COPPE
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia
UFRJ

CODIAGNOSABILITY OF NETWORKED DISCRETE EVENT SYSTEMS SUBJECT TO COMMUNICATION DELAYS AND INTERMITTENT LOSS OF OBSERVATION

Carlos Eduardo Viana Nunes

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Elétrica.

Orientadores: João Carlos dos Santos Basilio
Marcos Vicente de Brito Moreira

Rio de Janeiro
Outubro de 2016

CODIAGNOSABILITY OF NETWORKED DISCRETE EVENT SYSTEMS SUBJECT TO COMMUNICATION DELAYS AND INTERMITTENT LOSS OF OBSERVATION

Carlos Eduardo Viana Nunes

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. João Carlos dos Santos Basilio, Ph.D.

Prof. Marcos Vicente de Brito Moreira, D.Sc.

Profa. Patrícia Nascimento Pena, D.Sc.

Prof. José Eduardo Ribeiro Cury, Docteur d'Etat

Prof. Antônio Eduardo Carrilho da Cunha, D.Eng.

RIO DE JANEIRO, RJ – BRASIL
OUTUBRO DE 2016

Nunes, Carlos Eduardo Viana

Codiagnosability of Networked Discrete Event Systems subject to communication delays and intermittent loss of observation/Carlos Eduardo Viana Nunes. – Rio de Janeiro: UFRJ/COPPE, 2016.

XII, 102 p.: il.; 29, 7cm.

Orientadores: João Carlos dos Santos Basilio

Marcos Vicente de Brito Moreira

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2016.

Referências Bibliográficas: p. 94 – 101.

1. Discrete Events Systems. 2. Communication network. 3. Failure diagnosis. I. Basilio, João Carlos dos Santos *et al.*. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

"Satisfaction lies in the effort, not in the attainment, full effort is full victory. "

Mahatma Gandhi

Acknowledgments

First, I thank God, creator of all things, by the infinite love deposited in all humanity and by the opportunity to be on Earth in order to learn and evolve.

I thank Jesus, by his teachings of love and humility that guide all humanity to the path of good.

I thank my parents Avany and Hélio, by their unconditional support and because they provide me all love and education necessary to live with dignity.

I thank my brothers Hélio Jr, André, and Daniela, by their support and wonderful happy moments that they always provide me.

I thank my wife Carina, a wonderful woman that help me whenever I need. Thank you dear by trust and love that you always have for me.

I thank deeply my advisors João Carlos Basilio and Marcos Vicente Moreira for trusting me in execution fo this thesis and by patience and dedication to teach me.

A special thank to my friend Marcos Vinicius, that provided me his time to discuss many aspects of this thesis and by his help whenever I needed.

I cannot forget the friends that, in some way, help me during all doctoral course: Cristiano Carvalho, Dayro Barahona, Gustavo Viana, Ingrid Antunes, Felipe Cabral, Lilian Kawakami, Leonardo Bermeo and Félix Gamarra. Thanks everyone.

To CNPq by financial support.

Carlos Eduardo Viana Nunes

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

CODIAGNOSTICABILIDADE DE SISTEMAS A EVENTOS DISCRETOS EM REDE SUJEITOS A ATRASOS DE COMUNICAÇÃO E PERDA INTERMITENTE DE OBSERVAÇÃO

Carlos Eduardo Viana Nunes

Outubro/2016

Orientadores: João Carlos dos Santos Basilio
Marcos Vicente de Brito Moreira

Programa: Engenharia Elétrica

No diagnóstico de falhas de Sistemas a Eventos Discretos distribuídos, é usualmente considerado que na comunicação entre os dispositivos não há perdas nem atrasos na comunicação da ocorrência de eventos para os diagnosticadores. No entanto, os canais de comunicação reais são sujeitos a atrasos e a perdas intermitentes de pacotes que podem levar o diagnosticador a observar eventos fora da ordem de ocorrência, proporcionando um incorreto diagnostico da falha. Neste trabalho, investigamos a codiagnosticabilidade de um sistema em rede com atrasos de comunicação e perdas intermitentes de observação. Introduzimos a definição de codiagnosticabilidade em rede contra atrasos de comunicação e perdas intermitentes de observação, apresentamos uma condição necessária e suficiente para a codiagnosticabilidade em rede e propomos um algoritmo para a verificação desta propriedade.

Palavras-chave: Sistemas a eventos discretos, diagnóstico de falhas, comunicação em rede.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

CODIAGNOSABILITY OF NETWORKED DISCRETE EVENT SYSTEMS SUBJECT TO COMMUNICATION DELAYS AND INTERMITTENT LOSS OF OBSERVATION

Carlos Eduardo Viana Nunes

October/2016

Advisors: João Carlos dos Santos Basilio
Marcos Vicente de Brito Moreira

Department: Electrical Engineering

In failure diagnosis of networked Discrete Event Systems, it is usually assumed that communication among devices is lossless and without delay in the communication of event occurrences to the diagnosers. However, real communication channels are subject to transportation delays and intermittent loss of packets which can make the diagnosers observe events out of their order of occurrence, leading to an incorrect fault diagnosis. In this work, we investigate the codiagnosability of networked systems with communication delays and intermittent losses of observation. We introduce the definition of network codiagnosability against communication delays and intermittent losses of observation, present a necessary and sufficient condition for network codiagnosability and propose an algorithm for the verification of this property.

Keywords: Discrete event systems, failure diagnosis, network communication.

