QUANTITATIVELY-OPTIMAL COMMUNICATION PROTOCOLS FOR DECENTRALIZED SUPERVISORY CONTROL OF DISCRETE-EVENT SYSTEMS

Md Waselul Haque Sadid

A thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy

Concordia University

Montréal, Québec, Canada

April 2014

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Mr. Md Waselul Haque Sadid**

Entitled: **Quantitatively-Optimal Communication Protocols for Decentralized Supervisory Control of Discrete-Event Systems**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Dr. X, Chair

_____ Dr. John Thistle, External Examiner

_____ Dr. Shahin Hashtrudi Zad, Thesis Supervisor

_____ Dr. Laurie Ricker, Thesis Supervisor

_____ Dr. Khashayar Khorasani, Examiner

_____ Dr. Amir Aghdam, Examiner

_____ Dr. Mingyuan Chen, Examiner

Approved _____

Chair of Department or Graduate Program Director _____

20 _____ _____

Dr. C. Trueman, Dean

Faculty of Engineering and Computer Science

Abstract

Quantitatively-Optimal Communication Protocols for Decentralized Supervisory Control of Discrete-Event Systems

Md Waselul Haque Sadid, Ph.D.

Concordia University, 2014

In this thesis, decentralized supervisory control problems which cannot be solved without some communication among the controllers are studied. Recent work has focused on finding minimal communication sets (events or state information) required to satisfy the specifications. A quantitative analysis for the decentralized supervisory control and communication problem is pursued through which an optimal communication strategy is obtained. Finding an optimal strategy for a controller in the decentralized control setting is challenging because the best strategy depends on the choices of other controllers, all of whom are also trying to optimize their own strategies. A locally-optimal strategy is one that minimizes the cost of the communication protocol for each controller. Two important solution concepts in game theory, namely Nash equilibrium and Pareto optimality, are used to analyze optimal interactions in multi-agent systems. These concepts are adapted for the decentralized supervisory control and communication problem.

A communication protocol may help to realize the exact control solution in decentralized supervisory control problem; however, the cost may be high. In certain circumstances, it can be advantageous, from a cost perspective, to reduce communication, but incur a penalty for synthesizing an approximate control solution. An exploration of the trade-off between the cost and accuracy of a decentralized discrete-event control solution with synchronously communicating controllers in a multi-objective optimization problem is presented. A widely-used evolutionary algorithm (NSGA-II) is adapted to examine the set of Pareto-optimal solutions that arise for this family of decentralized discrete-event systems (DES).

The decentralized control problem is synthesized first by considering synchronous communication among the controllers. In practice, there are non-negligible delays in communication channels which lead to undesirable effects on controller decisions. Recent work on modeling communication delay between controllers only considers the case when all observations are communicated. When this condition is relaxed, it may still be possible to formulate communicating decentralized controllers that can solve the control problem with reduced communications. Instead of synthesizing reduced communication protocols under bounded delay, a procedure is developed for testing protocols designed for synchronous communications (where not all observations are communicated) for their robustness under conditions when only an upper bound for channel delay is known.

Finally a decentralized discrete-event control problem is defined in timed DES (TDES) with known upper-bound for communication delay. It is shown that the TDES control problem with bounded delay communication can be converted to an equivalent problem with no delay in communication. The latter problem can be solved using the algorithms proposed for untimed DES with synchronous communication.

Dedicated

To

My Parents

Acknowledgments

I would like to express my deep gratitude to my academic supervisors, Dr. L. Ricker and Dr. S. Hashtrudi Zad for their guidance, support and enthusiasm throughout my research work. I am greatly indebted to them for their erudite supervision, constructive criticism and invaluable advice during the time of my research. Their suggestions, comments and encouragement at all stages of my work have made it possible to complete this research. I would also like to remember late Dr. P. Gohari whose initial guidance and support built my research interest in this area.

I would like to express my thanks to all of the committee members for their valuable feedback on my thesis during my Ph.D. proposal and seminar. Their comments, questions and suggestions have been very useful to improve my research work. I also like to extend my special thanks to the external examiner for his valuable comments and suggestions.

I would like to convey my sincere thanks to my colleagues of Rajshahi University of Engineering and Technology for their cordial cooperation throughout this work. I also like to thank Dr. Mahmudul Hasan to help for implementing NSGA-II algorithm.

I cannot thank my parents, parents-in-law, brothers and sisters enough for their moral support and inspiration, which always act as a driving force behind me. Lastly and most importantly, I would like to thank my wife for her support, encouragement and care during all these years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Discrete-event systems (DES) are used to model systems whose dynamics can be described by transitions among a set of finite states. These models, for instance, can be used in the analysis and design of the sequences of (high-level) commands in many control systems. In the supervisory *control* of DES, the objective is to design a *controller* to restrict the system behaviour to a set of *design specifications*. The basic architecture of a DES is shown in Figure 1.1.

In decentralized DES control problems, a set of controllers, each of which only *controls* and *observes* part of the system, must collectively achieve the given specification. In the basic problem of decentralized DES control, no communication among the controllers is assumed. We consider the class of decentralized DES control problems in which the information about the system is distributed among the controllers in such a way that the problem cannot be solved without some degree of *communication* among them. The communications take place at a cost and we would like to characterize the minimal cost communication.

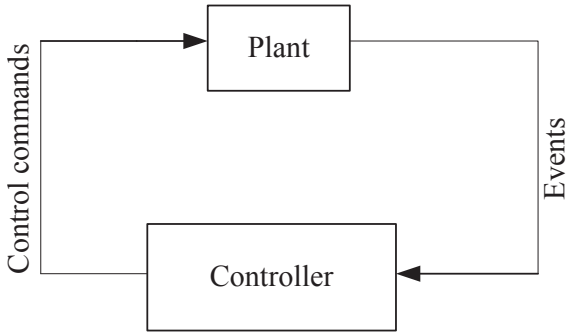


Figure 1.1: Basic architecture of a DES.

1.1 Motivation

In the synthesis of solutions for the decentralized control and communication problem, we want to compare different control and communication options and choose one with a *minimal* cost. Existing approaches do not give a *total ordering* on the controlled behaviour due to the lack of measure on the system states or events. So we are interested in a *quantitative analysis* of decentralized DES control and communication problems to permit us to compare different communication protocols that allow the controllers to achieve the control objectives.

The decentralized supervisory control and communication problem deals with multiple controllers that interact with each other as multi-agent systems. There has been increasing interest in the use of game theory for quantitative analysis of multi-agent systems. Many studies in *multi-agent* systems have analyzed multi-agent interactions, especially those involving negotiation and co-ordination. In most multi-agent systems, the overall outcome depends critically on the choices made by all agents. To optimize the outcome, an agent takes into account the decisions that other agents take and assumes they act so as to optimize their own outcome. Game theory is a way of formalizing and analyzing such concerns. Hence, the concept of game theory can be adapted for this class of control problems.

In the synthesis of *synchronous* or *zero-delay* communication protocols, it is assumed that communication occurs with zero-delay among the controllers. But, in practice, non-negligible *delays* occur in the communication channel may adversely affect the local controllers' decisions. Hence, we are interested in extending our analysis to the problem of decentralized supervisory control under such communication delays.

1.2 Problem Statement

In the supervisory control of DES, the goal is to synthesize a control policy for an uncontrolled system that must satisfy a given specification. The decentralized DES control and communication problem is concerned with cases in which the control objectives are not satisfied in the absence of communication among the controllers. In these problems, the optimality of communication is a major issue [4, 36, 55, 56]. Most of the existing approaches do not use a quantitative measure, rather, they use a *logical notion* of optimality. Without a quantitative metric, it is not always possible to compare different protocols.

Sometimes communication that solves the control problem may be prohibitively expensive. Therefore, we may have to sacrifice the exact control solution for a cheaper communication policy. Hence, we need to optimize the cost of a communication protocol and that of the control objective simultaneously, giving rise to a *multi-objective optimization* problem [50].

In real-life problems, the communication exchange among the controllers is affected by communication delays. Hence, we may verify how resilient a communication protocol designed based on zero-delay assumption is under the condition of such delay. Finally, we are interested in extending the decentralized control and communication problem to timed DES (TDES) models by including a timing feature. This would

result in a direct procedure for designing communication protocols that take communication delay into account at the design stage. A TDES differs from untimed DES by including the *time bounds* of event occurrence, which are defined by tick events synchronized by a global clock. Hence, optimal communication policy in TDES model can be synthesized using the approach developed for the untimed model.

1.3 Objectives

In decentralized control and communication problems, we seek communication protocols that not only allow for a solution to the control problem, but also have minimal cost. We use quantitative analysis using the cost functions adapted from those used in centralized DES [24, 49].

- We are interested in exploring two key concepts of distributed optimality in game theory, *Nash equilibrium* and *Pareto optimality*. These will give us different ways of evaluating solutions in decentralized supervisory control problems with communication. Specifically, they provide a mathematical means of examining the interactions of independent controllers in the decentralized case. We design the controllers to communicate and cooperate with each other to solve a control problem and synthesize synchronous communication protocols for this class of problems. An algorithm for calculating Nash equilibrium of multi-agent systems has been adapted for the quantitative analysis of communication protocols in these problems. We also consider that a protocol must be *coherent*. A communication protocol is coherent if, when a controller communicates after its partial observation and the received messages from other controllers through a sequence s, it also communicates after observing all sequences that are considered to be observationally equivalent to s.

- When communication occurs with a large cost, there may be a cost advantage to realizing only part of the specification, instead of realizing the entire specification with a costly communication protocol. In that case, we incur a *penalty* for synthesizing an approximate control solution. To that end, we want to investigate the *trade-off* between the cost of an exact control solution achieved with communication and that of an approximate solution where penalties are assessed for settling for sublanguage of the specification, with a cheaper communication policy. We are interested in a widely-used concept of Pareto-optimality for the multi-objective optimization problem in decentralized DES.

- We will extend our work to consider the robustness of zero-delay synchronous communication when operated under *bounded* delay. Starting with a communication protocol that solves the control problem with zero delay, we want to verify if this protocol is robust enough to withstand timing effects associated with a more realistic communication network and the timing characteristics of the plant.

- Finally, we will examine the decentralized control problem in TDES with a bounded but unknown delay in the communication channel. Since synchronous communication protocols are synthesized for decentralized untimed DES control problem, communication protocols can be synthesized for TDES control problem with bounded delay communication by converting it to an equivalent problem with zero-delay in the communication, but with time bounds on message sending, assuming that the tick events are observable to all controllers.

1.4 Assumptions and Limitations

We consider that the system we deal with satisfies the notion of observability: because of this, we can always find a communication protocol to solve the control problem, if (in the worst case) all observations are communicated to all controllers. We assume that there is no communication loss in the network, so that no message sent by a controller is lost in the communication channel. The communication happens in first-in-first-out basis. Initially, we also assume that communication occurs without delay in the decentralized DES control and communication problem. This assumption is relaxed when we deal with the control problem with bounded delay communication. We assume that the set of controllers has perfect information about the system model that they must control. Their local view is built from a known model of the complete system. Uncertainty in the system model is beyond the scope of the thesis.

1.5 Literature Review

This literature review includes works on quantitative optimal centralized control of DES, minimal communication and communication delay in decentralized DES, multi-objective optimization, and TDES.

1.5.1 Quantitative Optimal Control in DES

Quantitative optimal control has been examined from the perspective of centralized discrete-event control using various cost models [8, 18, 24, 49]. Costs are assigned to control decisions and the goal is to synthesize a controller with an overall minimal cost with respect to the control strategy. An alternate technique for measuring the cost of centralized control was introduced in [34]. Although not developed with control theory applications in mind, a new class of quantitative languages (based on weighted

automata) has also been proposed in [7]. In most cases, two cost functions are defined that are used to realize the quantitative performance of a language: the *event cost* function and the *control cost* function [24, 49]. The event cost is associated with an event executed by the system, and the control cost is associated with the disabling of an event by the controller. These functions are used to induce a cost on the sequences generated by the system. A sequence cost is simply the summation of costs of those events belong to the sequence. The control cost is associated with the events that are disabled by a controller through a sequence. The event cost can only be reduced by increasing the control cost. Thus there is a trade-off between the control cost and control objectives which sets up an optimization problem.

A centralized partial observation problem is considered in [52] where the authors propose an algorithm to approximate an optimal observation policy. A cost is incurred when identifying an occurrence of an observable event, and the objective is to minimize the total cost incurred by the controller.

In problems that finding an exact solution results in exponential time complexity, language measure has significant role to play in the search for an approximate solution. A signed real measure of regular languages is introduced in [8, 34]. This measure formalizes the synthesis of the supervisory control systems in finite state automata (which recognize regular languages). This measure is considered for quantitative evaluation of every sequence of the system. It makes an exact quantitative comparison of the controlled system under different controllers. The cost of disabling events can also be considered for the performance measure of supervised sublanguages. A controller can easily be obtained by considering the event disabling cost but with slightly lower performance [33]. The optimal control problem is to find a sublanguage that has the minimum worst-case behaviour among all sublanguages and is finite.

1.5.2 Communication in Decentralized DES

The decentralized control of DES was introduced in [9, 41]. In the absence of communication, an exact control solution may not be achievable. Communication was first incorporated into the model of decentralized DES content in [57]. There the focus is on the introduction of a minimal amount of communication so that the exact control solution is achieved. A basic architecture of communicating controllers in decentralized DES is shown in Figure 1.2.

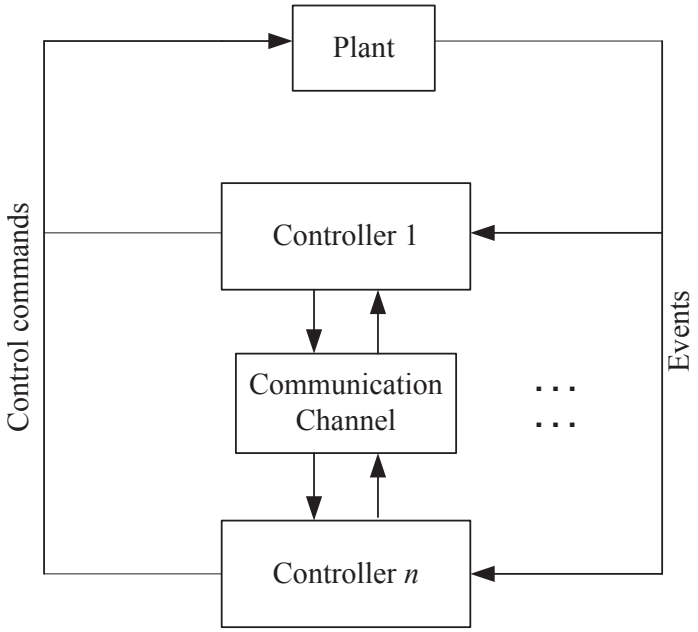


Figure 1.2: Communication between controllers in decentralized DES architecture.

Communication in a decentralized supervisory control system has been considered with a variety of models [4, 23, 35, 36, 56, 57]. If an event takes the system outside of the specification and there is at least one controller that can disable that event, then the system is *co-observable* and we can synthesize decentralized controllers that correctly reach the control objective. When the system is not co-observable, i.e., no controllers can distinguish a behaviour in the specification from behaviours outside of the specification, it may still be possible to find a correct control decision if the

controllers communicate with each other. A controller can communicate either state estimates or an observation of an event to other controllers. In [4, 38, 57], observable events are not directly communicated by a controller, rather, the state estimate based on the sender's local observation is communicated. In [4], the authors introduce a strategy for synthesizing a decentralized communication policy where communication is postponed as long as possible along a sequence that is about to leave the specification. In the context of this model, the resulting communication protocol is deemed optimal. Also communication is performed via a broadcast. In [23], a slightly different strategy is presented: the protocol design for non co-observable desired behaviours is reduced to the synthesis of communicating decentralized controllers in the presence of ideal communication channels.

The equilibrium of asymmetric communication policies for decentralized diagnosis of discrete-event systems was examined in [5]. Nash equilibrium for communication between two diagnosers was computed assuming a *uniform* cost for communication. The asymmetry arises from imposition of the constraint that only one diagnoser had the ability to communicate. In contrast, the format for communication in [4] allows any controller to initiate communication.

1.5.3 Optimal Synchronous Communication

The prevailing definition of minimal communication is *set-theoretic*: remove any one of the communications from the protocol and either the control problem cannot be solved or the notion of coherency is violated. Although there is no control or diagnosis objective in [40], the authors describe a *locally-optimal* communication protocol for a two-agent system based on a set-theoretic definition of minimality. Here each agent needs to distinguish each state from every other state. The computational complexity of the algorithm is exponential in time, but an approximate solution can be developed

in polynomial time. A generalization of the approach is described for decentralized control problems in [56]. Both approaches are limited to acyclic systems. The issue of optimality arises when cycles are involved.

A *greedy* algorithm is developed in [35] for minimal communication. In [36], finding a *globally-optimal* communication policy is reduced to an optimization problem over a set of *Markov chains*. The difference in the definition of minimality provides the motivation for finding a quantitative measure. Quantitative measures are crucial in order to compare communication protocols. Two significant aspects of this finite-state model are the identification of violations of co-observability and the presence, by construction, of potential communications. Such violations are called *illegal* configurations and a communication protocol is synthesized by selecting communication transitions that allow the system to avoid reaching an illegal configuration. The protocol(s) with a minimal cost is then easily determined.

1.5.4 Communication with Delay

Decentralized control problem with delayed communication has been studied in [16, 51,53]. The delay in the communication is defined as the number of events that occur in the system before the reception of the message by other controllers. Delays can be either bounded or unbounded. Since the length of the delay is unpredictable in the real-life problems, *k-bounded* delay communication is a natural model for communication with delay. When the delay is bounded by a given constant k, the system can execute at most k events between the transmission and the reception of a message. Every problem solved by decentralized controllers with k-bounded delay communication can also be solved by $(k-1)$-bounded delay communication, but the reverse is not true [54]. In the unbounded delay communication, any number of events can

be executed between the transmission of a message and its reception. When communication occurs with delay, either bounded or unbounded, existing approaches of communication protocols only consider the case when *all* observations are communicated among decentralized controllers [16, 54], diagnosers [30] or prognosicators [51]. Checking the existence of controllers for unbounded delay communication is *undecidable*, even when all observations are communicated, but a related problem with bounded delay communication is *decidable* [54].

A distributed diagnosis problem under bounded delay communication has been formulated in [30] where a failure can be diagnosed by a local diagnoser based on its own observation as well as the communication received from other diagnosers with k-bounded delay. In a similar prognosis problem [51], local prognosicators exchange their observations over a bounded delay communication channel to capture the condition under which any failure can be predicted by some local prognosicators prior to its occurrence. In these methods, it is also assumed that each prognosicator transmits all of its observations to other prognosicators.

In [16], each decentralized controller manipulates a different set of events: *communication events*, which uniquely corresponds to a system event. For each observation of system event, the corresponding communication event (message) is broadcast to all other agents. A network of sequential processes communicating with each other is identified in [22]. They also assume an upper bound k for the delay such that if a process executes k events without receiving any message from the other process, then it never receives any message from that process.

1.5.5 Multi-Objective Optimization

Multi-objective optimization deals with the problems that require the optimization of several possibly competing criteria. In principle, multi-objective optimization is

different from single-objective optimization. One can define and obtain the best solution in the single objective optimization problem, i.e., the globally minimal or maximal depending on the optimization problem [3, 50]. But there may not exist a single best solution in the multi-objective optimization problem. In general, the objectives are in *conflict* in such a way that no single solution optimizes all the objectives simultaneously. Therefore, we are interested in finding a set of solutions with the property that no objectives can be improved without worsening any other objectives. These are optimal in the sense that no one is better than the others considering all objective functions, and they are known as Pareto-optimal solutions or non-dominated solutions [10, 17, 59].

Optimal decentralized control in the absence of communication was studied in [27], using Nash equilibrium as the optimization criterion. The *maximal* solutions of a set of controllers is obtained for decentralized supervisory control from a set-theoretic perspective. The solutions are based on the set of languages of the controlled system.

The notion of fictitious play is employed to find quantitatively decentralized control strategies as an optimization of intruder/detection problem in [52]. The approach will detect the presence of interference which may damage the system, so that it can prevent an intruder from reaching to the *undesirable* states. A terminal cost is assigned in the analysis which guarantees a positive measure for the sequences reaching desirable states, and a negative measure for those ending in the undesirable states. The terminal cost is combined with the event cost in determining the quantitative analysis. As detailed in [12], the execution cost to optimize fault-tolerant systems and the quality of service are taken into account as multi-criteria optimization in synthesizing a discrete controller. We are interested in a class of decentralized control problems, where we want to optimize two objective functions: communication cost

and control cost, introducing a multi-objective optimization problem to decentralized DES. Evolutionary algorithms are well-suited for optimization problems when (i) exhaustive search is computationally prohibitive, and (ii) there are multiple objectives to optimize. So, we examine our multi-objective optimization problem using *evolutionary algorithms* [3].

1.5.6 Evolutionary Algorithms

Evolutionary algorithms are search methods that simulate the process of natural selection using the concept of "survival of the fittest". An initial population of possible solutions are considered, and a measure of their *fitness* determines whether a member of the population will be involved in the formulation of the next generation of the population. Just as in natural adaptation, over a period of many generations, a population of solutions evolve that are "closer" to an optimal solution than their predecessors. Some of the well-known evolutionary algorithms are Pareto-archived evolution strategy (PAES) [17], strength Pareto evolutionary algorithm (SPEA) [59, 61], non-dominated sorting genetic algorithm (NSGA) [10,11]. Most work in the area of evolutionary multi-objective optimization has concentrated on the approximation of the Pareto set. Knowles and Cornes [17] suggested an multi-objective optimization evolutionary algorithm using an evolution strategy in their PAES. An archive of non-dominated solutions is maintained to persist the *diversity*. An *elitist* multi-objective evolutionary algorithm is proposed based on a systematic comparison of different solutions in [42]. Zitzler and Thiele [61] suggested an elitist multi-criteria evolutionary algorithm using the concept of non-domination in their SPEA. A fast elitist multi-objective evolutionary algorithm is developed by modifying NSGA, which outperforms most of the other evolutionary algorithms [10]. A multi-criteria optimization problem has been introduced by [1] to determine simultaneous maxima of a set of real functions

over some domain, where they also use the concept of Pareto optimality. For multi-objective optimization problem in decentralized DES, we use NSGA-II (a modified version of NSGA), which has already proven useful for a diverse range of control problems (e.g., [15, 58]).

1.5.7 Timed DES (TDES)

The supervisory control framework that describes the system behaviour in TDES was developed in [6], and extended to decentralized architectures in [26]. A model allowing *asynchronous* communication between controllers in a TDES is presented in [39], where a timing constraint is incorporated to an event with respect to another event to define how many clock ticks have elapsed between these two events. In [28], the decentralized supervisory control problem in untimed DES is extended to a new structure in TDES. In contrast, we are interested in recasting the decentralized control and communication in synthesizing a controller in decentralized TDES with a bounded but unknown delay.

1.6 Contributions of the Thesis

The contributions of the thesis are as follows:

1. Quantitative analysis in synthesizing an optimal communication protocol for the control and communication problem in decentralized control of DES. In contrast to existing work, the communication protocol is locally-optimal, which minimizes the communication cost for each controller.

    - Synthesizing a communication protocol in decentralized DES is cast as a natural-form game.

- For finding optimal solutions, an existing algorithm calculating Nash equilibrium is used. The algorithm is modified by adding a step to the algorithm to ensure the final solution is observationally-equivalent(see Definition 2.7). To the best of our knowledge, this has not been done before.

2. The trade-off between the cost of exact control solution with costly communication protocol and an approximate solution with cheaper communication protocol is studied, where penalties are assessed for not achieving the exact control solution. This sets up a multi-objective optimization problem.

   - To obtain the optimal solutions (Pareto-front) of the multi-objective optimization problem, a widely-used evolutionary algorithm, NSGA-II, is adapted by adding a step to the algorithm to ensure the final communication protocols and control policies are observationally-equivalent.

3. A method is developed to verify the robustness of a synchronous communication protocol under conditions of (i) fixed delay, (ii) unknown but bounded delay.

   - *Rational transducers* have been designed for propagating the delay of messages.

   - A *product automaton* is introduced to verify the robustness of a synchronous communication protocol under the conditions mentioned above.

4. A control and communication problem in decentralized TDES with unknown but bounded delay communication has been studied.

   - It is shown that the problem can be converted to an equivalent problem with synchronous communication.

- The solution of TDES problem with synchronous communication is obtained using the approach of synthesizing synchronous communication protocol in untimed DES.
- A product automaton has been defined to detect violations of co-observability in TDES.

1.7 Outline of the Thesis

The rest of this thesis is organized as follows. Background definitions and results relevant to this research are described in Chapter 2. Minimal communication in decentralized DES is explored in Chapter 3. Two game theory concepts of Nash equilibrium and Pareto optimality are used to find minimal cost communication protocols. The concept of Pareto optimality has been used for multi-objective optimization problem in decentralized DES, and is presented in Chapter 4. The use of an existing evolutionary algorithm to solve the decentralized DES control and communication problem is discussed and shown that when the problem is solved, a set of Pareto-optimal solutions is obtained. Chapter 5 contains an examination of robustness of synchronous communication protocols under a fixed and bounded but unknown delay. Chapter 6 formalizes a decentralized control problem with a known upper-bound delay communication using TDES models. The TDES control problem with an unknown but bounded delay is converted to an equivalent problem with zero-delay communication. Chapter 7 concludes the thesis with a summary and an outline of future research.

Chapter 2

Background

This chapter introduces the relevant background from the theory of supervisory control of DES, as developed by [31], and the decentralized architecture [41]. It also reviews the concepts of Nash Equilibrium and Pareto optimality from game theory, and multi-objective optimization problems.

2.1 Supervisory Control of DES

We employ the supervisory control framework of [31, 32] which is concerned with the behaviour of a DES requiring control, assuming the desired system behavior (specification) is given as regular language over an alphabet Σ. We denote the behaviors of the DES plant and the design specification by regular languages L and K, respectively. In the theory of supervisory control of DES, a centralized controller keeps the system in K by issuing control directives to prevent the system from performing behaviour in $L \setminus K$, the set of sequences in L that are not in K. The controller issues a control decision (e.g., enable or disable an event) in response to sequence (generated from L) and the control objective is reached when correct pattern of control decisions are issued to keep the system in K.

A finite-state automaton is a five-tuple

$$M_L = (Q, \Sigma, T_L, q_0, Q_m),$$

where Q is a finite set of *states*; Σ is a finite *alphabet* of symbols called events; $T_L \subseteq Q \times \Sigma \times Q$ is the *transition relation*; $q_0 \in Q$ is the *initial state*; and $Q_m \subseteq Q$ is a set of *marked* states. We will write $q_1 \xrightarrow{\sigma} q_2$ if there is a transition labeled σ from state q_1 to state q_2 (i.e., $(q_1, \sigma, q_2) \in T_L$). We can compose transitions and write $q_1 \xrightarrow{\sigma_1\sigma_2\ldots\sigma_m} q_r$ if there is a sequence of events $\sigma_1\sigma_2\ldots\sigma_m \in \Sigma^*$ that begins in state q_1 and ends in state q_r, where Σ^* is called the Kleene closure of Σ. An empty sequence is denoted by ε, where $\varepsilon \in \Sigma^*$.

For all sequences $s, t \in \Sigma^*$, we say that t is a *prefix* of s if $\exists w \in \Sigma^*$ such that $s = tw$. If $L \subseteq \Sigma^*$, then the *prefix-closure* of L, consisting of all prefixes of sequences of L, is defined as $\overline{L} := \{s \in \Sigma^* | \exists s' \in \Sigma^* \text{ such that } ss' \in L\}$. If L is prefix-closed, then $L = \overline{L}$. Unless specifically mentioned, we work exclusively with prefix-closed languages.

We begin with some basic concepts from control theory as applied to finite-state automata. The language generated by an automaton is defined as its closed behaviour, and the marked behaviour is the subset of sequences of the language, which end at the marked states of automaton.

Definition 2.1. *The **closed** behaviour of M_L, denoted by L, is defined as:*

$$L := \{s \in \Sigma^* | (\exists q_1 \in Q) q_0 \xrightarrow{s} q_1\}.$$

Definition 2.2. *The **marked** behaviour of M_L, denoted by L_m, is defined as:*

$$L_m := \{s \in L | (\exists q_1 \in Q) q_0 \xrightarrow{s} q_1 \wedge q_1 \in F\}.$$

The supervisory control problem assumes disjoint partitions of the elements of Σ based on the set of controllable events Σ_c and the set of uncontrollable events Σ_{uc} (i.e., $\Sigma = \Sigma_c \uplus \Sigma_{uc}$). By assumption, the supervisor may prevent only controllable events from occurring (i.e., only controllable events can be disabled), and uncontrolled events are considered to be permanently enabled, as they cannot be prevented from occurring. We want to synthesize a supervisor or *controller* such that the uncontrolled system, under the control decisions of the controller, performs only behaviour in a given *specification* language $K \subseteq L$ (with a corresponding automaton $M_K \sqsubseteq M_L$). Thus a controller, upon detecting that a behaviour in $L \setminus K$ is about to occur, will disable controllable events that take the system from K to $L \setminus K$. A language K is *controllable* if no behaviour in the system exits the specification via an uncontrollable event.

Definition 2.3. *The language K is said to be **controllable** [31, 32] w.r.t. L and Σ_{uc} iff*

$$\overline{K}\Sigma_{uc} \cap L \subseteq \overline{K}.$$

Another aspect of the control problem involves the notion of partial observation. Only a subset of the events Σ is observed by a controller, so Σ can be partitioned into the disjoint sets of observable events Σ_o and unobservable events Σ_{uo} (i.e., $\Sigma = \Sigma_o \uplus \Sigma_{uo}$). A controller's view of the system behaviour is modeled by a *natural projection* $\pi : \Sigma^* \to \Sigma_o^*$. This operator removes those events σ from a sequence in Σ^* that are not found in Σ_o :

$$\pi(\sigma) = \begin{cases} \sigma, & \text{if } \sigma \in \Sigma_o; \\ \varepsilon, & \text{otherwise.} \end{cases}$$

The above definition can be extended to sequences as follows: $\pi(\varepsilon) = \varepsilon$, and $\forall s \in \Sigma^*, \forall \sigma \in \Sigma,\ \pi(s\sigma) = \pi(s)\pi(\sigma)$. The *inverse projection* of π is a mapping $\pi^{-1} : \Sigma_o^* \to 2^{\Sigma^*}$ such that for $s' \in \Sigma_o^*$, $\pi^{-1}(s') = \{u \in \Sigma^* \mid \pi(u) = s'\}$. For readability, we use the notation $[\![s]\!]$ to refer to $\pi^{-1}[\pi(s)]$.

Based only on the partial view of a sequence, if the controller can make the correct control decision, i.e., determine when the system leaves K via a controllable event, then K is called *observable*.

Definition 2.4. *A language K is said to be* ***observable*** *[20] w.r.t. L, π, and Σ_c iff*

$$(\forall s \in \overline{K})(\forall \sigma \in \Sigma_c)\ s\sigma \in L \setminus \overline{K} \Rightarrow [\![s]\!]\sigma \cap \overline{K} = \emptyset. \tag{2.1}$$

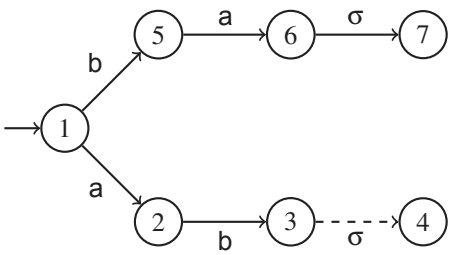


Figure 2.1: A finite-state automaton representing a DES.

Example 2.1. *A DES is shown in Figure 2.1. Suppose that L is the language generated by the DES and let K, the design specification, be the language generated by only solid line transitions. Let $\Sigma_c = \Sigma_o = \{\mathsf{a}, \mathsf{b}, \sigma\}$. The system leaves the specification via a controllable event σ; therefore K is controllable. Now consider two sequences $s = \mathsf{ba}$ and $s' = \mathsf{ab}$. Here $\pi(s) \neq \pi(s')$ since $\pi(s) = \mathsf{ab}$ and $\pi(s') = \mathsf{ba}$. That means two sequences are distinguishable to the controller based on observations. In case the sequence ab is generated, the controller can disable σ. Hence, K is observable.* ◇

A supervisory control or controller for the system can be defined as a function from the projection of closed behaviour of the system to the power set of Σ as

$\Gamma : \pi(L) \rightarrow 2^{\Sigma}$. It defines the set of events that should be *enabled* based on the controller's partial view of the system behaviour. The control decision for a sequence $s \in L$ is $\Gamma(\pi(s))$. While the elements in Σ_c are enabled or disabled by the controller according to the specification, all uncontrollable events that are eligible to occur should remain enabled. Without loss of generality, we assume that all events in Σ_{uc} are enabled:

$$(\forall s \in L)\Gamma(\pi(s)) = \{\gamma \in Pwr(\Sigma) \mid \gamma \supseteq \Sigma_{uc}\}.$$

Let Γ/M_L denote the system M_L under the supervision of Γ. The closed behaviour of Γ/M_L is defined as a language $L(\Gamma/M_L) \subseteq L$ such that

$$(i)\ \varepsilon \in L(\Gamma/M_L),\ \text{and}$$
$$(ii)(\forall s \in L(\Gamma/M_L))(\forall \sigma \in \Gamma(s))\ s\sigma \in L \Rightarrow s\sigma \in L(\Gamma/M_L).$$

The marked behaviour of Γ/M_L is

$$L_{\mathsf{m}}(\Gamma/M_L) = L(\Gamma/M_L) \cap L_{\mathsf{m}}.$$

Theorem 1. *There exists a controller Γ for the system M_L such that Γ/M_L is non-blocking and the closed behaviour of Γ/M_L is limited to the specification K (i.e., $L(\Gamma/M_L) = K$) iff*

(i) K is controllable w.r.t. L and Σ_{uc},

(ii) K is observable w.r.t. L, π and Σ_c, and

(iii) K is L_{m}-closed [32].

A controlled system is represented by a set of sublanguages of the unsupervised

(open loop) system, which is, in general, partially ordered. Our interest lies in quantitative measures which provide total ordering on respective performances of the sublanguages.

2.1.1 Quantitative Analysis of DES

A cost function may be considered for control actions on DES. Two cost functions are introduced in [24,49] for the quantitative analysis of a supervised system. In the basic setup, a finite (to be explained later) cost is considered for all legal sequences $s \in K$ and an infinite cost is applied when an illegal sequence $s \notin K$ occurs. It is assumed that unobservable events are uncontrollable, $\Sigma_{uo} \subseteq \Sigma_{uc}$, which implies that $\Sigma_c \subseteq \Sigma_o$. The event cost function is defined by $c_e : \Sigma \to \mathbb{R}^+$, the cost incurred for *executing* an event. This is extended to a sequence $s \in \Sigma^*$ by adding the respective event costs of the sequence. Hence, $c_e(s) = \sum_{i=1}^{m} c_e(\sigma_i)$ for $s = \sigma_1 ... \sigma_m$. The event cost is further extended to a sublanguage K by the summation of costs of all possible sequences in K as $c_e(K) = \sum_{s \in K} c_e(s)$. The control cost function is defined as $c_c : \Sigma \to \mathbb{R}^+ \cup \{0, \infty\}$. An infinite cost is incurred for all $\sigma \in \Sigma_{uc}$. The control cost is associated with the events that must be *disabled* to keep the system within the desired behaviour. The cost of a sequence in a controlled system is obtained by adding the event costs in the sequence with the control costs of the events that are disabled on the way to remain in the desired behaviour. Initially a *one-stage* cost function is defined for an event $\sigma \in \Gamma(s)$, for $s \in \Sigma^*$, as a function $c : \Sigma^* \times \Gamma \times \Sigma \to \mathbb{R} \cup \{0, \infty\}$, such that

$$c(\varepsilon, \Gamma(\varepsilon), \sigma) = c_e(\sigma) + \sum_{\sigma' \in L \setminus \Gamma(\varepsilon)} c_c(\sigma'),$$

which is extended to $s = \sigma_1\sigma_2 \ldots \sigma_m \in L$ as

$$c(s, \Gamma(s), \sigma_m) =$$
$$c(\varepsilon, \Gamma(\varepsilon), \sigma_1) + c(\sigma_1, \Gamma(\sigma_1), \sigma_2) + \ldots + c(\sigma_1 \ldots \sigma_{m-1}, \Gamma(\sigma_1 \ldots \sigma_{m-1}), \sigma_m).$$

The purpose of quantitative analysis is to remove the sequences with high event costs, which is accompanied by rising control costs. Thus, the trade-off in the optimization problem finds a balance between the cost function and the control objective.

