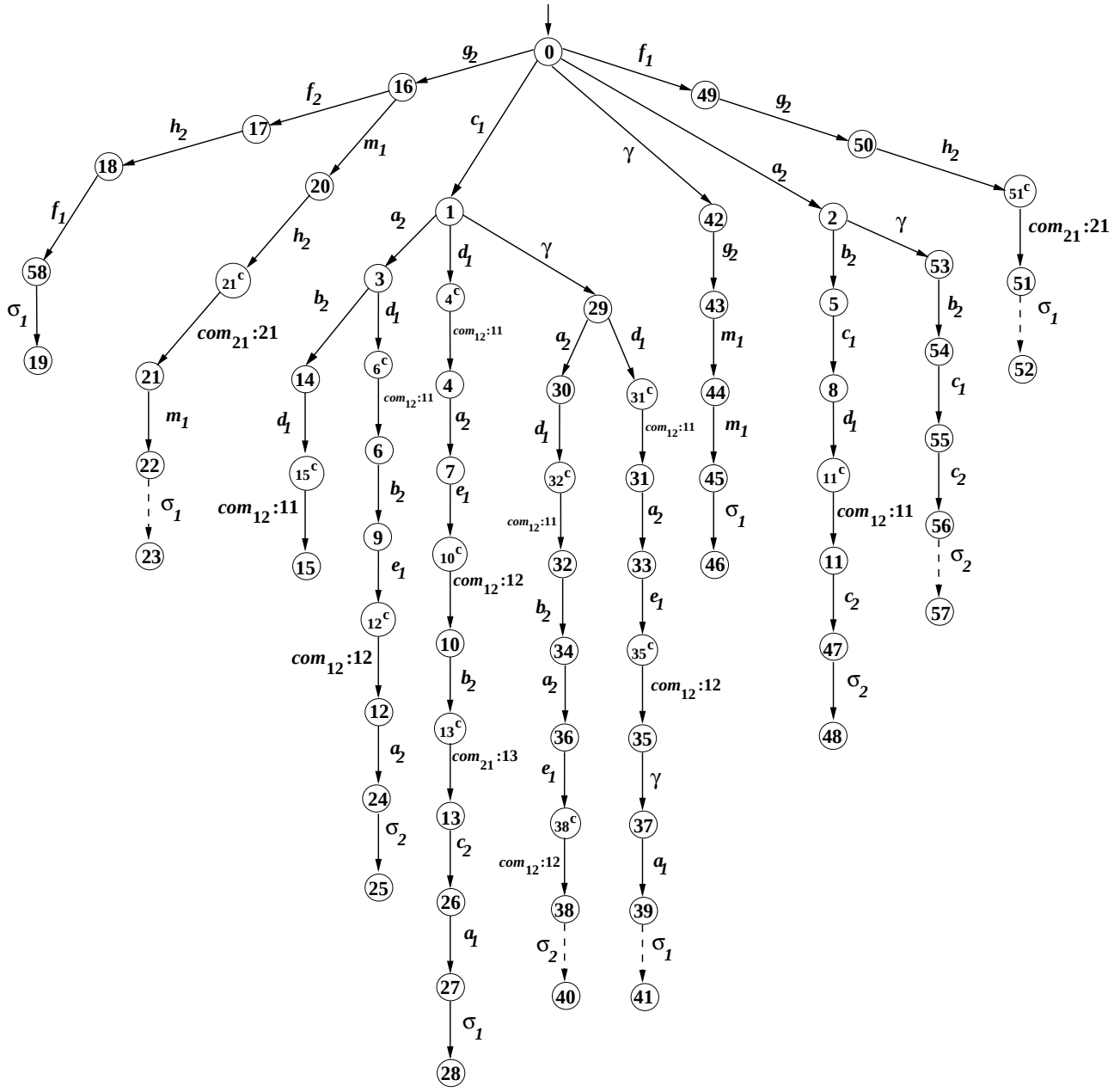

Figure 4.20: G^{com}, from G in figure 4.18 after completing Algorithms 4.1 and 4.2.