Contents

List of Figures **x**

List of Tables **xii**

1 Introduction **1**

2 Discrete Event Systems: theory and fundamentals **6**

2.1 Modeling of discrete event systems 7

2.2 Languages 8

2.2.1 Language of Discrete Event Systems 8

2.2.2 Operations on languages 10

2.3 Automata 12

2.3.1 Deterministic automata 12

2.3.2 Nondeterministic automata 13

2.3.3 Observer automata 16

2.3.4 Operations on automata 18

2.4 Failure diagnosis 23

2.4.1 Algorithms of failure diagnosis 25

2.4.2 Failure diagnosis of DES 26

2.5 Diagnoser 27

2.5.1 Centralized diagnosis 28

2.5.2 Decentralized diagnosis 29

2.5.3 Codiagnosability verification 32

2.6 Final remarks 34

3 Communication Networks Subject to Delays and Losses 37

3.1 Communication networks 38

3.1.1 OSI model 39

3.1.2 Circuit switching and packet switching 41

3.1.3 Components of a network 42

3.2 Delays and losses 43

3.3 Specification of a communication network 47

3.4 Final remarks 54

4 Codiagnosability of Networked Discrete Event Systems 55

4.1 Problem formulation 56

4.2 Model of the plant subject to communication delays 58

4.3 Modeling of intermittent loss of events 73

4.4 Model of the plant subject to communication delays and intermittent loss of observations 75

4.5 Definition of network codiagnosability of discrete-event systems . . . 78

4.6 Verification of network codiagnosability of discrete-event systems . . . 80

4.7 Complexity analysis of Algorithm 4.2 84

4.8 Concluding Remarks 90

5 Conclusion and Future Works 92

Bibliographic References 94

List of Figures

2.1 Automaton G. 13
2.2 Nondeterministic automaton. 14
2.3 Nondeterministic automaton with ϵ-transition. 15
2.4 Automaton G. 17
2.5 Observer automaton of G. 17
2.6 Automaton H 20
2.7 Coaccessible part of automaton H 20
2.8 Automaton $Tim(H)$ 20
2.9 Comp[$Trim(H)$] 21
2.10 A general diagnosis framework [1]. 23
2.11 Classification of diagnosis algorithms [1]. 26
2.12 Label automaton A_l for building diagnoser 28
2.13 (a) Automaton G (b) Parallel composition between G and A_l, (c) $G_d = Obs(G||A_l)$ 30
2.14 Codiagnosis structure. 30
2.15 Automaton G of example 2.10 35
2.16 Automaton A_l^N 35
2.17 Automaton G_N 35
2.18 Automaton G_F 35
2.19 Automaton G_{R_1} 35
2.20 Automaton G_{R_2} 35
2.21 Verifier automaton G_V 36

3.1 OSI model. 39
3.2 Resume and application of OSI model. 41

3.3 Types of delays of a network communication. 44
3.4 Convoy analogy. 46
3.5 Peer-to-peer topology. 48
3.6 Bus topology. 49
3.7 Ring topology. 50
3.8 Star topology. 51
3.9 Tree topology. 51
3.10 Simplex mode 52
3.11 Half−duplex mode 52
3.12 Full−duplex mode 52

4.1 Network decentralized diagnosis architecture. 57
4.2 Automaton G 63
4.3 Network codiagnosis scheme of example 4.1. 64
4.4 Construction of automaton D_1 step-by-step. 65
4.5 Automaton D_2. 66
4.6 (a) Automaton G_1; (b) Automaton G_2. 72
4.7 Automaton H of Example 4.3. 74
4.8 Automaton H_{dil}. 75
4.9 (a) automaton G'_1; (b)automaton G'_2. 77
4.10 (a) automaton $\bar{G}'_1$; (b)automaton $\bar{G}'_2$. 79
4.11 (a) Automaton $G'_{1,\rho}$, (b) Automaton $G'_{1,F}$. 84
4.12 (a) Automaton $G'_{2,\rho}$, (b) Automaton $G'_{2,F}$. 85
4.13 (a) Path of V_1 with cyclic path cl_1 embedded, (b) path of V_2 with cyclic path cl_2 embedded. 86
4.14 Path of G_V with an embedded cyclic path cl that violates the network codiagnosability of L. 86
4.15 (a) Automaton $\bar{G}'_{1,\rho}$, (b) Automaton $\bar{G}'_{1,F}$. 87
4.16 (a) Automaton $\bar{G}'_{2,\rho}$, (b) Automaton $\bar{G}'_{2,F}$. 88
4.17 (a) Path of $\bar{V}_1$ with cyclic path cl_1 embedded, (b) path of $\bar{V}_2$ with cyclic path cl_2 embedded. 89
4.18 Part of verifier $\bar{G}_V$. 89

List of Tables

2.1 States and events of the DES composed of machines M_1, M_2 and a robot. 9

Chapter 1

Introduction

Industrial systems are becoming more complex with the advance of technology, thus, failure diagnosis in components of these systems becomes a complex task to be solved by using only the experience and knowledge of the operator of the system. In this context, the improvement of automatic failure diagnosis systems becomes an important area to be developed.

The importance of research in the area of failure diagnosis is reflected in the number of works published in international conferences. In [2], it is presented the statistics of published works in WODES (Workshop on Discrete Event Systems) and DCDS (Workshop on Dependable Control of Discrete Systems) between 1998–2012 and 2007–2011, respectively. In WODES, around 12% of published works are related with failure diagnosis and in DCDS this statistic is 22%. Furthermore, there are other contributions to fault diagnosis of DES that have been published in control journals, such as Automatica, IEEE Transactions on Automatic Control, Control Engineering Practice, and others.

Since the first publications that addressed the problem of failure diagnosis in Discrete Event Systems (DES) [3, 4, 5], in which the fundamental concepts of diagnosability of DES were presented, many issues related to this problem were presented in the literature such as: (i) failure prediction, (ii) selection of sensors and dynamic activation and, (iii) robust diagnosis. In (i), the problem is to predict the occurrence of failure events based on the observation of events and then prevent these failure event occurrence [6, 7, 8, 9, 10, 11, 12, 13]. In (ii), the problem is to ensure the diagnosability of the language of the system with respect to a set of observable

events with smallest cardinality, thus, minimizing the cost of the diagnosis system associated with the use of sensors [14, 15, 16, 17, 18, 19, 20] and, finally, in (*iii*), the problem is to detect the occurrence of unobservable fault events using a set of sensors that themselves are subject to failures, such as, intermittent or permanent malfunction [21, 22, 23, 24, 25, 26, 27, 28, 29].

The problem of failure diagnosis was also considered in [30, 31, 32, 33, 34, 35, 36], with several applications in [37, 38, 39, 40, 41, 42], where it is assumed that communication between the sensors of the system and the diagnoser is perfectly reliable and the sensors work perfectly, *i.e.*, there is neither loss of observation nor delays in the communication of the events to the diagnosers. It is realistic to assume that if diagnoser and plant communicate via a dedicated communication link, also called point-to-point link [43, 44] since there is a wire for each sensor or actuator point that connects them to the central control computer. This kind of communication is complex and expensive and the whole system is difficult to maintain and diagnose due to the large number of connectors and cables [45]. Thus, due to the complexity of the plants, the diagnosers are often implemented in a distributed way and, consequently, with the development of network technology, there is a trend in industries to implement communication systems by using shared (wired or wireless) communication networks. In a shared network, communication problems such as loss of observation of events or communication delays are unavoidable [44].

In [46], [26] and [27], the problem of loss of observation is addressed considering the malfunctioning of sensors of the plant. In [46], it has been developed a probabilistic methodology for failure diagnosis in finite state machines based on a sequence of unreliable observations, and in [26, 27], it has been developed a deterministic methodology for failure diagnosis in Discrete Event Systems. In these works, the problem of communication delays between sensors and diagnosers was not considered.

In networked failure diagnosis systems, communication delay can make the diagnoser to receive signals out of the original order of occurrence, and the diagnoser can erroneously detect the occurrence of a failure event. One way to solve this problem is to insert clocks and time stamps in the communication protocol. Thus, the protocol will be able to reorganize the events by using the times inserted in the

pack of transmission of each event. However, in order to add a time information in the communication protocol, it is necessary to synchronize the clocks of the devices, which is not a simple task, since each device must have exactly the same time. Another problem is maintenance because the synchronization process must be executed periodically on the whole plant, which increases the cost of implementation [47]. In the literature there are some works about DES with network communication addressing the problem of communication delays, and most of them are related to supervisory control, [48, 49, 50, 51, 52, 53]. The problem of failure diagnosis of DES with communication delay is addressed in [47, 54, 55, 56].