The concept of language measure is introduced in [34], which ensures a total ordering on the controlled behaviours under different controllers. A partially observable system is considered for the quantitative analysis of a language in [8]. A signed real measure is constructed for a sublanguage of a system as a function $\mu : Pwr(L) \to \mathbb{R}$, where $\mathbb{R} = (-\infty, \infty)$. Relative to μ, a sublanguage can be classified as null, positive and negative sublanguages. When a sublanguage ends in a marked state with a sequence $s \in K$, μ is positive, otherwise it is negative. Here a characteristic function is defined which assigns a signed real weight [-1, 1] to a sequence, based on whether it ends with behaviours lie in or outside of the specification. A cost function for a sequence $s \in L$ can be obtained by the product of respective costs formed on the state from which the events of the sequence are generated. It makes an exact quantitative comparison of the controlled system under different controllers.

2.2 Supervisory Control of Decentralized DES

The decentralized supervisory control problem [9, 41] considers the synthesis of $n \geq 2$ controllers that cooperatively intend to keep the system in $\overline{K}$ by issuing control decisions to prevent the system from performing behaviour in $L \setminus \overline{K}$. Here we use $I = \{1, \ldots, n\}$ as an index set for the decentralized controllers. The ability to achieve

a correct control policy relies on the existence of at least one controller that can make the correct control decision to keep the system within $\overline{K}$.

The basic decentralized control architecture is shown in Figure 2.2. In this setup, each local controller makes decision based on its own observation. The decisions of local controllers are fused to determine the global control decision. A *conjunctive fusion rule* ($\wedge$) is applied to determine whether an event $\sigma \in \Sigma$ is globally enabled after a sequence $s \in L$. In the context of the decentralized supervisory control problem, Σ is partitioned into two sets for each controller $i \in I$: controllable events $\Sigma_{c,i}$ and uncontrollable events $\Sigma_{uc,i} := \Sigma \backslash \Sigma_{c,i}$. The overall set of controllable events is $\Sigma_c := \bigcup_{i=1} \Sigma_{c,i}$. Let $I_c(\sigma) = \{i \in I | \sigma \in \Sigma_{c,i}\}$ be the set of controllers that control event σ. We will also partition T_L based on controllability status of the events, as $T_{c,i} = \{(q, \sigma, q') \in T_L \mid \sigma \in \Sigma_{c,i}\}$, and $T_{uc,i} = T_L \setminus T_{c,i}$.

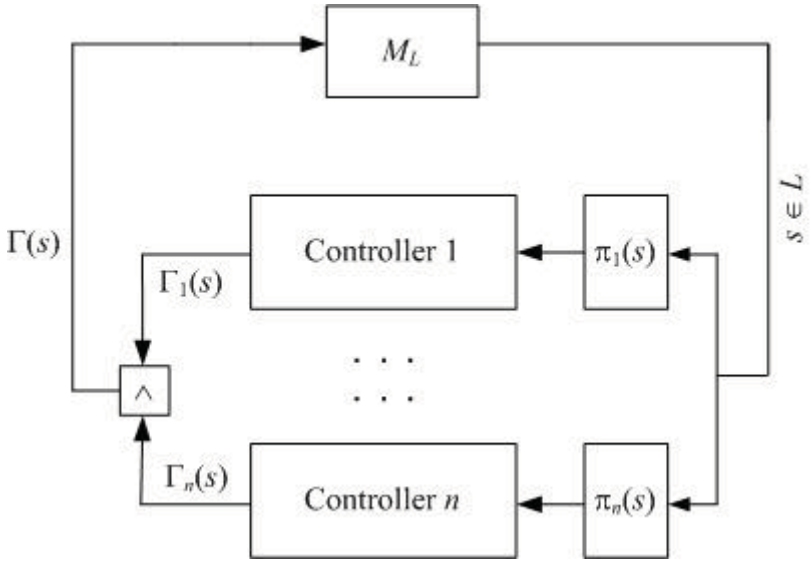


Figure 2.2: Decentralized DES architecture.

Each controller $i \in I$ also has a set of observable events, denoted by $\Sigma_{o,i}$, and unobservable events $\Sigma_{uo,i} = \Sigma \backslash \Sigma_{o,i}$. Similarly, the set of transitions T_L is partitioned as $T_{o,i} = \{(q, \sigma, q') \in T_L \mid \sigma \in \Sigma_{o,i}\}$ and $T_{uo,i} = T_L \setminus T_{o,i}$. To formally capture the notion of partial observation in decentralized supervisory control problems, the

natural projection is extended to each controller $i \in I$ as $\pi_i : \Sigma^* \to \Sigma_{o,i}^*$. Thus for $s = \sigma_1\sigma_2 \ldots \sigma_m \in \Sigma^*$, the partial observation $\pi_i(s)$ will contain only those events $\sigma \in \Sigma_{o,i}$:

$$\pi_i(\sigma) = \begin{cases} \sigma, & \text{if } \sigma \in \Sigma_{o,i}; \\ \varepsilon, & \text{otherwise,} \end{cases}$$

which is extended to sequences as follows: $\pi_i(\varepsilon) = \varepsilon$, and $\forall s \in \Sigma^*$, $\forall \sigma \in \Sigma$, $\pi_i(s\sigma) = \pi_i(s)\pi_i(\sigma)$. The inverse projection of π_i is a mapping $\pi_i^{-1} : \Sigma_{o,i}^* \to 2^{\Sigma^*}$ such that for $s' \in \Sigma_{o,i}^*$, $\pi_i^{-1}(s') = \{u \in \Sigma^* \mid \pi_i(u) = s'\}$. As before, we use the notation $[\![s]\!]_i$ to refer to $\pi_i^{-1}[\pi_i(s)]$.

When a global control decision is made, there may exist at least one decentralized controller that can take a correct control decision (i.e., determine that $s\sigma \in L\backslash\overline{K}$) based on its partial observation of a sequence, by disabling a controllable event through which the sequence leaves the specification K. In that case, K is called *co-observable*.

Definition 2.5. *A language K is **co-observable** [41] w.r.t. L, $\Sigma_{o,i}$, and $\Sigma_{c,i}$ $(i \in I)$ iff*

$$(\forall s \in \overline{K})(\forall \sigma \in \Sigma_c)\ s\sigma \in L\backslash\overline{K} \Rightarrow (\exists i \in I_c(\sigma))\ [\![s]\!]_i\sigma \cap \overline{K} = \emptyset.$$

When it is clear from the context, we just say that K is co-observable. Also note that when $I = \{1\}$, co-observability reduces to observability (Definition 2.1).

Example 2.2. *Continuing with Example 2.1, let $n = 2$ and suppose that $\Sigma_{o,1} = \{\mathsf{a}, \sigma\}$, and $\Sigma_{o,2} = \{\mathsf{b}, \sigma\}$. Further, let $I_c(\sigma) = \{1, 2\}$. Consider $s = \mathsf{ba}$ and $s' = \mathsf{ab}$. Note*

that K is not co-observable since $\pi_1(s) = \pi_1(s') = \mathsf{a}$, and $\pi_2(s) = \pi_2(s') = \mathsf{b}$. Hence, $(\forall i \in I_c(\sigma))\ \pi_i(s) = \pi_i(s')$ and, thus, no single controller can take the correct control decision regarding σ. $\diamond$

A decentralized control law for controller i is a mapping $\Gamma_i : \pi_i(L) \rightarrow Pwr(\Sigma)$ that defines the set of events that controller i believes should be *enabled* based on its partial view of the system behaviour. While controller i can choose to enable or disable events in $\Sigma_{c,i}$, as in the centralized case, all events in $\Sigma_{uc,i}$ must be enabled.

$$(\forall i \in I)(\forall s \in L)\ \Gamma_i(\pi_i(s)) = \{\gamma \in Pwr(\Sigma) \mid \gamma \supseteq \Sigma_{uc,i}\}.$$

To find a solution to the decentralized control problem, we want to find controllers Γ_i $(\forall i \in I)$ such that $\forall s \in \overline{K}$:

$$(\forall \sigma \in (\Sigma))\ s\sigma \in \overline{K} \Rightarrow \exists \sigma \in \bigcap_{i \in I} \Gamma_i(\pi_i(s)) \text{ and}$$
$$(\forall \sigma \in \Sigma_c)\ s\sigma \in L \setminus \overline{K} \Rightarrow \exists \sigma \notin \bigcap_{i \in I} \Gamma_i(\pi_i(s)).$$

That means, an event σ must be enabled after a sequence s by all supervisors if $s\sigma \in \overline{K}$. Otherwise, it is disabled by at least one controller. From the results of [41], such supervisors exist if the specification K is co-observable, controllable, and L_{m}-closed.

When K is not co-observable, it may still be possible to find a control solution by introducing communication between decentralized controllers, so that with the additional information provided through the content of received messages, all the correct control decisions can be taken.

2.3 Synchronous (Zero-Delay) Communication

There are a variety of strategies for introducing communication between decentralized controllers: sending messages as early as possible [38], as late as possible [4] or possibilities in-between [37]. Strategies are further distinguished by the content of the messages sent: state-estimates [4, 38, 57], event occurrences [37, 56], and information related to control decisions [23]. Figure 2.3 shows controllers communicating with each other in decentralized DES.

The synthesis of communication protocols requires additional information provided through the content of received messages, denoted here by $\Sigma_i^? \subseteq \Sigma_o \backslash \Sigma_{o,i}$ so that at least one controller can take the correct control decision after receiving the communicated information.

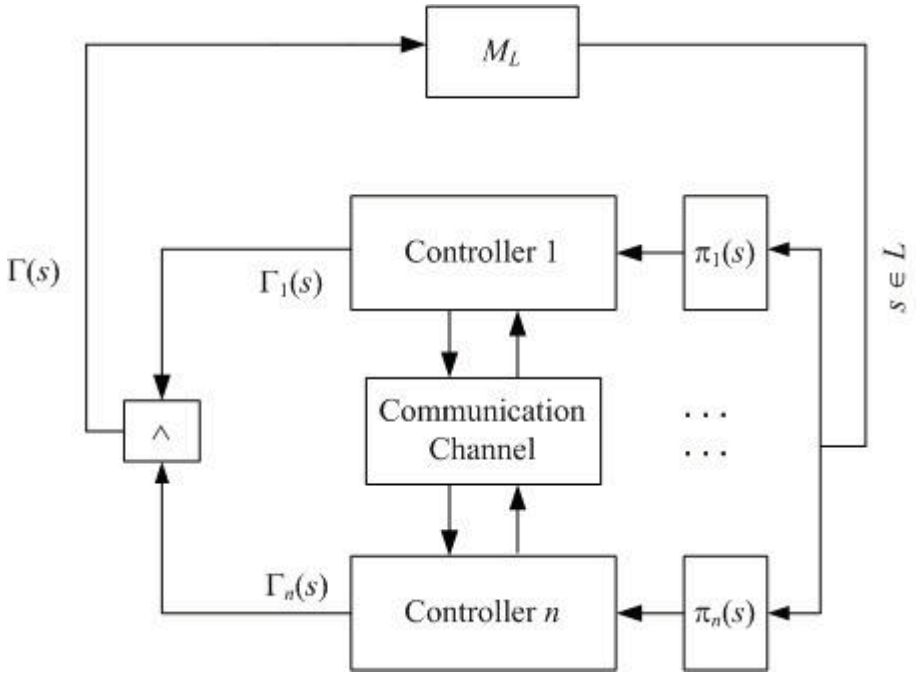


Figure 2.3: Communication between controllers in decentralized DES architecture.

Let us consider the strategy of [36] to identify a communication protocol. The set of messages sent from one controller to another is derived from a set of communication

transitions, denoted here by $T_i^! = \bigcup_{j \in I} T_{i,j}^! \subseteq T_{o,i}$, where $T_{i,j}^!$ identifies transitions that are coming from controller i to j on at least one system trajectory. The goal is to allow the controllers to make the correct control decisions not just based on their partial observation of the system, but also taking into account the information received from other communicating controllers. There are many options for choosing when communication should occur: each controller/sender can communicate everything it observes followed by subsequent refinement based on information it has received from the others [16,30,51,54]; or specific events/transitions can be identified by the sender as providing useful information to a receiver [35,36,56]. While it is assumed that the protocol is identified as a set of specific transitions to incorporate this into K, we will replace $(q, \sigma, q') \in T_{i,j}^!$ by σ in controller i's of K. We define the content of the message as $\Sigma_i^! = \{\sigma \in \Sigma_{o,i} | \exists (q, \sigma, q') \in T_i^!\}$.

Definition 2.6. *Consider a set of communication transitions for controller i, $T_i^! = T_{i,1}^! \cup \ldots \cup T_{i,n}^!$. A **communication protocol between** i **and** j, $\phi_{i,j} : \overline{K} \to \Sigma_i^! \cup \{\varepsilon\}$ $(j \in I \backslash \{i\})$, is defined as follows:*
$(\forall s \in \overline{K})$ $(\forall \sigma \in \Sigma)$, if $q_0 \xrightarrow{s} q' \xrightarrow{\sigma} q'' \in T_L$, then

$$\phi_{i,j}(s\sigma) = \begin{cases} \sigma, & \text{if } (q', \sigma, q'') \in T_{i,j}^!, \\ \varepsilon, & \text{otherwise.} \end{cases}$$

*Hence, $\phi_i = \langle \phi_{i,1}, \ldots, \phi_{i,n} \rangle\}$. Then for every $i \in I$, the **set of communication protocols for controller** i is $\Phi_i = \{\phi_i | \phi_i = \langle \phi_{i,1}, \ldots, \phi_{i,n} \rangle\}$. The overall set of communication protocols is then defined as $\Phi = (\Phi_i)_{i \in I}$.* □

The most recent information that a controller has about a sequence is defined as $\psi_i : L \to \Sigma_{o,i} \cup (\bigcup_{j \in I \backslash \{i\}} \Sigma_{o,j})$. When $s\sigma \in L$ occurs, each controller i keeps track of

communication it receives about $s\sigma$ along with its own observations of $s\sigma$.

$$\psi_i(s\sigma) = \begin{cases} \sigma, & \text{if } \sigma \in \Sigma_{o,i} \text{ or } (\sigma \notin \Sigma_{o,i} \text{ and } \exists j \in I \text{ s.t. } \phi_{j,i}(s\sigma) \neq \varepsilon); \\ \varepsilon, & \text{otherwise.} \end{cases}$$

The natural projection π_i is extended to $\pi_i^? : L \to (\Sigma_{o,i} \cup \Sigma_i^?)^*$ to include received messages as follows:

$$\pi_i^?(\varepsilon) = \varepsilon,$$
$$\pi_i^?(s) = \psi_i(\sigma_1)\psi_i(\sigma_1\sigma_2)\dots\psi_i(\sigma_1\dots\sigma_m) \text{ for } s = \sigma_1\dots\sigma_m.$$

Finally, communication must occur in an observationally-equivalent manner, i.e., the same message should be sent after the occurrence of all sequences that have the same extended natural projection.

Definition 2.7. *A communication protocol $\phi_{i,j}$ is* ***coherent***[1] *if*

$$(\forall s, s' \in L)(\forall i \in I)\ \pi_i^?(s) = \pi_i^?(s') \Rightarrow (\forall j \in I \setminus \{i\})\ \phi_{i,j}(s) = \phi_{i,j}(s').$$

When K is not co-observable, there may exists a controller i that can make the correct control decision (i.e., determine that $s\sigma \in L\backslash\overline{K}$) based on its partial observation of the system behaviour and the information received through the communication protocol ϕ_i $(i \in I)$.

Definition 2.8. *We say that K is* ***communication observable*** *w.r.t. L, $\pi_i^?$, $\Sigma_{c,i}$*

[1] *In the discrete-event system literature, this property is referred to as feasibility, but we will use this word in the sequel as it has a particular meaning in the context of finding Nash equilibrium points.*

and ϕ_i $(i \in I)$ *iff*

$$(\forall s \in \overline{K})(\forall \sigma \in \Sigma_c)\ s\sigma \in L \setminus \overline{K} \Rightarrow (\exists i \in I_c(\sigma))\ [\![s]\!]_i^? \sigma \cap \overline{K} = \emptyset.$$

When it is clear from the context, we just say that K is communication observable.

We extend the decentralized control law to a communicating controller i as follows $\Gamma_i^? : \pi_i^?(L) \to Pwr(\Sigma)$. To find a solution to the decentralized control problem with synchronous communication protocols $\phi = \{\phi_{i,j}\}$ for all controllers $i, j \in I$, we have to find $\Gamma_i^?$ $(\forall i \in I)$ such that $\forall s \in \overline{K}$:

$$(\forall \sigma \in (\Sigma_c \cup \Sigma_{uc}))\ s\sigma \in \overline{K} \Rightarrow \sigma \in \bigcap_{i \in I} \Gamma_i^?(\pi_i^?(s)),$$
$$(\forall \sigma \in \Sigma_c)\ s\sigma \in L \setminus \overline{K} \Rightarrow \sigma \notin \bigcap_{i \in I_c(\sigma)} \Gamma_i^?(\pi_i^?(s)).$$

It is already shown in [41] that we can find such Γ_i if K is co-observable, controllable, and L_{m}-closed.

The decentralized control and (synchronous) communication problem (DCCP) is formally described below.

Problem 2.1. *Consider two regular languages K, L defined over a common alphabet Σ, controllable events $\Sigma_{c,1}, ..., \Sigma_{c,n} \subseteq \Sigma$, observable events $\Sigma_{o,1}, \ldots, \Sigma_{o,n} \subseteq \Sigma$, and a set of communication transitions $T_1^!, \ldots, T_n^!$ (for $i \in I$), where $K \subseteq L \subseteq \Sigma^*$ is observable w.r.t. L, Σ_o, Σ_c and controllable w.r.t. L, Σ_{uc}, but is not co-observable w.r.t. L, $\Sigma_{o,i}$, $\Sigma_{c,i}$. Construct communication protocols $(\phi_i)_{i \in I}$, such that K is communication observable.*

We can synthesize communicating controllers when K is communication observable w.r.t. L, $\pi_i^?$, $\Sigma_{c,i}$ $(i \in I)$ and $(\phi_i)_{i \in I}$.

Example 2.3. *In the ongoing Example 2.1, let the set of communication transitions*

for Controller 1 be $T^{!}_{1,2} = \{(1,a,2), (5,a,6)\}$ *(i.e.,* $\Sigma^{?}_{2} = \{a\}$*) and for Controller 2 be* $T^{!}_{2,1} = \{(1,b,5), (2,b,3)\}$ *(i.e.,* $\Sigma^{?}_{1} = \{b\}$*). Hence, we have the following coherent communication protocol* $\phi = (\phi_1; \phi_2)$*:*

$$\begin{aligned}
\phi_1 \;&:\; \phi_{1,2}(ba) = \phi_{1,2}(a) = a, \\
&\quad\; (\forall s \in L \setminus \{ba,a\})\phi_{1,2}(s) = \varepsilon; \\
\phi_2 \;&:\; \phi_{2,1}(b) = \phi_{2,1}(ab) = b, \\
&\quad\; (\forall s \in L \setminus \{b,ab\})\phi_{2,1}(s) = \varepsilon.
\end{aligned}$$

While $\pi_1(ab) = \pi_1(ba)$*, by extending controller 1's information to* $\Sigma_{o,1} \cup \Sigma^{?}_{1}$ *via* ϕ*, we have* $\pi^{?}_{1}(ab) = ab$ *whereas now* $\pi^{?}_{1}(ba) = ba$*. Thus* K *is communication observable w.r.t.* L*,* $\pi^{?}_{i}$*,* $\Sigma_{c,i}$ *and* ϕ_i *(*$i \in I$*).* $\diamond$

A violation of co-observability can be detected by a product automaton $\mathcal{U}$ [36]. A special product is used to compose more than one automaton, which is called *synchronized composition*.

Synchronized composition of automata: The synchronized composition, denoted by $\times_{\mathcal{S}}$, is defined as follows. Assume that we have m finite-state automata $M_1, \ldots, M_m$, where $M_j = (Q_j, \Sigma_j, T_j, q_{0,j}, F_j)$, for $j = 1, 2, \ldots, m$, where self-loops of ε are added to every state $q_j \in Q_j$, for $j \in \{1, \ldots, m\}$, to facilitate the composition. Then

$$M = M_1 \times_{\mathcal{S}} M_2 \times_{\mathcal{S}} \ldots \times_{\mathcal{S}} M_n = (Q_{\mathcal{S}}, \Sigma_{\mathcal{S}}, T_{\mathcal{S}}, \langle q_{0,1}, q_{0,2}, \ldots, q_{0,n} \rangle, F_{\mathcal{S}}),$$

where $Q_{\mathcal{S}} \subseteq Q_1 \times Q_2 \times \ldots \times Q_n$; $\Sigma_{\mathcal{S}} \subseteq \Sigma_1 \times \Sigma_2 \times \ldots \times \Sigma_n$; $T_{\mathcal{S}} \subseteq Q_{\mathcal{S}} \times \Sigma_{\mathcal{S}} \times Q_{\mathcal{S}}$; and $F_{\mathcal{S}} \subseteq F_1 \times F_2 \times \ldots \times F_n$. The state set $Q_{\mathcal{S}}$ is a set of state vectors of the form $q_{\mathcal{S}} = (q_1, \ldots, q_n)$ and we will occasionally refer to the j^{th} component of $q_{\mathcal{S}}$ as $q_{\mathcal{S}}(j)$,

where $j \in \{1, \ldots m\}$. We can think of these automata as running concurrently, and, thus, there is some synchronization of events in each of the alphabets. A transition $(q_{\mathcal{S}}, \sigma_{\mathcal{S}}, q'_{\mathcal{S}}) \in T_{\mathcal{S}}$, where $q_{\mathcal{S}} = (q, q_1, \ldots, q_n)$ and $q'_{\mathcal{S}} = (q', q'_1, \ldots, q'_n)$ with transition label $\sigma_{\mathcal{S}} = \langle \sigma_S(0), \ldots, \sigma_{\mathcal{S}}(n) \rangle$, iff $(q_j, \sigma(j), q'_j) \in T_j$, for $j \in \{1, \ldots, m\}$.

The $\mathcal{U}$-structure is constructed by composition of M_L with n copies of M_K.

$$\mathcal{U} = (X, \Sigma^{\mathcal{U}}, T_{\mathcal{U}}, x_0, F_{\mathcal{U}}) = M_L \times_{\mathcal{S}} \Pi_{i=1}^{n} (M_K)_i \tag{2.2}$$

The alphabet of $\Sigma^{\mathcal{U}}$ is a set of vector labels from [2]. We have two types of labels, corresponding to the occurrence and observation of an event in Σ_o (and thus $\Sigma_{o,i}$) and events that are not officially observed, i.e., events in $\Sigma_{uo,i}$ and $\Sigma \setminus \Sigma_o$. Let $I_o(\sigma) = \{i \in I \mid \sigma \in \Sigma_{o,i}\}$. We build the following set of labels for $\Sigma^{\mathcal{U}}$: for all $\sigma \in \Sigma_o$, $\ell(0) = \sigma$ and for all $i \in I_o(\sigma)$, $\ell(i) = \sigma$, and for all $j \in I \setminus I_o(\sigma)$, $\ell(j) = \varepsilon$; for all $\sigma \in \Sigma \setminus \Sigma_{o,i}$, $\ell(i) = \sigma$ and for all $j \neq i \in I$, $\ell(j) = \varepsilon$; for all $\sigma \in \Sigma \setminus \Sigma_o$, $\ell(0) = \sigma$ and for all $i \in I$, $\ell(i) = \varepsilon$.

A transition $(x, \ell, x') \in T_{\mathcal{U}}$, where $x = (q, q_1, \ldots, q_n)$ and $x' = (q', q'_1, \ldots, q'_n)$ with label $\ell = \langle \ell(0), \ldots, \ell(n) \rangle$ iff $(q, \ell(0), q') \in T_L$, and for all $i \in I$, $(q_i, \ell(i), q'_i) \in T_K$.

The set of marked transitions is defined as follows:

$$F_{\mathcal{U}} = \{(x, \ell, x') \mid ((x(0), \ell(0), x'(0)) \in T_L \setminus T_K \wedge$$
$$\forall i \in I_c(\ell(0)) : (x(i), \ell(i), x'(i)) \in T_K)\}.$$

Example 2.4. *The $\mathcal{U}$-structure of the ongoing example is shown in Fig 2.4. It has 42 states, 8 labels, and 66 transitions.*

The set of atoms for this example is $A = \{\langle \mathsf{a}, \mathsf{a}, \varepsilon \rangle, \langle \varepsilon, \varepsilon, \mathsf{a} \rangle, \langle \mathsf{b}, \varepsilon, \mathsf{b} \rangle, \langle \varepsilon, \mathsf{b}, \varepsilon \rangle, \langle \sigma, \sigma, \sigma \rangle\}$. *Since* a *is not an event observable to Controller 2, we have the label* $\ell = \langle \varepsilon, \varepsilon, \mathsf{a} \rangle \in A$.

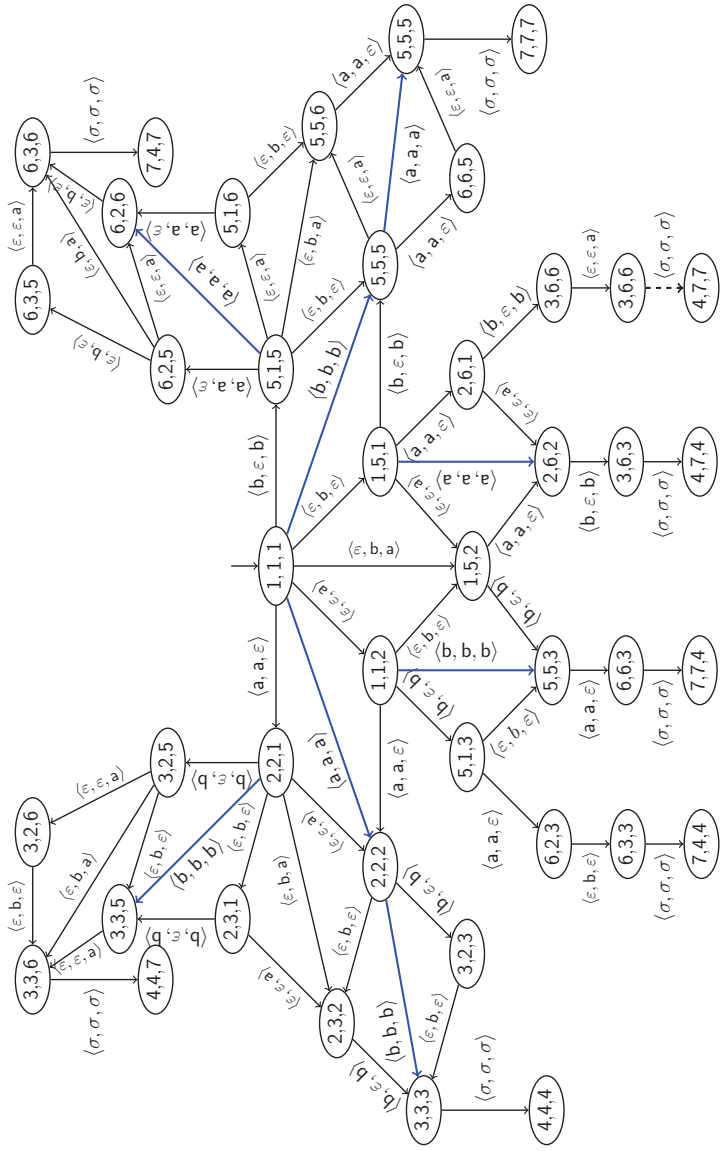


Figure 2.4: Automaton $\mathcal{U}$ for the ongoing example with Figure 2.1. Marked transition is denoted by dashed line. Potential communication transitions are indicated in blue.

We interpret this as follows: Controller 2 has no idea whether or not a *has just occurred in the plant (i.e.,* $\ell(0) = \varepsilon$*), but it guesses that* a *could have occurred (i.e.,* $\ell(2) = \mathsf{a}$*) and it makes no assumptions about the observations of Controller 1 (i.e.,* $\ell(1) = \varepsilon$*). But* a *is observable to Controller 1, so we have the label* $\ell' = \langle \mathsf{a}, \mathsf{a}, \varepsilon \rangle \in A$*. This means that* a *has occurred in the plant (i.e.,* $\ell'(0) = \mathsf{a}$*) and its occurrence was observed by Controller 1 (i.e.,* $\ell'(1) = \mathsf{a}$*); however, the event was not observed by Controller 2 (i.e.,* $\ell'(2) = \varepsilon$*). Finally,* $\Sigma^{\mathcal{U}} = A \cup \{\langle \mathsf{a}, \mathsf{a}, \mathsf{a} \rangle, \langle \mathsf{b}, \mathsf{b}, \mathsf{b} \rangle, \langle \varepsilon, \mathsf{b}, \mathsf{a} \rangle\}$.

In the example, $F^{\mathcal{U}} = \{(\mathsf{3}, \mathsf{6}, \mathsf{6}) \xrightarrow{\langle \sigma, \sigma, \sigma \rangle} (\mathsf{4}, \mathsf{7}, \mathsf{7})\}$*. Let this marked transition be denoted by* ζ*, whose label is denoted by* ℓ_ζ *and which is reached via* $(\mathsf{1}, \mathsf{1}, \mathsf{1}) \xmapsto{w} (\mathsf{3}, \mathsf{6}, \mathsf{6})$*. We use* w *to identify the way in which* ζ *corresponds to a violation of co-observability. The true system trajectory is the sequence formed by* $w(0)\ell_\zeta(0)$*, namely* ba*, which is in* $L \setminus \overline{K}$*. Both controllers control* σ*, so we examine* $w(1)\ell_\zeta(1)$ *and* $w(2)\ell_\zeta(2)$ *to see what each controller considers possible sequences if* $w(0)\ell_\zeta(0)$ *had occurred in the system. In this case,* $w(1)\ell_\zeta(1) = w(2)\ell_\zeta(2) = \mathsf{ab}\sigma \in \overline{K}$*. Hence, the transition* $(3, 6, 6) \xrightarrow{\langle \sigma, \sigma, \sigma \rangle} (4, 7, 7)$ *is marked because* σ *must be disabled according to* M_L *whereas both controllers believe that* σ *should be enabled.*

We can also illustrate the set of communications, shown in blue color in the $\mathcal{U}$ *structure, which makes the system co-observable. For example, Controller* 1 *communicates the occurrence of* a *to Controller 2* $((1, 5, 1), \langle \mathsf{a}, \mathsf{a}, \mathsf{a} \rangle, (2, 6, 2))$*. In that case, the transitions* $((1, 5, 1), \langle \varepsilon, \varepsilon, \mathsf{a} \rangle, (1, 5, 2))$ *and* $((1, 5, 2), \langle \mathsf{a}, \mathsf{a}, \varepsilon \rangle, (2, 6, 2))$ *are pruned from the* $\mathcal{U}$*. The reception of* a *forces Controller 2 to follow the plant behavior, and avoid to reach to the marked transition. Hence Controller 2 takes correct control decision with communication received from Controller 1.*

In synthesizing synchronous communication, no delay is assumed in the communication. But, in reality, communication occurs with some delay. In that case, we can consider time bounds for the occurrence of events which inspire to use TDES.

2.4 Supervisory Control of TDES

Classical DES are concerned with the order of occurrences of events in the system. The exact time at which each event occurs is unimportant. In many applications, however, the exact time each event occurs is important. We use the supervisory control framework of [6] that describes the system behaviour of a TDES, denoted here by L^τ. We start with an automaton to model a TDES:

$$M_{\mathsf{act}} = (A, \Sigma_{\mathsf{act}}, T_{\mathsf{act}}, a_0, A_{\mathsf{m}}).$$

The components of M_{act} are defined in the usual way, except that states are now called *activities*. Here A is a finite set of *activities*; Σ_{act} is the alphabet of event labels; $T_{\mathsf{act}} \subseteq A \times \Sigma_{\mathsf{act}} \times A$ is the transition relation; a_0 is the initial activity; and $A_{\mathsf{m}} \subseteq A$ is a set of marked activities. Two maps are defined for each event in Σ_{act}: (1) a lower bound $\mathsf{l} : \Sigma_{\mathsf{act}} \to \mathbb{N}$ and (2) an upper bound $\mathsf{u} : \Sigma_{\mathsf{act}} \to \mathbb{N} \cup \{\infty\}$. Each $\sigma \in \Sigma_{\mathsf{act}}$ can occur in the interval $[\mathsf{l}(\sigma), \mathsf{u}(\sigma)]$, where $\mathsf{l}(\sigma)$ is the lower or minimum delay after which σ can occur, and $\mathsf{u}(\sigma)$ is the upper or hard deadline before which σ must occur. It is required that $(\forall \sigma \in \Sigma_{\mathsf{act}})\mathsf{l}(\sigma) \leq \mathsf{u}(\sigma)$. A TDES can be fully specified by $(M_{\mathsf{act}}, \mathsf{l}, \mathsf{u})$, where time is implicitly modeled. The transition graph associated with M_{act} is called an *activity transition graph* (ATG).

While M_{act} has a compact representation, it is converted to an automaton M^τ before a supervisory control or communication protocol is designed. The automaton M^τ describing the timed system is a five-tuple

$$M^\tau = (Q, \Sigma^\tau, T^\tau, q_0, Q_{\mathsf{m}}),$$

where time is explicitly modeled. Here Q is a finite set of states; Σ^τ is a finite set of events; $T^\tau \subseteq Q \times \Sigma^\tau \times Q$ is the transition relation; q_0 is the initial state; and $Q_{\mathsf{m}} \subseteq Q$

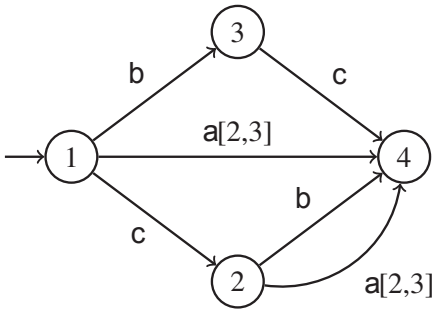


Figure 2.5: A finite-state automaton representing ATG.

is a set of marked states. The set of events is composed of $\Sigma^\tau = \Sigma_{\mathsf{act}} \cup \{\tau\}$, where τ denotes the passage of one unit of time. We assume that we have a global digital clock for measuring time. The transition graph of M^τ is called a *timed transition graph* (TTG). The specification, denoted by $K^\tau \subseteq L^\tau$, describes the desired behaviour of the system.

Example 2.5. *The example illustrates how a system is represented by an ATG and a TTG. In ATG given in Figure 2.5,* $\Sigma_{\mathsf{act}} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$*;* $A = A_m = \{1, 2, 3, 4\}$*;* $a_0 = 1$*. The lower and upper time bounds of* a *are 2 and 3 respectively. Time bounds are omitted of* $\sigma \in \Sigma_{\mathsf{act}}$*, when* $\mathsf{l}(\sigma) = 0$ *and* $\mathsf{u}(\sigma) = \infty$*. The events can occur anytime between the lower and upper time bounds in TTG. Next we convert the ATG to the corresponding TTG, shown in Figure 2.6 which describes the occurrence of* a *and* b *with respect to clock ticks.*

In the context of decentralized TDES, Σ^τ is partitioned into three subsets for each controller $i \in I$. The set of *controllable* events and *uncontrollable* events are $\Sigma_{c,i}$ and $\Sigma_{uc,i}$ as before, and a set of *forcible* events denoted by $\Sigma_{f,i}$ that a controller can force to happen before time progresses. The overall set of forcible events, $\Sigma_f = \cup_{i=1}^n \Sigma_{f,i}$, and the set of controllers for which σ is forcible is $I_f(\sigma) = \{i \in I \mid \sigma \in \Sigma_{f,i}\}$. We will also partition T^τ accordingly.