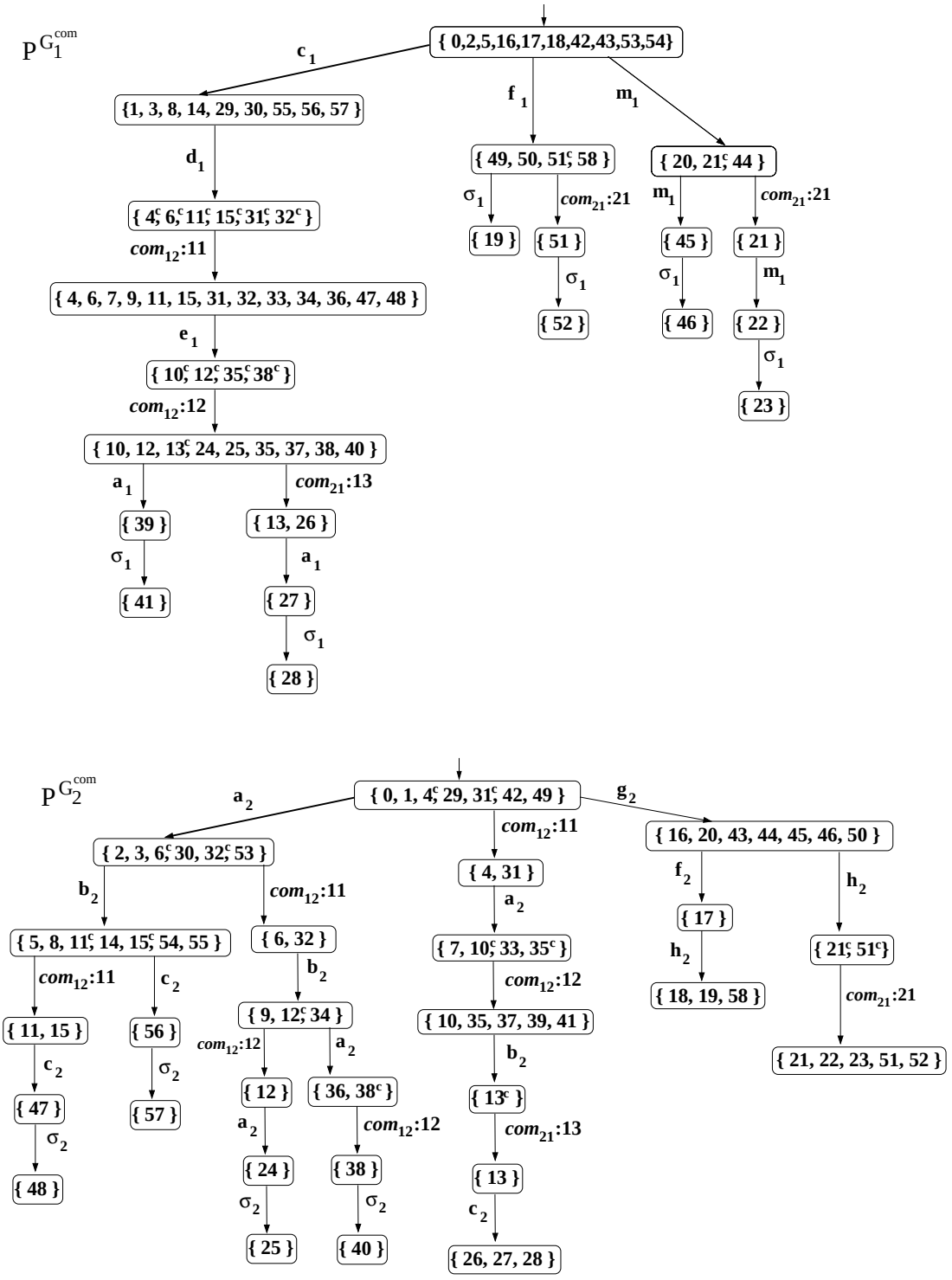

Figure 4.21: The projection automaton for agent 1 (the top of the figure) and agent 2 (bottom of the figure) for G^{com} in figure 4.20.

Chapter 5

Conclusions and Future Work

5.1 General Conclusions

Several general conclusions can be drawn from this work. The results from chapter 3 show that it is possible to describe decentralized discrete-event control problems using knowledge theory. It is feasible to ascribe knowledge to a supervisor or controller, reason about what an agent needs to know to solve the control problem and find a control solution. We also used our knowledge models 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 the correct control decisions.

We use the underlying structure of our knowledge model to locate places for agents to communicate. More communication injects knowledge into the system, which allows supervisors to solve a larger class of decentralized control problems.