The problem of decentralized supervisory control with communication delays is presented in [48]. Two types of delays are defined: unbounded delay and bounded delay by a constant $k \in \mathbb{N}$ (k-bounded delay). In the unbounded delay approach, the plant can execute any number of events between the transmission of the occurred event and its reception by the supervisor. On the other hand, in the k-bounded delay approach, the plant can execute at most k events between the transmission and reception of an event by the supervisor. In [48], it is assumed that the communication is lossless, that is, all messages are eventually delivered within a finite delay and there is no change of order in the observation by the supervisor.

The problem of centralized supervisory control with communication delays and partial observations is addressed in [49], where a nonblocking supervisor is designed to reach a specification even if there are communication delays between plant and supervisor. It is important to remark that there is no change of order among events in [49]. The problem of decentralized supervisory control with communication delays based on conjunctive and permissive structure is addressed in [50] assuming that uncontrollable events can occur unexpectedly before any control action takes place on the plant, in order to find conditions of existence of a nonblocking decentralized supervisor. The notion of delay co-observability for a given specification is also presented, and it is shown that co-observability is the key condition for the existence of a decentralized supervisor that can reach the specification.

A strategy of control for a system with network communication is proposed in [51], whose goal is to compute a supervisor that is tolerant to delays and events observation losses. In [51], it is assumed that supervisor and plant are networked,

and there are two channels connecting both: observation channel and control channel, and, in both channels, communication delays and loss of information can occur. The communication delays are assumed to be k-bounded, as in [48], and there is no change of order of event observation. As a consequence, an event that has occurred in the plant may not be seen by the supervisor until the occurrence of the next event in the plant, and the supervisor may not be capable of disabling this event. In [52], the work presented in [51] is continued, assume the same assumptions as in [51]. Based on these assumptions, the authors show how to solve the problem of centralized control of Discrete Event Systems with network communication. In [53] the problem addressed in [52] is extended to decentralized supervisory control of networked systems. Control policies are defined for all local supervisor in order to satisfy the design specifications.

In [47], the problem of decentralized failure diagnosis with communication delays is addressed considering protocols 1 and 2 of [30], and assuming that there are communication delays between local diagnosers and the coordinator, which is responsible to reorder the events and to infer the occurrence of failure events. It is also assumed in [47] that each communication channel that connects a diagnoser and the coordinator is FIFO (Fist-in-First-out), so that, only events that come from different diagnosers can be observed in a order different from that executed in the plant.

In [54], the problem of distributed failure diagnosis in DES subject to communication delays is presented. It is assumed that, the diagnosis system does not have a coordinator and, each local diagnoser sends its observation to other local diagnosers. As in [48], the observation delay of an event σ that occurs in the plant is bounded. In [54], it is assumed that the communication delay takes place between local diagnosers. Moreover, it is assumed that both communication channels that connect the local diagnosers have the same maximum delay, the communication is lossless and *first-in-first-out* (FIFO), *i.e*, there is no change of order on the event observations that are transmitted through the same channel.

The problem of decentralized failure diagnosis with communication delays is addressed in [55] considering protocol 3 of [30], *i.e.*, (i) there is no communication between the local diagnosers; (ii) each local diagnoser infers the occurrence of the

failure event based on its own observations, and; (*iii*) the failure event is diagnosed when at least one of the local diagnosers identifies its occurrence. It is assumed in [55] that there exist communication delays between measurement sites and local diagnosers, resulting in an out of order observation, by local diagnosers, of the events executed by the system. The problem of loss of observation was not addressed in [55].

In this work, we extend the problem considered in [55] to also take into account loss of observation. Since the communication channels are independent, the delays generated by each channel can be different. In this work, we assume that the delays are k-bounded as presented in [48]. Based on that, we present the definition of network codiagnosability against communication delays and loss of observation. Moreover, we propose an automaton model that describes all possible delays that the system is subject and, consequently, all possible orders of observation by the local diagnosers. We also propose an algorithm for the verification of network codiagnosability against communication delays and loss of observation, and we show that the approach presented in [54] can be considered as a particular case of the problem formulated in this work.

This thesis is organized as follows. In chapter 2, the fundamental concepts of DES modeled by automata including the notion of failure diagnosis are presented. We also review the verifier algorithm proposed in [57]. In chapter 3, we present the main reasons of delays and losses in communication networks. In chapter 4, we introduce the definition of network codiagnosability against communication delays and loss of observation, present a necessary and sufficient condition for network codiagnosability and propose an algorithm for its verification. The final remarks and suggestions of future research works are presented in chapter 5.

Chapter 2

Discrete Event Systems: theory and fundamentals

Discrete Event Systems (DES) are dynamic systems with discrete states space where the transitions between states are made by the occurrence, in general asynchronous, of discrete events. The fact that the states of the systems are discrete implies that it can assume symbolic values, for example, (on, off), (green, yellow, red), or numeric values that belong to sets $\mathbb{N}$ or $\mathbb{Z}$, or be formed by a subset of enumerable elements of $\mathbb{R}$. Events can be associated to specific actions (for example, someone presses a button, an aircraft takes off, etc), or be the result of several conditions that are satisfied (*e.g*, an object reaches a point of a production line, a liquid that reaches a determined height). Although it is possible to model any physical system as a DES according to some level of abstraction that we can consider, some systems are naturally discrete and their evolution are determined by the occurrence of events.

This chapter aims to present the basic definitions and notions about DES and intends to introduce the main concepts on DES for novice readers. The theory presented is based on [58]. This chapter is structured as follows: in Section 2.1, we present the main formalisms for DES. In Section 2.2, some definitions and concepts related to DES are addressed. In Section 2.4, we introduce the basic concepts about failure and failure diagnosis in DES. Finally, in Section 2.5 the fundamental concepts about centralized diagnosers, decentralized diagnosers and the verifier proposed in [57] are reviewed.

2.1 Modeling of discrete event systems

The model of a DES must be capable of reproducing, within some prespecified tolerance, the behavior of the system. In continuous variable dynamic systems (CVDS), the states are expressed as functions of time, while in DES, the system behavior is described in terms of traces of events. All traces that can be generated by a given DES describe the language of this system, which is defined over a set of events (alphabet) of the system. Thus, when we consider the evolution of states of a DES, the main concern is the trace associated with visited states and with the events that cause the corresponding state transitions, *i.e.* the model of a DES consists basically of two elements: state and transition.

The DES models take into account some characteristics in order to represent a determined system:

- The model agreement with reality regarding the purpose of the model.
- The complexity of the model.

In the literature several formalisms to model DES are presented, whose choice is related to the accuracy and complexity. The model formalisms more frequently used in the literature are:

- Automata;
- Petri nets;
- Hybrid automata.

The automaton modeling has the following advantages: several problems found in industries can be modeled with it, the computational implementation is simple and, algorithms and operations among automata are well known. On the other hand, simultaneous tasks are better modeled by Petri nets.