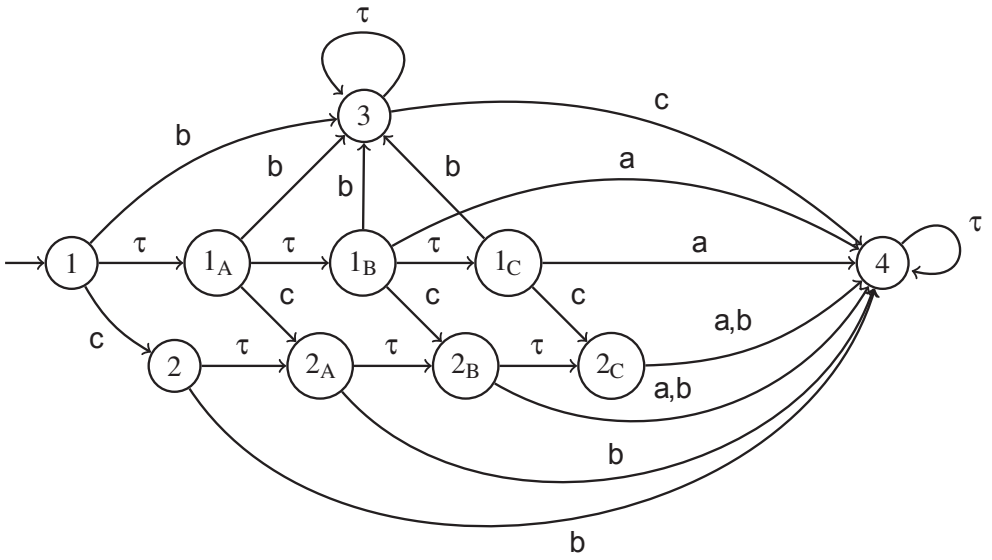


Figure 2.6: A finite-state automaton representing TTG of Figure 2.5.

Definition 2.9. *A language K^τ is **controllable** w.r.t. L^τ and Σ_{uc} in TDES [26] iff*

$$(\forall s \in \overline{K^\tau})(\forall \sigma \in \Sigma_{uc}) s\sigma \in L^\tau \Rightarrow s\sigma \in \overline{K^\tau}.$$

We also consider the notion of partial observation in TDES, which can be formally described by a natural projection as $\pi_i^\tau : \Sigma^{\tau *} \to \Sigma_{o,i}^*$. The inverse projection $\pi_i^{\tau^{-1}} : \Sigma_{o,i}^* \to 2^{\Sigma^{\tau *}}$ is defined for $s' \in \Sigma_{o,i}^*$ as $\pi_i^{\tau^{-1}}(s') = \{u \in \Sigma^{\tau *} \mid \pi_i^\tau(u) = s'\}$. Note that $\tau \in \Sigma_{o,i}$ for all $i \in I$.

Definition 2.10. *K^τ is **co-observable** w.r.t. L^τ, $\Sigma_{o,i}$, and $\Sigma_{c,i}$ $(i \in I)$ in TDES [26] iff*

$$(\forall s \in \overline{K^\tau})(\forall \sigma \in \Sigma_c) s\sigma \in L^\tau \backslash \overline{K^\tau} \Rightarrow (\exists i \in I_c(\sigma))[\![s]\!]_i^\tau \sigma \cap \overline{K^\tau} = \emptyset.$$

A decentralized control law for controller i in TDES is defined as a map $\Gamma_i^\tau : \pi_i^\tau(L^\tau) \to Pwr(\Sigma^\tau) \times Pwr(\Sigma_{f,i})$, such that for a sequence $s \in \pi_i^\tau(L^\tau)$, $\Gamma_i^\tau(s) :=$

$(\Gamma^{\tau}_{e,i}(s), \Gamma^{\tau}_{f,i}(s))$, where $\Gamma^{\tau}_{e,i}(s) = \{\gamma \in \Sigma^{\tau} | \gamma \supseteq \Sigma_{uc,i}\}$, and $\Gamma^{\tau}_{f,i}(s) = \{\gamma \in \Sigma^{\tau} | \gamma \in \Sigma_{f,i}\}$ [26]. $\Gamma^{\tau}_{e,i}(s)$ defines the set of events that are enabled by controller i and $\Gamma^{\tau}_{f,i}(s)$ denotes the set of events that are forced to occur by controller i after observation $\pi^{\tau}_{i}(s)$. The event τ has a double role in the control map: when there are forcible events present, it is treated as a controllable event and can be preempted; when no forcible event is present, it is treated as an uncontrollable event. In that case, $\tau \in \Gamma^{\tau}_{e,i}(s)$ if $(\nexists \sigma \in \Sigma_{f,i}) s\sigma \in L^{\tau}$.

2.5 Nash Equilibriun and Pareto Optimality

Equilibrium is a key idea in calculating optimal strategies for multi-agent systems. A multi-agent system in game theory models competition (or cooperation) among the agents. To optimize the outcome, an agent takes into account the decisions that other agents take and assumes they act so as to optimize their own outcome. Nash equilibrium and Pareto optimality, two important concepts in game theory, are used to find equilibria among multiple agents. They are used to analyze the outcome of the strategic interaction of multiple agent systems. A Nash equilibrium is a collection of strategies, one for each agent in the system, such that if all other agents adhere to their strategies, an agent's recommended strategy is strictly better than any other strategy it could execute.

For a system with N agents, let $A = A_1 \times \ldots \times A_n$, where A_i is a set of strategies of agent i. Let $u_i : A \to \mathbb{R}$ denote a real-valued cost function for agent i. Consider the problem of optimizing (minimizing) the cost functions u_i. A set of strategies $a^* = (a^*_1, \ldots, a^*_n) \in A$ is a *Nash equilibrium*, if

$$(\forall i \in N)(\forall a_i \in A_i)\ u_i(a^*_i, a^*_{\_i}) \leq u_i(a_i, a^*_{\_i}),$$

where $a_{\_i}$ denotes the set of strategies $\{a_k | k \in N \text{ and } k \neq i\}$. Intuitively, a Nash equilibrium represents each agent's best response to the strategies of the other agents.

The concept of Nash equilibrium is used for decentralized DES in [27]. A controller S^* is supremal if for all controllers S that solve the supervisory control problem $L(S/M_L) \subseteq L(S^*/M_L)$. The closed loop behaviours generated by the controllers is analogous to the cost function of a game. However, the resulting controllers are only *partially* ordered and incomparable because of the underlying optimality definition.

Various numerical methods for calculating Nash equilibrium have been proposed. For two-player games, the Lemke-Howson algorithm [19] is still the best-known among the combinatorial algorithms. Other algorithms to calculate a sample Nash equilibrium point for such two-player games include [19, 29, 47]. Finding a Nash equilibrium is an NP-hard problem. The Lemke-Howson algorithm to find a sample Nash equilibrium is based on linear programming and is exponential in time. In [29], a heuristic approach is presented for finding a sample Nash equilibrium in normal-form games. The algorithm is based on the *support space* and a notion of dominated actions that are pruned from the search space. The support specifies the subset of available actions that are assigned positive probability. The search space is ordered according to the support size profiles. In two-player games, the algorithm chooses the support sizes favoring those that are balanced and small. Then the algorithm prunes the search space by the conditional dominance, which instantiates each players' support. An action is conditionally dominated given a profile of sets of available actions of the remaining players, if the utility function of this player can be improved by choosing a different action. Two algorithms are proposed using the backtracking approach, one for two-player games and the other for n-player games, with $n > 2$.

On the other hand, Pareto optimality is a measure of efficiency. A set of strategies, one for each agent in the system, is *Pareto-optimal* if there is no other strategies that

make at least one agent strictly better off with making all other agents at least as well off [48]. A strategy $a^* = (a_1^*, \ldots, a_n^*) \in A$ is Pareto-optimal if there is no other strategies $a = (a_1, \ldots, a_n)$ such that

$$(\forall i \in N)\ u_i(a_i, a_{-i}) \leq u_i(a_i^*, a_{-i}^*).$$

When an agent gains by changing a strategy without worsening any other agent, this is called Pareto improvement. When a solution is Pareto-optimal, no further improvement in one cost function is possible without worsening another. In DES, we are interested in optimizing the communication cost of a controller (agent) considering the communication cost of all other controllers.

Pareto-optimal solutions do not necessarily form a Nash equilibrium or vice versa. The concept of Pareto optimality is widely-used in multi-objective optimization [14, 61]. A Pareto-optimal solution is not necessarily unique and we have to consider a Pareto-optimal set. This set forms the *Pareto front*, and constitutes a complete set of solutions for multi-objective optimization problem.

2.6 Multi-Objective Optimization

Multi-objective optimization deals with solving problems having multiple, often conflicting objectives. These problems arise naturally in most of the fields of science, engineering and business. A multi-objective optimization problem can be formally defined in terms of m decision variables $x_1, \ldots, x_m$ and n objective functions $f_1, \ldots, f_n$:

$$\begin{aligned} \mathsf{min}\ \mathbf{y} &= (f_1(\mathbf{x}), ..., f_n(\mathbf{x})) \\ \text{subject to } \mathbf{x} &= (x_1, ..., x_m) \in X \\ \mathbf{y} &= (y_1, ..., y_n) \in Y, \end{aligned}$$

where x is decision vector, y is objective vector, X is the decision space and Y is the objective space. A solution $\mathbf{x}_1$ is said to be *dominated* by another solution $\mathbf{x}_2$, if $\mathbf{x}_1$ is not better than $\mathbf{x}_2$ in all objective functions, and $\mathbf{x}_1$ is strictly worse than $\mathbf{x}_2$ in at least one objective function, i.e.,

$$(\exists i \in \{1, \ldots, n\})\ f_i(\mathbf{x}_2) < f_i(\mathbf{x}_1) \text{ and}$$
$$(\forall j \neq i)\ f_j(\mathbf{x}_2) \leq f_j(\mathbf{x}_1).$$

A solution is Pareto-optimal when no other solution dominates it.

There are different ways to approach a multi-objective optimization problem: (1) aggregating approaches, (2) population-based non-Pareto approaches, and (3) Pareto-based approaches [14]. Aggregating approaches combine the objectives into a single one, and are advantageous to produce a single objective optimization problem. Some of the popular aggregating approaches are the weighted-sum approach, target vector optimization, and the method of goal attainment.

Population-based non-Pareto approaches are able to find a set of different non-dominated solutions concurrently. The search is guided in different directions at the same time by modifying the selection criterion to generate multiple non-dominated solutions. But non-Pareto approaches are often sensitive to the non convexity of Pareto-optimal sets. Non-Pareto algorithms often use the method of multiple linear combinations.

Pareto-based approaches explicitly use the concept of Pareto optimality to select individuals for the next generation. Most work in the area of multi-objective optimization has concentrated on the approximation of the Pareto set [60]. But generating the Pareto set is computationally expensive and is often infeasible. A number of stochastic search strategies, like evolutionary algorithms, Tabu search, simulated annealing have been developed, but these do not yield exact solutions, rather, they

find a good approximation. Evolutionary algorithms are seem to be mostly suitable for multi-objective optimization problems, because they process a set of solutions of the problem in parallel.

Chapter 3

Equilibria for Communication in Decentralized DES

Previous investigations for optimal communication policies consider set-theoretic definitions of optimality [4, 35], and quantitative approach for globally-optimal communication protocols [36]. Finding optimal communication strategies for a controller in a decentralized control setting is challenging because the best strategy depends on the choices of other controllers, all of whom are also trying to optimize their own strategies. We are interested in applying the concepts from game theory to investigate the locally-optimal communication policies. Applications of game theory try to find equilibria, where each player of the game chooses a strategy that is unlikely to change. More specifically, a game is in equilibrium if no player can improve its outcome unilaterally in the game. In this chapter, optimal strategies are considered that minimize the cost of the communication protocol for each controller. An example where communication among the controllers is necessary is given in the following.

Example 3.1. *Let us consider a problem in the space science where a number of robots navigate to explore an area of a planet. The area map is divided into square*

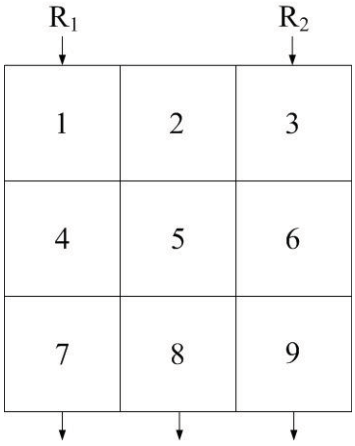
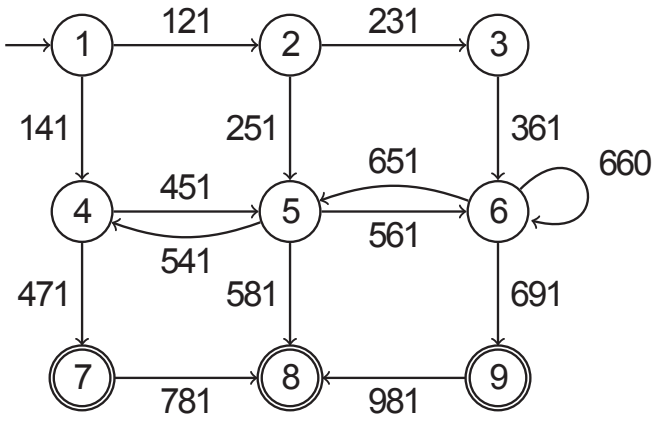


Figure 3.1: Robot navigation to explore a fixed area.

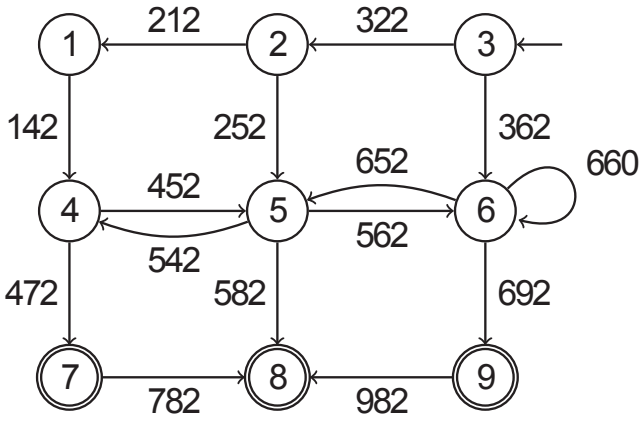
blocks, where the robots can move from one block to another, either horizontally (left-right), or vertically (up-down). Each movement is represented by a transition, and each transition occurs at a cost. The transition cost in one direction may be higher than the other direction, e.g., if the surface is steep in one direction, then the robots need more energy to move than in the other direction. In general, we can divide the area into $m \times m$ square blocks. Suppose there are n robots to explore the area, and more than n target states where the robots want to reach. Furthermore, suppose an antenna is placed in a block, which must be activated by one robot. The robot movements are subject to the following constraints:

- *no two robots can occupy the same block at any time, and*
- *no two robots activate the antenna through the same navigation.*

For simplicity, in this example, we consider a 3×3 map ($m = 3$) and $n = 2$ robots, shown in Figure 3.1. The automaton for each robot is shown in Figure 3.2. Each square block is represented as a state, and the movement from one block to another, either horizontally or vertically, is represented by a transition. We do not consider



(a)



(b)

Figure 3.2: The automaton model for (a) R_1; (b) R_2.

all possible transitions to simplify the problem. The robots are denoted by R_1 and R_2, having 3 target states (7, 8, 9) to reach.

An event xyi $\in \Sigma_i$ corresponds to a transition from state x to state y by R_i. Suppose the antenna is placed in Block 6, and antenna activation is represented by a common event 660*, which is observable and controllable by both robot controller. All other events are locally controllable (e.g., R_i controls only events that end in i). Similarly, the events are locally observable (e.g., R_i observes only events that end in i). On the way to the target states, only one robot activates the antenna. No two robots activate it at the same time, because they cannot occupy state 6 at the same time according to the first constraint.*

Robot R_1 and R_2 start from state 1 and 3 respectively, and their target states are 7,8 and 9. The system behavior L is generated by the synchronous product of $R_1||R_2$. The corresponding automaton M_L has 81 states and 234 transitions. According to the first constraint noted above, they cannot be in the same state at the same time, so that $(1,1)$, $(2,2)$, ..., $(9,9)$ *are illegal states in M_L. The specification automaton M_K is a subautomaton of M_L, minus the illegal states from M_L and the transitions associated with these states.*

The robots have a map of the area, but no robot knows about the position of other robot. As we will see later, if R_1 reaches state 7 as its target state, then R_2 must be informed about the position of R_1, so that R_2 can move to end up in either state 8 or 9. Similarly R_1 should be informed about the position of R_2. To avoid the situation when both R_1 and R_2 are at the same state, it is necessary that both robots communicate with each other about their position throughout the navigation. The example will be explored next chapter where we solve a multi-objective optimization problem to examine the trade-offs between communication and control costs.

3.1 Nash Equilibrium for Communication Protocols

The result in this section is predicated on the fact that we can express the decentralized control and synchronous communication problem as a *normal-form* game. In a normal-form game, each player has a finite set of strategies. Further, strategies are associated with a payoff function. That means, the normal-form representation specifies the players' strategy spaces and their payoff functions.

Definition 3.1. *(From [29]) A (finite, n-person)* normal-form *game is a tuple* (N, A, u)*, where:*

- *N is a finite set of n players, indexed by i;*
- *$A = A_1 \times \ldots \times A_n$, where A_i is a finite set of actions available to player i, each vector $a = \langle a_1, ..., a_n \rangle \in A$ is called an action profile;*
- *$u = (u_1, ..., u_n)$ where $u_i : A \to \mathbb{R}$ is a real-valued cost function for player i.*

□

We consider a decentralized discrete-event control and communication problem where

- I is a finite set of n controllers, indexed by i;
- $\Phi = \Phi_1 \times \ldots \times \Phi_n$, where Φ_i is a finite set of communication protocols for controller i, and $\phi = \langle \phi_1, \ldots, \phi_n \rangle \in \Phi$ is an action profile with $\phi_i = \langle \phi_{i,1}, \ldots, \phi_{i,n} \rangle$, $\phi_{i,j} : \overline{K} \to \Sigma_i^! \cup \{\varepsilon\}$; and
- $u = (u_1, ..., u_n)$ where $u_i : \Phi \to \mathbb{R}$ is a real-valued cost function for each controller i.

In this chapter and the next, we assume that the DES plant can be modeled as an acyclic automaton. Since the language of an acyclic automaton is finite, the set of actions (communication protocols) is also finite. The decentralized control and communication problem DCCP (Problem 2.1) augmented with a cost function is described below.

Problem 3.1. *Consider two regular languages K, L defined over a common alphabet Σ, where $K \subseteq L \subseteq \Sigma^*$ is observable w.r.t. L, Σ_o, Σ_c and controllable w.r.t. L, Σ_{uc}, but is not co-observable. Given a set of communication protocols $\Phi = \Phi_1 \times \ldots \times \Phi_n$ with a cost function $u_i : \Phi \to \mathbb{R}$ for each $i \in I$, find a communication protocol $\phi \in \Phi$, such that ϕ solves DCCP and for every $i \in I$ and for every communication protocol $\phi' \in \Phi$ solving DCCP that is obtained from ϕ by replacing ϕ_i with ϕ'_i, $u_i(\phi_i, \phi_{-i}) \leq u_i(\phi'_i, \phi_{-i})$.*

Note that ϕ_{-i} is the set of all communication protocols $\{\phi_k \mid k \in I \backslash \{i\}\}$. Similarly, Φ_{-i} denotes the set $\{\Phi_k \mid k \in I \backslash \{i\}\}$.

Nash equilibrium is a widely-used solution approach in game theory of predicting the outcome of strategic interactions among the players. It defines a non-cooperative multiple objective optimization strategy, where each agent optimizes its own criterion given that all other criteria of other agents are fixed. In other words, a Nash equilibrium is a collection of strategies, one for each agent in the system, where each agent knows the equilibrium strategies of the other agents, and no agent can gain anything independently by only changing its strategy. Intuitively, a Nash equilibrium represents each agent's best response to the strategies of the other agents.

We assume a uniform cost for communication to simplify the verification (i.e., same cost for every message). The cost for a communication is defined as a mapping $u_i : \Phi_i(K) \to \mathbb{R}$ such that $(\forall s \in \overline{K})$ $(\forall \sigma \in \Sigma)$,

$$u_i(\phi_{i,j}(s\sigma)) = \begin{cases} C_\sigma, & \text{if } (q_0 \xrightarrow{s} q' \xrightarrow{\sigma} q) \in T_L \text{ and } (q', \sigma, q) \in T_i^!; \\ 0, & \text{otherwise,} \end{cases}$$

where C_σ is the cost to communicate σ.

Then the total communication cost for all $s \in \overline{K}$ controller i is $\sum_{s \in \overline{K}} u_i(\phi_{i,j}(s))$.

We take all communications sent by controller i over all sequences $s \in \overline{K}$ and add the cost to get the total communication cost. For acyclic systems, this corresponds to finding protocols that have an overall minimal number of communications. In an alternative way, we can calculate the cost for each sequence $s \in \overline{K}$ and take the maximum communication cost among the sequences.

We focus on two ways in which a controller can choose its communication protocol: (i) select a single communication protocol for a controller and execute it; (ii)

randomize over the set of available protocols for a controller according to some probability distribution. The former case is called a *pure* strategy, while the latter case is called a *mixed* strategy.

A mixed strategy for a controller specifies the probability distribution used to select the protocol that a controller will use to solve the control problem. The probability distribution for a controller i is denoted by $\mathcal{P}_i : \Phi_i \to [0, 1]$, such that $\sum_{\phi_i \in \Phi_i} \mathcal{P}_i(\phi_i) = 1$.

Definition 3.2. *(Adapted from [29].) The* ***support*** *of a mixed strategy is the set of all communication protocols $\phi_i \in \Phi_i$ such that $\mathcal{P}_i(\phi_i) > 0$.*

A pure strategy is a special type of mixed strategy when the support is a single communication protocol.

We can now define Nash equilibrium in the context of Problem 3.1 [43].

Definition 3.3. *A communication protocol $\phi^* = \langle\, \phi_1^*, \ldots, \phi_n^* \,\rangle$ is a* Nash equilibrium *for decentralized supervisory control problem if*

- *for all $i \in I$, $u_i(\phi_i^*, \phi^*{}_{-i}) \leq u_i(\phi_i, \phi^*{}_{-i})$ for every communication protocol $\phi_i \in \Phi_i$;*
- *$\phi^* = (\phi_i^*, \phi^*{}_{-i})$ and $(\phi_i, \phi^*{}_{-i})$ are coherent; and*
- *ϕ^* and $(\phi_i, \phi^*{}_{-i})$ solve Problem 2.1.*

Note that each controller plays its best response to the other controllers simultaneously. That means Nash equilibrium seeks a least-costly best response communication protocol for each controller $i \in I$.

Theorem 3.2 (Adapted from [25])**.** *Every normal-form game has at least one Nash equilibrium.*

Therefore, we have the following result.

Theorem 3.3. *A decentralized discrete-event control and communication problem has at least one Nash equilibrium.*

Proof. Since a decentralized discrete-event control and communication problem can be recast as a normal-form game, the result follows from Theorem 3.1. □

3.1.1 Nash Equilibrium for Two Communicating Controllers

In [29], a novel approach to finding a sample Nash equilibrium for normal-form games is presented. This algorithm, called SEM (Support-Enumeration Method), introduces heuristics based on the space of supports and a notion of *dominated* actions that are pruned from the search space. It may still be the case that an exponential number of iterations are required to find a sample Nash equilibrium, but as noted in the literature [29], when tested on large sets of random games, SEM outperformed the standard algorithms because of the heuristics used to refine the search space.

In the acyclic case, the brute force approach would examine $2^{T_{o,i}}$possibilities of communication protocols for each controller i. In the cyclic case, we use the size of the power set of the finite transition set of $\mathcal{U}$ (discussed in Section 2.3) as an upper-bound. We will use this set of communication protocols as input to the Nash equilibrium algorithm and establish that we have (i) made the protocols coherent; (ii) selected a set of protocols such that the control problem can be solved; and (iii) submitted the now-coherent set of protocols to ensure that we have a feasible solution w.r.t. criteria for finding a Nash equilibrium.

The search for a sample Nash equilibrium requires a lexicographic ordering on the *sizes* of the supports of prospective solutions. For instance, if we identify sets of transitions that range in size from 1 (a single transition) to k_1 that controller 1 could communicate to controller 2 that would allow the latter controller to make all

of its correct control decisions, then the support size profile for controller 1 will be $1, 2, 3, \ldots, k_1$. We also include the possibility that no communication occurs. Assume that we have a similar range for controller 2 (i.e., 0 to k_2). To check all the support size profiles of the two controllers, we must check all possible pairs $(x_1, x_2) \in \{0, 1, \ldots, k_1\} \times \{0, 1, \ldots, k_2\}$. To ensure that balanced supports are examined first, the lexicographic ordering is based on the increasing order of the difference between the support sizes, and in the event of a tie, followed by the increasing order of the sum of the support sizes. Note that this approach ensures that all support sizes are considered and that the search for an equilibrium does not overlook any part of the valid solution space.

Another innovation of the approach of [29] is the elimination of solutions that will never be Nash equilibrium points, because the estimated utility function is always dominated by other solutions. Because we are seeking a minimal-cost communication protocol, we want to eliminate solutions that have larger cost than other solutions.

Definition 3.4. *A coherent communication protocol $\phi_i \in \Phi_i$ is* conditionally dominated *given the sets of available (coherent) protocols $\hat{\Phi}_{-i} \subseteq \Phi_{-i}$ for the remaining controllers, if there exists $\phi_i' \in \Phi_i$ such that for any $\phi_{-i} \in \hat{\Phi}_{-i}$, $u_i(\phi_i, \phi_{-i}) > u_i(\phi_i', \phi_{-i}) > 0$.*

We assume that when determining conditional domination, all communication protocols are coherent, or are made coherent for the purpose of testing conditional domination.

In adapting SEM for the decentralized communication and control problem, Algorithm 3.1 contains two additional steps: since we initially consider ϕ that are not coherent, we must make the prospective communication protocols coherent (where coherent versions of ϕ are denoted by φ) (Line 7) and a check to ensure that the protocols being considered for Nash equilibrium actually solve the control problem (Line 8). In the worst case, there are an exponential number of supports (in the number of

Algorithm 3.1 SEM for DES (n=2)

for all support size profiles $x = (x_1, x_2)$ sorted in increasing order $|x_1 - x_2|$, and in the event of a tie, sorted in increasing order by $x_1 + x_2$ **do**
  $\Phi_{x_1} = \{\langle \phi_{1,1}, \phi_{1,2} \rangle \in \Phi_1 \mid |\phi_{1,2}| = x_1\}$
  $\Phi_2' \leftarrow \{\phi_2 \in \Phi_2$ not conditionally dominated, given $\Phi_{x_1}\}$
  if $\forall \phi_1 \in \Phi_{x_1}$, ϕ_1 is not conditionally dominated, given Φ_2' **then**
    $\Phi_{x_2} = \{\langle \phi_{2,1}, \phi_{2,2} \rangle \in \Phi_2' \mid |\phi_{2,1}| = x_2\}$
    if $\forall \phi_1 \in \Phi_{x_1}$, ϕ_1 is not conditionally dominated, given Φ_{x_2} **then**
      $\Phi \leftarrow \{(\varphi_1, \varphi_2) \mid (\varphi_1, \varphi_2) \leftarrow$ *coherent* $(\phi_1, \phi_2), (\phi_1, \phi_2) \in \Phi_{x_1} \times \Phi_{x_2}\}$
      $\Phi \leftarrow \Phi \setminus \{(\varphi_1, \varphi_2) \mid \varphi$ does not solve the control problem$\}$
      if Program 3.1 is satisfiable for Φ **then**
        return found NE ϕ^*
      end if
    end if
  end if
end for

possible communication protocols) and thus, Algorithm 3.1 has exponential running time.

We also need a feasibility program to determine whether or not a potential solution is a true Nash equilibrium. A standard feasibility program from [29], adapted for our notation, is described by Program 3.1. The input is a set of coherent communication protocols that solve Problem 3.1 and the output is a protocol ϕ that satisfies Nash equilibrium. The first two constraints ensure that the controller has no preference for one protocol over another within the input set and it must not prefer a protocol that is not part of the input set. The third and fourth constraints check that the protocols in the input set are chosen with a non-zero probability, whereas any protocols outside of the input set are chosen with zero probability. The last constraint simply determines that there is a valid probability distribution over the communication protocols.

Note that, the structure on which we will reason about communication is isomorphic to the plant automaton in the case of acyclic systems and isomorphic to the $\mathcal{U}$-structure defined in [36] in the case of cyclic systems.

Program 3.1 Feasibility Program TGS (Test Given Supports)

Input: $\Phi = \Phi_1 \times \ldots \times \Phi_n$

Output: ϕ is a Nash equilibrium if there exist both $\phi = (\phi_1, \ldots, \phi_n)$ and $v = (v_1, \ldots, v_n)$ such that:

1: $\forall i \in I, \phi_i \in \Phi_i : \sum_{\phi_{-i} \in \Phi_{-i}} \mathcal{P}(\phi_{-i}) u_i(\phi_i, \phi_{-i}) = v_i$

2: $\forall i \in I, \phi_i \notin \Phi_i : \sum_{\phi_{-i} \in \Phi_{-i}} \mathcal{P}(\phi_{-i}) u_i(\phi_i, \phi_{-i}) \geq v_i$

3: $\forall i \in I, \phi_i \in \Phi_i : \mathcal{P}_i(\phi_i) \geq 0$

4: $\forall i \in I, \phi_i \notin \Phi_i : \mathcal{P}_i(\phi_i) = 0$

5: $\forall i \in I : \sum_{\phi_i \in \Phi_i} \mathcal{P}_i(\phi_i) = 1$

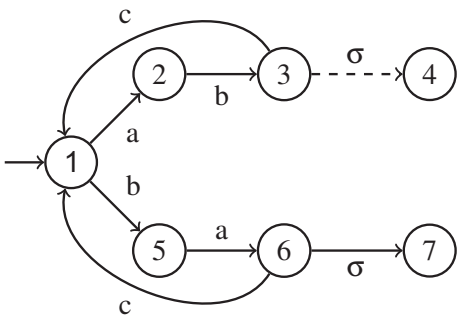


Figure 3.3: A joint M_L (all transitions) and M_K (only solid line transitions).

Example 3.2. *We illustrate Algorithm 3.1 using the automaton in Figure 3.3. Suppose that L is the language generated by the collection of all transitions, and K is the language generated by transitions with solid lines. Let $\Sigma_{o,1} = \{\mathsf{a}\}$, $\Sigma_{o,2} = \{\mathsf{b}\}$ and $\Sigma_{c,1} = \{\mathsf{a}, \sigma\}$, $\Sigma_{c,2} = \{\mathsf{b}, \sigma\}$. Note that K is not co-observable as there is no controller that controls σ that can distinguish between $\mathsf{ab}\sigma$ and $\mathsf{ba}\sigma$ or between $\mathsf{abcab}\sigma$ and $\mathsf{bacba}\sigma$.*

The input to Algorithm 3.1 w.r.t. the transition relation: $\Phi_1 = \{\langle\emptyset, \emptyset\rangle, \langle\emptyset, \{(1, \mathsf{a}, 2)\}\rangle, \langle\emptyset, \{(5, \mathsf{a}, 6)\}\rangle, \langle\emptyset, \{(1, \mathsf{a}, 2), (5, \mathsf{a}, 6)\}\rangle\}$ and $\Phi_2 = \{\langle\emptyset, \emptyset\rangle, \{\langle\{(1, \mathsf{b}, 5)\}, \emptyset\rangle, \langle\{(2, \mathsf{b}, 3)\}, \emptyset\rangle, \langle\{(1, \mathsf{b}, 5), (2, \mathsf{b}, 3)\}, \emptyset\}\rangle$. No controller sends a message to itself, so that $\phi_{i,i} = \emptyset$ for $i \in I$.

The smallest support size for Φ_1 is 0, corresponding to controller 1 sending no

information at all to controller 2, whereas the largest support size is 2, when controller 1 communicates all of its observations to controller 2. Similarly, the smallest and largest support sizes for Φ_2 *are 0 and 2. Thus, we begin by searching profiles where* $x = (0,0)$*, followed by* $x = (1,1)$*,* $x = (2,2)$*,* $x = (0,1)$*,* $x = (1,0)$*,* $x = (1,2)$*,* $x = (2,1)$*,* $x = (0,2)$ *and* $x = (2,0)$*. We will ignore the case when* $x = (0,0)$ *as this is the situation when no communication occurs. By assumption, the communication protocol corresponding to this situation,* $\Phi = (\langle\emptyset,\emptyset\rangle, \langle\emptyset,\emptyset\rangle)$*, does not solve the control problem.*

Iteration 1: $\mathbf{x} = (\mathbf{1},\mathbf{1})$*. Line 2:* $\Phi_{x_1} = \{\langle\emptyset, \{(1,\mathsf{a},2)\}\rangle, \langle\emptyset, \{(5,\mathsf{a},6)\}\rangle\}$.

Line 3: We can then determine the set Φ_2' *by calculating those elements of* Φ_2 *that are not conditionally dominated by the elements of* Φ_{x_1}*. No elements of* Φ_2 *are conditionally dominated by the elements of* Φ_{x_1}*, thus* $\Phi_2' = \Phi_2$*. Note that conditional domination is tested based on coherent communication policies. We temporarily transform elements of* Φ_2 *and* Φ_{x_1} *so that they satisfy coherency. For example, when* $\phi_1 = \langle\emptyset, \{(5,\mathsf{a},6)\}\rangle$ *and* $\phi_2 = \langle\{(2,\mathsf{b},3)\},\emptyset\rangle$*, first make* ϕ_2 *coherent w.r.t.* ϕ_1*, so that* ϕ_2 *becomes* $\langle\{(1,\mathsf{b},5),(2,\mathsf{b},3)\},\emptyset\rangle$ *and now* ϕ_1 *is already coherent w.r.t. the coherent* ϕ_2*. Line 4: No elements of* Φ_{x_1} *are conditionally dominated given* Φ_2'*. Line 5:* $\Phi_{x_2} = \{\langle\{(1,\mathsf{b},5)\},\emptyset\rangle, \langle\{(2,\mathsf{b},3)\},\emptyset\rangle\}$*. Line 6: None of the elements of* Φ_{x_1} *are conditionally dominated by the elements of* Φ_{x_2}*. Line 7:* Φ *contains the following coherent communication protocols:*

- $(\langle\emptyset, \{(1,\mathsf{a},2)\}\rangle, \langle\{(1,\mathsf{b},5)\},\emptyset\rangle)$,
- $(\langle\emptyset, \{(1,\mathsf{a},2),(5,\mathsf{a},6)\}\rangle, \langle\{(2,\mathsf{b},3)\},\emptyset\rangle)$,
- $(\langle\emptyset, \{(5,\mathsf{a},6)\}\rangle, \langle\{(1,\mathsf{b},5),\ (2,\mathsf{b},3)\},\emptyset\rangle)$.

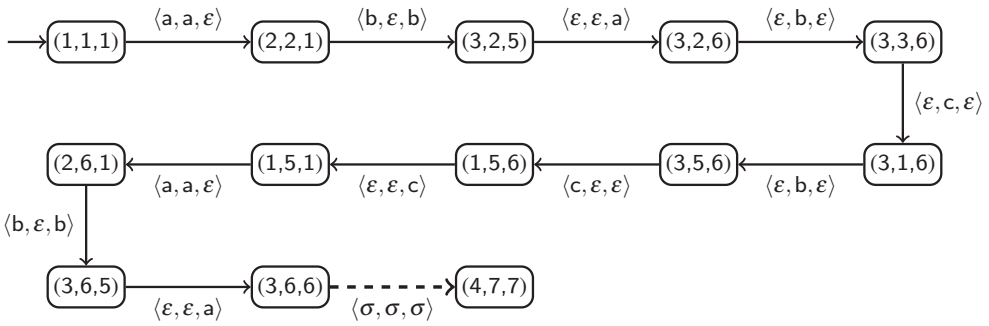


Figure 3.4: Automaton $\mathcal{U}$ for the example shown in Figure 3.3. The marked transition is denoted with a thick dashed line, where no controller can take the correct control decision.