We considered that when an individual supervisor cannot make a correct control decision it might be possible for the group of supervisors to pool their collective information to solve the problem. Our strategy of distributed observability did not adequately capture the idea of eliminating the deficit of knowledge for the knowledge-deficient supervisors. To make the correct control decision, a supervisor must be able to distinguish between a circumstance when an event must be disabled and one where the event is enabled. Waiting for this information until just prior to making the control decision can lead to an incorrect control solution, as was suggested in section 3.4.1.

We are, subject to certain assumptions, able to identify places in the knowledge model where one supervisor provides the other with enough information to solve the control problem. We find these places based on an understanding of the underlying structure of the plant language. As we discuss in the next section, translating this strategy into reasoning about places to communicate based strictly on what each supervisor knows remains a difficult problem.

5.2 Future Work

In chapter 3 we described how to translate a DES plant into a knowledge model. If the knowledge model does not satisfy Kripke-observability, then in chapter 4 we described how to update the plant with communication events. The new plant is subsequently translated into a knowledge domain, and we showed that a control solution is generated.

What knowledge could possibly have been involved in determining a place for agents to communicate information to solve the control problem? We would like agents to choose places to communicate based on reasoning about the knowledge of other agents. The knowledge model of section 4.5 is updated in this section.

In chapter 4, we assumed that we could identify a place where an agent does not know whether it is along a "bad" path or a "good" path but where it would know if information were pooled. It seems reasonable to assume that we are able to find such a place because the other agent "knew" whether the system was along a bad path (or equivalently, along a good path). Therefore, an overall solution to the control problem would exist if the group of agents had enough information to determine whether the system was about to generate an illegal or a legal sequence.

We extend our knowledge model from chapter 3 so that agents reason about the lack of knowledge other agents have regarding the control problem. An agent that

knows another agent does not have enough information to make the correct control decision will communicate the missing knowledge.

The set of primitive propositions contains the propositions $\sigma_G, \sigma_E \in \mathsf{R}_\Sigma$. To deal with the idea that a state is either "good" or "bad" with respect to an agent's view of a particular sequence in $L(G)$, we introduce new propositions (two for each event $\sigma \in \Sigma$) for each agent $i \in \mathsf{G}$: $goodstate_i(\sigma), badstate_i(\sigma)$.

$$\pi^{DES'}(w)(goodstate_i(\sigma)) := \begin{cases} \textbf{true} & \text{if } \exists u', v', t \in \Sigma^* \text{ such that} \\ & \delta^G(u', q_0^G) = w_e, \\ & \delta^G(v', w_e)!, \text{ and } P_i(u'v') = P_i(t) \text{ and} \\ & u'v'\sigma, t\sigma \in L(G) \text{ and } u'v'\sigma \in L(E), \\ & t\sigma \notin L(E) \\ \textbf{false} & \text{otherwise.} \end{cases}$$

Suppose w represents a state that occurs along a particular sequence in $L(G)$ that leads to another state represented by $\tilde{w}$ such that σ is legal and controllable by agent i. We denote w as "good" with respect to agent i by setting $goodstate_i(\sigma) = \textbf{true}$ if at $\tilde{w}$, σ is legal.

$$\pi^{DES'}(w)(badstate_i(\sigma)) := \begin{cases} \textbf{true} & \text{if } \exists u, v, t' \in \Sigma^* \text{ such that} \\ & \delta^G(u, q_0^G) = w_e, \\ & \delta^G(v, w_e)!, \text{ and } P_i(uv) = P_i(t') \text{ and} \\ & uv\sigma, t'\sigma \in L(G) \text{ and } uv\sigma \notin L(E), \\ & t'\sigma \in L(E) \\ \textbf{false} & \text{otherwise.} \end{cases}$$

Suppose w represents a state that occurs along a sequence that leads to another state represented by $\tilde{w}$ where σ is illegal and controllable by agent i. We want to denote w as a "bad" state if at $\tilde{w}$, agent i does not know whether σ is legal.