In order to illustrate the modeling of a system, consider the following example [59].

Example 2.1 *Consider a manufacturing cell composed of two machines (M_1 and M_2) and a robot which transports parts from M_1 to M_2. Machine M_1*

receives raw parts, and, processes them. After being processed, they are collected by the robot. In case the robot is busy, the machine M_1 holds the part until the robot becomes available. In case another part arrives while machine M_1 is processing some part, it rejects the part. When the robot takes a part from M_1, it transports it to M_2. When the part arrives at M_2, the robot will deliver it to M_2, only if M_2 is free. Otherwise, the robot holds the part until M_2 becomes free. After the robot delivers the part to M_2, it returns to M_1. Finally, machine M_2 receives the part from the robot and processes it.

Table 2.1 describes the states and events of machines M_1, M_2 and the robot. Notice that, events e_1 (part delivered to robot) and a_2 (part delivered to M_2) belong to two subsystems: machine M_1 and robot, and robot and machine M_2, respectively. It is important to notice that, in order to event e_1 to occur, machine M_1 must be in state H_1 and the robot in state I; in order to event a_2 to occur, the robot must be in state H and the machine M_2 must be in state I_2. For other states of the system, i.e., those that are in only one of the subsystems, the occurrence is determined only by the current state of the subsystem; for example, the occurrence of event t_1 (end of processing) depends only on machine M_1 to be in state P_1, independently of which states the robot and machine M_2 are.

□

2.2 Languages

2.2.1 Language of Discrete Event Systems

One of the formal ways to study the logical behavior of a DES is based on the theories of language and automata. The starting point is the fact that any DES has an associated event set Σ. The event set Σ is the "alphabet" and the traces are the "words" of a language. We will assume that Σ is finite. The length of a word is the number of events it contains. We denote the length of a trace s by $\|s\|$. The word that does not contain events is called the empty trace, and is denoted by ϵ. The length of the empty trace ϵ is zero.

Definition 2.1 (Language) *A language defined over an event set Σ is a set of traces (words) with finite length formed by events of Σ.*

Table 2.1: States and events of the DES composed of machines M_1, M_2 and a robot.

Element	States	Events
Machine M_1	M_1 available: I_1	Arrival of a part to M_1: a_1
	M_1 processing: P_1	End of processing: t_1
	M_1 holding a processed part: H_1	Part delivered to robot: e_1
	$X_1 = \{I_1, P_1, H_1\}$	$E_1 = \{a_1, t_1, e_1\}$
Robot	Robot available: I	Part delivered to robot: e_1
	Transporting $M_1 - M_2$: T_{12}	Arrival at M_2: c_2
	Waiting at M_2: H	Part delivered to M_2: a_2
	Returning to M_1: R	Robot returned to M_1: r_1
	$X_r = \{I, T_{12}, H, R\}$	$E_r = \{e_1, c_2, a_2, r_1\}$
Machine M_2	M_2 available: I_2	Part delivered to M_2: a_2
	M_2 processing: P_2	End of processing: t_2
	$X_2 = \{I_2, P_2\}$	$E_2 = \{a_2, t_2\}$

Example 2.2 *Let $\Sigma = \{a, b, g\}$ be a set of events. Language L_1, defined as $L_1 = \{\epsilon, a, abb\}$, consists of only three traces. Language L_2 that contains all possible traces of length 3 that start with event a, can also be listed, namely $L_2 = \{aaa, aab, aag, aba, aga, abb, agg, abg, agb\}$.*

Let us denote by Σ^* the set of all finite traces formed with events $\sigma \in \Sigma$, including the empty trace ϵ. Σ^* is also referred to as the Kleene-closure of Σ. Notice that the set Σ^* is countably infinite since it contain traces of arbitrarily long length. For example, if $\Sigma = \{a, b, c\}$, then

$$\Sigma^* = \{\epsilon, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, \ldots\}.$$

A language L defined over an event set Σ is, therefore, a subset of Σ^*.

The key operation to construct traces, is the *concatenation*. The trace abb in Example 2.2 is the concatenation of trace ab with event b; consequently, trace ab is the concatenation of events a and b. The empty trace ϵ is the identity element of concatenation, *i.e.*, $\epsilon u = u\epsilon = u$ for every trace u.

Some terminology about traces can be considered. If $tuv = s$ with t, u, $v \in \Sigma^*$, then t is a prefix of s, u is a subtrace of s and v is a suffix of s.

2.2.2 Operations on languages

The usual set operations, such as union, intersection, difference, and complement with respect to Σ^*, are applicable to languages since languages are sets. Besides these operations, four other operations can be defined for languages: concatenation, prefix-closure, Kleene-closure and natural projection.

- **Concatenation:** Let $L_a, L_b \subseteq \Sigma^*$, then

$$L_a L_b := \{s \in \Sigma^* : (s = s_a s_b) \wedge (s_a \in L_a) \wedge (s_b \in L_b)\}.$$

In words, a trace is in $L_a L_b$ if it can be written as the concatenation of a trace in L_a with another trace in L_b.

- **Prefix-closure:** Let $L \subseteq \Sigma^*$, then

$$\bar{L} := \{s \in \Sigma^* : (\exists t \in \Sigma^*)[st \in L]\}.$$

In words, the prefix closure $\overline{L} \subseteq L$ is formed by all prefixes of all traces in L.

- **Kleene closure:** Let $L \subseteq \Sigma^*$, then

$$L^* := \{\epsilon\} \cup L \cup LL \cup LLL \cup ...$$

This is the same operation as that defined for set Σ, except, now, that it is applied to set L whose elements may be traces of finite length.

- **Natural projection:** The natural projection, or simply projection, is a mapping from a set of events, Σ_l, to a smaller set of events, Σ_s, where $\Sigma_s \subset \Sigma_l$ and, is denoted by P; a subscript is usually added to specify either Σ_s or both Σ_l and Σ_s, for the sake of clarity, when dealing with multiple sets. We extended the projection operation to traces as follows:

$$\begin{aligned} P: \quad & \Sigma_l^* \to \Sigma_s^* \\ & s \mapsto P(s) \end{aligned}$$

with the following properties

$$P(\epsilon) = \epsilon$$

$$P(\sigma) = \begin{cases} \sigma, & \text{if } \sigma \in \Sigma_s \\ \epsilon, & \text{if } \sigma \in \Sigma_l \setminus \Sigma_s \end{cases}$$

$$P(s\sigma) = P(s)P(\sigma),\ \forall s \in \Sigma_l^*,\ \forall \sigma \in \Sigma_l.$$

According to the previous definition, the projection operation erases the events of $s \in \Sigma_l^*$ that do not belong to the smaller event set, Σ_s. This operation can be used to obtain the observed language of a system.