Line 8: Each element of Φ *solves the control problem. Line 9: Since each controller has three choices that are equally likely, let* $\mathcal{P}_i(\phi_i) = \frac{1}{3}$ *for each* $\phi_i \in \Phi_i$*. Program TGS returns the Nash equilibrium communication protocol* $\phi^* = (\langle \emptyset, \{(1, a, 2)\}\rangle, \langle\{(1, b, 5)\}, \emptyset\rangle)$.

The set of communications that makes the system co-observable can be illustrated from the $\mathcal{U}$ structure. A part of $\mathcal{U}$-structure is shown in Figure 3.4. The set of alphabets for the example is $\Sigma^{\mathcal{U}} = \{\langle \mathsf{a}, \mathsf{a}, \varepsilon\rangle,\ \langle \varepsilon, \varepsilon, \mathsf{a}\rangle,\ \langle \mathsf{b}, \varepsilon, \mathsf{b}\rangle,\ \langle \varepsilon, \mathsf{b}, \varepsilon\rangle,\ \langle \mathsf{c}, \varepsilon, \varepsilon\rangle,\ \langle \varepsilon, \mathsf{c}, \varepsilon\rangle,\ \langle \varepsilon, \varepsilon, \mathsf{c}\rangle,\ \langle \sigma, \sigma, \sigma\rangle,\ \langle \mathsf{a}, \mathsf{a}, \mathsf{a}\rangle,\ \langle \mathsf{b}, \mathsf{b}, \mathsf{b}\rangle,\ \langle \varepsilon, \mathsf{b}, \mathsf{a}\rangle\}$. In the $\mathcal{U}$-structure, $F^{\mathcal{U}} = \{(\mathsf{3}, \mathsf{6}, \mathsf{6}) \xrightarrow{\langle \sigma, \sigma, \sigma\rangle} (\mathsf{4}, \mathsf{7}, \mathsf{7})\}$, shown as dashed line in Figure 3.4. The transition $(3, 6, 6) \xrightarrow{\langle \sigma, \sigma, \sigma\rangle} (4, 7, 7)$ is marked because σ must be disabled according to M_L whereas both controllers believe that σ should be enabled. An occurrence of communication in $\mathcal{U}$ is shown in Figure 3.5 (highlighted in blue color). Here Controller 1 communicates the occurrence of a to Controller 2 $((1, 5, 1), \langle \mathsf{a}, \mathsf{a}, \mathsf{a}\rangle, (2, 6, 2))$. In that case, the transitions $((1, 5, 1), \langle \varepsilon, \varepsilon, \mathsf{a}\rangle, (1, 5, 2))$ and $((1, 5, 2), \langle \mathsf{a}, \mathsf{a}, \varepsilon\rangle, (2, 6, 2))$ are pruned from the $\mathcal{U}$. Controller 2 will follow the plant behavior with the reception of a from Controller 1. That means Controller 2 believes that σ should be disabled, and it takes correct control decision regarding σ through the transition $((3, 6, 3), \langle \sigma, \sigma, \sigma\rangle, (4, 7, 4))$.

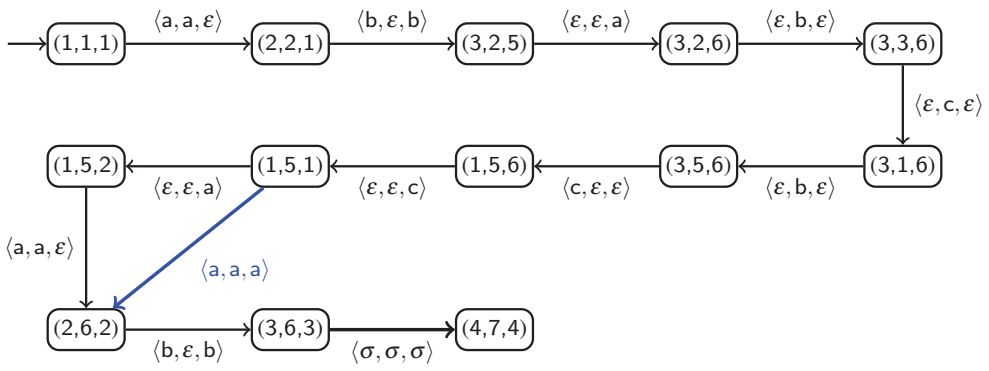


Figure 3.5: A communication occurs from $(1,5,1)$ to $(2,6,2)$, shown in blue color. Then Controller 2 takes correct control decision through the transition $((3,6,3),\langle\sigma,\sigma,\sigma\rangle,(4,7,4))$.

3.1.2 Nash Equilibrium for More Than Two Controllers

Algorithm 3.2 is a modification of Algorithm 3.1 to accommodate the case of more than two controllers. One subtle difference in Algorithm 3.2 is the change in the ordering of the support sizes: sorted first by size and then by balance. The justification for this decision comes from [29]: when there are more than two players (controllers), balance is not as important a criterion when finding a sample Nash equilibrium. Additionally, the algorithm relies on recursive backtracking (Procedure 1) to explore the search space.

Algorithm 3.2 SEM for DES when $n > 2$

1: **for all** $x = (x_1, \ldots, x_n)$ sorted in increasing order of first $\sum_{i \in I} x_i$ followed by $max_{i,k \in I}(|x_i - x_k|)$ **do**
2: $\quad \forall i\ \Phi'_i \leftarrow \emptyset$ // *uninstantiated supports*
3: $\quad \forall i\ D_{x_i} \leftarrow \{\phi_i \in \Phi_i \mid \sum_{k \in I} |\phi_{i,k}| = x_i\}$ // *domain of supports*
4: $\quad$ **if** RecursiveBacktracking$(\Phi', D_x, 1)$ returns NE ϕ^* **then**
5: $\quad\quad$ **return** ϕ^*
6: $\quad$ **end if**
7: **end for**

Procedure 1 RecursiveBacktracking

Input: $\Phi' = \Phi'_1 \times \ldots \times \Phi'_n$; $D_x = (D_{x_1}, \ldots, D_{x_n})$; i
Output: Nash equilibrum ϕ^* or failure

1: **if** $i = n + 1$ **then**
2: $\quad \Phi' \leftarrow \{(\varphi_1, \ldots, \varphi_n) \mid (\varphi_1, \ldots \varphi_n) \leftarrow \mathit{coherent}(\phi_1, \ldots, \phi_n), \forall \phi_i \in \prod_{i \in I} \Phi'_i\}$
3: $\quad \Phi' \leftarrow \Phi' \setminus \{(\varphi_1, \ldots, \varphi_n) \mid \varphi$ does not solve the control problem$\}$
4: $\quad$ **if** Program 3.1 is satisfiable for Φ' **then**
5: $\quad\quad$ **return** found NE ϕ^*
6: $\quad$ **else**
7: $\quad\quad$ **return** failure
8: $\quad$ **end if**
9: **else**
10: $\quad \Phi'_i \leftarrow D_{x_i}$
11: $\quad D_{x_i} \leftarrow \emptyset$
12: $\quad$ **if** IRDCP$(\Phi'_1, \ldots, \Phi'_i, D_{x_{i+1}}, \ldots, D_{x_n})$ succeeds **then**
13: $\quad\quad$ **if** RecursiveBacktracking$(\Phi', D_x, i + 1)$ returns NE ϕ^* **then**
14: $\quad\quad\quad$ **return** found NE ϕ^*
15: $\quad\quad$ **end if**
16: $\quad$ **end if**
17: **end if**
18: **return** failure

For the case of more than two controllers, it is slightly more complicated to remove dominated communication protocols. The input to Procedure 2 (line 11 in Procedure 1) is now the set of domains for support of each controller. When the support for a controller is instantiated, the domain contains only the instantiated support. For all other controllers, the domain contains the supports of size x_i that were not previously eliminated by earlier calls to this procedure.

Procedure 2 Iterated Removal of Dominated Communication Protocols (IRDCP)

Input: $D_x = (D_{x_1}, \ldots, D_{x_n})$
Output: Updated domains or failure

```
1: repeat
2:    dominated← false
3:    for all i ∈ I do
4:       for all φ_i ∈ D_{x_i} do
5:          for all φ'_i ∈ Φ_i do
6:             if φ_i is conditionally dominated by φ'_i given D_{-x_i} then
7:                D_{x_i} ← D_{x_i} \ {φ_i}
8:                dominated← true
9:                if D_{x_i} = ∅ then
10:                  return failure
11:               end if
12:            end if
13:         end for
14:      end for
15:   end for
16: until dominated = false
17: return D_x
```

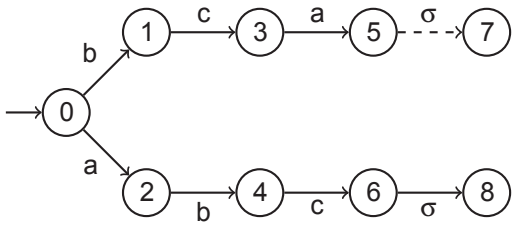


Figure 3.6: A finite-state automaton for Example 3.2.

Example 3.3. *We illustrate Algorithm 3.2 using the automaton in Figure 3.6. As the previous examples, let L be the language generated by the collection of all transitions,*

and K be the language of only the solid-line transitions. Suppose that $\Sigma_{o,1} = \{a\}$, $\Sigma_{o,2} = \{b\}$, and $\Sigma_{o,3} = \{c\}$, while $\Sigma_{c,1} = \{a, \sigma\}$, $\Sigma_{c,2} = \{b, \sigma\}$, and $\Sigma_{c,3} = \{c, \sigma\}$. K is not co-observable since none of the controllers can make the correct control decisions regarding σ.

The support sizes for each controller ranges from 0 to 2, thus when checking the possible support sizes there are 16 possibilities, beginning with $x = (0, 0, 0)$, which we have previously indicated we would ignore, and ending with $x = (2, 2, 2)$. The first three supports (all with a cumulative sum of 1 and a max difference of 1) to examine are of size $(1, 0, 0)$, $(0, 1, 0)$ and $(0, 0, 1)$. We will begin with $(0, 1, 0)$.

Algorithm 2, iteration 1: *$x = (0, 1, 0)$. Line 2: $\Phi'_1 = \Phi'_2 = \Phi'_3 = \emptyset$. Line 3: $D_{x_1} = \{\langle\emptyset, \emptyset, \emptyset\rangle\}$, $D_{x_2} = \{\langle\{(0, b, 1)\}, \emptyset, \emptyset\rangle, \langle\{(2, b, 4)\}, \emptyset, \emptyset\rangle, \langle\emptyset, \emptyset, \{(0, b, 1)\}\rangle, \langle\emptyset, \emptyset, \{(2, b, 4)\}\rangle\}$, $D_{x_3} = \{\langle\emptyset, \emptyset, \emptyset\rangle\}$. Line 4: Call to Procedure 1 with Φ', D_x and 1.*

Procedure 1, call for $i = 1$. *Line 9: $\Phi'_1 = \{\langle\emptyset, \emptyset, \emptyset\rangle\}$. Line 10: $D_{x_1} = \emptyset$. Line 11: Call to Procedure 2 with Φ'_1, D_{x_2} and D_{x_3}.*

Procedure 2, call for *$D_{x_1} = \{\langle\emptyset, \emptyset, \emptyset\rangle\}$, $D_{x_2} = \{\langle\{(0, b, 1)\}, \emptyset, \emptyset\rangle, \langle\{(2, b, 4)\}, \emptyset, \emptyset\rangle, \langle\emptyset, \emptyset, \{(0, b, 1)\}\rangle, \langle\emptyset, \emptyset, \{(2, b, 4)\}\rangle$, $D_{x_3} = \{\langle\emptyset, \emptyset, \emptyset\rangle\}$. For all three iterations of Lines 4 - 10, no communication protocols are removed from the three domains. The vector D_x is returned unchanged.*

Procedure 1, return to line 11. *Line 12: Recursive call to Procedure 1 with Φ', D_x and 2.*

When i =2 and 3, no communication protocols are removed by these procedure calls. Then make the recursive call to Procedure 1 with Φ', D_x and 4.

Procedure 1, call for $i = 4$. *Line 2: At this point, there are six different possible communication protocols: $\{\langle\emptyset, \emptyset, \emptyset\rangle\} \times \{\langle\{(0, b, 1)\}, \emptyset, \emptyset\rangle, \langle\{(2, b, 4)\}, \emptyset, \emptyset\rangle, \langle\emptyset, \emptyset, \{(0, b, 1)\}\rangle, \langle\emptyset, \emptyset, \{(2, b, 4)\}\rangle\} \times \{\langle\emptyset, \emptyset, \emptyset\rangle\}$. When we make these combinations coherent, Φ' contains two unique protocols:*

- $(\langle\emptyset,\emptyset,\emptyset\rangle, \langle\{(0,\mathsf{b},1),\ (2,\mathsf{b},4)\},\emptyset,\emptyset\rangle,\ \langle\emptyset,\emptyset,\emptyset\rangle)$,
- $(\langle\emptyset,\emptyset,\emptyset\rangle,\ \langle\emptyset,\emptyset,\{(0,\mathsf{b},1),(2,\mathsf{b},4)\}\rangle,\ \langle\emptyset,\emptyset,\emptyset\rangle)$.

Line 3: Only one of these communication protocols solves the control problem, so now $\Phi' = \{(\langle\emptyset,\emptyset,\emptyset\rangle,\ \langle\{(0,\mathsf{b},1),(2,\mathsf{b},4)\},\emptyset,\emptyset\rangle,\ \langle\emptyset,\emptyset,\emptyset\rangle)\}$. *Line 4: Call to Program 1 with* Φ' *returns NE* $\phi^* = \{(\langle\emptyset,\emptyset,\emptyset\rangle,\ \langle\{(0,\mathsf{b},1),(2,\mathsf{b},4),\emptyset,\emptyset\}\rangle,\ \langle\emptyset,\emptyset,\emptyset\rangle)\}$. *This is a pure strategy.*

The algorithms for finding sample Nash equilibrium for communication protocols produce locally-optimal solutions. Both algorithms terminate when a first Nash equilibrium point is found.

3.2 Pareto Optimality for Communication Protocols

Pareto optimality is another important concept in game theory. When a strategy is Pareto-efficient or Pareto-optimal, no player can be better off without making at least one player worse off w.r.t the payoff function. We use the same cost function as of Nash equilibrium, and then we define Pareto optimality in decentralized DES according to the formulation of a *normal-form* game as below.

Definition 3.5. *A communication protocol* $\phi^* = (\phi_1^*,\ \ldots,\phi_n^*)$ *is Pareto-optimal in decentralized supervisory control problem if there exists no other communication protocol* $\phi = (\phi_1,\ \ldots,\phi_n)$ *such that*

$$u_i(\phi_i, \phi_{-i}) \leq u_i(\phi_i^*, \phi^*{}_{-i}) \quad \textit{for all } i \in I, \tag{3.1}$$

with at least one inequality strict, subject to the communication protocols Φ^* *and* Φ *being coherent and solve Problem 2.1.*

□

In other words, a communication protocol ϕ^* is Pareto-optimal if

- $(\forall i \in I) u_i(\phi_i, \phi^*_{-i}) \leq u_i(\phi^*_i, \phi^*_{-i}) \Rightarrow (\exists j \in I) u_j(\phi_j, \phi^*_{-j}) \geq u_j(\phi^*_j, \phi^*_{-j})$;
- ϕ^*, (ϕ_i, ϕ^*_{-i}), and (ϕ_j, ϕ^*_{-j}) are coherent; and
- ϕ^*, (ϕ_i, ϕ^*_{-i}), and (ϕ_j, ϕ^*_{-j}) solve Problem 2.1.

While Nash equilibrium are also Pareto-optimal in some problems, they do not necessarily coincide in decentralized control and communication problem. We consider Example 3.1 to show that Nash equilibrium does not imply Pareto optimality in the decentralized control and communication problem. Using the cost functions defined in the example, the communication cost for both controllers are shown in Table 3.1. Note that when a controller communicates an event σ after a sequence s, it must also communicate σ after all sequences s' indistinguishable to s (due to coherency). Then we consider this to be a single communication and therefore a unit cost is incurred. An infinite cost is assumed if the communication protocols are not coherent or they do not solve the control problem. For example, for the communication protocol $(\langle \emptyset, \{(1, \mathsf{a}, 2), (5, \mathsf{a}, 6)\} \rangle, \langle \emptyset, \emptyset \rangle)$, Controller 1 communicates both of its observations through the sequences $s = \mathsf{ab}$ and $s' = \mathsf{ba}$, but Controller 2 communicates nothing. Since there is no communication from Controller 2, s and s' are indistinguishable to Controller 1. So that it seems to be a single communication to Controller 1, and a unit cost is incurred. For the communication protocol $(\langle \{(1, \mathsf{b}, 5)\}, \emptyset \rangle, \langle \emptyset, \{(1, \mathsf{a}, 2), (5, \mathsf{a}, 6)\} \rangle)$, Controller 2 also communicates b through s'. In that case, s and s' are no longer indistinguishable to Controller 1 and a cost of 2 is incurred.

Table 3.1: Communication cost of two controllers for the decentralized DES shown in Figure 3.1, appears as communication cost of Controller 1, Communication cost of Controller 2.

Φ_1 \ Φ_2	$\langle\emptyset,\emptyset\rangle$	$\langle\{(1,\mathsf{b},5)\},\emptyset\rangle$	$\langle\{(2,\mathsf{b},3)\},\emptyset\rangle$	$\langle\{(1,\mathsf{b},5),(2,\mathsf{b},3)\},\emptyset\rangle$
$\langle\emptyset,\emptyset\rangle$	∞,∞	∞,∞	∞,∞	0,1
$\langle\emptyset,\{(1,\mathsf{a},2)\}\rangle$	∞,∞	1,1	∞,∞	1,2
$\langle\emptyset,\{(5,\mathsf{a},6)\}\rangle$	∞,∞	∞,∞	∞,∞	1,1
$\langle\emptyset,\{(1,\mathsf{a},2),(5,\mathsf{a},6)\}\rangle$	1,0	2,1	1,1	2,2

With the cost functions defined above, it is straightforward to see that not all Nash equilibrium points are Pareto-optimal. We have three Nash equilibrium points:

- $(\langle\emptyset,\{(1,\mathsf{a},2),(5,\mathsf{a},6)\}\rangle,\ \langle\emptyset,\emptyset\rangle)$,
- $(\langle\emptyset,\emptyset\rangle,\ \langle\{(1,\mathsf{b},5),(2,\mathsf{b},3)\},\emptyset\rangle)$,
- $(\langle\emptyset,\{(1,\mathsf{a},2)\}\rangle,\ \langle\{(1,\mathsf{b},5)\},\emptyset\rangle)$.

Among the solutions, the first two are Pareto-optimal. The last point is not Pareto-optimal according to Definition 3.5, because if Controller 1 changes its communication from $\{(1,\mathsf{a},2)\}$ to $\emptyset$ (or Controller 2 changes it communication from $\{(1,\mathsf{b},5)\}$ to $\emptyset$), the cost function is improved from $(1,1)$ to $(0,1)$ (or $(1,0)$). Hence, the inequality of Equation 3.1 is strict for at least one controller.

We use both Nash equilibrium and Pareto-optimal solutions to analyze the outcome of strategic interaction among several controllers. When more than one controller communicates with another to make a local decision in a decentralized DES control problem, each controller takes into account the decision of other controllers. Nash equilibrium and Pareto-optimal are also used for multi-objective optimization problems. Many real-life problems contain multiple conflicting objectives. For example, we may want to adjust the energy usage of several mobile robots so that they will arrive to a previously-identified place at a specified time. This cannot be solved

by locally-optimal methods because there could be several local optima which are not promising solutions. The objectives are in conflict with each other if an improvement in one objective deteriorates the other objective. Therefore, Pareto-optimal is widely used in solving such type of problems. In the next chapter, we will use the concept of Pareto-optimal solution to solve multi-objective optimization problem in decentralized DES.

Chapter 4

Multi-Objective Optimization for Decentralized DES

When incorporating communication into the decentralized control problem, there may be a cost advantage to synthesizing only part of the specification, instead of realizing the entire specification with a costly communication protocol. Hence we want to investigate the trade-off between the cost of an exact control solution achieved with communication and an approximate solution, where penalties are assessed for achieving a sublanguage of a desired controllable and observable specification, with a possibly cheaper communication policy. To that end we are interested in a class of quantitative decentralized discrete-event control problems where we have more than one function or *objective* to optimize simultaneously, which leads to a *multi-objective optimization* problem [50].

Evolutionary algorithms are well-suited for addressing multi-objective optimization problems, since they are based on biological processes which are inherently multi-objective [3]. They are found to be highly effective in finding a set of promising solutions. They converge very fast to the set of optimal solutions because of ease of implementation. Each iteration of an evolutionary algorithm involves a competitive

selection that eliminates poor solutions. Existing good solutions are combined with the new potential solutions for the next iteration. An initial population of possible solutions are considered, and a measure of their *fitness* determines whether a member of the population will be involved in the formulation of the next generation of the population. Just as in natural adaptation, over a period of many generations, a population of solutions evolves that is "closer" to an optimal solution than their predecessors. We use a modified version of the *Non-dominated Sorting Genetic Algorithm* (NSGA-II) [10], which has already proven useful for a diverse range of control problems (e.g., [15, 58]).

4.1 Multi-Objective Optimization: Decentralized Control with Communication

A multi-objective optimization problem is characterized by the requirement to optimize multiple competing objectives. For a decentralized DES control and communication problem, we have two objectives: (i) the cost of a control law; and (ii) the cost of a communication policy. Ideally, we would like the joint decisions of the controllers in the presence of the full communication protocol to allow exactly the specification K to occur; however, in the presence of a costly communication protocol, it might be more efficient to allow some subset of K to occur. But it may be the case that the penalty for disabling certain sequences within K is more expensive than the communication required to enable the same sequences. We are interested in a quantitative analysis of the trade-off between the cost of imperfectly controlling the system by removing some (potentially costly) communications and the cost of taking the exact control solution with the full communication protocol.

4.1.1 Control Cost Function

We adapt the centralized control cost function of [49] to the case of the control cost function for a decentralized controller $i \in I$. We consider three basic costs that controller $i \in I$ can incur to control a system:

- We assume that there is a basic cost for a transition, which can be considered to be the cost to enable the transition, denoted by $e_i : T \to \mathbb{R}^+ \cup \{0\}$.
- There is a cost to disable a transition that would otherwise take the system out of K, $d_i : T \to \mathbb{R}^+ \cup \{0, \infty\}$.
- Since our control objective is to have the fusion of $\Gamma_i^?$ (for $i \in I$) allow exactly K to occur, when a transition is disabled that would otherwise keep the system in K, the cost to disable is incurred, plus an additional penalty is assessed: $p_i^K \in \mathbb{R}^+ \cup \{0\}$.

We assume that a disablement (and any associated penalty) or an enablement cost lies in the range of $[0, \infty)$. When controller i tries to disable an uncontrollable event $\sigma \in \Sigma \backslash \Sigma_{c,i}$, a penalty of ∞ is levied. Recall, when that controller i is not sure whether or not the system leaves K via a controllable transition, the default decision is to enable the transition. In this case, although controller i does not know the correct control decision, the cost incurred is that of enablement. Because we will consider only control laws that under the full communication protocol Φ keep the system within K, it is not possible that all controllers enable a transition that takes the system out of K.

The control cost $v_i : \Gamma_i^? \times \overline{K} \times T \to \mathbb{R}^+ \cup \{0, \infty\}$ describes the cost incurred by controller i for the occurrence of transition $(q, \sigma, q') \in T$ after the sequence $s \in \overline{K}$

such that $q_0 \xrightarrow{s} q \xrightarrow{\sigma} q'$:

$$v_i(\Gamma_i^?, s, (q,\sigma,q')) = \begin{cases} e_i(q,\sigma,q'), & \text{if } (q,\sigma,q') \in \Gamma_i^?(\pi_i^?(s)); \\ d_i(q,\sigma,q'), & \text{if } (q,\sigma,q') \notin \Gamma_i^?(\pi_i^?(s)) \text{ and} \\ & [\![s]\!]_i^? \sigma \cap \bar{K} = \emptyset; \\ d_i(q,\sigma,q') + p_i^K, & \text{if } (q,\sigma,q') \notin \Gamma_i^?(\pi_i^?(s)) \text{ and} \\ & [\![s]\!]_i^? \sigma \cap \bar{K} \subseteq \bar{K}; \\ \infty, & \text{otherwise.} \end{cases} \tag{4.1}$$

When controller i enables a transition after $s \in \overline{K}$ according to its control law, the basic enablement cost is incurred. It incurs only a disable cost if it disables a transition which takes the system outside K, otherwise a penalty is imposed in addition to disable the transition if it keeps the system in K.

The total control cost for all $s \in \overline{K}$ for controller $i \in I$ is then

$$V_i(\Gamma_i^?, \overline{K}, T) = \sum_{s \in \overline{K}} \sum_{(q,\sigma,q') \in T} v_i(\Gamma_i^?, s, (q,\sigma,q')).$$

Similar to Chapter 3, we study finite languages, i.e., acyclic automata. Therefore, the total control cost is finite.

4.1.2 Communication Cost Function

Each decentralized controller i has a communication protocol $\phi_i = \langle \phi_{i,1}, \ldots, \phi_{i,j}, \ldots, \phi_{i,n} \rangle$. We assume that a basic cost for communication is incurred each time controller i sends a message to controller j, denoted by $\mathsf{com}_i : T \to \mathbb{R}^+ \cup \{0\}$. The cost of controller i's communication protocol $u_i : \Phi_i \times \overline{K} \times T \to \mathbb{R}^+ \cup \{0\}$ assumes that a cost is incurred only when a communication is sent by controller i through a sequence

$s = s'\sigma \in \Sigma^*$ with $q_0 \xrightarrow{s'} q \xrightarrow{\sigma} q'$:

$$u_i(\Phi_i, s, (q, \sigma, q')) = \begin{cases} \mathsf{com}_i(\sigma), & \text{if } (\exists j \in I)\ \phi_{i,j}(\pi_i^?(s)) = \sigma; \\ 0, & \text{otherwise.} \end{cases}$$

It is possible that two identical messages sent by different controllers incur different local costs. There is no cost for the reception of a message and it may be the case that the cost for a point-to-point communication differs from that of a broadcast. For simplicity, we assume that when controller i communicates the same message to more than one controller, a single cost is incurred, regardless of the number of recipients.

The communication cost for all $s \in \overline{K}$ for controller $i \in I$ is then

$$U_i(\Phi_i, \overline{K}, T) = \sum_{s \in \overline{K}} \sum_{(q,\sigma,q') \in T} u_i(\Phi_i, s, (q, \sigma, q')).$$

The communication cost could include, for instance, the required power consumption, bandwidth or CPU time.

4.2 Objective Functions and Multi-Objective Optimization Problems in DES

To ensure that the objective functions are defined across the same domain, we adjust the definition of U_i and V_i accordingly, so that both functions are defined over $\Gamma_i^? \times \Phi_i \times \overline{K} \times T$. We define the following two objective functions for our first two problems.

Objective 1. *The first objective function is the cost of the control decisions each controller $i \in I$ makes for its observation of $\overline{K}$:*

$$O_{1,i}(\Gamma_i^?, \Phi_i, \overline{K}, T) = V_i(\Gamma_i^?, \Phi_i, \overline{K}, T).$$

Objective 2. *The second objective function is the cost of the communication protocol that each controller uses to assist the other controllers in reaching the control objective for* $\overline{K}$*:*

$$O_{2,i}(\Gamma_i^?, \Phi_i, \overline{K}, T) = U_i(\Gamma_i^?, \Phi_i, \overline{K}, T).$$

4.2.1 Optimization w.r.t. the Cost Functions of Each Controller

We consider an optimization problem with a finite set of control laws $\Gamma_i^?$, and a finite set of communication protocols Φ_i. In this problem, each decentralized controller i must optimize the cost of its local control law v_i and the cost of its local communication policy u_i. Note that the costs considered here are associated with a controller's local decision regarding the occurrence of a transition, and not for the eventual fusion of the control decisions [44].

Problem 4.1. *Given two regular languages* K*,* L *defined over a common alphabet* Σ*, where* $K \subseteq L$*. Find* $\Gamma_i^?$ *and* Φ_i *(*$\forall i \in I$*) to*

$$\min_{\Gamma_i^? \times \Phi_i} f_i(\Gamma_i^?, \Phi_i, \overline{K}, T) = [O_{1,i}(\Gamma_i^?, \Phi_i, \overline{K}, T), O_{2,i}(\Gamma_i^?, \Phi_i, \overline{K}, T)]^T, \tag{4.2}$$

subject to $\emptyset \subset \cap_{i=1}^n \Gamma_i^?(\pi_i^?(\overline{K})) \subseteq \overline{K}$*, and* $\Gamma^? = \langle \Gamma_1^?, \ldots, \Gamma_n^? \rangle, \Phi = \langle \Phi_1, \ldots, \Phi_n \rangle$ *are coherent.*

4.2.2 Evolutionary Algorithms Applied to Decentralized DES

Evolutionary algorithms are used for solving multi-objective optimization problems. The idea of such algorithms is the following: beginning with an initial population of possible solutions, each solution is assigned a fitness value indicating its quality.

The fitness value determines which solutions will be selected for breeding the next *generation*. These candidates are mutated and combined to produce new "children" candidate solutions. The evolutionary process continues until either an optimal set of solutions is determined or a pre-determined number of generations is exceeded.

There may not exist a single best solution in the multi-objective optimization problem. Instead, evolutionary algorithms define a *set* of best solutions for these problems. The class of evolutionary algorithms that we are using produces a *Pareto front* of the candidate solutions. Solutions that comprise the front are said to be *Pareto-optimal* or *non-dominated*.

Most evolutionary multi-objective optimization approaches such as strength Pareto evolutionary algorithm (SPEA) [59], non-dominated sorting genetic algorithms (NSGA-II) [10], and the Pareto-archived evolution strategy (PAES) [17] use the concept of domination. We solve Problem 4.1 by applying the evolutionary algorithm NSGA-II [10]. Unlike some of the other approaches, NSGA-II keeps an archive of the best b solutions generated so far: all children of generation k compete for membership in generation $k+1$ with generation k. In this way, good solutions from a previous generation are preserved. The algorithm also features a strong fitness assignment procedure for each solution, based on the number of solutions dominated by it and it is dominated by. The main algorithms required to implement NSGA-II are presented in [21]. Here we describe how the algorithms work.

We create an initial population of pairs of possible control laws and communication protocols $\langle \Gamma_i, \Phi_i \rangle$. In accordance with NSGA-II, each member of the population is assigned a fitness value, calculated w.r.t. the values of the two objective functions. From the initial population, candidate members for the Pareto front are calculated: those members of the population that are non-dominated. The next generation is calculated following a "breeding" process of elements from the preceding generation.

Algorithm 4.1 *NSGA-II* Algorithm

$P \leftarrow \{P_1, \ldots, P_m\}$ // Build initial population
AssessFitness(P) // Compute the objective values for P
$R \leftarrow \langle \ldots \rangle$ Pareto front ranks of P
for each front rank $R_i \in R$ **do**
  Compute Sparsities of Individuals in R_i
end for
BestFront $\leftarrow$ Pareto Front of P
repeat
  $W \leftarrow$ Breed(P) // use Algorithm 4.5 for selection (typically with tournament size of 2)
  AssessFitness(W) // Compute the objective values for W
  $W \leftarrow W \cup P$
  $P \leftarrow \emptyset$
  $R \leftarrow$ Compute Pareto front ranks of W
  BestFront $\leftarrow$ Pareto f of W
  for each front rank $R_i \in R$ **do**
    Compute Sparsities of Individuals in $\langle R_i \rangle$
    if $||P|| + ||R_i|| \geq m$ **then**
      $P \leftarrow P \cup$ the Sparsest $m - ||P||$ individuals in R_i, breaking ties arbitrarily;
      break from the **for** loop
    else
      $P \leftarrow P \cup R_i$
    end if
  end for
until *BestFront* is the ideal Pareto front or we have run out of time
for each individual $p_i \in$ *BestFront* **do**
  $\Gamma_i \leftarrow \{(\delta_1, \ldots, \delta_n) \mid (\delta_1, \ldots, \delta_n) \leftarrow \mathit{coherent}(\gamma_1, \ldots, \gamma_n),\ \gamma_i \in \prod_{i \in I} \Gamma_i\}$ // Make the control actions coherent
  $\Phi_i \leftarrow \{(\varphi_1, \ldots, \varphi_n) \mid (\varphi_1, \ldots \varphi_n) \leftarrow \mathit{coherent}(\phi_1, \ldots, \phi_n), \phi_i \in \prod_{i \in I} \Phi_i\}$ // Make the communication protocol coherent
  $P' \leftarrow P \setminus \{\langle \delta_1, \varphi_1 \rangle, \ldots, \langle \delta_n, \varphi_n \rangle \mid \langle \delta, \varphi \rangle$ does not solve the control problem$\}$ // Ensure all individuals in the *BestFront* solve the control problem
end for
return *BestFront*

Coherency of potential control and communication solutions is determined during breeding. Those members of the previous and current population with the best fitness values are then ranked and reorganized into a new candidate set for the Pareto front. An archive of non-dominated solutions will maintain the diversity. This process continues until either we exceed the number of pre-specified generations or the ideal Pareto front is found.

In adapting *NSGA-II* for the decentralized control and communication problem, Algorithm 4.1 contains few additional steps. Since the control decisions and communication protocols are not coherent in the initial population, we must make them coherent to satisfy the constraints of Problem 4.1. We make the prospective control decisions coherent in Line 23 (where coherent versions of γ are denoted by δ). Similarly we make the prospective communication protocols coherent (where coherent versions of ϕ are denoted by φ) (Line 24). We also check to ensure that the solutions being considered for Pareto-optimal actually solve the control problem (Line 25).

The individuals in population P are ranked according to the level of non-domination using Algorithms 4.2 and 4.3. Each solution is compared with every other solution in the population to find whether it is dominated. The solutions which are not dominated by any other solutions are ranked as first front. The same procedure is repeated to find the individuals of the subsequent fronts. We then define sparsity to assign a distance measure of individuals in the same Pareto front using Algorithm 4.4.