An agent i will communicate its local state to agent j if agent i is not confused about whether the plant is in a good or bad state (for a particular σ controlled by

agent j) and it knows that agent j is confused:

$$\begin{aligned}(\mathcal{I}^{DES'}, w) \quad \models \quad & (K_i(goodstate_j(\sigma) \vee \neg badstate_j(\sigma)) \wedge \\ & K_i \neg K_j(goodstate_j(\sigma) \vee \neg badstate_j(\sigma))) \vee \\ & (K_i(\neg goodstate_j(\sigma) \vee badstate_j(\sigma)) \wedge \\ & K_i \neg K_j(\neg goodstate_j(\sigma) \vee badstate_j(\sigma))).\end{aligned}$$

For a given σ and a state q along a sequence t such that $t\sigma \notin L(G)$ we consider that states along t are neither "good" nor "bad". Therefore, at the global states corresponding to the states along t, from the above definition of $\pi^{DES'}$ the following truth assignments hold at such an w: $\pi^{DES'}(w)(goodstate_i(\sigma)) = \textbf{false}$ and similarly $\pi^{DES'}(w)(badstate_i(\sigma)) = \textbf{false}$. This is why, in checking the knowledge of agent i at w we do not require the agent to know that w is definitely a good state. Rather, we want an agent to know that a state is either "good" or that a state is "not bad".

An agent i communicates if it can tell the difference between the good and bad states for agent j at w and agent i knows that agent j cannot determine if the system is at a good or bad set of states that lead to places in the interpreted system where agent j must have knowledge to make the correct control decision.

We conjecture that a control solution exists if the system satisfies distributed observability in the following sense: before communication, the knowledge of the group must be enough to determine that an agent making a control decision is along a good path or a bad path; and after communication there must exist an agent that knows to make the correct control decision.

One of the problems with our modal logic approach is that we introduce an overwhelming number of propositions to translate the intuition "agent i knows it is along a path where it must disable something". Instead, one may want to incorporate temporal operators and describe the idea that agent j will "eventually" have to communicate

along a given sequence of worlds so that agent i will know to disable σ. Or it would be interesting to determine if we could describe communication as "eventually" being necessary to make a control decision. If no communication is required along a given sequence of worlds, we would describe the notion of agent i "always" knowing to disable σ. In such a scenario perhaps instead of communicating their local states, agents build up more complicated formulas (such as "agent i knows that agent j does not know p") to share.

We have implemented our communication strategy using the programming language C. Future work on our existing strategy for communicating in decentralized control problems includes relaxing our assumption that the agents do not jointly control events. Can we now characterize a place for agents to communicate information so that at least one agent can make the correct control decision? In addition, we want to extend the model for communication to systems with more than two agents.

In some situations, it is not possible to synthesize the complete legal language. What we seek is the maximal (since there is no maximum) subset of $L(E)$ such that $L(E)$ can be synthesized. Constructing such a subset for decentralized control is a difficult problem. It would also be of interest to use knowledge to identify a controllable and co-observable subset of legal behaviour. Knowledge could play a significant role in this class of problems: we could "roll back" from the global states where Kripke-observability fails and identify the "last" place in the system where we did not know to make the correct control decision.

In real distributed systems, dealing with delays in communication (latency) is a significant problem. Considering latency in our model would further complicate the relationship between communication and control. To address more realistic problems of distributed systems, such as latency, would require a model that allows for delays in communication and allows for communication redundancy in the event that a message from an agent does not get through in time.

5.3 Summary

We have presented a novel setting in which to reason about the knowledge that decentralized discrete-event supervisors need to solve control problems. In addition, we have provided a strategy for introducing communication into the problem-solving process. It is our belief that formalizing the knowledge of decentralized communicating supervisors will continue to play a significant role in identifying new strategies to address control issues in distributed systems.

Appendix A

This appendix contains several tables of notation. Table A.1 contains notation that is standard in the DES literature (with the exception of the notation for the monitoring and projection automata). This notation is introduced and defined in chapter 2, but appears in other parts of the dissertation.

Table A.2 contains notation that appears in the knowledge models presented in [12]. Again, the concepts are consistent with those in the knowledge logic literature, except that we have chosen to identify global states with a w.

Table A.3 contains notation that we use to describe both our sequence-based and state-based knowledge models for DES. Most of this notation is presented in chapter 3 and chapter 4.

Table A.4 contains notation that we use to describe decentralized DES with communication. These concepts are defined throughout chapter 4.