We can also work with the corresponding inverse mapping, called inverse projection, which is defined as follows:

$$P^{-1} : \Sigma_s^* \to 2^{\Sigma_l^*}$$

$$s \mapsto P^{-1}(s) = \{t \in \Sigma^* : P(t) = s\}$$

The projection P can be extended to a language $L \subseteq \Sigma_l^*$ by applying it to all the traces in L. Thus,

$$P(L) = \{t \in \Sigma_s^* : (\exists s \in L)[P(s) = t]\}.$$

The inverse projection can also be extended to a language $L_s \subseteq \Sigma_s^*$ as follows:

$$P^{-1}(L_s) = \{s \in \Sigma_l^* : (\exists t \in L_s)[P(s) = t]\}.$$

In order to illustrate the notion of projection, consider the following example [58].

Example 2.3 *Consider set* $\Sigma = \{a, b, c\}$ *and the two proper subsets* $\Sigma_1 = \{a, b\}$ *and* $\Sigma_2 = \{b, c\}$ *of* Σ. *Consider, in addition, the following language:*

$$L = \{c, ccb, abc, cacb, cabcbbca\} \subset \Sigma^*$$

Let the projections $P_i : \Sigma^* \to \Sigma_i^*$, $i = 1, 2$. *Thus,*

$$\begin{aligned} P_1(L) &= \{\epsilon, b, ab, abbba\} \\ P_2(L) &= \{c, ccb, bc, cbcbbc\} \\ P_1^{-1}(\{\epsilon\}) &= \{c\}^* \\ P_1^{-1}(\{b\}) &= \{c\}^*\{b\}\{c\}^* \\ P_2^{-1}(\{bc\}) &= \{a\}^*\{b\}\{a\}^*\{c\}\{a\}^* \end{aligned}$$

□

2.3 Automata

An automaton is a device that is capable of representing a language according to well defined rules. The formal definition of an automaton is presented in the sequel [58].

2.3.1 Deterministic automata

The deterministic automaton is formally defined as follows:

Definition 2.2 *(Deterministic automata) A deterministic automaton, denoted by G, is a six-tuple*

$$G = (X, \Sigma, f, \Gamma, x_0, X_m),$$

where X is the set of states, Σ is the finite set of events, $f : X \times \Sigma \to X$ is the transition function, where $f(x, \sigma) = y$ means that there is a transition labeled by event $\sigma \in \Sigma$ from state $x \in X$ to state $y \in X$, $\Gamma : X \to 2^{\Sigma}$ is the active event function, i.e., $\Gamma(x)$ is the set of events $\sigma \in \Sigma$ for which $f(x, \sigma)$ is defined, x_0 is the initial state, and $X_m \subseteq X$ is the set of marked states.

Automata are graphically represented by state transitions diagrams. In these diagrams, the states are circles that are connected by arcs labeled with events. The marked states are identified by two concentric circles and, in general, are related with the conclusion of a task. The initial state is indicated by an arrow pointing to it.

Example 2.4 *Let $G = (X, \Sigma, f, \Gamma, x_0, X_m)$ be the automaton shown in the Figure 2.1. Thus, $X = \{1, 2, 3, 4, 5\}$, $\Sigma = \{a, b, c, d\}$, $x_0 = 1$, $X_m = \{3\}$, the transition function f is defined as $f(1, a) = 2$, $f(1, b) = 4$, $f(2, b) = 3$, $f(3, c) = 3$, $f(4, d) = 5$ and $f(5, c) = 5$, the active event function is given by: $\Gamma(1) = \{a, b\}$, $\Gamma(2) = \{b\}$, $\Gamma(3) = \{c\}$, $\Gamma(4) = \{d\}$, and $\Gamma(5) = \{c\}$.*

An automaton is capable of representing languages. Two kinds of languages can be associated with the behavior of an automaton G: the generated language and the marked language. The generated language, denoted by $L(G)$, is formed by all traces that can be executed by G, starting at the initial state. The marked language,

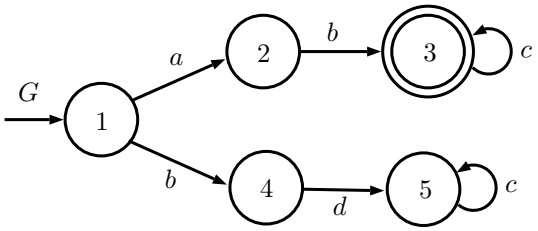


Figure 2.1: Automaton G.

denoted by $L_m(G)$, is a subset of the generated language $L(G)$ and consists of all traces that finish in a marked state in the state transition diagram of G.

Definition 2.3 *(Generated language and marked language). The generated language of* $G = (X, \Sigma, f, \Gamma, x_0, X_m)$ *is defined as:*

$$L(G) = \{s \in \Sigma^* : f(x_0, s)!\}.$$

where $f(x_0, s)!$ *denotes that* $f(x_0, s)$ *is defined, i.e.,* $\exists y \in X$ *such that* $f(x_0, s) = y$. *The marked language of* G *is defined as:*

$$L_m(G) = \{s \in L(G) : f(x_0, s) \in X_m\}.$$

Notice that the definition of the generated language implies that $\epsilon \in L(G)$.

The language generated by automaton G, shown in Figure 2.1, is $L(G) = \{\epsilon, a, b, abc^n, bdc^n\}$, where $n \in \{0, 1, 2, ...\}$. Since $X_m = \{3\}$, the marked language is $L_m = abc^*$.

2.3.2 Nondeterministic automata

A nondeterministic automaton is defined by $G_{nd} = (X, \Sigma, f_{nd}, \Gamma, X_0, X_m)$, where X is the state set, Σ is the set of events, $f_{nd} : X \times \Sigma \to 2^X$, where 2^X is the set of all subsets of X, Γ is the set of feasible events, $X_0 \subseteq X$ and X_m is the set of marked states. Notice that, differently from deterministic automata, automaton G_{nd} can have more than one initial state and the codomain of the transition function f_{nd} is a subset of X, not a single state.

In order to illustrate a nondeterministic automaton, consider the following example.

Example 2.5 *Let the nondeterministic automaton,* $G_{nd} = (X, \Sigma, f_{nd}, \Gamma, X_0, \emptyset)$, *shown in Figure 2.2. Notice that, the transition function assumes values in* 2^X,

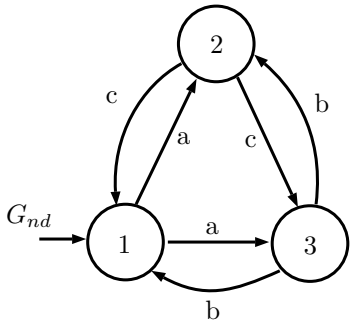


Figure 2.2: Nondeterministic automaton.

for $x \in X$. For instance, $f_{nd}(1, a) = \{2, 3\}$, $f_{nd}(2, c) = \{1, 3\}$ and $f_{nd}(3, b) = \{1, 2\}$. Thus, this type of configuration suggests uncertainty in the dynamic evolution of the system. □

Automata with ϵ-transitions

Differently from the deterministic and nondeterministic automata seen before, in an automaton with ϵ-transitions, states can change spontaneously without detecting any event. This class of automata are important when we want to model problems in the plant, for example, failure and missing of sensors. Since, one of the characteristics of these automata is the uncertainty of dynamic evolution of the system, they are considered also nondeterministic automata.