Algorithm 4.5 uses sparsity to find the crowding distance of each solution in the Pareto front. It guides the selection process of the algorithm towards a uniformly spread out Pareto-optimal front. The algorithm defines a tournament selection which breaks ties in the Pareto front rank using sparsity. We prefer to select an individual with a lower rank when the individuals are in different fronts. Otherwise, if two individuals belong to the same front, then we prefer one with more sparsity (which is

Algorithm 4.2 Computing a Pareto Non-Dominated Front

1: $G \leftarrow \{G_1, \ldots, G_m\}$ // Group of individuals to compute the front among:often the population
2: $O \leftarrow \{O_1, \ldots, O_n\}$ // objectives to assess with
3: $F \leftarrow \emptyset$ // The front
4: **for** each individual $G_i \in G$ **do**
5: $F \leftarrow F \cup \{G_i\}$ // Assume G_i will be in the front
6: **for** each individual $F_j \in F$ other than G_i **do**
7: **if** F_j Pareto dominates G_i given O **then**
8: $F \leftarrow F - \{G_i\}$ // G_i will not stay in the front
9: **break** from inner **for** loop
10: **else**
11: **if** G_i Pareto dominates F_j given O **then**
12: $F \leftarrow F - \{F_j\}$ // An existing front member knocked out
13: **end if**
14: **end if**
15: **end for**
16: **end for**
17: **return** F

Algorithm 4.3 Front Rank Assignment by Non-Dominated Sorting

1: $P \leftarrow$ Population
2: $O \leftarrow \{O_1, \ldots, O_n\}$ // objectives to assess with
3: $P' \leftarrow P$ // Gradually remove individuals from P'
4: $R \leftarrow \langle\rangle$ // Initially empty ordered vector of Pareto Front Ranks
5: $i \leftarrow 1$
6: **repeat**
7: $R_i \Leftarrow$ Pareto non-dominated front of P' using O
8: **for** each individual $A \in R_i$ **do**
9: ParetoFrontRank(A) $\leftarrow i$
10: $P' \leftarrow P' - \{A\}$ // Remove the current front from P'
11: **end for**
12: $i \leftarrow i + 1$
13: **until** P' is empty
14: **return** R

Algorithm 4.4 Sparsity Assignment Algorithm

1: $R \leftarrow \langle \ldots \rangle$ // provided Pareto front ranks of individuals
2: $O \leftarrow \langle O_1, \ldots, O_n \rangle$ // objectives to assess with
3: Range(O_i) // function providing the range (max - min) of possible values for a given objective O_i
4: **for** each Pareto front rank $F \in R$ **do**
5: **for** each individual $F_j \in F$ **do**
6: Sparsity(F_j) $\leftarrow 0$
7: **end for**
8: **for** each objective $O_i \in O$ **do**
9: $F' \leftarrow F$ sorted by ObjectiveValue given objective O_i
10: Sparsity(F'_1) $\leftarrow \infty$
11: Sparsity($F'_{||F||}$) $\leftarrow \infty$ // Each end is really sparse.
12: **for** $j \leftarrow 2$ to $||F'|| - 1$ **do**
13: $$\text{Sparsity}(F'_j) \leftarrow \text{Sparsity}(F'_j) + \frac{\mathsf{ObjValue}(O_i, F'_{j+1}) - \mathsf{ObjValue}(O_i, F'_{j-1})}{\text{Range}(O_{i+1})}$$
14: **end for**
15: **end for**
16: **end for**
17: **return** R with Sparsities assigned

Algorithm 4.5 Non-Dominated Sorting Lexicographic Tournament Selection With Sparsity

1: $P \leftarrow$ Population with Pareto front ranks assigned
2: *Best* $\leftarrow$ individual picked at random from P with replacement
3: $t \leftarrow$ tournament size, $t \geq 1$
4: **for** $i \leftarrow 2$ to t **do**
5: *Next* $\leftarrow$ individual picked at random from P with replacement
6: **if** ParetoFrontRank(*Next*) $<$ ParetoFrontRank(*Best*) **then**
7: // Lower ranks are better.
8: *Best*$\leftarrow$ *Next*
9: **else**
10: **if** ParetoFrontRank (*Next*) = ParetoFrontRank (*Best*) **then**
11: **if** Sparsity(*Next*) $>$ Sparsity(*Best*) **then**
12: // Higher sparsities are better
13: *Best* $\leftarrow$ *Next*
14: **end if**
15: **end if**
16: **end if**
17: **end for**
18: **return** *Best*

located in a region with a fewer number of solutions).

Note that at the conclusion of the algorithm, we have a set of optimal solutions from which to choose. In particular, solutions to Problem 4.1 provide Pareto-optimal costs with respect to communicating controller i. Thus, the designer is free to choose a solution that favours one controller over another, based on the Pareto fronts produced for each controller.

Definition 4.1. $(\Gamma_i^?, \Phi_i)$ *is Pareto-optimal for controller* i *iff*
$\min\limits_{\Gamma_i^? \times \Phi_i} f_i(\Gamma_i^?, \Phi_i, \overline{K}, T) = [O_{1,i}(\Gamma_i^?, \Phi_i, \overline{K}, T),\ O_{2,i}(\Gamma_i^?, \Phi_i, \overline{K}, T)]^T$.

Example 4.1. *We consider Example 3.1 discussed in Chapter and impose the following cost functions. Three basic costs are assigned for the control cost function:*

$$e_1(q, \sigma, q') = \begin{cases} 50, & \text{if } q \in \{1, 2, 3\}; \\ 100, & \text{if } q \in \{4, 5, 6\}; \\ 150, & \text{if } q \in \{7, 9\}; \end{cases} \tag{4.3}$$

$$e_2(q, \sigma, q') = \begin{cases} 100, & \text{if } q \in \{1, 2, 3\}; \\ 150, & \text{if } q \in \{4, 5, 6\}; \\ 200, & \text{if } q \in \{7, 9\}; \end{cases} \tag{4.4}$$

$$d_1(q, \sigma, q') = \begin{cases} 500, & \text{if } q \in \{1, 2, 3\}; \\ 450, & \text{if } q \in \{4, 5, 6\}; \\ 400, & \text{if } q \in \{7, 9\}; \end{cases} \tag{4.5}$$

$$d_2(q,\sigma,q') = \begin{cases} 550, & \textit{if } q \in \{1,2,3\}; \\ 500, & \textit{if } q \in \{4,5,6\}; \\ 450, & \textit{if } q \in \{7,9\}; \end{cases} \tag{4.6}$$

and

$$p_1^K = p_2^K = 10^6. \tag{4.7}$$

The cost of a communication is defined below.

$$\mathsf{com}_1(q,\sigma,q') = \begin{cases} 50, & \textit{if } q \in \{1,2,3,4,6\}; \\ 500, & \textit{if } q \in \{5\}; \\ 10000, & \textit{if } q \in \{7\}; \\ 900, & \textit{if } q \in \{9\}. \end{cases} \tag{4.8}$$

$$\mathsf{com}_2(q,\sigma,q') = \begin{cases} 50, & \textit{if } q \in \{1,2,3,4,6\}; \\ 500, & \textit{if } q \in \{5\}; \\ 700, & \textit{if } q \in \{7\}; \\ 20000, & \textit{if } q \in \{9\}. \end{cases} \tag{4.9}$$

Additionally, a cost of 100 is assigned for activating the antenna in State 6 for both controllers.

We illustrate Algorithm NSGA-II for $\mathsf{R}_1||\mathsf{R}_2$. *The initial size of the population P is 40 and the algorithm was run for 125 generations. The first three ranks of the*

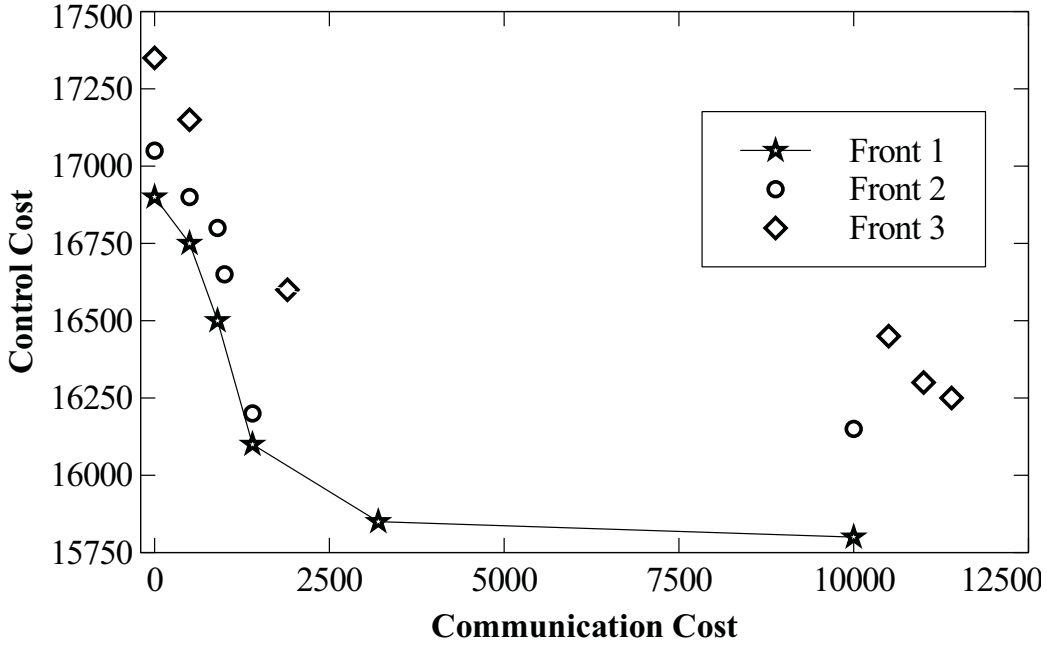


Figure 4.1: Pareto fronts of rank 1,2,3 for Controller 1 after 100 generations.

Table 4.1: Non-dominated solutions of Controller 1.

$u_1^*(\cdot)$	$v_1^*(\cdot)$	$u_2(\cdot)$	$v_2(\cdot)$
0	*16,900*	*127,100*	*7,039,000*
500	*16,750*	*107,600*	*5,041,200*
900	*16,500*	*127,100*	*6,040,200*
1,400	*16,100*	*127,600*	*7,039,750*
3,200	*15,850*	*127,600*	*7,039,750*
10,000	*15,800*	*127,100*	*6,040,200*

Pareto front for Controller 1 are shown in Figure 4.1. The non-dominated solutions for Controller 1 (the solutions in the front of rank 1) are shown in Table 4.1, which represents the best compromises for Robot 1. Since it gives the best solutions for Controller 1, the cost functions are indicated as $u_1^*(\cdot)$ *and* $v_1^*(\cdot)$ *for the communication cost and control cost of Controller 1. In particular, the six Pareto-optimal solutions offer a variety of possible costs for a communication policy for Controller 1, ranging from a cost of 0 up to a cost of 10,000. It is interesting to note that when Controller 1 communicates nothing to Controller 2, the control cost is 16,900. When the communication cost increases to 500, the control cost decreases by less than 1%, but when the communication cost increases to 1400, the control cost decreases by 4.7%. More interestingly, Controller 2 activates the antenna more times than Controller 1. This is because Robot 2 starts in closer to the antenna than Robot 1. On the other hand, since there is a cost to activating the antenna and we are optimizing the cost for Controller 1, it avoids activating the antenna to minimize its cost. The result shows that Controller 2 communicates more, since we only optimize the cost of Controller 1. The control actions are also more expensive for Controller 2 due to the fact that it receives less communication from Controller 1. Thus both the communication cost and control cost of Controller 2 are much higher.*

The algorithm converges to the set of Pareto-optimal solutions after 85 generations for Controller 2. The first three ranks of the Pareto front for Controller 2 are shown in Figure 4.2. The non-dominated solutions for Controller 2 (the solutions in the front of rank 1) are shown in Table 4.2, which are the best compromises for Robot 2. Again, it is interesting to examine the Pareto-optimal solutions for Controller 2: when it communicates nothing, the control cost is 23,150. When the communication cost increases to 500, the control cost decreases by 2%. When the communication cost increases more, the control cost does not go down significantly. Also it is interesting to

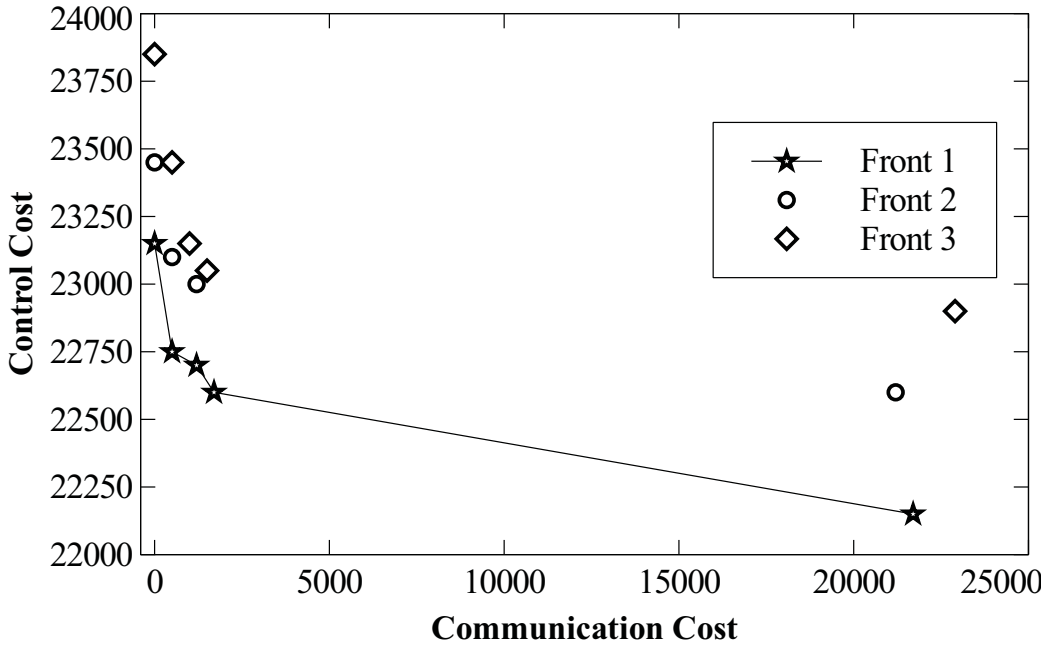


Figure 4.2: Pareto fronts of rank 1,2,3 for Controller 2 after 85 generations.

see that Controller 1 activates the antenna here more times than Controller 2. Since we are optimizing the cost for Controller 2, it minimizes the cost by not activating the antenna. As before, the communication cost and control cost of Controller 1 is much higher in that case, because Controller 1 communicates more, but receives less from Controller 2.

Table 4.2: Non-dominated solutions of Controller 2.

$u_1(\cdot)$	$v_1(\cdot)$	$u_2^*(\cdot)$	$v_2^*(\cdot)$
47,500	*4,030,700*	*0*	*23,150*
47,000	*5,030,650*	*500*	*22,750*
48,400	*5,030,650*	*1,200*	*22,700*
48,400	*7,029,250*	*1700*	*22,600*
47,500	*7,029,800*	*21,700*	*22,150*

4.2.3 Optimization w.r.t. the Cost Functions of All Controllers

We are given a finite set of control laws $\Gamma_i^?$, and a finite set of communication protocols Φ_i for all $i \in I$. Here we want to optimize the control cost and communication cost functions of *all* controllers. One such problem is given below with the cost functions defined as for the previous problem. The control cost and communication cost functions of all controllers are considered here as an objective function.

Problem 4.2. *Given two regular languages K, L over a common alphabet Σ, where $K \subseteq L$. Find $\Gamma_i^?$ and Φ_i $(\forall i \in I)$ to*

$$\underset{\Gamma_1^? \times \Phi_1 \times \ldots \times \Gamma_n^? \times \Phi_n}{min} \quad f(\Gamma_i^?, \Phi_i, \overline{K}, T) = \quad [O_{1,1}(\Gamma_1^?, \Phi_1, \overline{K}, T), \quad O_{2,1}(\Gamma_1^?, \Phi_1, \overline{K}, T), \ldots,$$
$$O_{1,n}(\Gamma_n^?, \Phi_n, \overline{K}, T), \quad O_{2,n}(\Gamma_n^?, \Phi_n, \overline{K}, T)]^T,$$

subject to $\emptyset \subset \cap_{i=1}^n \Gamma_i^?(\pi_i^?(\overline{K})) \subseteq \overline{K}$, and $\Gamma^? = \langle \Gamma_1^?, \ldots, \Gamma_n^? \rangle, \Phi = \langle \Phi_1, \ldots, \Phi_n \rangle$ are coherent.

Definition 4.2. *$(\Gamma_i^?, \Phi_i)$ $(\forall i \in I)$ is Pareto-optimal iff*

$$\underset{\Gamma_1^? \times \Phi_1 \times \ldots \times \Gamma_n^? \times \Phi_n}{min} \; f(\Gamma_i^?, \Phi_i, \overline{K}, T) = [O_{1,1}(\Gamma_1^?, \Phi_1, \overline{K}, T), \; O_{2,1}(\Gamma_1^?, \Phi_1, \overline{K}, T), \ldots,$$
$$O_{1,n}(\Gamma_n^?, \Phi_n, \overline{K}, T), \; O_{2,n}(\Gamma_n^?, \Phi_n, \overline{K}, T)]^T.$$

Example 4.2. *We use the same cost functions as for the previous example. We also consider the same initial size of the population as 40, but the algorithm was run for 200 generations. The non-dominated solutions w.r.t. both controllers (the solutions in the front of rank 1) are shown in Table 4.3, which represents the best compromises for both robots w.r.t. their communication cost and control cost functions. In particular, the five Pareto-optimal solutions offer a variety of possible costs for communication policies and control actions for both controllers. The communication cost and control*

Table 4.3: Non-dominated solutions of both controllers for Problem 4.2.

$u_1^*(\cdot)$	$v_1^*(\cdot)$	$u_2^*(\cdot)$	$v_2^*(\cdot)$
0	*21,200*	*0*	*29,150*
0	*19,450*	*500*	*28,650*
900	*19,800*	*0*	*29,150*
900	*18,850*	*500*	*29,100*
12,300	*18,850*	*21,200*	*29,050*

cost range from 0 to 12,300 and from 18,850 to 22,200 respectively for Controller 1, while for Controller 2 they range from 0 to 21,200 and 28,650 to 29,150 respectively. When the communication cost for Controller 1 increases from 0 to 12,300, the control cost goes down by 11.1%. On the other hand, the control cost for Controller 2 decreases by only 0.40% when the communication cost increases from 0 to 21,200. In the absence of communication, both controllers exhibit the highest control cost. But when Controller 1 communicates with a cost of 900, the overall control cost decreases by 2.7%, and when Controller 2 communicates with a cost of 500, the overall control cost decreases by 4.5%. When communication cost for both controllers are maximum, the overall control cost goes down by only 5%. The antenna is activated more times by Controller 2. This can be justified by noting that Controller 2 starts closer to State 6, where the antenna is placed.

4.2.4 Optimization w.r.t. the Global Cost Functions

We consider the cost functions from a global perspective here. We take the fusion of the control decisions made by all controllers for the occurrence of a transition. The fusion rule is conjunctive. Thus a transition is enabled if all controllers take the decision of enablement. In that case, we sum up the enable cost incurred by all controllers. The transition is disabled if any controller takes a decision to disable it,

and we take the minimal disable cost among the controllers who disable the transition.

The global control cost for the occurrence of a transition $(q, \sigma, q') \in T$ after the sequence $s \in \overline{K}$ such that $q_0 \xrightarrow{s} q \xrightarrow{\sigma} q'$, is defined according to the control cost defined in Equation 4.1:

$$v(\Gamma^?, s, (q, \sigma, q')) = \begin{cases} \sum_i e_i(q, \sigma, q'), & \text{if } (q, \sigma, q') \in \Gamma^?(\pi^?(s)); \\ \min_i(d_i(q, \sigma, q')), & \text{if } (q, \sigma, q') \notin \Gamma^?(\pi^?(s)) \text{ and} \\ & \cap_{i \in I} [\![s]\!]\sigma \cap \bar{K} = \emptyset; \\ \min_i(d_i(q, \sigma, q') + p_i^K), & \text{if } (q, \sigma, q') \notin \Gamma^?(\pi^?(s)) \text{ and} \\ & \cap_{i \in I} [\![s]\!]\sigma \cap \bar{K} \subseteq \bar{K}; \\ \infty, & \text{otherwise.} \end{cases}$$

The total control cost is then $V(\Gamma^?, \overline{K}, T) = \sum_{s \in \overline{K}} \sum_{(q,\sigma,q') \in T} v(\Gamma^?, s, (q, \sigma, q'))$.

Similarly the global communication cost takes communications into account sent by all controllers and sum up the cost incurred by each controller. Then we have the following two objective functions.

Objective 3. *The first objective function is the global control cost for the decisions made by all controllers $i \in I$ for their observation of $\overline{K}$:*

$$O_1(\Gamma^?, \Phi, \overline{K}, T) = V(\Gamma^?, \Phi, \overline{K}, T).$$

Objective 4. *The second objective function is the global communication cost for the communication protocol used by all controllers to assist other controllers in reaching the control objective for $\overline{K}$. The global communication cost is defined as $U(\Phi, \overline{K}, T)$*

Table 4.4: Non-dominated solutions of both controllers for Problem 6.2.

$u_1(\cdot)$	$v_1(\cdot)$	$u_2(\cdot)$	$v_2(\cdot)$	$(u_1+u_2)^*(\cdot)$	$(v_1+v_2)^*(\cdot)$
0	24,750	0	30,850	0	55,600
500	23,650	0	28,850	500	52,500
900	21,750	0	27,700	900	49,450
1,900	21,150	500	28,050	2,400	49,200
3,300	21,250	0	27,850	3,300	49,100
3,300	20,900	500	27,350	3,800	48,250
12,400	20,100	1,400	26,700	13,800	46,800

$= \sum_i U_i(\Phi_i, \overline{K}, T)$. *Hence the objective function is*

$$O_2(\Gamma^?, \Phi, \overline{K}, T) = U(\Gamma^?, \Phi, \overline{K}, T).$$

Problem 4.3. *Given two regular languages K, L over a common alphabet Σ, where $K \subseteq L$. Find $\Gamma_i^?$ and Φ_i $(\forall i \in I)$ to*

$$\min_{\Gamma^?, \Phi} \quad f(\Gamma^?, \Phi, \overline{K}, T) = [O_1(\Gamma^?, \Phi, \overline{K}, T), \\ O_2(\Gamma^?, \Phi, \overline{K}, T)]^T,$$

subject to $\emptyset \subset \cap_{i=1}^n \Gamma_i^?(\pi_i^?(\overline{K})) \subseteq \overline{K}$, *and* $\Gamma^? = \langle \Gamma_1^?, \ldots, \Gamma_n^? \rangle, \Phi = \langle \Phi_1, \ldots, \Phi_n \rangle$ *are coherent.*

Definition 4.3. $(\Gamma_i^?, \Phi_i)$ $(\forall i \in I)$ *is Pareto-optimal iff*

$$\min_{\Gamma^?, \Phi} f(\Gamma^?, \Phi, \overline{K}, T) = [O_1(\Gamma^?, \Phi, \overline{K}, T), O_2(\Gamma^?, \Phi, \overline{K}, T)]^T.$$

Example 4.3. *In this example, as before, we have two objectives: the control costs for both controllers, and the communication costs for both controllers. We again use the same cost functions defined in Equations* $4.3 - 4.9$*, and also run the algorithm for 200 generations with an initial population size of 40. The first three ranks of the Pareto*

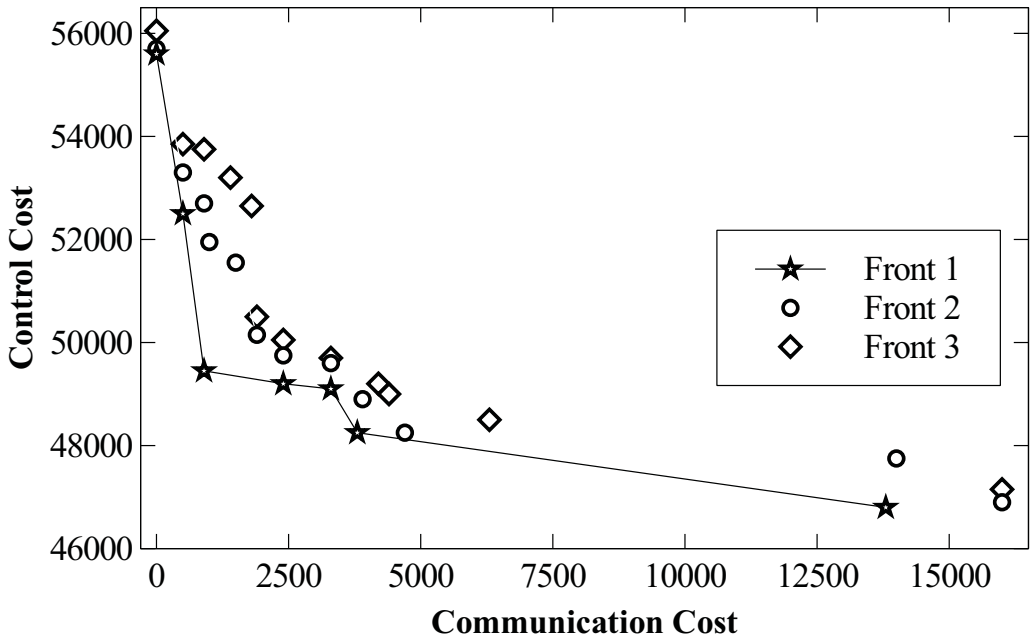


Figure 4.3: Pareto fronts of rank 1,2,3 for both controllers after 200 generations.

front w.r.t. the communication cost and control cost of both controllers are shown in Figure 4.3. The non-dominated solutions for both controllers are shown in Table 4.4, which represents the best compromises w.r.t. the total communication and control costs of Robot 1 and 2. We have seven Pareto-optimal solutions in this example, which gives a wide variety of possible costs for control actions and communication policies for both controllers. The communication cost and control cost range from 0 to 13,800 and from 46,800 to 55,600 respectively w.r.t. both controllers. In the absence of communication between the controllers, the controllers incur the highest control cost. When the communication cost increases to 500, the control cost decreases by 6%, whereas the control cost goes down by 16% when the communication cost goes up to 13,800. When controllers communicate all observations, the solution becomes equivalent to centralized supervision. The control cost is the lowest in that case.

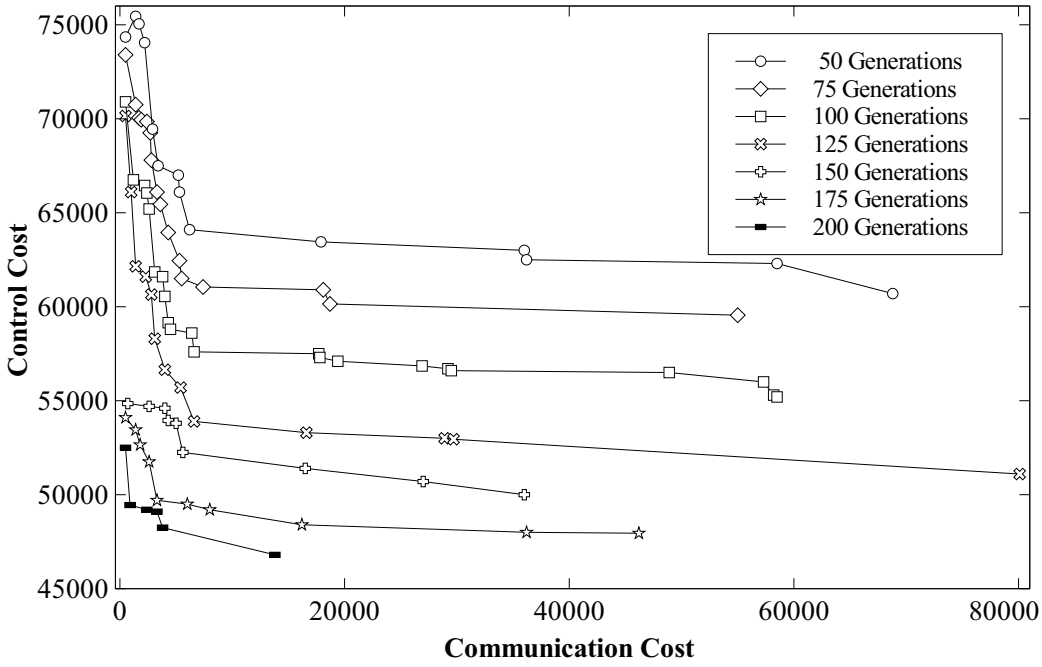


Figure 4.4: Convergence of Pareto front for Problem 4 in 200 generations.

How the solutions converge to the set of Pareto front is shown in Figure 4.4 for Problem 4.3. With the increased number of generations, the algorithm chooses better solutions w.r.t. the cost functions. For instance after 150 generations, the cost of the solutions (in the form of communication cost, and control cost) in the Pareto front of rank 1 are $(0, 55700)$, $(500, 54100)$, $(1400, 53450)$, $(1800, 52650)$, $(2600, 51750)$, $(3300, 49700)$, $(6000, 49500)$, $(8000, 49200)$, $(16200, 48400)$, $(36200, 48000)$ and $(46200, 47950)$ where the cost of the Pareto front solutions after 200 generations are $(0, 55600)$, $(500, 52500)$, $(900, 49450)$, $(2400, 49200)$, $(3300, 49100)$, $(3800, 48250)$ and $(13800, 46800)$. Figure 4.5 shows how many solutions are in the first two Pareto fronts in different generations. Initially the Pareto fronts with rank 1, 2 have 7 and 4 solutions respectively. After 100 generations they occupy 25 and 15 solutions, and after 200 generations they finally have 7 and 12 solutions respectively. It is interesting that first two ranks are present and make up most of the solutions between 50 and 150

generations.

Ultimately, we use NSGA-II as an initial guide to aid in the selection of Pareto-optimal communication and control policies for decentralized DES. For instance, there may be compelling physical arguments to insist that one decentralized site assumes the bulk of the communication during the operation of system tasks, despite the site incurring a high communication cost (w.r.t. other sites). Similarly, we may be willing for some degree of approximation on one or more sites to reduce the cost of communication to achieve a precise control decision, or we may want to reduce the total communication cost incurred by all controllers. Modeling the trade-off between the cost functions as an optimization problem gives us a better selection of quantitatively optimal solutions, from which to choose communication and control policies for this class of decentralized DES control problems.

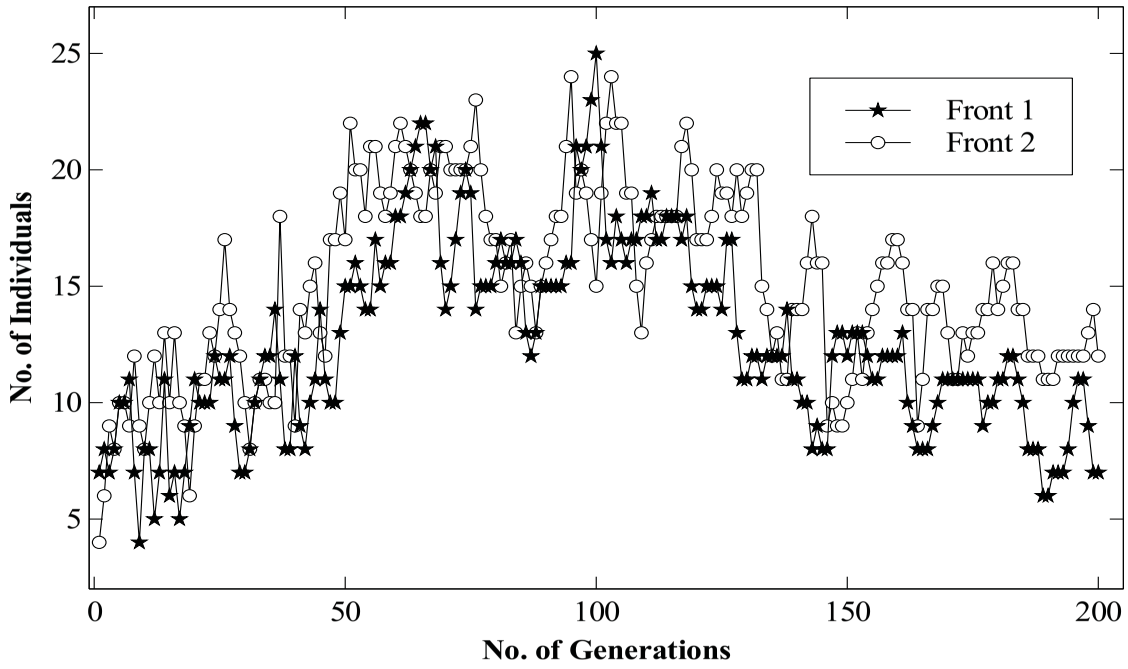


Figure 4.5: No. of solutions occupied in Rank 1 and 2 for Problem 4 in different generation.

Chapter 5

Robustness of a Synchronous Communication Protocol

Synchronous communication protocols are synthesized in various models including [4, 37, 38, 56, 57]. Given a synchronous communication protocol that is designed to solve the decentralized control and communication problem with zero delay, we want to verify if the protocol is robust enough to withstand timing delays associated with a more realistic communication network. Hence instead of directly synthesizing communication protocols under bounded and not necessarily fixed delay, we examine synchronous communication protocols (where not all observations are communicated) for their robustness under conditions when only the upper-bound for the delay is known. Ideally we would prefer the control actions taken by the controllers in presence of a communication protocol with delay result in the same closed-loop closed behaviour K.

5.1 Robust Synchronous Communication with Delay

We are interested in determining how robust a synchronous communication protocol is in the face of delay in the communication channels. There are two circumstances in which a controller that took the correct control decision with zero delay is now prevented from taking the correct control decision when its received messages are delayed. It may be the case that the late arrival of a message results in an incorrect estimate of the system behavior so that the controller can no longer definitively distinguish a sequence in $L \setminus K$ from one in K. Or the message may arrive after the control decision has to be taken, in which case the information simply arrives too late and the correct decision cannot be made. Note that, we do not limit our study to the robustness of *optimal* synchronous communication policies. Rather, we only insist that the protocol allows the controllers to solve the control problem correctly, however, we are assuming that the protocols that we are examining may represent some sort of reduced communication from the "communicate every observation" strategy. We start with Example 3.2 for $n = 3$ controllers, where a synchronous communication protocol solves the control problem.

Example 5.1. *We are given M_L (plant) and M_K (design specification) as shown in the following figure. Suppose that $\Sigma_{o,1} = \{\mathsf{a}\}$, $\Sigma_{o,2} = \{\mathsf{b}\}$, and $\Sigma_{o,3} = \{\mathsf{c}\}$. Further, let $I_c(\sigma) = \{1, 2, 3\}$. Consider $s = \mathsf{bca}$ and $s' = \mathsf{abc}$. Note that K is not co-observable w.r.t. $\Sigma_{o,i}$ since for all $i \in I_c(\sigma)$, $\pi_i(s) = \pi_i(s')$ and, thus, no single controller can take the correct control decision regarding σ. There are many different sets of communication transitions that give rise to synchronous communication protocols that will solve this problem (short of communicating all observations). We will examine the communication protocol constructed from the following sets of communication*

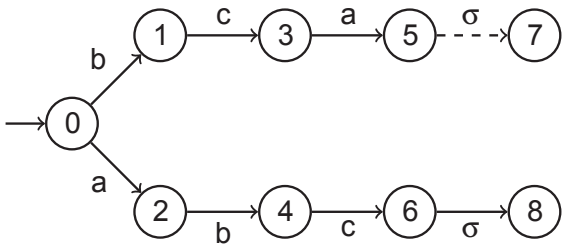


Figure 5.1: A joint M_L (all transitions) and M_K (only solid line transitions).

transitions $T^!_{1,3} = \{(0,a,2), (3,a,5)\}$ *(i.e.,* $\Sigma^?_3 = \{a\}$*) and* $T^!_{2,1} = \{(0,b,1), (2,b,4)\}$ *(i.e.,* $\Sigma^?_1 = \{b\}$*), to be the only non-empty sets of communication transitions. This gives rise to the following coherent communication protocol* $\phi = (\phi_1; \phi_2; \phi_3)$*:*

$$
\begin{aligned}
\phi_1 \;:\; & (\forall s \in L)\phi_{1,2}(s) = \varepsilon, \\
& \phi_{1,3}(bca) = \phi_{1,3}(a) = a, \\
& (\forall s \in L \setminus \{bca,a\})\phi_{1,3}(s) = \varepsilon; \\
\phi_2 \;:\; & \phi_{2,1}(b) = \phi_{2,1}(ab) = b, \\
& (\forall s \in L \setminus \{b,ab\})\phi_{2,1}(s) = \varepsilon, \\
& (\forall s \in L)\phi_{2,3}(s) = \varepsilon; \\
\phi_3 \;:\; & (\forall s \in L)\phi_{3,1}(s) = \phi_{3,2}(s) = \varepsilon.
\end{aligned}
$$

While $\pi_1(s) = \pi_1(s')$*, by extending controller 1's information to* $\Sigma_{o,1} \cup \Sigma^?_1$ *via* ϕ*, we have* $\pi^?_1(s) = ba$ *whereas now* $\pi^?_1(s') = ab$*. Similarly, controller 3 can also distinguish* s *from* s' *after receiving its messages. Thus* K *is communication observable w.r.t.* L*,* $\Sigma_{o,i} \cup \Sigma^?_i$*,* $\Sigma_{c,i}$ *and* ϕ_i *(*$i \in I$*).* $\diamond$

We will examine two cases involving delay: (i) when the (finite) delay is known exactly and never varies throughout the runs of the system; (ii) when the delay is unknown, but bounded.