	Automata $G = (Q^G, \Sigma, \delta^G, q_0^G)$
G	The automaton representing the plant
Q^G	The set of states in the plant
Σ	The alphabet or set of events defined for the plant
σ	An event of the alphabet
δ^G	The transition function for the plant
q_0^G	The initial state of the plant
E	The legal automaton
A	The monitoring automaton
$\delta(\sigma, q)!$	A transition of σ from state q is defined
a, c, ℓ, q, x	States of an automaton
	Formal Languages
ε	The empty string
Σ^*	The set of all finite sequences over Σ plus ε
$L(G)$	The closed behaviour of G (also the language generated by G)
$\overline{L}$	The prefix-closure of a language L
b, d, t, u, v	sequences of a language
	Supervisory Control
$\mathcal{S}$	A centralized supervisor
$\mathcal{S}_i$	The i^{th} decentralized supervisor
P	A canonical projection operator
P_i	A canonical projection operator with respect to $\mathcal{S}_i$
P^{G_i}	A projection automaton of G for $\mathcal{S}_i$
Σ_{uc},	The set of uncontrollable events
Σ_c,	The set of controllable events
$\Sigma_{i,c}$	The set of events controllable by $\mathcal{S}_i$
Σ_o	The set of observable events
$\Sigma_{i,o}$	The set of events observable by $\mathcal{S}_i$

Table A.1: Discrete-Event Systems Notation

M	A Kripke structure
K	A modal operator for knowledge
D	A modal operator for distributed knowledge
w	A global state or possible world
w_e	The state of the environment
w_i	The local state of agent i
π	An interpretation function
G	Group of agents for the knowledge model
$\mathcal{I}$	An interpreted system
p, ϕ	A primitive proposition
Φ	The set of primitive propositions

Table A.2: Knowledge Model Notation

G_σ	The group of agents that can control event σ
$\mathcal{I}^{DES}$	A sequence-based interpreted system
$\mathcal{I}^{DES'}$	A state-based interpreted system
$\mathcal{I}^{cDES'}$	A state-based interpreted system for communicating DES
σ_G	A primitive proposition for event σ in G
σ_E	A primitive proposition for event σ in E
$\Phi^{DES'}, \Phi^{cDES'}$	Sets of primitive propositions
$\pi^{DES}, \pi^{DES'}, \pi^{cDES'}$	Interpretation functions
$\mathcal{KP}$	A knowledge protocol

Table A.3: Knowledge Model for DES Notation

G^{com}	A plant that contains communication events
$Q^{G^{com}}$	The set of states for G^{com}
$\delta^{G^{com}}$	The transition function for G^{com}
$q_0^{G^{com}}$	The initial state of G^{com}
Σ^{com}	The set of communication events; together with Σ, this is the alphabet for G^{com}
com_{ij}:q	An event in the alphabet of G^{com} where agent i communicates its local view of state q to agent j
Σ_{ij}^{com}	A subset of Σ^{com} that contains communication events where agent i communicates to agent j
(q, t)	A control communication pair
(x, v)	A compatible communication pair for (q, t)
(u, u')	A maximal-P pair
$\mathcal{C}$	The set of control communication pairs
$\mathcal{C}_{ij}$	The set of control communication pairs where agent i communicates to agent j
$\mathcal{C}^{compat}$	The set of compatible communication pairs
$\mathcal{C}_{ij}^{compat}$	The set of compatible communication pairs where agent i communicates to agent j
D, $\hat{D}$	Matrix representation of dependency graphs
$\hat{P}$, $\hat{P}_i$	A canonical projection that "erases" communication events
$P^{\mathbf{c}}, P_i^{\mathbf{c}}$	A canonical projection that "erases" unobservable events
$t^{\mathbf{c}}$	A version of $t \in L(G)$ as it appears in $L(G^{com})$ after Procedure 4.1
$\hat{t}^{\mathbf{c}}$	A version of $t \in L(G)$ as it appears in $L(G^{com})$ after Procedure 4.2
$\tilde{t}^{\mathbf{c}}$	A version of $t \in L(G)$ as it appears in $L(G^{com})$ after Procedure 4.3
$\mathcal{XV}$	The set of all compatible communication pairs for elements in $\mathcal{C}$
$\mathcal{XV}_{4.2}$	The set of compatible communication pairs used in Procedure 4.2
$\mathcal{XV}_{4.3}$	The set of compatible communication pairs used in Procedure 4.3
$FinalCom$	A set of control communication pairs and some of their compatible communication pairs: output from Algorithm 4.2
$\mathcal{X}^{compat}$	The set of compatible communication pairs for the elements of $FinalCom$

Table A.4: Communication and DES Notation

Bibliography

[1] R. J. Aumann. Agreeing to disagree. *Annals of Statistics*, 4(6):1236–1239, 1976.