An automaton with ϵ-transitions, or simply, automaton-ϵ is defined as the sextuple

$$G_\epsilon = (X, \Sigma \cup \{\epsilon\}, f_\epsilon, \Gamma, X_0, X_m),$$

where each parameter of G_ϵ is similar to parameters of nondeterministic automaton G_{nd}. Notice that, the transition function f_ϵ is defined as $f_\epsilon = X \times \Sigma \cup \{\epsilon\} \to 2^X$.

Example 2.6 *Consider automaton G_ϵ shown in Figure 2.3. Notice that, this automaton does not generate trace bc, i.e., there does not exist x such that $x = f_\epsilon(x_0, bc)$. However, automaton G_ϵ generates trace $\epsilon b \epsilon c$ which, when observed, is equivalent to trace bc.* □

In order to define the generated and marked language by automaton G_ϵ, let us introduce the notion of ϵ-reach. The ϵ-reach of state x, denoted as $\epsilon R(x)$, is defined as the set of all states reached from state x by following only transitions labeled by

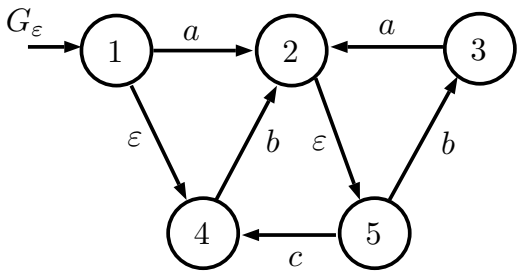


Figure 2.3: Nondeterministic automaton with ϵ-transition.

ϵ. By definition, $x \subseteq \epsilon R(x)$. The definition of ϵ-reach can be naturally extended to a set of states Y as follows

$$\epsilon R(Y) = \bigcup_{y \in Y} \epsilon R(y) \tag{2.1}$$

Consider, again, the ϵ-automaton shown in Figure 2.3. The ϵ-reach of states in G_ϵ are as follows: $\epsilon R(1) = \{1, 4\}$, $\epsilon R(2) = \{2, 5\}$, $\epsilon R(3) = \{3\}$, $\epsilon R(4) = \{4\}$, $\epsilon R(5) = \{5\}$.

The extended transition function $\tilde{f}_\epsilon$ is defined in a recursive way as follows. First we set

$$\tilde{f}_\epsilon(x, \epsilon) = \epsilon R(x). \tag{2.2}$$

Second, for $w \in \Sigma^*$ and $\sigma \in \Sigma$, we set

$$\tilde{f}_\epsilon(x, w\sigma) = \epsilon R[\{k : k \in f_\epsilon(y, \sigma) \text{ for some state } y \in \tilde{f}_\epsilon(x, w)].$$

We can now characterize the generated and marked languages of automaton G_ϵ. The language generated by G_ϵ is defined as:

$$L(G_\epsilon) = \{w \in \Sigma^* : (\exists x \in X_0)[\tilde{f}_\epsilon(x, w)!]\}, \tag{2.3}$$

and the language marked by automaton G_ϵ is defined as:

$$L_m(G_\epsilon) = \{w \in L(G_\epsilon)(\exists x \in X_0)[\tilde{f}_\epsilon(x, w) \cap X_m \neq \emptyset]\}. \tag{2.4}$$

As said earlier, an important application of nondeterministic automata with ϵ-transitions is in the modeling of failures in the plant and loss of information in sensors, so that some events become unobservable. However, a system with unobservable events can be modeled using a deterministic automata called Observer, which will be described in the next subsection.

2.3.3 Observer automata

Suppose that Σ is partitioned as $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$, where Σ_o is the set of observable events and Σ_{uo} is the set of unobservable events. An event is observable when its occurrence can be registered and communicated to the observer. The unobservable events are those that cannot be observed by sensors (including the failure events) or, even though there are sensors to register it, these events cannot be seen because of the distributed nature of the system. When $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$, the automaton is called automaton with unobservable events.

The dynamic behavior of an automaton with unobservable events can be described by a deterministic automaton called observer automaton, whose set of events is formed by observable events only. The observer for G, is denoted by $Obs(G)$, and it is defined as follows:

$$Obs(G) = (X_{obs}, \Sigma_o, f_{obs}, \Gamma_{obs}, x_{0_{obs}}, X_{m_{obs}}),$$

where $X_{obs} \subseteq 2^X$ and $X_{m_{obs}} = \{B \in X_{obs} : B \cap X_m \neq \emptyset\}$. In order to define $x_{0_{obs}}$, Γ_{obs}, and f_{obs}, it is necessary to introduce the concept of unobservable reach of a state $x \in X$, denoted as $UR(x, \Sigma_o)$:

$$UR(x, \Sigma_o) = \{y \in X : (\exists t \in \Sigma_{uo}^*)[f(x,t) = y]\}$$

The unobservable reach can be extended to a set $B \in 2^X$ as follows:

$$UR(B) = \bigcup_{x \in B} UR(x, \Sigma_o)$$

Thus, $x_{0,obs} = UR(x_0, \Sigma_o)$, and for all $x_{obs} \in X_{obs}$, $\Gamma_{obs}(x_{obs}) = \bigcup_{x \in x_{obs}} \Gamma(x)$, $f_{obs}(x_{obs}, \sigma) = \bigcup_{x \in (x_{obs}) \wedge (f(x,\sigma)!)} UR[f(x,\sigma), \Sigma_o]$, if $\sigma \in \Gamma_{obs}(x_{o,obs})$, or, undefined, otherwise.

In order to illustrate an observer automaton, consider the following example [59].

Example 2.7 *Let us consider automaton G shown in Figure 2.4. Suppose that the event set $\Sigma = \{a, b, c\}$ is partitioned in $\Sigma_o = \{b, c\}$ and $\Sigma_{uo} = \{a\}$. Thus, when the automaton starts, it is not possible to know if it is in the initial state $x_0 = 0$ or if it has changed to state $x = 1$, because the occurrence of event a cannot be registered. Thus, the initial state of $Obs(G, \Sigma_o)$ shown in Figure 2.5 is $\{0, 1\}$. In*

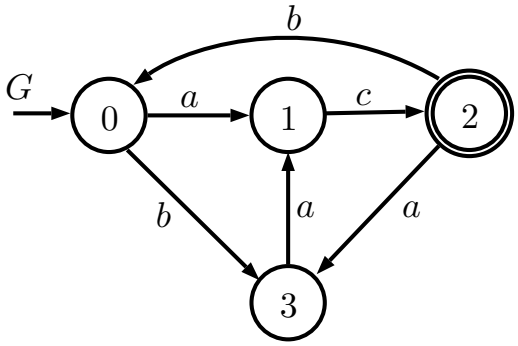


Figure 2.4: Automaton G.