5.1.1 Modeling Communicating Controllers

Unlike the synthesis of synchronous communication protocols, when considering the effect of delayed communication, we need to distinguish between the observation triggering a message, the sent message and the received message [45,46]. In particular, for an event occurrence σ in the plant, the corresponding sent message by one controller is denoted by $!\sigma$, and the received message by another controller is $?\sigma$. Hence, we introduce the notation for the event corresponding to a sent message for controller i as follows:

$$\Sigma_i^! := \{!\sigma \mid \exists (q, \sigma, q') \in T_i^!\}.$$

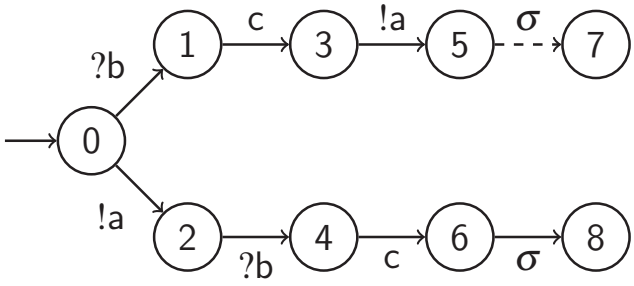
We similarly use the following notation for received messages:

$$\Sigma_i^? := \{?\sigma \mid \exists (q, \sigma, q') \in T_i^?\}.$$

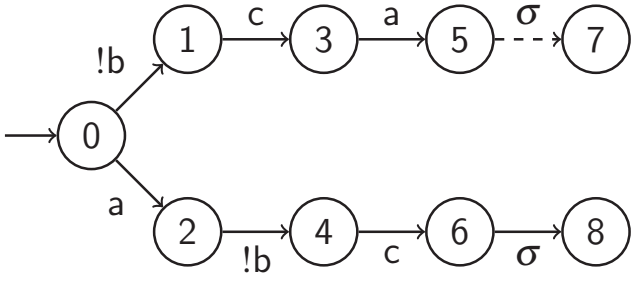
We assume, however, that the content of the message is simply the event σ. Additionally, we treat the events in $\Sigma_i^!$ and $\Sigma_i^?$ as private events for controller i (i.e., unobservable to all others including the uncontrolled system).

We describe a two-step process to transform M_L (for each controller $i \in I$) so that the sender of a message can distinguish between the observation of an event and the sending of a message regarding the occurrence of that event *and* the delayed reception of a message is encoded. We begin by incorporating events from $\Sigma^!$ and $\Sigma^?$ into a copy of the specification automaton, which we denote by $\mathcal{M}_i^{!?}$. In particular, for $\sigma \in \Sigma_i^?$, we replace σ with $?\sigma$, and similarly for events in $\Sigma_i^!$. The augmented automaton is defined as follows:

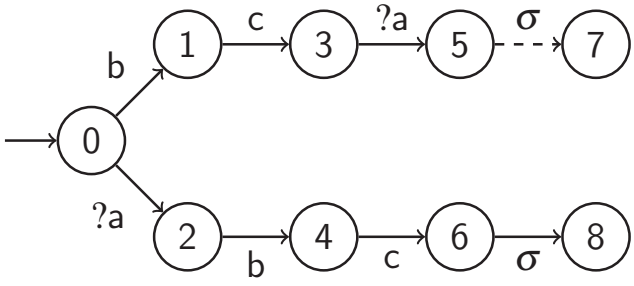
$$\mathcal{M}_i^{!?} = (Q, \Sigma \cup \Sigma_i^! \cup \Sigma_i^?, T_{K_i}^{!?}, q_0, F^{!?}),$$



(a) $\mathscr{M}_1^{!?}$



(b) $\mathscr{M}_2^{!?}$



(c) $\mathscr{M}_3^{!?}$

Figure 5.2: $\mathcal{M}_1^{!?}$, $\mathcal{M}_2^{!?}$ and $\mathcal{M}_3^{!?}$ with ϕ from Example 5.1.

where $T^{!?}_{K_i}$ is the updated transition relation after incorporating information from $\Sigma^!$ and $\Sigma^?$; and an additional set of transitions recording illegal transitions w.r.t. K, $F^{!?} := \{(q, \sigma, q') \in T_L \setminus T_K\}$.

We illustrate our strategy using Example 5.1 (see Figure 5.2) where the effects of $T^!_{1,3}$ and $T^!_{2,1}$, in addition to $T^?_3 = \{(0,a,2), (3,a,5)\}$ and $T^?_1 = \{(0,b,1), (2,b,4)\}$, have been directly incorporated into $\mathcal{M}^{!?}_1$, $\mathcal{M}^{!?}_2$ and $\mathcal{M}^{!?}_3$.

5.1.2 Rational Transducer for Delayed Messages

We need a way to transform the controller's view so that it reflects the delayed arrival of messages. Note that the behavior of the plant is not affected by the introduction of a communication protocol; it is each controller's perception of the plant behavior that is adjusted when messages are delayed. In that case, we define a rational transducer, which is a generalized finite automaton where the transition function has a domain consisting of a pair of states and a sequence. It allows the propagation of the delayed messages, whether we are concerned with fixed or bounded delay.

Formally, our first transducer $\mathcal{T}_0(d)$, which takes the language generated by $\mathcal{M}^{!?}_i$ as input, is defined as follows: $\mathcal{T}_0(d) = (Q_\mathcal{T}, \Sigma_\mathcal{T}, \Gamma, q^\mathcal{T}_0, E)$, where $Q_\mathcal{T}$ is the finite state set; the input alphabet $\Sigma_\mathcal{T} := \Sigma \cup \Sigma^!_i \cup \Sigma^?_i$; the output alphabet $\Gamma := \Sigma \cup \Sigma^!_i \cup \Sigma^?_i \cup \{(d)\}$; $q^\mathcal{T}_0$ is the initial state; and the transition relation $E : Q_\mathcal{T} \times \Sigma^*_\mathcal{T} \times Q_\mathcal{T} \to \Gamma^*$. As an example, consider Figure 5.3. Starting from the top of the figure, the first transition takes an instruction to send a message, i.e., $!\sigma$, and separates the occurrence of the event (i.e., σ) from the sending of the message that the event occurred (i.e., $!\sigma$). Continuing in a clockwise direction, the next transition of $\mathcal{T}_0(d)$ takes an input to receive a message (i.e., $?\sigma$), separates it from the occurrence of the event (i.e., σ), but prepends to the content of the message a delay of d, i.e, $?(d)\sigma$. The final transition just processes events in Σ, performing no transformation to the input whatsoever.

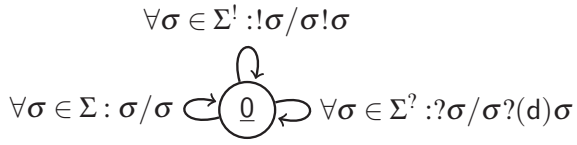


Figure 5.3: Transducer $\mathcal{T}_0(d)$, with initial state underlined.

Note that this transducer is used simply to demonstrate the events corresponding to received messages in the language of $\mathcal{M}_i^{!?}$ with a temporary event label that, depending on the context, indicates either the exact delay or the upper-bound of the delay. Figure 5.4 shows an automaton that generates $\mathcal{L}(\mathcal{M}_1^{!?} \circ \mathcal{T}_0(1))$ for Example 5.1 with $d = 1$.

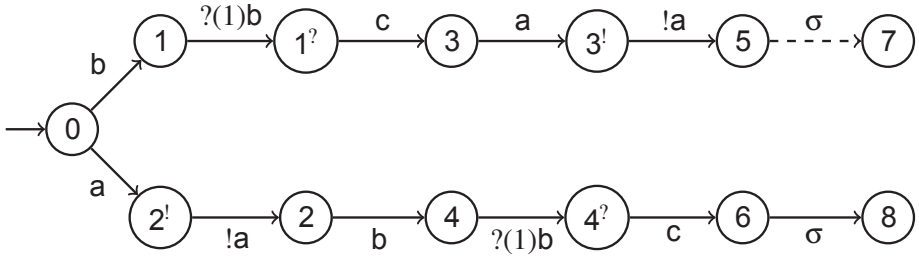


Figure 5.4: An automaton that generates $\mathcal{L}(\mathcal{M}_1^{!?} \circ \mathcal{T}_0(1))$ for Example 5.1.

5.1.3 Known and Fixed Delay

A synchronous communication protocol is *robust under a fixed delay d* if all the correct control decisions can be taken, even when messages are received exactly d clock cycles late.

We must transform the $\mathcal{M}_i^{!?}$, where necessary, to reflect the delayed reception of the messages for the exact delay d. We use an additional transducer for this.

The next transducer that we require, denoted by $\mathcal{T}_1(k)$ and shown in Figure 5.5, performs the deterministic propagation of a delayed message reception, where k represents a counter for the remaining delay. It takes as input the language generated by $\mathcal{M}_i^{!?} \circ \mathcal{T}_0(d)$ and outputs the propagation of a message by one position in the input

string. We assume that the initial value for this transducer is $k = d$. In particular, starting from the left-hand side of Figure 5.5, the first self-loop does nothing to events not involved in the communication protocol; continuing in a clockwise direction, the next self-loop does nothing to sent message events; and the final self-loop propagates the received message by one position and the counter k is decremented. Note that when $k = 1$, we interpret $?(0)\sigma$ as "σ is communicated with a delay of 0" and we write this as simply $?\sigma$.

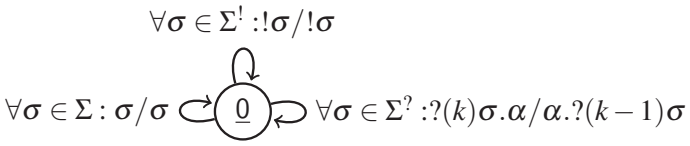


Figure 5.5: Transducer $\mathcal{T}_1(k)$, with initial state underlined.

We are interested in an automaton, denoted by $\mathcal{M}_i(= d)$, that generates the following language:

$$\mathcal{L}(\mathcal{M}_i^{!?} \circ \mathcal{T}_0(d) \circ \mathcal{T}_1(d) \circ \mathcal{T}_1(d-1) \circ \ldots \circ \mathcal{T}_1(1)), \tag{5.1}$$

where d is the fixed delay. We want to ensure that this structure generates a regular language. $\mathcal{M}_1(= 1)$ is shown in Figure 5.6 for Example 5.1 with $d = 1$. Because both $\mathcal{T}_0(d)$ and $\mathcal{T}_1(k)$ are finite transducers, we refer to the following result:

Theorem 5.1. *[Adapted from [13]] The composition of two finite transducers is also a finite transducer.*

Therefore, we have the following result.

Theorem 5.2. *The language generated by $\mathcal{M}_i(= d)$ is a regular language.*

Proof. Since $\mathcal{T}_0(d) \circ \mathcal{T}_1(d) \circ \mathcal{T}_1(d-1) \circ \ldots \circ \mathcal{T}_1(1)$ is the composition of finite transducers, by Theorem 5.1, $\mathcal{M}_i(= d)$ is a finite transducer and, thus, by the definition of finite transducers, generates a regular language. □

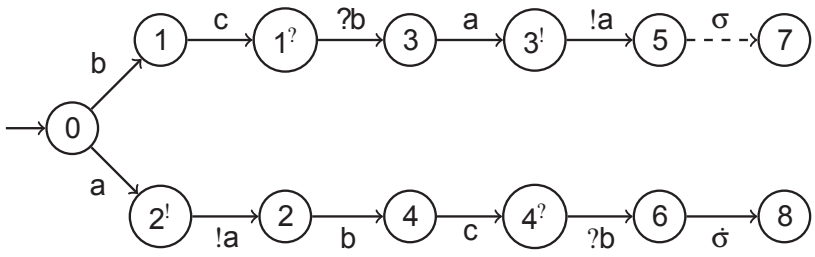


Figure 5.6: $\mathcal{M}_1(=1)$ for Controller 1 from Example 5.1 when $d = 1$.

We denote the language of $\mathcal{M}_i(=d)$ as $L_i^?(d)$ (or $K_i^?(d)$, depending on the context). This language reflects the fact that received messages have been displaced from the synchronous communication protocol by exactly d positions. Note that when $d = 0$, we can use the transducer $\mathcal{T}_0$ to generate $L_i^?(0) = L$, and similarly $K_i^?(0) = K$. When relevant, we will refer to this pair of languages as simply $L(0)$ and $K(0)$. Similarly a sequence $s \in L$ has been adjusted by a controller as $s(d)$ with a fixed delay d. We can now formally define what it means for a synchronous communication protocol to be robust under conditions of a fixed delay.

Definition 5.1. *A synchronous communication protocol $\phi = (\phi_i)_{i\in I}$ for $K \subseteq L$, where K is communication observable w.r.t. L, $\pi_i^?$,$\Sigma_{c,i}$ $(i \in I)$ is* ***robust w.r.t. fixed delay*** *$d \in \mathbb{N}^*$ if*

$$
\begin{aligned}
&(\forall s \in \overline{K})(\forall \sigma \in \Sigma_c) s\sigma \in L \setminus \overline{K} \Rightarrow \\
&\quad \exists i \in I_c(\sigma) : [\![s(d)]\!]_i^? \sigma \cap \overline{K_i^?(d)} = \emptyset \Leftrightarrow [\![s]\!]_i^? \sigma \cap \overline{K} = \emptyset, \textit{and} \\
&(\forall s \in \overline{K})(\forall \sigma \in \Sigma_c) s\sigma \in \overline{K} \Rightarrow \\
&\quad \forall i \in I_c(\sigma) : [\![s(d)]\!]_i^? \sigma \cap \overline{K_i^?(d)} \neq \emptyset \Leftrightarrow [\![s]\!]_i^? \sigma \cap \overline{K} \neq \emptyset.
\end{aligned}
$$

□

We want to verify that if there exists an $i \in I_c(\sigma)$ that can distinguish an illegal sequence $s(d)\sigma$ in $L_i^?(d) \setminus K_i^?(d)$ (or a legal sequence $s(d)\sigma$ in $K_i^?(d)$), then it can also

verify if $s\sigma$ is in $L \setminus \overline{K}$ (or in $\overline{K}$) with 0 delay.

We formally state the robustness problem for a synchronous communication protocol ϕ under conditions of a fixed delay d.

Problem 5.1. *Consider two regular languages K, L defined over a common alphabet Σ, with controllable events $\Sigma_{c,1}$, ..., $\Sigma_{c,n} \subseteq \Sigma$, observable events $\Sigma_{o,1}$, ..., $\Sigma_{o,n} \subseteq \Sigma$, a set of messages $T_1^!, \ldots, T_n^!$ (for $i \in I$). We assume that $K \subseteq L \subseteq \Sigma^*$ is controllable w.r.t. L, Σ_{uc}, observable w.r.t. L, π, Σ_c and communication observable w.r.t. $L, \pi_i^?, \Sigma_{c,i}$ ($i \in I$). Determine whether $\phi = (\phi_i)_{i \in I}$ is robust w.r.t. a fixed delay d.*

5.1.4 Finite and Bounded Delay

A synchronous communication protocol is *robust under a bounded delay* $[1..d]$ if messages received by controller i, anywhere up to d clock cycles late, continue to allow the correct control decisions to be taken. The significant difference for this problem is that we have no idea how long the delay is in the communication channel: we know only the upper-bound for the delay. That is, our model must take into account that the delay for each message can range between 1 and d.

While we will utilize transducer $\mathcal{T}_0(d)$ to identify the effect of the synchronous communication protocol ϕ on the uncontrolled system, we need an additional transducer, $\mathcal{T}_2(k)$, to perform the non-deterministic propagation of a delayed message reception. The transducer is shown in Figure 5.7. From the top of the figure, the first transition does not change labels in $\Sigma^!$ and the second transition clockwise is identical to that used in $\mathcal{T}_1(k)$. The next transition performs part of the non-deterministic propagation of $?\sigma$ and adds the immediate (i.e., zero delay) reception of a message. The final transition makes no changes to events in Σ.

Again, we are interested in an automaton, denoted by $\mathcal{M}_i(\leq d)$, that generates

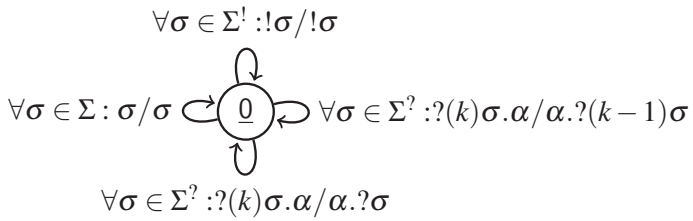


Figure 5.7: Transducer $\mathcal{T}_2(k)$, with initial state underlined.

the following language

$$\mathcal{L}(\mathcal{M}_i^{!?} \circ \mathcal{T}_0(d) \circ \mathcal{T}_2(d) \circ \mathcal{T}_2(d-1) \circ \ldots \circ \mathcal{T}_2(1)),$$

where d is the upper-bound of the delay. We denote the language of $\mathcal{M}_i(\leq d)$ as $L_i^?(\leq d)$ (or $K_i^?(\leq d)$, depending on the context). Similarly as defined in the case of a fixed delay, a sequence $s \in L$ has been adjusted by a controller as $s(\leq d)$ with a bounded delay $[1..d]$.

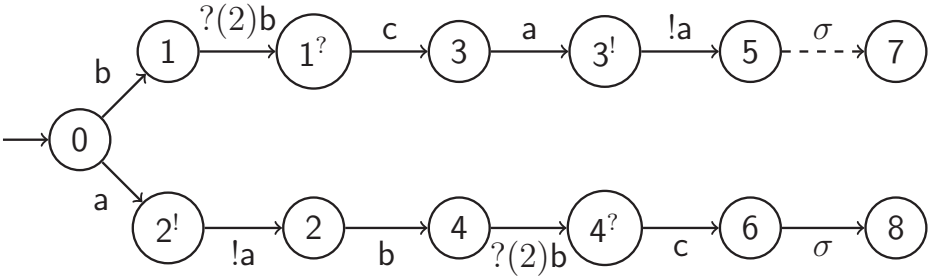
One important assumption we make is that there can be no loops of communication events in our original system. Here $\mathcal{M}_i(\leq d)$ should be a finite transducer to ensure that we are generating a regular language. When communication occurs in a loop, the delay becomes infinite. This leads to an unbounded transducer $\mathcal{T}_2^*(k)$, in which case we are no longer working with a regular language.

Theorem 5.3. *The language generated by $\mathcal{M}_i(\leq d)$ is a regular language.*

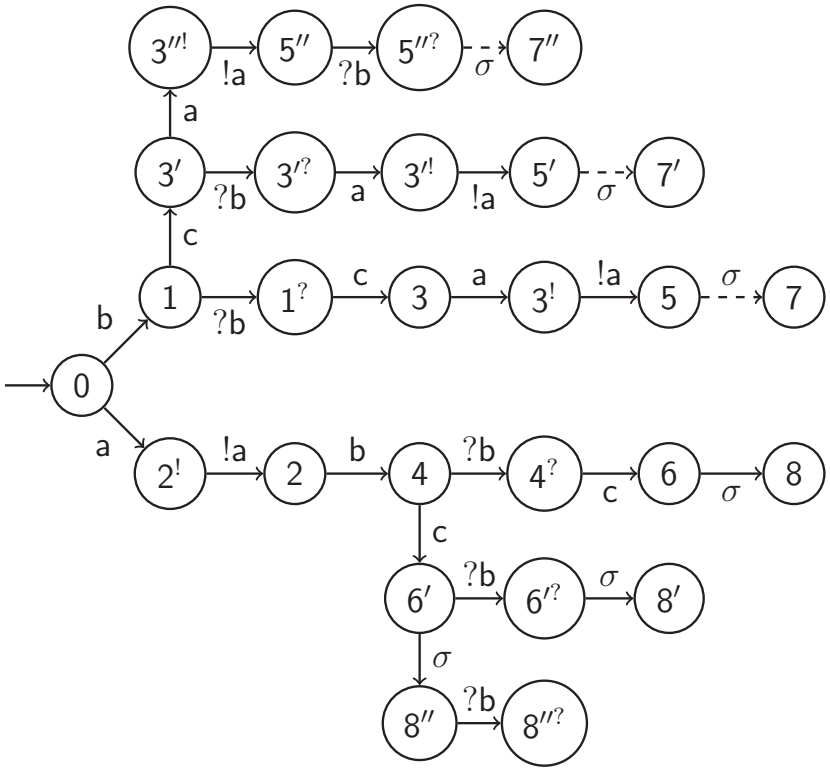
Proof. Since $\mathcal{M}_i^{!?} \circ \mathcal{T}_0(d) \circ \mathcal{T}_2(d) \circ \ldots \circ \mathcal{T}_2(1)$ is a composition of finite transducers, it follows from Theorem 5.1 that $\mathcal{M}_i(\leq d)$ is a finite transducer, which generates a regular language. □

We illustrate the result of these transducer compositions in Figure 5.8 for a bounded delay of 2 from Figure 5.2(a).

When we consider communication with a bounded delay $[1..d]$, we must check to see if the correct control decisions can be made for *any* delays between 1 and d.



(a) An automaton that generates $\mathcal{L}(\mathcal{M}_1^{!?} \circ \mathcal{T}_0(2))$.



(b) $\mathcal{M}_1(\leq 2)$.

Figure 5.8: Building of $\mathcal{M}_1(\leq 2)$ for a bounded delay of 2 w.r.t. Controller 1.

Definition 5.2. *A synchronous communication protocol $\phi = (\phi_i)_{i\in I}$ for $K \subseteq L$, where K is communication observable w.r.t. L, $\pi_i^?$,$\Sigma_{c,i}$ $(i \in I)$ is* ***robust w.r.t. bounded delay*** $[1..d]$ *if*

$$
\begin{aligned}
&(\forall s \in \overline{K})(\forall \sigma \in \Sigma_c) s\sigma \in L \setminus \overline{K} \Rightarrow \\
&\quad \exists i \in I_c(\sigma) : [\![s(\leq d)]\!]_i^? \sigma \cap \overline{K_i^?(\leq d)} = \emptyset \Leftrightarrow [\![s]\!]_i^? \sigma \cap \overline{K} = \emptyset, \textit{and} \\
&(\forall s \in \overline{K})(\forall \sigma \in \Sigma_c) s\sigma \in \overline{K} \Rightarrow \\
&\quad \forall i \in I_c(\sigma) : [\![s(\leq d)]\!]_i^? \sigma \cap \overline{K_i^?(\leq d)} \neq \emptyset \Leftrightarrow [\![s]\!]_i^? \sigma \cap \overline{K} \neq \emptyset.
\end{aligned}
$$

□

Problem 5.2. *Consider two regular languages K, L defined over a common alphabet Σ, with controllable events $\Sigma_{c,1}$, ..., $\Sigma_{c,n} \subseteq \Sigma$, observable events $\Sigma_{o,1}$, ..., $\Sigma_{o,n} \subseteq \Sigma$, a set of messages $T_1^!, \ldots, T_n^!$ (for $i \in I$). We assume that $K \subseteq L \subseteq \Sigma^*$ is controllable w.r.t. L, Σ_{uc}, observable w.r.t. L, π, Σ_c and communication observable w.r.t. $L, \pi_i^?, \Sigma_{c,i}$ $(i \in I)$. Determine whether $\phi = (\phi_i)_{i\in I}$ is robust w.r.t. a bounded delay* $[1..d]$.

In the next section, we present a finite-state structure that we use to verify robustness of a synchronous communication protocol under both types of delay.

5.2 Verifying Robust Synchronous Communication Under Conditions of Delay

We want to build a finite structure with which we can determine the robustness of a given synchronous communication protocol ϕ. We want to have a way of indicating that delay has occurred without using a fully timed model. Although we are not explicitly modeling time, we assume that the uncontrolled system and the controllers

all have access to a global clock. To that end, we will use a special event τ to mark the end of a clock cycle. Next we update M_L and $\mathcal{M}_i(= d)$, respectively $\mathcal{M}_i(\leq d)$, by incorporating τ into the transition relation. Note that the verification strategy for robustness of communication protocols under fixed delay or bounded delay is the same; however, we use $\mathcal{M}_i$(=d) when verifying the former, and $\mathcal{M}_i(\leq d)$ when verifying the latter.

5.2.1 Incorporating τ into the Behavior of the Uncontrolled System

We begin with $M_L = (Q, \Sigma, T_L, q_0, F^T)$, the automaton model of L with no timing information. We use τ to indicate how many events occur within a clock cycle. It will be of considerable use to learn if the uncontrolled system generates events faster than the controllers can inform each other of relevant observations, under a given communication protocol and a bounded delay. Formally we have

$$M_L^\tau = (Q \cup Q^\tau, \Sigma \cup \{\tau\}, T_L^\tau, q_0, Q_\mathsf{m}),$$

where the event τ is placed at the end of a sequence of events that occur during a clock cycle; Q^τ are the additional states needed to incorporate the τ events into M_L; and T_L^τ is the updated transition relation.

In Figure 5.9, τ events have been added to M_L from Figure 5.1 to indicate that one event in Σ occurs within each clock cycle.

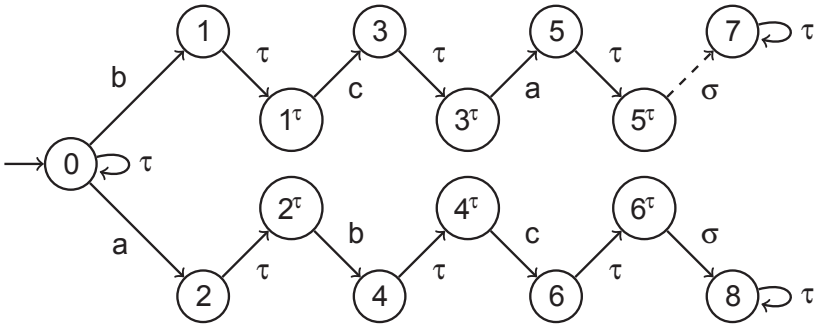


Figure 5.9: M_L^τ for M_L in Figure 5.1, where one event per clock cycle occurs.

5.2.2 Incorporating Message Delay and τ into the Behavior of the Controllers

To facilitate the construction of our finite-state structure, we must update $\mathcal{M}_i(= d)$, respectively $\mathcal{M}_i(\leq d)$, $(i \in I)$ to reflect the effect of a clock cycle (noted by the τ event) and the idea that, for a receiver, messages may arrive with delay. We denote the augmented automata by $\mathcal{M}_i^\tau(= d)$ and $\mathcal{M}_i^\tau(\leq d)$, respectively.

The controllers are (unaware of and) unconcerned with the timing details of the plant, assumed to be designed based on the untimed behavior of the plant. Therefore, we simply add self-loops of τ at each state in $\mathcal{M}_i(= d)$, respectively $\mathcal{M}_i(\leq d)$, with the exception that all sub-sequences of the form $\mathsf{m!m}$ must occur within a single clock cycle. This represents the idea that a message is sent immediately after an observation is made: the delay we are modeling is that between the sending and the reception of a message. Thus to update $\mathcal{M}_i(\leq d)$ to $\mathcal{M}_i^\tau(\leq d)$, for $i \in I$, (see Figure 5.8(b)), we would add self-loops of τ at every state except states $\mathsf{2}^!$, $\mathsf{3}^!$, $\mathsf{3}'^!$, $\mathsf{3}''^!$, where we assume a message is sent in the same clock cycle as the observation that triggers the communication. We will also associate with each component, a set $B_i^{!?\tau} := \{(q, \sigma, q') \in T_L \setminus T_K\}$, the set of transitions recording illegal transitions w.r.t. K.

5.2.3 Building $\mathcal{U}_1$ and $\mathcal{U}_2$

We build an automaton $\mathcal{U}_1$, built with synchronized composition, to verify the robustness of ϕ w.r.t. a fixed delay d as follows.

$$\mathcal{U}_1(d,\phi) = (X, A, T^{\mathcal{U}}, x_0, B) = M_L^\tau \times_{\mathcal{S}} \mathcal{M}_1^\tau(=d) \times_{\mathcal{S}} \ldots \times_{\mathcal{S}} \mathcal{M}_n^\tau(=d).$$

Further, we build a second automaton $\mathcal{U}_2$, also via synchronized composition, to verify the robustness of ϕ w.r.t. a delay in the range $[0..d]$ as follows.

$$\mathcal{U}_2(d,\phi) = (X, A, T^{\mathcal{U}}, x_0, B) = M_L^\tau \times_{\mathcal{S}} \mathcal{M}_1^\tau(\leq d) \times_{\mathcal{S}} \ldots \times_{\mathcal{S}} \mathcal{M}_n^\tau(\leq d).$$

Prior to taking either of the products, we add selfloops of ε at each state of all of the components.

The alphabet of $\mathcal{U}_m(d,\phi)$ ($m \in \{1,2\}$) is similar to the set of vector labels defined in 2.2. In addition, the events in $\Sigma_i^?$ and $\Sigma_i^!$ are considered to be private (i.e., not part of the system behavior), so that the labels associated with events in these sets are generated as follows: for all $?\sigma \in \Sigma_i^?$ (equivalently for $\Sigma_i^!$) $\ell(i) = ?\sigma$, otherwise for all $r \in I \setminus \{i\}$, $\ell(r) = \varepsilon$. Finally, we have a label for τ: for all $i \in 0, \ldots, n$, $\ell(i) = \tau$. We will denote this as $\overrightarrow{\tau}$ since it is an event on which all components synchronize.

A transition $(x, \ell, x') \in T^{\mathcal{U}}$, where $x = (q, q_1, \ldots, q_n)$ and $x' = (q', q_1', \ldots, q_n')$ with label $\ell = \langle \ell(0), \ldots, \ell(n) \rangle$ iff $(q, \ell(0), q') \in T_L^\tau$, and for all $i \in I$, $(q_i, \ell(i), q_i') \in T_i^{!?\tau}$, where $T_i^{!?\tau}$ is the transition system for $\mathcal{M}_m^\tau(=d)$, respectively, $\mathcal{M}_m^\tau(\leq d)$.

We define the set of "bad" transitions as follows:

$$\begin{aligned}
&B = \{(x, \ell, x') \mid \\
&((x(0), \ell(0), x'(0)) \in T_L \setminus T_K \wedge \forall i \in I_c(\ell(0)) : (x(i), \ell(i), x'(i)) \in T_i^{!?\tau}) \vee \\
&((x(0), \ell(0), x'(0)) \in T_K \wedge \exists i \in I_c(\ell(0)) : (x(i), \ell(i), x'(i)) \notin T_i^{!?\tau})\}.
\end{aligned}$$

A transition (x, ℓ, x') is in B if either (1) this encodes a transition where event $\ell(0)$ should be disabled in L and $\forall i \in I_c(\ell(0))$ the controllers believe that $\ell(0)$ should be enabled; or (2) this encodes a transition where event $\ell(0)$ should be enabled but $\exists i \in I_c(\ell(0))$ mistakenly believe that $\ell(0)$ should be disabled.

The set of events corresponding to received messages now have a special form $\Sigma_i^?([0,\infty))$ such that each event $?(\tau_\sigma)\sigma$ has a counter that increases with every τ event. Here τ_σ counts the number of τ events before receiving the event $?\sigma$. Note that to avoid the construction of an infinite-state $\mathcal{U}_m$, we annotate the received messages after taking the product to construct $\mathcal{U}_m$, $m \in \{1, 2\}$.

To provide a sense of the construction of $\mathcal{U}_2(d, \phi)$, Figure 5.10 contains a portion of $\mathcal{U}_2(d, \phi)$ for our example. In particular, notice how M_L^τ and each of the copies of $\mathcal{M}_i^\tau$, for $i \in I$, synchronize on τ and that we annotate a received message with its counter τ_σ.

Theorem 5.4. *$B = \emptyset$ in $\mathcal{U}_1(d, \phi) \Leftrightarrow \phi$ is robust w.r.t. fixed delay d.*

Proof. Follows the proof of Theorem 5.5. (See below.) □

Theorem 5.5. *$B = \emptyset$ in $\mathcal{U}_2(d, \phi) \Leftrightarrow \phi$ is robust w.r.t. delay $[1..d]$.*

Proof. ($\Rightarrow$) Suppose that $B = \emptyset$ but ϕ is not robust w.r.t. $[1..d]$. By definition, there exists $s \in \overline{K}$ and $\sigma \in \Sigma_c$ such that $s\sigma \in L \setminus \overline{K}$ and there exists $d > 0$ $(\forall i \in I_c(\sigma))$: $\exists s_i'(d) \in \overline{K_i^?(\leq d)}$ such that $s_i'(d)\sigma \in [\![s(\leq d)]\!]_i^?\sigma \cap \overline{K_i^?(\leq d)}$, or $s\sigma \in \overline{K}$ and there exists $d > 0$ $(\exists i \in I_c(\sigma)) : \exists s_i'(d) \in [\![s(\leq d)]\!]_i^?$ such that $[\![s(\leq d)]\!]_i^?\sigma \cap K_i^?(\leq d) = \emptyset$. So we need to consider the following two cases.

Case 1: In M_L^τ, we have $q_0 \xrightarrow{s} q \xrightarrow{\sigma} q' \in T_L^\tau$, where $(q, \sigma, q') \in T_L \setminus T_K$ and in $\mathcal{M}_i^\tau(\leq d)$ we have $q_0 \xrightarrow{s_i'(d)} q_i \xrightarrow{\sigma} q_i' \in T_i^{!?\tau}$, for all $i \in I_c(\sigma)$.

From the construction of $\mathcal{U}_2(d, \phi)$, we know that $(\forall i \in I_c(\sigma))\ \pi_i^?(s) = \pi_i^?(s_i'(d))$. Thus there exists a trace in $\mathcal{U}_2(d, \phi)$ such that $x_0 \xrightarrow{w} x$, where $w(0) = s$ and $w(i) =$

$s'_i(d)$, for all $i \in I_c(\sigma)$. In addition, we have $(x, \ell, x') \in T^{\mathcal{U}}$ where $x(0) = q$, $\ell(0) = \sigma$, $x'(0) = q'$ and $x(i) = q_i$, $\ell(i) = \sigma$, $x'(i) = q'_i$. By the definition of B, we have $(x, \ell, x') \in B$, thereby reaching a contradiction.

Case 2: In M_L^τ, we have $q_0 \xrightarrow{s} q \xrightarrow{\sigma} q' \in T_L$, where $(q, \sigma, q') \in T_K$ and $\exists i \in I_c(\sigma)$, we have $q_0 \xrightarrow{s'_i(d)} q_i \xrightarrow{\sigma} q'_i \notin T_i^{!?\tau}$ in $\mathcal{M}_i^\tau(\leq d)$.

In $\mathcal{U}_2(d, \phi)$, there exists a trace such that $x_0 \xrightarrow{w} x$, where $w(0) = s$ and $w(i) = s'_i(d)$, for all $i \in I_c(\sigma)$. We also have $(x, \ell, x') \in T^{\mathcal{U}}$ where $s\ell(0) \in \overline{K}$, but $s'_i(d)l(i) \notin K_i^?(d)$ with $x(0) = q$, $\ell(0) = \sigma$, $x'(0) = q'$ and $x(i) = q_i$, $\ell(i) = \sigma$, $x'(i) = q'_i$. Then $(x(i), l(i), x'(i)) \notin T_i^{!?\tau}$. By the definition of B, we have $(x, \ell, x') \in B$, arriving at a contradiction.

($\Leftarrow$) Suppose that $B \neq \emptyset$ but ϕ is robust w.r.t. $[1..d]$. Let $(x, \ell, x') \in B$. By definition, we have two cases to consider:

Case 1: By definition, $(x(0), \ell(0), x'(0)) \in T_L \setminus T_K$ and $\forall i \in I_c(\ell(0))$, $(x(i), \ell(i), x'(i)) \in T_{K_i}$ and $\ell(i) = \ell(0)$. Let $w = \ell_1 \ell_2 \ldots \ell_{|w|} \in A^*$ such that $x_0 \xrightarrow{w} x \in T^{\mathcal{U}}$. Let $s = w(0)$ and $\forall i \in I_c(\ell(0))$, there exists $d > 0$ such that $s'_i(d) = w(i)$. Since $x_0 \xrightarrow{w} x \xrightarrow{\ell} x' \in T^{\mathcal{U}}$, we have $s\ell(0) \in L \setminus \overline{K}$ and $s'_i(d)\ell(i) \in \overline{K_i^?(d)}$, $\forall i \in I_c(\sigma)$. The only labels on which s and s'_i synchronize are those of the form $\ell = \langle \ell(0) = \gamma, \ldots, \ell(i) = \gamma, \ldots \rangle$ in which $\gamma \in \Sigma_o$ and $\gamma \in \Sigma_{o,i}$, therefore $\pi_i^?(s) = \pi_i^?(s'_i(d))$. Thus we have $s'_i(d)\ell(i) \in [\![s]\!]_i^? \ell(i) \cap \overline{K_i^?(d)}$, arriving at a contradiction that ϕ is robust w.r.t. $[1..d]$.