[2] M. Barbeau, F. Kabanza, and R. St-Denis. Supervisory control synthesis from metric temporal logic specifications. In *Proceedings of the Thirty-third Annual Allerton Conference on Communication, Control and Computing*, pages 96–105, 1995.

[3] 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.

[4] R. I. Brafman and Y. Shoham. Knowledge considerations in robotics and distribution of robotic tasks. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1995.

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

[6] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, 1990.

[7] C. G. Cassandras, S. Lafortune, and G. J. Olsder. Introduction to the modelling, control and optimization of discrete-event systems. In A. Isidori, editor, *Trends in Control: A European Perspective*, pages 217–291. Springer-Verlag, 1995.

[8] 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.

[9] 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.

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

[11] J. M. Davoren. On hybrid systems and the modal μ-calculus. In P. Antsaklis, W. Kohn, M. D. Lemmon, A. Nerode, and S. Sastry, editors, *Hybrid Systems V*, Lecture Notes in Computer Science. Springer-Verlag, 1999. To appear.

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

[13] J. Glasgow, G. MacEwan, and P. Panangaden. A logic for reasoning about security. *ACM Trans. Comp. Systems.*, 10(3):226–264, 1992.

[14] V. Hadzilacos. A knowledge-theoretic analysis of atomic commitment protocols. In *Proceedings of the 6th ACM Symposium on Principles of Database Systems*, pages 129–134, 1987.

[15] J. Y. Halpern and R. Fagin. Modelling knowledge and action in distributed systems. *Distributed Computing*, 3:159–177, 1989.

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

[17] J. Y. Halpern and L. D. Zuck. A little knowledge goes a long way: knowledge-based derivations and correctness proofs for a family of protocols. *Journal of the ACM*, 39(3), 1992.

[18] L. Holloway, B. Krough, and A. Giua. A survey of Petri net methods for controlled discrete event systems. *Journal of Discrete Event Dynamic Systems*, 6(2):151–190, 1997.

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

[20] 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.

[21] Y. Li and W. M. Wonham. Control of vector discrete-event systems I—the base model. *IEEE Transactions on Automatic Control*, 38(8):1214, 1993.

[22] Y. Li and W. M. Wonham. Control of vector discrete-event systems II—controller synthesis. *IEEE Transactions on Automatic Control*, 39(3):512, 1994.

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

[24] F. Lin and W. M. Wonham. Decentralized control and coordination of discrete-event systems with partial observation. *IEEE Transactions on Automatic Control*, 35(12):1330–1337, 1990.

[25] B. L. Lipman. An axiomatic approach to the logical omniscience problem. In R. Fagin, editor, *Theoretical Aspects of Reasoning about Knowledge: Proceedings of the Fifth Conference*, pages 182–196. Morgan Kaufmann, 1994.

[26] J. S. Ostroff and W. M. Wonham. A temporal logic approach to real time control. In *Proceedings of the 24th Conference on Decision and Control*, pages 656–657, 1985.

[27] A. Radiya and R. Sargent. A logic-based foundation of discrete event modeling and simulation. *ACM Transactions on Modeling and Computer Simulation*, 4(1):3–51, 1994.

[28] 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.

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

[30] S. L. Ricker and K. Rudie. Know means no: Incorporating knowledge into decentralized discrete-event control. In *Proceedings of the 1997 American Control Conference*, pages 2348–2353, 1997.

[31] 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.

[32] 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.

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

[34] K. G. Rudie. Software for the control of discrete event systems: A complexity study. Master's thesis, Department of Electrical Engineering, University of Toronto, 1988.

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

[36] S. Takai and S. Kodama. Decentralized state feedback control of discrete event systems. *Systems and Control Letters*, 22(5):369–375, 1994.

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

[38] J. G. Thistle and W. M. Wonham. Control problems in a temporal logic framework. *International Journal of Control*, 44:943–976, 1986.

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

[40] Y. Willner and M. Heymann. Supervisory control of concurrent discrete-event systems. *International Journal of Control*, 54(5):1143–1169, 1991.

[41] 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.