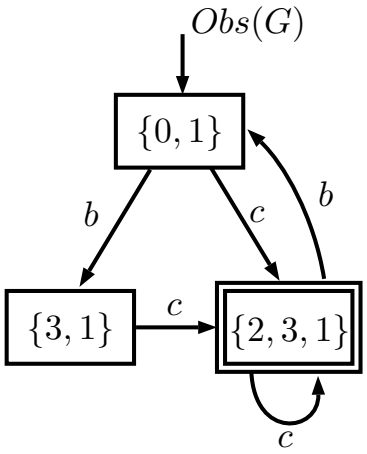


Figure 2.5: Observer automaton of G.

case event b occurs, we can see that the automaton reaches state $\{3\}$*, but, since event a is unobservable, it can also change to state* $\{1\}$ *without realizing the occurrence of event a. Therefore, the occurrence of event b in the observer automaton leads to state* $\{3, 1\}$*, from the initial state. There are also transitions labeled with event c that reach state* $\{2, 3, 1\}$ *from initial state* $\{0, 1\}$ *and state* $\{3, 1\}$*. Finally, when the occurrence of event c is registered, the observer automaton will keep in state* $\{2, 3, 1\}$*. However, if event b occurs, the observer will return to the initial state.* □

Let us consider the projection $P_o : \Sigma^* \to \Sigma_o^*$. Then, by construction, observer $Obs(G)$ has the following properties:

- $Obs(G)$ is a deterministic automaton.
- $L(Obs(G)) = P_o[L(G)]$.
- $L_m(Obs(G)) = P_o[L_m(G)]$.

- Let $B(t) \subseteq X$ be the state of $Obs(G)$ that is reached after trace $t \in P_o[L(G)]$, i.e., $B(t) = f_{obs}(x_{0,obs}, t)$. Then, a state $x \in B(t)$ iff x is reachable in G by a trace in $P_o^{-1}(t) \cap L(G)$.

2.3.4 Operations on automata

In order to analyze a DES modeled by an automaton, we can use a set of operations on a single automaton. These operations modify appropriately the state transition diagram according to some language operation that we wish to perform. We also need to define operations that allow us to combine, or compose, two or more automata, so that models of complete systems can be built from models of individual system components.

The operations that modify a single automaton are called unary operations. They alter state transition diagram of an automaton, but the set of events Σ remains unchanged. The main unary operations are: Accessible part, Coaccessible part, Trim operation and Complement. The composition operations are operations that combine more than one automaton. The main composition operations are: parallel and product composition.

- **Accessible part:** If we delete from G all the states that are not reachable from x_0 by some trace in $L(G)$ and their related transitions, without affecting the languages generated and marked by G, then we take the accessible part of automaton G. We will denote this operation by $Ac(G)$, where Ac stands for taking the accessible part, and is defined as follows:

$$Ac(G) \quad = \quad (X_{ac}, \Sigma, f_{ac}, x_0, X_{ac,m})$$

  where $X_{ac} = \{x \in X : (\exists s \in \Sigma^*)[f(x_0, s) = x]\}$, $X_{ac,m} = X_m \cap X_{ac}$, and $f_{ac} = f|_{X_{ac} \times \Sigma \to X_{ac}}$.

  The notation $f|_{X_{ac} \times \Sigma \to X_{ac}}$ means that we are restricting f to the smaller domain of the accessible states X_{ac}.

- **Coaccessible part:** A state $x \in X$ of G is said to be coaccessible if there exists a path in the state transition diagram of G from state x to a marked state. We take the coaccessible part by deleting all states of G that are not coaccessible

and their related transitions. This operation is denoted by $CoAc(G)$, where $CoAc$ stands for taking the coaccessible part, and is defined as follows:

$$CoAc(G) \;=\; (X_{coac}, \Sigma, f_{coac}, x_{0,coac}, X_m),$$

where $X_{coac} = \{x \in X : (\exists s \in \Sigma^*)[f(x,s) \in X_m]\}$, $x_{0,coac} = x_0$, if $x_0 \in X_{coac}$, undefined, otherwise, and $f_{coac} = f|_{X_{coac}\times\Sigma\to X_{coac}}$.

The $CoAc$ operation may shrink $L(G)$, since we may delete states that are accessible from x_0. However, the $CoAc$ operation does not affect $L_m(G)$, since a deleted state cannot be on any path from x_0 to X_m. If $G = CoAc(G)$, then G is said to be coaccessible; in this case, $L(G) = \overline{L_m(G)}$.

- **Trim operation:** An automaton that is both accessible and coaccessible is said to be trim. We define the Trim operation to be

$$Trim(G) = CoAc[Ac(G)] = Ac[CoAc(G)].$$

- **Complement:** Consider a deterministic automaton $G = (X, \Sigma, f, \Gamma, x_0, X_m)$ whose marked language is $L_m(G)$. The complement of G os an automaton G_{comp} such that $L_m(G_{comp}) = \Sigma^* \setminus L_m(G)$, where $G_{comp} = Comp(G)$. The operation $Comp(\cdot)$ is denoted as Complement. Automaton G_{comp} is obtained as follows.

  First, complete the transition function f of G. Thus, G will become a complete automaton whose generated language is $L(G) = \Sigma^*$. In order to do it, let us denote the new transition function by f_t. Then, a new state x_d is added to X. After that, all undefined $f(x,\sigma)$ in G are then assigned to x_d. Thus,

$$f_t(x,\sigma) = \begin{cases} f(x,\sigma), \text{ if } \sigma \in \Gamma(x) \\ x_d, \text{ otherwise} \end{cases}$$

  The new automaton $G_t = (X \cup \{x_d\}, \Sigma, f_t, x_0, X_m)$ is such that $L(G_t) = \Sigma^*$ and $L_m(G_t) = L(G)$.

  After we obtain automaton G_t, we change the marking status of all states of G_t, *i.e.*, we mark all unmarked states (including x_d) and remove the marking of the marked states. Thus,

$$Comp(G) = (X \cup \{x_d\}, \Sigma, f_{tot}, x_0, (X \cup x_d) \setminus X_m).$$

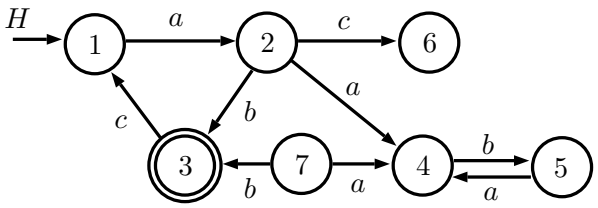


Figure 2.6: Automaton H

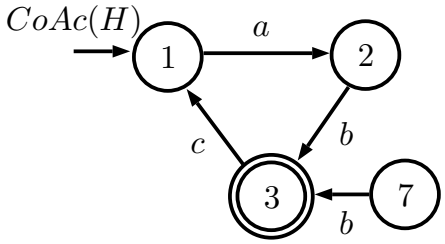


Figure 2.7: Coaccessible part of automaton H

The following example illustrate the unary operations [58].