Case 2:$(x(0), \ell(0), x'(0)) \in T_K$ and $\exists i \in I_c(\sigma)$, $(x(i), \ell(i), x'(i)) \notin T_i^{!?\tau}$ and $\ell(i) = \ell(0)$. Let there be a trace as before $x_0 \xrightarrow{w} x \in T^{\mathcal{U}}$, where $s = w(0)$, and there exists $d > 0$ such that $s'_i(d) = w(i)$ for $i \in I_c(\ell(0))$. We have $s\ell(0) \in \overline{K}$, since $x_0 \xrightarrow{w} x \xrightarrow{\ell} x' \in T_K$, but $\exists i \in I_c(\ell(0))$ such that $s'_i(d)\ell(i) \notin \overline{K_i^?(d)}$. Therefore $s'_i(d)\ell(i) \cap \overline{K_i^?(d)} = \emptyset$, contradicts our assumption that ϕ is robust w.r.t. $[1..d]$. $\square$

The two types of bad transitions are illustrated in Figure 5.11. In the top portion

of the figure, the transition $(5^{\tau}, 6', 6, 6) \xrightarrow{\langle\sigma,\sigma,\sigma,\sigma\rangle} (7, 8'', 8, 8)$ is marked because σ must be disabled according to M_L (colored here in red) whereas all the controllers in $I_c(\sigma)$ believe that σ should be enabled (colored in green). Under a synchronous communication protocol, controller 1 would have received the information that b occurred with zero delay, thus being able to take the correct control decision about σ; however, when the message is delayed, in this case the delay is $\tau = 2$ clock cycles, as evident from the event label $?\mathsf{b}(2)$ that occurs after the marked transition, controller 1 cannot take the correct control decision. Another bad transition $(6^{\tau}, 5''^{?}, 5, 5) \xrightarrow{\langle\sigma,\sigma,\sigma,\sigma\rangle} (8, 7, 7, 7)$, in the bottom of the figure, causes the opposite problem. With a synchronous communication policy, controller 3 would know immediately when a occurs; however in this case, controller 3 cannot distinguish between a message arriving with 0 delay and one with a delay of 2. Thus it mistakenly believes that it is issuing a control decision for σ after bca instead of the actual sequence that has occurred, namely abc.

Our strategy verifies the robustness of all synchronous communication protocols under conditions of fixed or finitely-bounded delay, not just optimal communication protocols. We do, though, assume that the original communication protocol does not require the communication of all observations. Verifying delay-robustness of previously-synthesized synchronous communication protocols is easily achieved by adapting existing techniques for verifying properties of decentralized discrete-event systems. This seems simpler than directly synthesizing communication protocols w.r.t. a fixed or a finitely-bounded delay. Since the communication protocol is already synthesized for zero delay, and thus we have communicating controllers, it is simpler to verify the robustness of a communication protocol rather than synthesize a new protocol and a set of new communicating controllers.

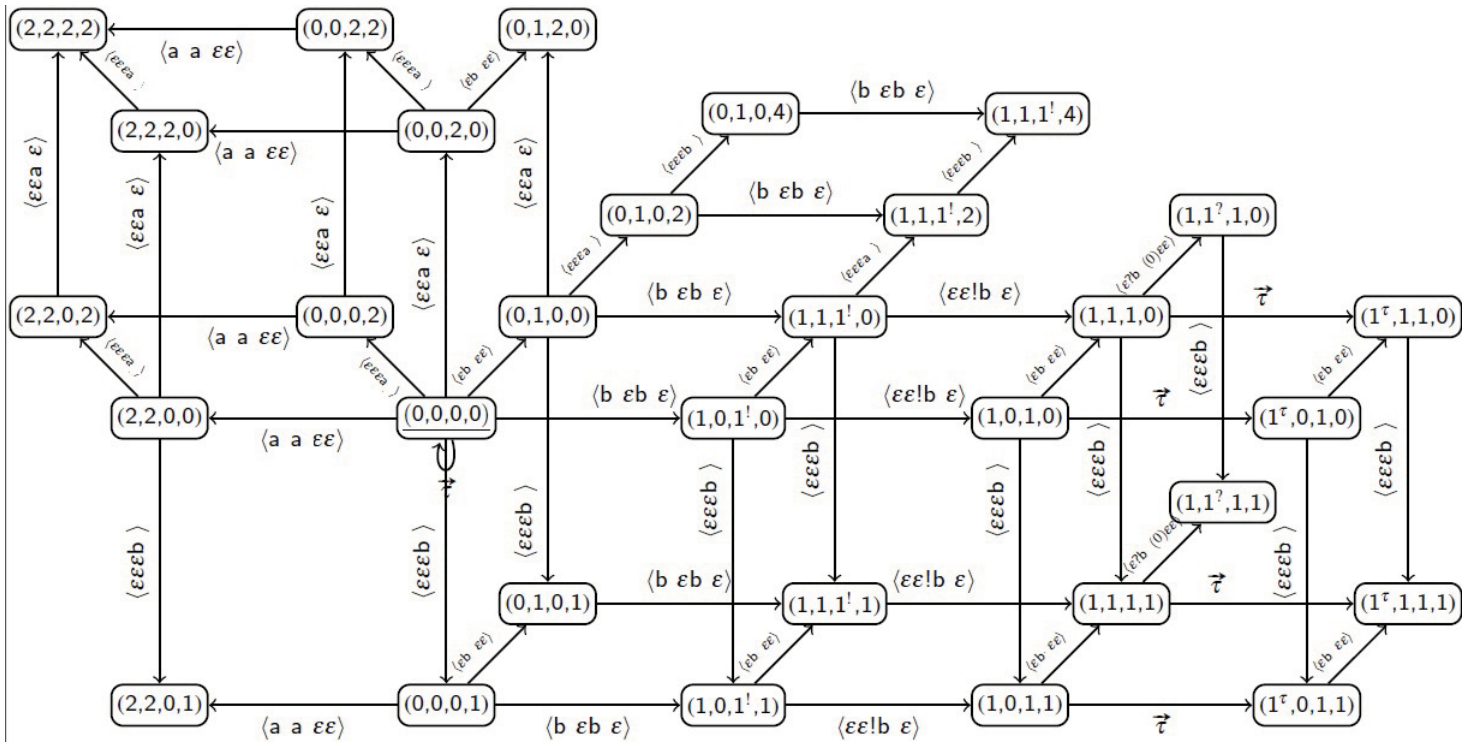


Figure 5.10: A portion of $\mathcal{U}_2(d,\phi) = M_L^\tau \times_{\mathcal{S}} \mathcal{M}_1^\tau(\leq 2) \times_{\mathcal{S}} \mathcal{M}_2^\tau(\leq 2) \times_{\mathcal{S}} \mathcal{M}_3^\tau(\leq 2)$ (initial state is underlined for readability).

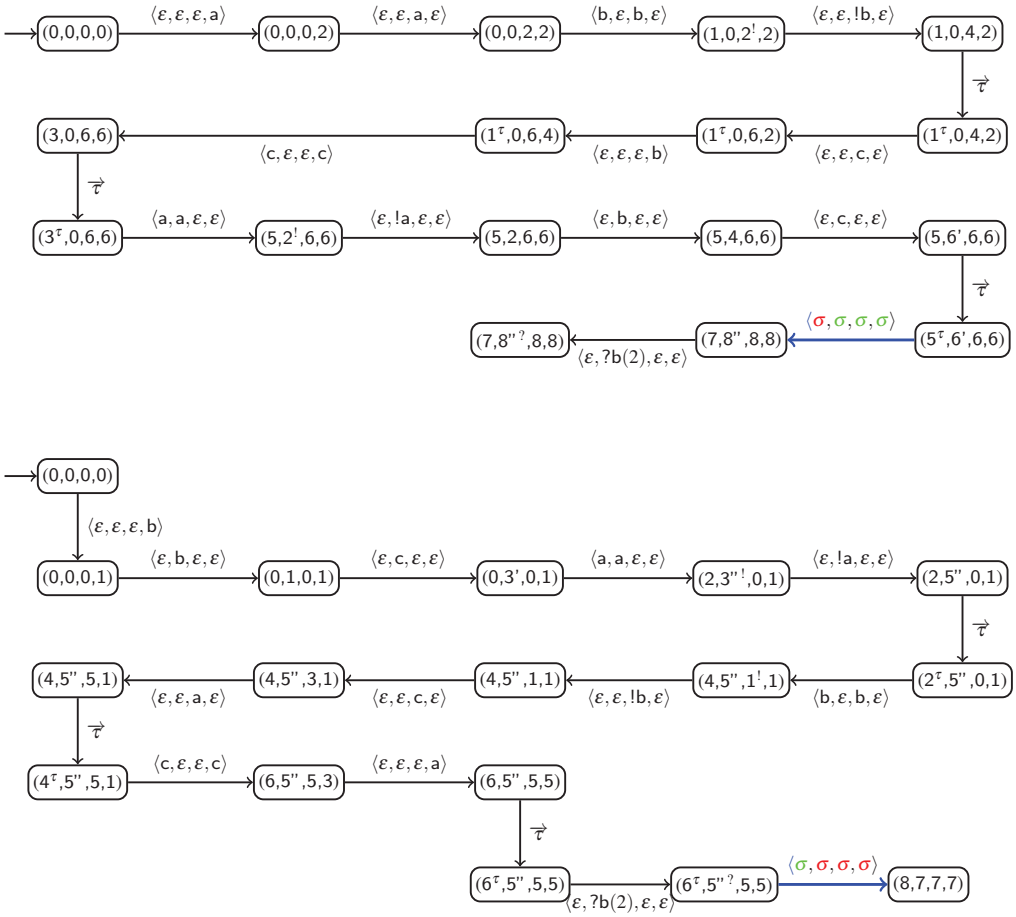


Figure 5.11: A portion of $\mathcal{U}_2(d,\phi)$ with bad transitions highlighted in blue (top) $(5^\tau, 6', 6, 6) \xrightarrow{\langle\sigma,\sigma,\sigma,\sigma\rangle} (7, 8'', 8, 8)$ where no controller can take the correct control decision and (bottom) $(6^\tau, 5''^?, 5, 5) \xrightarrow{\langle\sigma,\sigma,\sigma,\sigma\rangle} (8, 7, 7, 7)$ where all controllers incorrectly believe that σ should be disabled.

Chapter 6

Decentralized TDES Control with Communication

Untimed DES models are concerned with whether each event occurs at the logical order in the plant. The exact time at which each event occurs is not explicitly modeled. In many applications, however, the exact time that each event occurs is important. This chapter deals with the decentralized supervisory control and communication problem in timed DES (TDES), in which the passage of time is modeled using a tick event.

Synchronous communication protocols have been synthesized for untimed decentralized discrete-event control problems where controllers transmit their information through a zero-delay communication channel. But, in practice, communication occurs with some delay in the channel. Since the only difference between a TDES and an untimed DES is the occurrence of tick events, the same approach for untimed DES can be used to synthesize a communication protocol in a TDES with synchronous (instantaneous) communication. Motivated by the above observation, instead of synthesizing communication protocols for decentralized TDES control problems with bounded delay communication, in this chapter we will discuss a procedure for converting the

problem to an equivalent TDES control problem with (zero-delay) synchronous communication, and synthesize communication protocols for the converted problem. The proposed approach is developed for acyclic TDES.

6.1 Control and Communication Problem in TDES

In reality, communication occurs with some delay among the controllers. We consider a channel with a delay of known upper-bound d in the communication. In delayed communication with upper-bound d, at most d tick events can occur in the plant between sending a message by one controller and the reception of that message by another controller. Timing information of the occurrence of an event has to be captured in a useful way in the model of TDES.

We assume that controllers send messages through communication channels (not necessarily FIFO)[1]. To that end, we start with a TDES model shown in Figure 6.1. In the system, we have a plant (M_{act}) and a communication channel ($\mathcal{C}^d$). For brevity, throughout this chapter, we will assume there are only two controllers and that only Controller 1 can communicate to Controller 2.

The automaton for representing the plant is modeled as an activity transition graph (ATG):

$$M_{\mathsf{act}} = (A, \Sigma_{\mathsf{act}}, T_{\mathsf{act}}, a_0, A_{\mathsf{m}}).$$

M_{act} is assumed to be acyclic. Let $\Sigma = \Sigma_{\mathsf{act}} \cup \{\tau\}$, and L be the closed behaviour of the corresponding TTG.

We assume that when communicating an event observation, Controller 1 time stamps the event and transmits the number of tick events passed (since the initial state) when the event occurred in the plant.

[1] As an example, communication using a computer network using TCP/IP protocol is not necessarily FIFO

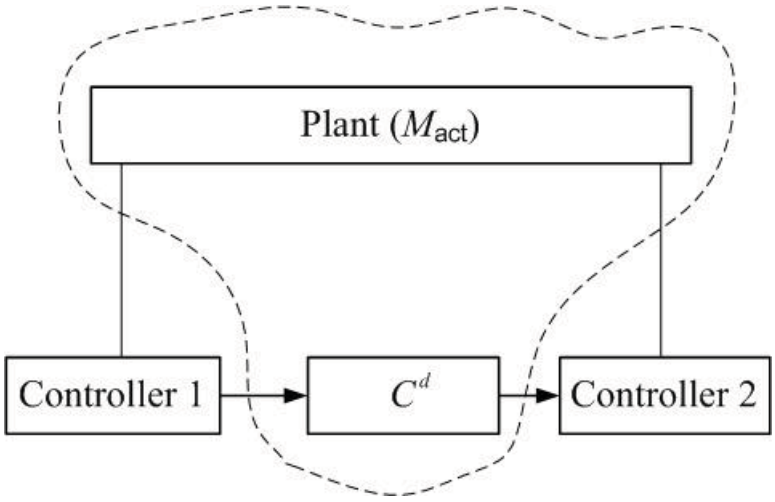


Figure 6.1: A TDES model with a communication channel between controllers 1 and 2.

Definition 6.1. *A **communication protocol** for Controller* 1 $\phi_{1,2} : L \to ((\Sigma_{o,1} - \Sigma_{o,2}) \times \mathbb{N})) \cup \{\varepsilon\}$ *is defined as follows.*

($\forall s \in L$*) (*$\forall \sigma \in \Sigma$*) such that* $s\sigma \in L$

$$\phi_{1,2}(s\sigma) = \begin{cases} \langle \sigma, \tau_{s\sigma} \rangle, & \text{if } \sigma \in \Sigma_{o,1} \setminus \Sigma_{o,2} \text{ and } s\sigma \in L_{1,2} \subseteq L, \\ \varepsilon, & \text{otherwise.} \end{cases}$$

Here $L_{1,2} \subseteq L$ is the set of sequences after which Controller 1 communicates to Controller 2, and $\tau_{s\sigma}$ is the number of tick events through the sequence $s\sigma$ before σ occurs.

We want communication to occur in an observationally-equivalent manner.

Definition 6.2. *In TDES with a delay of upper-bound d in the communication, a communication protocol* $\phi_{1,2}$ *is **coherent** if for all sequences* $s, s' \in L$,

$$\pi_1(s) = \pi_1(s') \Rightarrow \phi_{1,2}(s) = \phi_{1,2}(s'),$$

where π_1 is the natural projection $\pi_1 : \Sigma^* \to \Sigma_{o,1}^*$. It must be the case that the

same communication occurs after all sequences that result in the same observation in Controller 1.

Let $\Sigma^? = \{?\sigma | \sigma \in \Sigma_{o,1} - \Sigma_{o,2}\}$ be the set of message delivery labels. Once a message is received by Controller 2, given the time stamp and the current time, Controller 2 can determine the delay experienced by the message in the channel. Therefore, without loss of generality, we assume that Controller 2 receives the event with the delay experienced. Hence, the set of "message delivery events" is $\Sigma^? \times [0, d]$. Now the plant M_{act} and channel $\mathcal{C}^d$ can be viewed as the system-to-be-controlled with event set $\Sigma_{\mathsf{tot}} = \Sigma \cup (\Sigma^? \times [0, d])$. The natural projection $\pi_\Sigma : \Sigma^*_{\mathsf{tot}} \to \Sigma^*$ is later considered which removes those events from sequences in Σ^*_{tot} that do not belong to the plant behaviour.

6.1.1 Decentralized Control Law with Communication

In decentralized TDES, each controller decides which events are enabled and which are forced to occur after each sequence based on its own observation and the messages received from other controllers. Then a decentralized control law, with a delay of upper-bound d in the communication channel for controller 1, is a mapping

$$\Gamma_1 : \pi_1(L) \to Pwr(\Sigma) \times Pwr(\Sigma_{f,1}),$$

which defines the set of events that Controller 1 believes should be *enabled* and the set of events *forced* to occur based on its partial view. According to the definition, we have two components of Γ_1 as $(\Gamma_{e,1}, \Gamma_{f,1})$, where

$$(\forall i \in I)(\forall s \in L)\ \Gamma_{e,1}(\pi_1(s)) = \{\gamma \in \Sigma \mid \gamma \supseteq \Sigma_{uc,1}\}.$$

$\Gamma_{f,1}$ defines the set of events to be forced by Controller 1 after its partial observation.

As discussed in Section 6.1, Controller 2 makes decisions based on its partial observations and "message delivery events". The natural projection modeling these observations is $\pi_2^?$: $\Sigma_{\mathsf{tot}}^* \to (\Sigma_{o,2} \cup (\Sigma^? \times [0,d]))^*$. Now, the control map for Controller 2 is

$$\Gamma_2 : \Sigma_{\mathsf{tot}}^* \to Pwr(\Sigma_{\mathsf{tot}}) \times Pwr(\Sigma_{f,2}),$$

subject to the condition that observationally-equivalent sequences in Σ_{tot}^* must result in the same control decision.

The decentralized control and communication problem in TDES with known upper-bound delay in the communication channel is described as follows.

Problem 6.1. *Consider two regular languages K, L over a common alphabet Σ. Assume $K \subseteq L$ is controllable w.r.t. L, Σ_{uc}, observable w.r.t. L, $\pi : \Sigma^* \to (\Sigma_{o,1} \cup \Sigma_{o,2})^*$ and Σ_c, but not co-observable w.r.t. L, π_i and $\Sigma_{c,i}$ $(i = \{1,2\})$. Suppose there is a communication channel with a delay of upper-bound d for transmitting messages from Controller 1 to 2. Construct a coherent communication protocol $\phi_{1,2}$, such that $\pi_\Sigma(L(\Gamma_1 \wedge \Gamma_2 / M_{\mathsf{tot}})) \subseteq K$, where M_{tot} refers to the DES to be controlled (plant and communication channel).*

Example 6.1. *The ATG model of the plant and the specification is shown in M_{act} in Figure 6.2. The corresponding TTG is shown in Figure 6.3. In the TTG, 3 tick events occur between States 3 and 3^τ, and 7 and 7^τ, which is denoted as $\tau^{(3)}$. Suppose the legal behaviour can also be modeled by the ATG with solid activities only. Let $I = \{1,2\}$, $\Sigma_{o,1} = \Sigma_{c,1} = \{\mathsf{a},\mathsf{c}\}$, $\Sigma_{o,2} = \Sigma_{c,2} = \{\mathsf{b},\sigma\}$ and $I_c(\sigma) = \{2\}$. Consider $s = \tau\mathsf{a}\tau\mathsf{b}\tau\tau\tau\tau\mathsf{c}\tau\sigma$, and $s' = \tau\tau\mathsf{b}\tau\mathsf{a}\tau\tau\tau\mathsf{c}\tau\sigma$. Note that K is not co-observable, since $\pi_2(s) = \pi_2(s') = \tau\tau\mathsf{b}\tau\tau\tau\tau\tau\sigma$. In the next section, we will discuss a solution to this problem assuming communication channel with delay bounded by 2 ticks.*

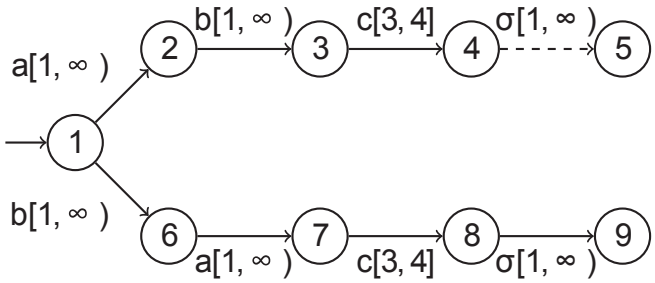


Figure 6.2: M_{act} is the collection of all transitions, and $M_{K,\mathsf{act}}$ is the collection of only solid-line transitions.

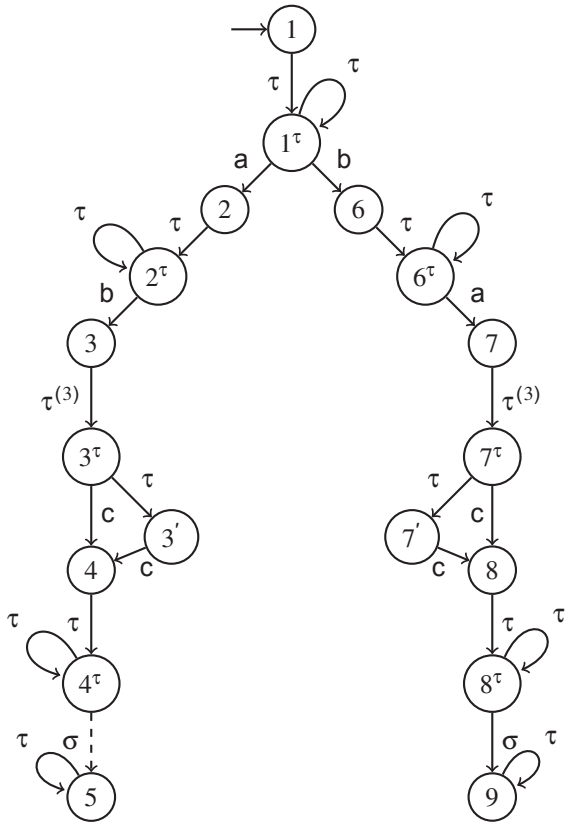


Figure 6.3: TTG of M_{act} shown in Figure 6.2.

6.2 Conversion to an Equivalent Problem with Synchronous Communication

In this section, we describe a procedure for converting Problem 6.1 posed in the previous section, which involved control and communication over a channel with bounded delay, to an equivalent problem of control and synchronous communication. The resulting problem can be solved using various procedures such as those discussed in Chapters 3 and 4.

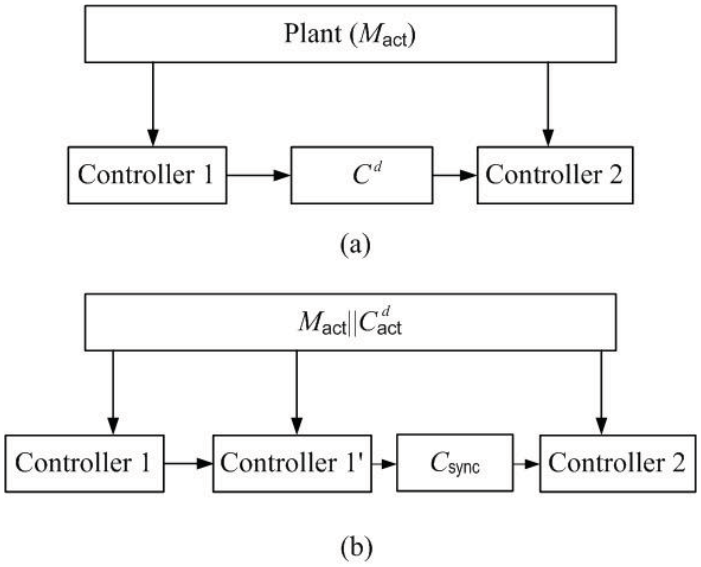


Figure 6.4: Block diagram shows information flow of the decentralized control problem in TDES (a) with a delayed communication of upper-bound d between controllers 1 and 2, and (b) with synchronous communication between Controllers $1'$ and 2.

The problem and the converted version are displayed in Figure 6.4(a) and (b). In the original problem, Controller 1 based on its communication policy transmits some events from $\Sigma_{o,1} - \Sigma_{o,2}$ over the communication channel $\mathcal{C}^d$. An event $\sigma \in \Sigma_{o,1} - \Sigma_{o,2}$ transmitted over the channel is delivered in between 0 and d clock ticks. As before, $?\sigma$ denotes the delivery of σ (i.e., reception by Controller 2). In the converted

problem (Figure 6.4(b)), the transmission of events over the communication channel is modeled using an ATG $\mathcal{C}^d_{\mathsf{act}}$. $\mathcal{C}^d_{\mathsf{act}}$ models the transmission of *every* occurrence of events in $\Sigma_{o,1} - \Sigma_{o,2}$. Construction of $\mathcal{C}^d_{\mathsf{act}}$ will be discussed shortly. Controller 1 similar to the original problem based on its observations enables or disables certain events in the plant. The transmission of some its observation to Controller 2 is done by a dummy controller $1'$. Controller $1'$ makes all observations that Controller 1 can, in addition to "delivery" events $?\sigma \in \Sigma^?$. Thus $\Sigma_{o,1'}=\Sigma_{o,1} \cup \Sigma^?$. Controller $1'$ based on its communication policy (to be designed) transmits certain events $?\sigma \in \Sigma^?$ synchronously (instantaneously) over a fictitious communication channel $\mathcal{C}_{\mathsf{sync}}$.

Next we consider how $\mathcal{C}^d_{\mathsf{act}}$ can be constructed. $\mathcal{C}^d_{\mathsf{act}}$ models the generation of all $?\sigma \in \Sigma^?$. For each $?\sigma \in \Sigma^?$, we construct an ATG shown in Figure 6.5.

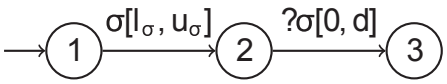


Figure 6.5: ATG $\mathcal{C}^d_\sigma$.

We let l_σ and u_σ be the lower and upper time bounds of $\sigma \in \Sigma_{o,1} - \Sigma_{o,2}$ (in case of remote events $[l_\sigma, \infty)$). Note that the lower and upper time bounds of "message delivery" events are 0 and channel upper bound delay d. Thus the ATG $||\{\mathcal{C}^d_\sigma | \sigma \in \Sigma_{o,1} - \Sigma_{o,2}\}$ models the generation of events σ and delivery events $?\sigma$.

Remark: The above procedure allows for one instance of transmission for every $\sigma \in \Sigma_{o,1} - \Sigma_{o,2}$. For $\sigma \in \Sigma_{o,1} - \Sigma_{o,2}$, we define

N_σ = max of the number of σ in every sequence of activities in M_{act}.

Since by assumption, M_{act} is acyclic, N_σ is finite. If $N_\sigma > 1$, in the process of building $\mathcal{C}^d_{\mathsf{act}}$, the successive occurrences of σ are labeled starting from 1 up to N_σ. An example is shown in Figure 6.6, where a can occur up to $N_{\mathsf{a}} = 2$, and all occurrences of a are labeled according to the order of occurrence. In building $\mathcal{C}^d_{\mathsf{act}}$, each labeled version of σ will be treated as unique event. So for Figure 6.6, we construct $\mathcal{C}^d_{\mathsf{a}_1}$ and

$\mathcal{C}^d_{\mathsf{a}_2}$. Once $\mathcal{C}^d_{\mathsf{act}}$ is built, the labels will be removed. □

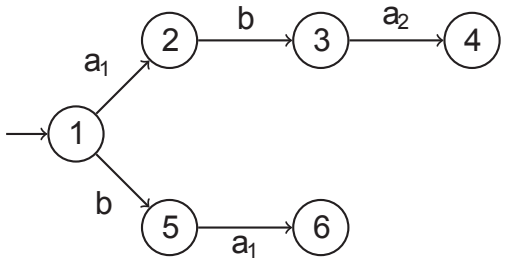


Figure 6.6: An ATG with $N_{\mathsf{a}} = 2$.

In the converted problem (Figure 6.4(b)), the system-to-be-controlled is modeled by the extended ATG $M_{\mathsf{act,ext}} = M_{\mathsf{act}}||\mathcal{C}^d_{\mathsf{act}}$ and the objective is to design control policies for Controller 1 and 2, and communication policy for Controller 1′.

Let M_{ext} be the TTG obtained for the extended system from the ATG $M_{\mathsf{act,ext}}$. As mentioned in Section 6.1, the transmitted events in the original problem is time-stamped and that is equivalent to Controller 2 receiving the information about the delay each message experiences in the channel. To include this information in the converted problem, in M_{ext} for every $?\sigma$ transition, we replace $?\sigma$ as the label with $(?\sigma, n_\sigma)$, where n_σ is the number of ticks between occurrence of σ (that resulted in "delivery" event $?\sigma$) and $?\sigma$. Thus the event set of M_{ext} becomes $\Sigma_{\mathsf{tot}} = \Sigma \cup (\Sigma^? \times [0, d])$ (as defined previously in Section 6.1).

Let the closed behaviour of M_{ext} be denoted by L_{ext} and the specification be K_{ext} $= \pi_\Sigma^{-1}(K)$. The communication protocol $\phi_{1',2}$ is a map $\phi_{1',2} : L_{\mathsf{ext}} \to (\Sigma^? \times [0, d]) \cup \{\varepsilon\}$.

We want the communication to occur in an observationally-equivalent manner. Since in the original problem, communication decisions were based on the local observations of Controller 1, in the converted problem that must be the case too. In other words, similar to Definition 6.2, two sequences are considered equivalent if they result in the same observation for Controller 1 (not Controller 1′, otherwise Controller 1′ would know the delay experienced in the channel by each message which is of course

unknown before transmission).

Definition 6.3. *In the converted problem, a communication protocol* $\Phi_{1',2}$ *is* ***coherent*** *if for all sequences* $s, s' \in L_{ext}$,

$$\pi_{1,ext}(s) = \pi_{1,ext}(s') \Rightarrow \phi_{1',2}(s) = \phi_{1',2}(s'),$$

where $\pi_{1,\mathsf{ext}}$ is the natural projection $\pi_{1,\mathsf{ext}}$: $\Sigma^*_{\mathsf{tot}} \rightarrow \Sigma^*_{o,1}$.

In the converted problem, K_{ext} is communication observable if there exists a set of controllers that can take correct control decision based on its observation and the information received from the dummy controller.

Definition 6.4. K_{ext} *is* ***communication observable*** *w.r.t.* L_{ext}, $\Sigma_{o,1}$, $\Sigma_{o,2} \cup \Sigma^?$, $\Sigma_{c,i}$ $(i \in \{1,2\})$, *and* $\phi_{1',2}$ *iff*

$$(\forall s \in \overline{K_{ext}})(\forall \sigma \in \Sigma_c)\ s\sigma \in L_{ext} \setminus \overline{K_{ext}} \Rightarrow (\exists i \in \{1,2\})\ [\![s]\!]^?_i \sigma \cap \overline{K_{ext}} = \emptyset.$$

A decentralized control law in TDES with synchronous communication is defined as a mapping for Controller 1:

$$\Gamma_1 : \pi_{1,\mathsf{ext}}(L_{\mathsf{ext}}) \rightarrow Pwr(\Sigma_{\mathsf{ext}}) \times Pwr(\Sigma_{f,1}),$$

which defines the set of events to be *enabled* and the set of events *forced* to occur by controller 1, based on its partial observation. As before, Γ_1 is decomposed as $(\Gamma_{e,1}, \Gamma_{f,1})$, where

$$(\forall s \in L_{\mathsf{ext}})\ \Gamma_{e,1}(\pi_{1,\mathsf{ext}}(s)) = \{\gamma \in Pwr(\Sigma_{\mathsf{ext}}) | \gamma \supseteq \Sigma_{uc,1}\}.$$

$\Gamma_{f,1}$ defines the set of events to be forced by controller 1 after its partial observation.

As before, Controller 2 makes decisions based on its partial observations and "message delivery events". The control map for Controller 2 is

$$\Gamma_2 : \Sigma^*_{\mathsf{tot}} \to Pwr(\Sigma_{\mathsf{ext}}) \times Pwr(\Sigma_{f,2}),$$

subject to the condition that observationally-equivalent sequences in Σ^*_{tot} must result in the same control decision.

We consider in the above equation that the "message delivery events" are incorporated in M_{ext} with delay $\leq d$ and the message is received without any delay in the fictitious communication channel. For example, if a message $\delta \in \Sigma_{o,1} \backslash \Sigma_{o,2}$ is sent after a sequence $s \in L_{\mathsf{ext}}$ by Controller 1 to Controller 2 with a delay of upper-bound 2, then there exists $s' \in \Sigma^*_{\mathsf{tot}}$ containing up to 2 tick events such that controller 2 receives $?\delta \in \Sigma^?$ after ss'. In M_{ext}, the received message $?\delta$ is added after ss'. Since there is no delay in the communication, controller 2 receives $?\delta$ instantaneously after ss' occurs.

The decentralized control and communication problem in TDES with synchronous communication is described below.

Problem 6.2. *Given two regular languages K_{ext}, L_{ext} over a common alphabet Σ_{tot}, where $K_{ext} \subseteq L_{ext}$ is observable w.r.t. L_{ext}, π and Σ_c, but not co-observable w.r.t. L_{ext}, π_i and $\Sigma_{c,i}$ $(i = \{1, 2\})$. Construct a coherent communication protocol $\phi_{1',2}$, such that K_{ext} is communication observable w.r.t. L_{ext}, $\Sigma_{o,1}$, $\Sigma_{o,2} \cup \Sigma^?$, $\Sigma_{c,i}$, $(i = \{1, 2\})$ and $\phi_{1',2}$.*

For Problem 6.1 and 6.2, the set of observable events $\Sigma_{o,i}$, and controllable events $\Sigma_{c,i}$ are the same for all $i \in I$. Similarly a received message, $?\sigma \in \Sigma^?$ is observable to Controller 2 in both problems. For instance, if $\sigma \in \phi_{1,2}$, this means that Controller 1 communicates the occurrence of event σ to Controller 2, then the corresponding

"message delivery event" is received by Controller 2, i.e., $?\sigma \in \Sigma^{?}$. The received messages are also observable to the corresponding dummy Controller $1'$ in Problem 6.2, but the dummy controller is not able to disable or force any event.

Example 6.1 (continued). Next we consider $\mathcal{C}^d_{\mathsf{act}}$ and the extended plant $M_{\mathsf{act,ext}}$.

The ATG models for events in $\Sigma_{o,1} - \Sigma_{o,2} = \{a, c\}$ are given in Figure 6.7.

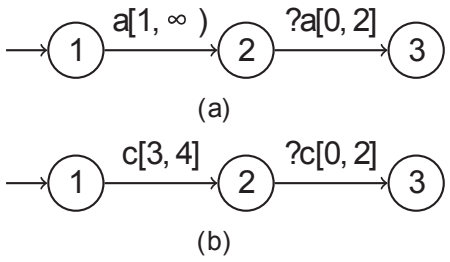


Figure 6.7: (a) $\mathcal{C}^d_{\mathsf{a}}$ and (b) $\mathcal{C}^d_{\mathsf{c}}$.

Finally $\mathcal{C}^d_{\mathsf{act}} = \mathcal{C}^d_{\mathsf{a}} || \mathcal{C}^d_{\mathsf{c}}$ and $M_{\mathsf{act,ext}} = M_{\mathsf{act}} || \mathcal{C}^d_{\mathsf{act}}$. $M_{\mathsf{act,ext}}$ has 24 states and 36 transitions. Let M_{ext} denote the TTG of $M_{\mathsf{act,ext}}$. Part of M_{ext} is shown in Figure 6.8. In M_{ext}, each delivery message $?\mathsf{a}$ is replaced with $(?\mathsf{a}, n_d)$, where n_d is the delay in clock ticks from the generation of a to occurrence of $?\mathsf{a}$. We consider no forcible events in the plant, so that τ is uncontrollable in that case. While M_{ext} is the collection of all transitions, $M_{K,\mathsf{ext}}$ is the collection of only solid-line transitions. Let consider $s_1 = \tau\mathsf{a}\tau\tau\mathsf{b}?\mathsf{a}\tau\tau\tau\tau\mathsf{c}\tau\tau$ (with a communication delay of 2), $s_2 = \tau\mathsf{a}\tau?\mathsf{a}\tau\mathsf{b}\ \tau\tau\tau\tau\mathsf{c}\tau\tau$ (with a communication delay of 1), and $s'_1 = \tau\tau\tau\mathsf{b}\tau\mathsf{a}\tau?\mathsf{a}\tau\tau\tau\mathsf{c}\tau$, $s'_2 = \tau\tau\tau\mathsf{b}\tau\mathsf{a}\tau?\mathsf{a}\tau\tau\tau\mathsf{c}\tau$. Here K_{ext} is not co-observable w.r.t. L_{ext} and $\Sigma_{o,2}$, since $\pi_2(s_1) = \pi_2(s'_1) = \tau\tau\tau\mathsf{b}\tau\tau\tau\tau\tau$ and $\pi_2(s_2) = \pi_2(s'_2) = \tau\tau\tau\mathsf{b}\tau\tau\tau\tau\tau$, i.e., Controller 2 cannot take correct control decision regarding σ.

The set of observable events for dummy controllers are $\Sigma_{o,1'} = \{?\mathsf{a}, ?\mathsf{c}\}$. We

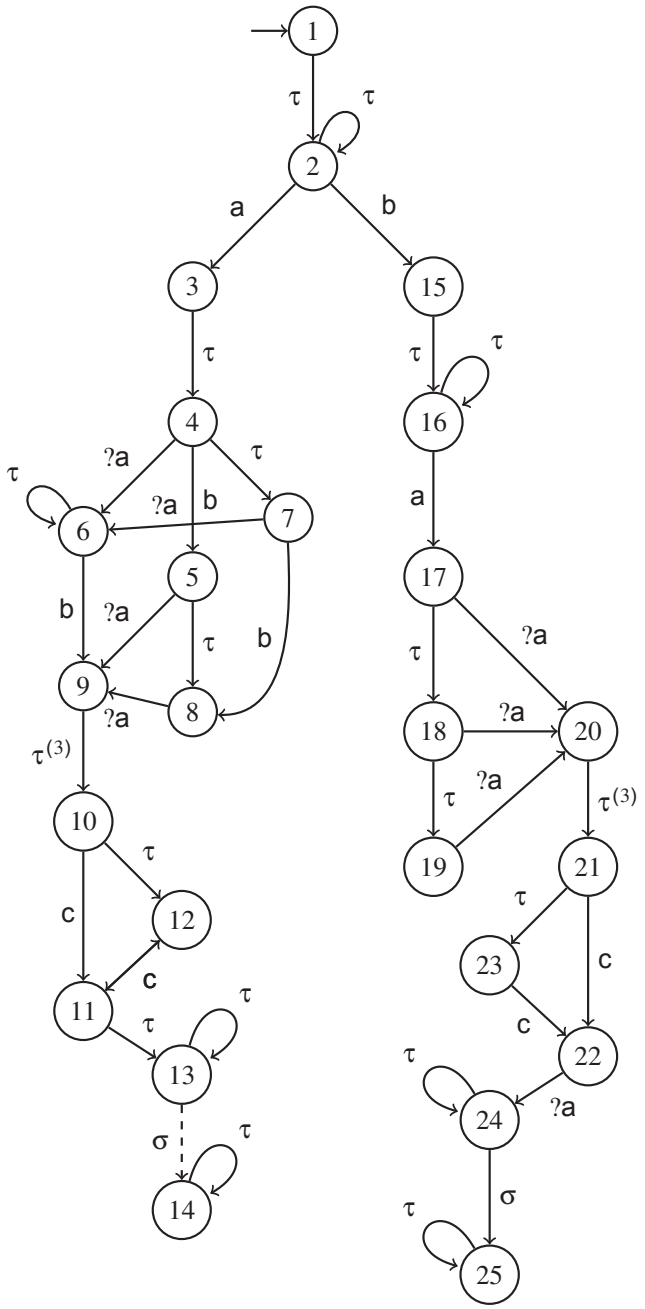


Figure 6.8: A portion of TTG of the extended plant, M_{ext}.

consider a corresponding communication protocol

$$\phi_{1',2} = \begin{cases} ?\mathsf{a}, & \text{if } s \text{ ends in } ?\mathsf{a}, \\ \varepsilon, & \text{otherwise,} \end{cases}$$

where a dummy controller $1'$ sends $?\mathsf{a}$ to Controller 2 through a synchronous communication channel.

This is equivalent to

$$\phi_{1,2} = \begin{cases} ?\mathsf{a}, & \text{if } s \text{ ends in } \mathsf{a}, \\ \varepsilon, & \text{otherwise.} \end{cases}$$

Then, $\pi_2(s_1) = \tau\varepsilon\tau\tau\mathsf{b}\,\langle ?\mathsf{a}, 2\rangle\,\tau\tau\tau\tau\varepsilon\tau\tau \neq \pi_2(s_1') = \tau\tau\tau\mathsf{b}\tau\varepsilon\tau\,\langle ?\mathsf{a}, 1\rangle\,\tau\tau\tau\varepsilon\tau$. Note the time stamp allows us to know that in s_1, event a occurred before b.

Similarly $\pi_2(s_2) = \tau\varepsilon\tau\,\langle ?\mathsf{a}, 1\rangle\,\tau\mathsf{b}\tau\tau\tau\tau\varepsilon\tau\tau \neq \pi_2(s_2') = \tau\tau\tau\mathsf{b}\tau\varepsilon\tau\,\langle ?\mathsf{a}, 1\rangle\,\tau\tau\tau\varepsilon\tau$. It can be verified that K_{ext} is communication observable w.r.t. L_{ext}, $\Sigma_{o,2} \cup \Sigma^?$, $\Sigma_{c,2}$ and $\Phi_{1',2}$.

The following proposition shows that a solution for Problem 6.2 gives a solution for Problem 6.1 and vice versa.

Proposition 6.1. *$\pi_\Sigma(L(\Gamma_1 \wedge \Gamma_2/M_{tot})) \subseteq K$ with $\phi_{1,2}$ iff K_{ext} is communication observable w.r.t. L_{ext}, $\Sigma_{o,1}$, $\Sigma_{o,2} \cup \Sigma^?$, $\Sigma_{c,i}$, ($i \in \{1,2\}$) and $\phi_{1',2}$.*

Proof. ($\Rightarrow$) Suppose that $\pi_\Sigma(L(\Gamma_1 \wedge \Gamma_2/M_{\mathsf{tot}})) \subseteq K$ with $\phi_{1,2}$. Then $(\forall s \in L)$, $(\forall \sigma \in \Sigma_c)$, $s\sigma \in L\backslash\overline{K}$, there exists $i \in \{1,2\}$ that can take correct control decision. That means controller i disables σ after the sequence $s \in L$ with the communication protocol $\phi_{1,2}$ and any delay not longer than d. Hence, there exists no $s' \in K$ observationally equivalent to $s\sigma$.

In M_{ext}, the received messages $?\sigma \in \Sigma^?$ are incorporated with all possible delays between 0 and d. So for a sequence $s \in L$ with $\phi_{1,2}$ and any delay not longer than

d such that $[\![s_1]\!]_i^? \in L_{\mathsf{ext}}$, there exists a corresponding sequence $s_1 \in L_{\mathsf{ext}}$ with $\phi_{1',2}$. Since there exists no $s' \in L$ observationally equivalent to $s\sigma$, we conclude that there exists no $s'_1 \in L_{\mathsf{ext}}$ such that $s'_1 \in K_{\mathsf{ext}}$ and $[\![s_1]\!]_i^?\sigma = s'_1$ with $\phi_{1',2}$. That means $[\![s_1]\!]_i^?\sigma \cap \overline{K_{\mathsf{ext}}} = \emptyset$. Then K_{ext} is communication observable w.r.t. L_{ext}, $\Sigma_{o,1}$, $\Sigma_{o,2} \cup \Sigma^?$, $\Sigma_{c,i}$, $(i \in \{1,2\})$ and $\phi_{1',2}$.

($\Leftarrow$) Suppose that K_{ext} is communication observable, then $(\forall s \in L_{\mathsf{ext}})$, $(\forall \sigma \in \Sigma_c)$, $s\sigma \in L_{\mathsf{ext}} \backslash \overline{K_{\mathsf{ext}}} \Rightarrow (\exists i \in \{1,2\})$. $[\![s]\!]_i^?\sigma \cap \overline{K_{\mathsf{ext}}} = \emptyset$. That means controller i disables σ after the sequence $s \in L_{\mathsf{ext}}$ with the communication protocol $\phi_{1',2}$. Hence there exists no $s' \in L_{\mathsf{ext}}$ such that $s' \in K_{\mathsf{ext}}$ and $[\![s]\!]_i^?\sigma = s'$.

To synthesize communication protocols in M_L, the communication messages are received through a sequence s with a delay between 0 and d. Since the received messages are already incorporated in M_{ext}, there exists a corresponding sequence $s_1 \in L$ (with $\phi_{1,2}$ and a bounded delay $[0..d]$) of $s \in L_{\mathsf{ext}}$ (with $\phi_{1',2}$). Since there exists no $s' \in L_{\mathsf{ext}}$, there also exists no $s'_1 \in L$ such that $s'_1 \in K$ and s'_1 is observationally equivalent to $s_1\sigma$ for either controllers with $\phi_{1,2}$ and a delay between 0 and d. □

Finally as explained in the next section we construct a product automaton, denoted by $\mathcal{V}^\tau$, similar to $\mathcal{U}_1$ and $\mathcal{U}_2$ of the previous chapter. $\mathcal{V}^\tau$ finds the violations of co-observability in the extended plant with synchronous communication in TDES and to specify the communication protocol.

6.2.1 Building $\mathcal{V}^\tau$

The automaton $\mathcal{V}^\tau$ is generated by composing M_{ext} with $n = 2$ versions of $(M_{K,\mathsf{ext}})_i$, one for each controller $i \in I = \{1,2\}$:

$$\mathcal{V}^\tau = (Z, \Sigma^{\mathcal{V}}, T_{\mathcal{V}}, z_0, F_{\mathcal{V}}) = M_{\mathsf{ext}} \times_{\mathcal{S}} \Pi_{i=1}^{n} (M_{K,\mathsf{ext}})_i.$$

The alphabet of $\mathcal{V}^{\tau}$ is similar to the set of vector labels defined in $\mathcal{U}_m (m \in \{1,2\})$ in the previous chapter.

A transition $(z, l, z') \in T_{\mathcal{V}}$, where $z = (q, q_1, \ldots, q_n)$ and $z' = (q', q'_1, \ldots, q'_n)$ with label $l = \langle l(0), l(1), \ldots, l(n) \rangle$ iff $(q, l(0), q') \in T_{\mathsf{ext}}$, and also for all $i \in I, (q_i, l(i), q'_i) \in T_{K,\mathsf{ext}}$.

The set of marked transitions is defined as follows:

$$F_{\mathcal{V}} = \{(z, l, z') \in T_{\mathcal{V}} \mid$$
$$((z(0), l(0), z'(0)) \in T_{\mathsf{ext}} \backslash T_{K,\mathsf{ext}} \wedge \forall i \in I_c(l(0)) : (z(i), l(i), z'(i)) \in T_{K,\mathsf{ext}})\}$$

This means a marked transition in $\mathcal{V}^{\tau}$ defines a situation where an event $\sigma \in L_{\mathsf{ext}} \backslash K_{\mathsf{ext}}$ must be disabled, but all controllers believe that σ should be enabled. Thus, a marked transition encodes a violation of co-observability.

We assume that each controller has prior knowledge about the communication: (i) only the "message delivery events" are being communicated, and (ii) Controller 2 receives a message $?\sigma$ from a dummy controller $1'$ if $\sigma \in \Sigma_{o,1} \backslash \Sigma_{o,2}$. A communication may occur in $\mathcal{V}^{\tau}$, when a "message delivery event" $?\sigma \in \Sigma^{?}$ occurs in the plant, i.e., $\ell(0) = ?\sigma$ and for all $i \in I$, $\ell(i) = \varepsilon$; and in the consecutive transition $\exists j \in I$ such that $\ell(j) = ?\sigma$, and $\ell(0) = \varepsilon$, $\forall i \in I \backslash \{j\}$, $\ell(i) = \varepsilon$. Then there is a communication sent by a dummy controller $1'$ (where $?\sigma \in \Sigma^{?}$) to controller 2, defined by a transition where $\ell(0) = ?\sigma$, $\ell(2) = ?\sigma$ and $\ell(1) = \varepsilon$. In that case, if controller $1'$ makes a decision to communicate the occurrence of $?\sigma \in \Sigma^{?}$ to controller 2, the previous transitions that ensures the communication are pruned from $\mathcal{V}^{\tau}$.

Lemma 1. *$F_{\mathcal{V}} = \emptyset$ iff K_{ext} is communication observable w.r.t. L_{ext}, $\Sigma_{o,1}$, $\Sigma_{o,2} \cup \Sigma^{?}$, $\Sigma_{c,i}$, $(i \in I)$, $\phi_{1',2}$.*

Proof. ($\Rightarrow$) Suppose that $F_{\mathcal{V}} = \emptyset$, but K_{ext} is not communication observable w.r.t.

L_{ext}, $\Sigma_{o,1}$, $\Sigma_{o,2} \cup \Sigma^?$, $\Sigma_{c,i}$ $(i \in I)$, $\phi_{1',2}$. By definition, there exists $s \in \overline{K_{\mathsf{ext}}}$ and $\sigma \in \Sigma_c$ such that $s\sigma \in L_{\mathsf{ext}} \setminus \overline{K_{\mathsf{ext}}}$ and there exists $s' \in \overline{K_{\mathsf{ext}}}$ such that $(\forall i \in \{1,2\}) s'\sigma \in [\![s]\!]_i^? \sigma \cap \overline{K_{\mathsf{ext}}}$.

In M_{ext}, we have $q_0 \xrightarrow{s} q \xrightarrow{\sigma} q' \in T_{\mathsf{ext}}$, where $(q, \sigma, q') \in T_{\mathsf{ext}} \setminus T_{K,\mathsf{ext}}$ and in each $M_{K,\mathsf{ext}}$, we have $q_0 \xrightarrow{s'} q_i \xrightarrow{\sigma} q_i' \in T_{K,\mathsf{ext}}$, for all $i \in \{1,2\}$.

From the construction of $\mathcal{V}^\tau$, we know that $(\forall i \in \{1,2\})\ \pi_i(s) = \pi_i(s')$. Thus there exists a trace in $\mathcal{V}^\tau$ such that $z_0 \xrightarrow{w} z$, where $w(0) = s$ and for all $i \in \{1,2\}$, $\exists s' \in \overline{K_{\mathsf{ext}}}\ .\ w(i) = s'$. In addition, we have $(z, \ell, z') \in T_{\mathcal{V}}$ where $z(0) = q$, $\ell(0) = \sigma$, $z'(0) = q'$ and $z(i) = q_i$, $\ell(i) = \sigma$, $z'(i) = q_i'$. By the definition of $F_{\mathcal{V}}$, we have $(z, \ell, z') \in F_{\mathcal{V}}$, thereby reaching a contradiction.

$(\Leftarrow)$ Suppose that K_{ext} is communication observable w.r.t. L_{ext}, $\Sigma_{o,1}$, $\Sigma_{o,2} \cup \Sigma^?$, $\Sigma_{c,i}$ $(i \in I)$, $\phi_{1',2}$, but $F_{\mathcal{V}} \neq \emptyset$.

By the definition, $(z(0), \ell(0), z'(0)) \in T_{\mathsf{ext}} \setminus T_{K,\mathsf{ext}}$ and $\forall i \in I_c(\ell(0))$, $(z(i), \ell(i), z'(i)) \in T_{K,\mathsf{ext}}$ and $\ell(i) = \ell(0)$. Let $w = \ell_1 \ell_2 \ldots \ell_{|w|} \in \Sigma_{\mathcal{V}}^*$ such that $z_0 \xrightarrow{w} z \in T_{\mathcal{V}}$. Let $s = w(0)$ and there exists $s' \in \overline{K_{\mathsf{ext}}}$ for all $i \in I_c(\ell(0))$ such that $s' = w(i)$. Since $z_0 \xrightarrow{w} z \xrightarrow{\ell} z' \in T_{\mathcal{V}}$, we have $s\ell(0) \in L_{\mathsf{ext}} \setminus \overline{K_{\mathsf{ext}}}$ and $s'\ell(i) \in \overline{K_{\mathsf{ext}}}$, $\forall i \in I_c(\ell(0))$. The only labels on which s and s' synchronize are those of the form $\ell = \langle \ell(0) = \gamma, \ldots, \ell(i) = \gamma, \ldots \rangle$ in which $\gamma \in \Sigma_o$ and $\gamma \in \Sigma_{o,i}$, therefore $\pi_i(s) = \pi_i(s')$. Thus we have $s'\ell(i) \in [\![s]\!]_i^? \ell(i) \cap \overline{K_{\mathsf{ext}}}$, arriving at a contradiction that K_{ext} is communication observable w.r.t. L_{ext}, $\Sigma_{o,1}$, $\Sigma_{o,2} \cup \Sigma^?$, $\Sigma_{c,i}$ $(i \in I)$, $\phi_{1',2}$. □

A portion of $\mathcal{V}^\tau$ is shown in Figure 6.9, where an example of marked transitions for the ongoing example is illustrated. The transition $((13, 13, 24), \langle \sigma, \sigma, \sigma \rangle, (14, 14, 25))$ is marked because σ must be disabled according to M_{ext} (colored here in red), whereas all the controllers in $I_c(\sigma)$ believe that σ should be enabled (colored in green). So that K_{ext} is not communication observable w.r.t. L_{ext}, $\Sigma_{o,1}$, $\Sigma_{o,2} \cup \Sigma^?$, $\Sigma_{c,i}$, $(i \in I)$ and $\phi_{1',2}$.

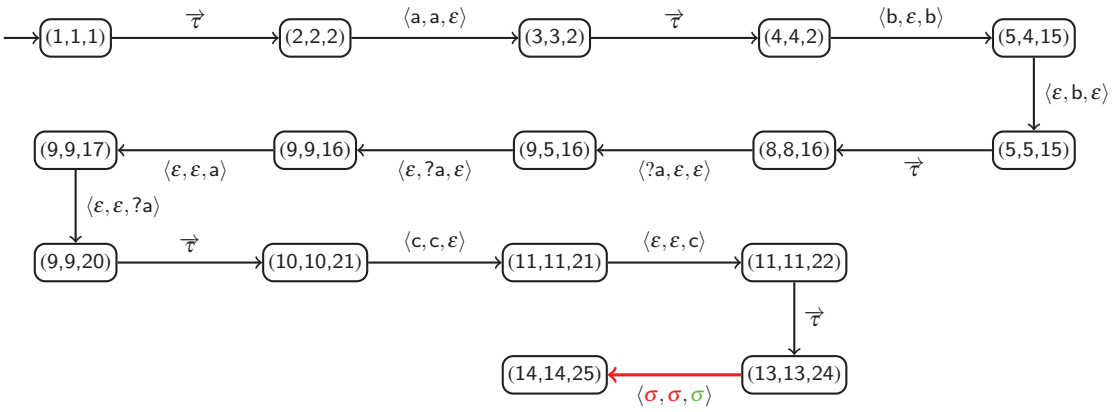


Figure 6.9: A portion of $\mathcal{V}^{\tau}$: a marked transition is highlighted in red where no controller $i \in I_c(\sigma)$ can take the correct control decision.

A communication occurs in $\mathcal{V}^{\tau}$ is shown in Figure 6.10 (highlighted in blue color). Here Controller $1'$ communicates the occurrence of ?a to Controller 2 $((5,5,5), \langle ?\sigma, \varepsilon, ?\sigma\rangle, (9,5,9))$. In that case, the transitions $((5,5,5), \langle ?\sigma, \varepsilon, \varepsilon\rangle, (9,5,5))$ and $((9,5,5), \langle \varepsilon, \varepsilon, ?\sigma\rangle, (9,5,9))$ are pruned from the $\mathcal{V}^{\tau}$. The reception of ?a forces Controller 2 to follow the plant behavior. Hence Controller 2 believes that σ should be disabled (colored in red), and it takes correct control decision regarding σ through the transition $((13,13,13), \langle \sigma, \sigma, \sigma\rangle, (14,14,14))$. Then K_{ext} is communication observable w.r.t. L_{ext}, $\Sigma_{o,2} \cup \Sigma^{?}$, $\Sigma_{c,i}$ $i = \{1,2\}$ and $\phi_{1',2}$.

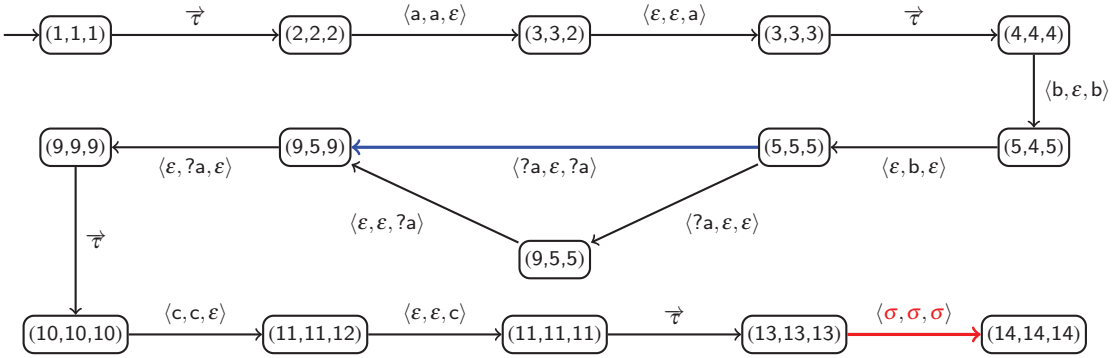


Figure 6.10: A communication occurs from $(5,5,5)$ to $(9,5,9)$, shown in blue. Controller 2 then takes correct control decision through the transition $((13,13,13),\langle\sigma,\sigma,\sigma\rangle,(14,14,14))$.

When tick events are observable to the controllers, a TDES is similar to the

corresponding untimed DES from the behavioural perspective. Hence, we convert the timed decentralized supervisory control problem with bounded-delay communication to an equivalent problem with synchronous communication, and synthesize the controllers using the approach modeled for synthesizing synchronous communication protocols in untimed DES.

Chapter 7

Conclusions and Future Work

This chapter summarizes research contributions. It concludes with some future research directions.

7.1 Concluding Remarks

We perform quantitative analysis for the decentralized discrete-event control and communication problem by finding locally-optimal communication policies. An optimal strategy is one that minimizes the cost of the communication protocol for each controller. Communication policies in decentralized DES have been initially examined in the context of Nash equilibrium. A recent algorithm for efficiently calculating a Nash equilibrium point for multi-agent systems in a game-theoretic setting is adapted for the problem of incorporating communication into decentralized discrete-event systems. We present two algorithms for exploring Nash equilibrium in the decentralized DES control problem: one for two controllers, and the other for more than two controllers.

We also extend our analysis to Pareto optimality, which is typically used when there are multiple objectives to optimize. The trade-off between the cost and the

accuracy of a decentralized discrete-event control solution with synchronously communicating controllers was explored as a multi-objective optimization problem. We examine a class of problems where communication is necessary to achieve the exact control solution. However, in some circumstances, it may be advantageous to reduce communication from a cost perspective, but incur a penalty for synthesizing an approximate control solution. A widely-used evolutionary algorithm, namely non-dominated sorting genetic algorithm (NSGA-II), is adapted to examine the set of Pareto-optimal solutions that arise for this family of decentralized discrete-event systems.

Synchronous communication in untimed decentralized supervisory control problems has been explored in the previous model, where controllers communicate with each other without delay via a communication channel. In reality, delays in communication play a significant role in controllers' decisions, when some events may occur in the plant between sending a message by one controller and receiving that message by another controller. For this reason, we extend our work to the case of communication channels with bounded delay. Instead of synthesizing communication protocols w.r.t. a fixed or a bounded (but not necessarily fixed) delay communication, first we verify the robustness of synchronous communication protocols that are already synthesized for supervisory control problems. We do not limit our study to just optimal communication protocols and assume that the given protocol does not necessarily communicate all of their observations to all the other controllers.

Finally, we consider direct solution of decentralized control and communication with bounded delay using timed discrete event models. The communication delay was illustrated by a special tick event, denoted by τ. We assumed that a global digital clock is available to the controllers. To solve the problem, we show that it can be converted to an equivalent problem with synchronous communication for which

solution can be synthesized using the same procedure for solving decentralized control and communication problems in untimed DES.

7.2 Future Research Directions

The following directions are suggested for research on optimal solutions to the decentralized DES control and communication problem.

7.2.1 Synthesizing Optimal Communication Protocol with Fixed and Bounded Delay

The synthesis of optimal communication protocols for either fixed or bounded delay can be done with minor adaptations to the robustness techniques outlined in Chapter 5. However, when a synchronous communication protocol is not robust with a fixed delay or a known upper-bounded delay, it will be a valuable research to optimize communication protocols under the condition of fixed and bounded delay in communication. For the quantitative analysis, one may impose a cost as a penalty for certain delay. In addition, it would be interesting to apply the concepts of game theory to optimize the communication protocol with bounded delay communication.

7.2.2 Multi-Objective Optimization in Control with Communication under Bounded Delay

We may update the cost functions defined in case of synchronous communication to cope with the effect of delay in making the control decisions. In a similar fashion, the multi-objective optimization problem can be defined in decentralized DES by taking the communication delay into account. Then we may apply the same concept of Pareto-optimal solution and adapt an evolutionary algorithm to solve the problem.

It would also be interesting to apply our strategy to a practical application, such as smart grid or smart buildings.

7.2.3 Synthesizing TDES Control Solutions

It might be an interesting research topic for synthesizing communication protocols and the communicating controllers for supervisory control problems in TDES. The quantitative analysis can be done using a similar approach to that of untimed DES. Since τ is only used to represent a clock tick, one may incur a cost of zero for this event. In addition, it would be feasible to adjust the algorithms for synchronous communication protocols in untimed DES, so that the protocols will also work in a TDES. It would be also interesting to extend our work to cyclic systems, since our research was restricted to acyclic systems for TDES models.

7.2.4 Synthesizing Asynchronous Communication for Distributed System

An interesting research topic is the synthesis of truly asynchronous communication protocols for distributed architectures. It would be more realistic to consider a local clock for each controller. We may measure the exact time when an event occurs in the plant, similarly the local clock measures the time when a controller observes an event. We assume that when an event is transmitted by a controller, it is sent in the same clock cycle as the controller observes it. When the message is received by another controller, it also counts the exact time of receiving the message. Then we need to find out how many events occur in the plant between sending an event by one controller and receiving that event by another controller. Hence, a communication protocol has to be synthesized with an unknown delay to solve the problem. It might be a good idea to adapt the algorithms of finding optimal synchronous communication

protocols to synthesize optimal asynchronous communication protocols.

7.2.5 Real-World Applications

We may use NSGA-II algorithm to small real-world application. It would be interesting to implement the algorithms to find Nash equilibrium and Pareto optimality for some practical applications. In addition, it would be useful to apply the our results on delay-robustness of a synchronous communication protocol and solutions to the control problem using TDES models to real-world applications.

Bibliography

[1] S. Alexandra. A note on global Pareto optimality in multicriteria optimization problems. *Nonlinear Analysis*, 69:1321–1324, 2008.

[2] A. Arnold. *Finite Transition Systems*. Prentice-Hall, 1994.

[3] T. Back. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, 1996.

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

[5] R.K. Boel and J.H. van Schuppen. Decentralized failure diagnosis for discrete-event systems with costly communication between diagnosers. In *Proceedings of International Workshop on Discrete-Event Systems (WODES)*, pages 175–181, Saragoza, Spain, 2002.

[6] B.A. Brandin and W.M. Wonham. Supervisory control of timed discrete-event systems. *IEEE Transactions on Automatic Control*, 39(2):329–342, 1994.

[7] K. Chatterjee, L. Doyen, and T.A. Henzinger. Quantitative languages. *ACM Transactions on Computational Logic*, 11(4), 2010.

[8] I. Chattopadhyay and A. Ray. A language measure for partially observed discrete event systems. *International Journal of Control*, 79(9):1074–1086, September 2006.

[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] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA - II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[11] K. Deb and N. Srinivas. Multi-objective function optimization using non-dominated sorting genetic algorithm. *Evolutionary Computation*, 2(3):221–248, 1995.

[12] E. Dumitrescu, A. Girault, H. Marchand, and E. Rutten. Multicriteria optimal reconfiguration of fault-tolerant real-time tasks. *Proceedings of WODES*, pages 366–373, 2010.

[13] C.C. Elgot and J.E. Mezei. On relations defined by generalized finite automata. *IBM Journal of Research and Development*, 9(1):47–68, 1965.

[14] C.M. Fonseca and P.J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms-part II: Application example. *IEEE Transactions on Systems, Man, and Cybernetics: Part A: Systems and Humans*, pages 38–47, 1998.

[15] A. Gambier and E. Badreddin. Multi-objective optimal control: An overview. In *Proceedings of 16^{th} IEEE International Conference on Control Applications*, pages 170–175, Singapore, 2007.

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

[17] J. Knowles and D. Corne. The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 98–105, New Jersey, 1999.

[18] R. Kumar and V.K. Garg. Optimal supervisory control of discrete event dynamical systems. *SIAM Journal of Applied Mathematics*, 33:419–439, 1995.

[19] C. Lemke and J. Howson. Equilibrium points of bi matrix games. *SIAM Journal of Applied Mathematics*, 12(2):413–423, 1964.

[20] F. Lin and W. Wonham. On observability of discrete-event systems. *Information Sciences*, 44(3):173–198, 1988.

[21] S. Luke. *Essentials of Metaheuristics*. Lulu, 2009, Available at http://cs.gmu.edu/ sean/book/metaheuristics/.

[22] P. Madhusudan, P.S. Thiagarajan, and Shaofa Yang. The MSO theory of connectedly communicating processes. *Lecture Notes in Computer Science, Springer*, 3821:201–212, 2005.

[23] A. Mannani and P. Gohari. Decentralized supervisory control of discrete-event systems over communication networks. *IEEE Transactions on Automatic Control*, 53(2):547–559, 2008.

[24] H. Marchand, O. Boivineau, and S. Lafortune. On optimal control of a class of partially-observed discrete event systems. *Automatica*, 38(11):1935–1943, October 2002.

[25] J. Nash. Equilibrium points in n-person games. In *Proceedings of National Academy of Sciences*, volume 36, pages 48–49, 1950.

[26] M. Nomura and S. Takai. Decentralized supervisory control of timed discrete event systems. *IEICE Transactions on Fundamentals*, E94-A(12), 2802–2809 2011.

[27] A. Overkamp and J.H. van Schuppen. Maximal solutions in decentralized supervisory control. *SIAM Journal of Control and Optimization*, 39:492–511, 2000.

[28] S.J. Park and H.L. Cho. Time-coobservability in the decentralized supervisory control of timed discrete event systems. *Journal of Institute of Control, Robotics and Systems*, 15(4):396–399, April 2009.

[29] R. Porter, E. Nudelman, and Y. Shoham. Simple search methods for finding a Nash equilibrium. *Games and Economic Behavior*, 63(2):642–662, 2008.

[30] W. Qiu and R. Kumar. Distributed diagnosis under bounded-delay communication of immediately forwarded local observations. *IEEE Transactions on Systems, Man, and Cybernetics: Part A: Systems and Humans*, 38(3):628–643, 2008.

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

[32] P.J. Ramadge and W. Wonham. The control of discrete-event systems. In *Proceedings of IEEE*, volume 77, pages 81–98, 1989.

[33] A. Ray, J. Fu, and C. Lagoa. Optimal supervisory control of finite state automata. *International Journal of Control*, 77(12):1083–1100, July 2004.

[34] A. Ray, A. Surana, and S. Phoha. A language measure for supervisory control. *Applied Mathematics Letters*, 16:985–991, February 2003.

[35] S.L. Ricker. Knowledge and communication in decentralized discrete-event control. *PhD Dissertation, Queen's University*, December 1999.

[36] S.L. Ricker. Asymptotic minimal communication for decentralized discrete-event control. In *Proceedings of International Workshop on Discrete-Event Systems (WODES)*, pages 486–491, Goteburg, Sweden, 2008.

[37] S.L. Ricker and B. Caillaud. Mind the gap: Expanding communication options in decentralized discrete-event control. *Automatica*, 47:2364–2372, 2011.

[38] S.L. Ricker and K. Rudie. Incorporating communication and knowledge into decentralized discrete-event systems. In *Proceedings of IEEE Conference on Decision and Control*, pages 1326–1332, Phoenix, AZ, 1999.

[39] S.L. Ricker and J. van Schuppen. Asynchronous communication in timed discrete-event systems. *Proceedings of ACC*, pages 305–306, December 2001.

[40] K. Rudie, S. Lafortune, and F. Lin. Minimal communication in a distributed discrete-event system. *IEEE Transactions on Automatic Control*, 48(6):1965–1970, June 2003.

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

[42] R. Rudolph. Evolutionary search under partially ordered sets. *Technical report*, CI-67/99, Department of Computer Science, University of Dortmund, 1999.

[43] W.H. Sadid, S.L. Ricker, and S. Hashtrudi-Zad. Nash equilibrium for communication protocols in decentralized discrete-event systems. pages 3384–3389, Baltimore, MD, June 2010.

[44] W.H. Sadid, S.L. Ricker, and S. Hashtrudi-Zad. Multiobjective optimization in control with communication for decentralized discrete-event systems. pages 372–377, Orlando, FL, December 2011.

[45] W.H. Sadid, S.L. Ricker, and S. Hashtrudi-Zad. Robustness of synchronous communication protocols with bounded delay for decentralized discrete-event control. pages 181–186, Guadalajara, Mexico, October 2012.

[46] W.H. Sadid, S.L. Ricker, and S. Hashtrudi-Zad. Robustness of synchronous communication protocols with delay for decentralized discrete-event control. *Submitted for journal publication*, 2013.

[47] T. Sandholm, A. Gilpin, and V. Conitzer. Mixed-integer programming methods for finding Nash equilibria. In *Proceedings of AAAI Conference*, pages 495–501, Pittsburgh, PA, 2005.

[48] Y. Sawaragi, H. Nakayama, and T. Tanino. *Theory of Multiobjective Optimization.* Academic Press, Orlando, 1985.

[49] R. Sengupta and S. Lafortune. An optimal control theory for discrete event systems. *SIAM Journal of Control Optimization*, 36(2):488–541, March 1998.

[50] R.E. Steuer. *Multiple Criteria Optimization: Theory, Computation and Application.* John Wiley, New York, 1986.

[51] S. Takai and R. Kumar. Distributed prognosis of discrete event systems under bounded delay communications. *Proceedings of CDC*, pages 1235–1240, 2009.

[52] David P.L. Thorsley. Applications of stochastic techniques to partially observed discrete event systems. *PhD Dissertation, University of Michigan*, 2006.

[53] S. Tripakis. Decentralized control of discrete event systems with bounded or unbounded delay communication. *IEEE Transactions on Automatic Control*, 49(9):1489–1501, September 2004.

[54] S. Tripakis. Undecidable problems of decentralized observation and control on regular languages. *Information Processing Letters*, 90:21–28, 2004.

[55] J.H. van Schuppen. Decentralized control with communication between controllers. *Unsolved Problems in Mathematical Systems and Control Theory*, pages 144–150, 2004.

[56] W. Wang, S. Lafortune, and F. Lin. Minimization of communication of event occurrences in acyclic discrete event systems. *IEEE Transactions on Automatic Control*, 53(9):2197–2202, 2008.

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

[58] P. Woźniak. Multi-objective control systems design with criteria reduction. *Lecture Notes in Computer Science, Springer*, 6457:583–587, 2010.

[59] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

[60] E. Zitzler, M. Laumanns, and S. Bleuler. *A Tutorial on Evolutionary Multiobjective Optimization.* Lecture Notes in Economics and Mathematical Systems. Springer, 2004.

[61] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transcations on Evolutionary Computation*, 3(4):257–271, 1999.