Example 2.8 *Let automaton H shown in Figure 2.6. In order to obtain its coaccessible part, we need to delete all states from which it is not possible to reach marked state* 3. *Thus, states* 4, 5 *and* 6 *are deleted, leading to the automaton shown in Figure 2.7. Automata $Trim(H)$ and $Comp[Trim(H)]$ are shown in Figures 2.8 and 2.9.*

There are two composition operations called product and parallel composition. In order to describe these operations, consider the following two automata:

$$H_1 = (X_1, \Sigma_1, f_1, \Gamma_1, x_{0_1}, X_{m_1}) \text{ and } H_2 = (X_2, \Sigma_2, f_2, \Gamma_2, x_{0_2}, X_{m_2}).$$

- **Product composition:** The product operation between automata H_1 and

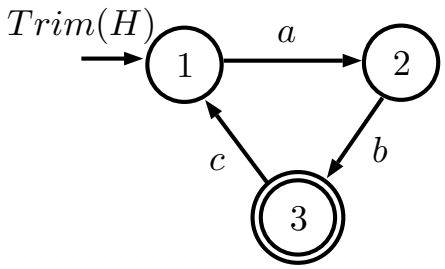


Figure 2.8: Automaton $Tim(H)$

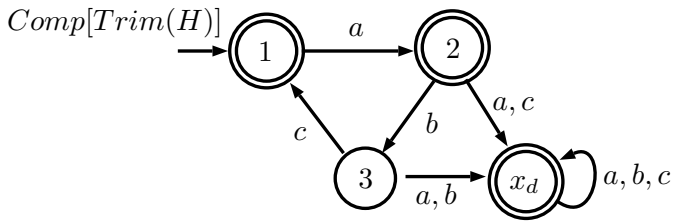


Figure 2.9: Comp[$Trim(H)$]

H_2 results in the automaton

$$H_1 \times H_2 = Ac\{X_1 \times X_2, \Sigma_1 \cup \Sigma_2, f_{1\times 2}, \Gamma_{1\times 2}, (x_{0_1}, x_{0_2}), X_{m_1} \times X_{m_2}\}$$

where

$$f_{1\times 2}((x_1, x_2), \sigma) = \begin{cases} (f_1(x_1, \sigma), f_2(x_2, \sigma)), \text{ if } \sigma \in \Gamma_1(x_1) \cap \Gamma_2(x_2), \\ \text{undefined, otherwise.} \end{cases}$$

and

$$\Gamma_{1\times 2}((x_1, x_2)) = \Gamma_1(x_1) \cap \Gamma_2(x_2)$$

In the product, the transitions of the two automata must always be synchronized on common events, *i.e.*, events in $\Sigma_1 \cap \Sigma_2$. Thus, it corresponds to intersection of the generated and marked languages:

$$\begin{aligned} L(H_1 \times H_2) &= L(H_1) \cap L(H_2), \\ L_m(H_1 \times H_2) &= L_m(H_1) \cap L_m(H_2). \end{aligned}$$

Properties of the product

1. Product is commutative up to a reordering of the state components in the composed states.
2. Product is associative and it can be defined as

$$G_1 \times G_2 \times G_3 = (G_1 \times G_2) \times G_3 = G_1 \times (G_2 \times G_3).$$

- **Parallel composition:** This composition operation is, usually, used to connect different components in order to model a unique system whose components work in synchrony. In general, when a system is composed by different components that interact with each other, the event set of each component

contain events that belong solely to itself, called as private events, and common events that are shared with each other. Thus, the parallel operation is suitable to model a entire system from individual components.

The parallel composition between automata H_1 and H_2 results in the following automaton

$$H_1||H_2 = Ac(X_1 \times X_2, \Sigma_1 \cup \Sigma_2, f_{1||2}, \Gamma_{1||2}, (x_{0_1}, x_{0_2}), X_{m_1} \times X_{m_2})$$

where

$$f_{1||2}((x_1, x_2), \sigma) = \begin{cases} (f_1(x_1, \sigma), f_2(x_2, \sigma)), \text{ if } \sigma \in \Gamma_1(x_1) \cap \Gamma_2(x_2), \\ (f_1(x_1, \sigma), x_2), \text{ if } \sigma \in \Gamma_1(x_1) \setminus \Sigma_2, \\ (x_1, f_2(x_2, \sigma)), \text{ if } \sigma \in \Gamma_2(x_2) \setminus \Sigma_1, \\ \text{not defined, otherwise.} \end{cases}$$

In the parallel composition, events in $\Sigma_1 \cap \Sigma_2$, can only be executed if both automata execute them at the same time, so that, the automata are synchronized on the common events. On the other hand, the private events, *i.e.*, events in $(E_2 \setminus E_1) \cup (E_1 \setminus E_2)$, can be executed whenever possible. In order to characterize the generated and marked languages of $G_1 \| G_2$ with respect to G_1 and G_2, we use the operation of language projection introduced earlier. Let $\Sigma_1 \cup \Sigma_2$ be the larger set of events and let Σ_1 or Σ_2 be the smaller sets of events. Thus, we can consider two projections

$$P_i : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_i^* \text{ for } i = 1, 2.$$

Now, we can characterize the languages resulting from a parallel composition:

1. $L(G_1 \| G_2) = P_1^{-1}[L(G_1)] \cap P_2^{-1}[L(G_2)]$,
2. $L_m(G_1 \| G_2) = P_1^{-1}[L_m(G_1)] \cap P_2^{-1}[L_m(G_2)]$.

Properties of parallel composition

1. $P_i[L(G_1 \| G_2)] \subseteq L(G_i)$, for $i = 1, 2$.

   The coupling of the two automata by common events may prevent some of the traces in their individual generated languages to occur, due to the constraints imposed in the definition of parallel composition regarding these common events.

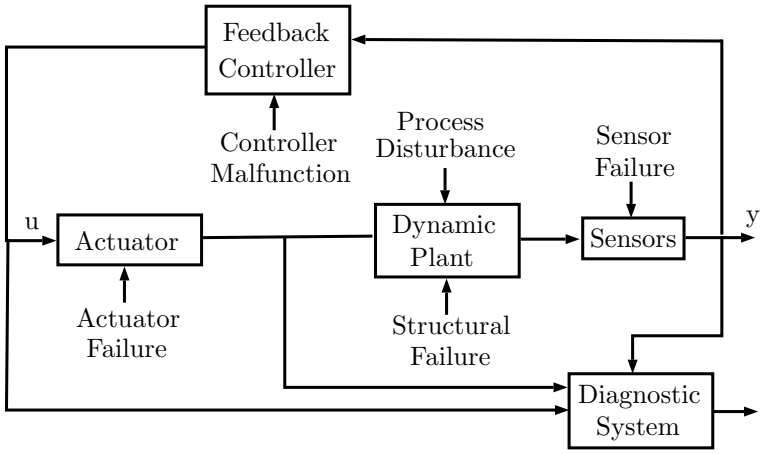


Figure 2.10: A general diagnosis framework [1].

2. Parallel composition is commutative up to a reordering of the state components in composed states.

3. Parallel composition is associative:

$$G_1||G_2||G_3 = (G_1||G_2)||G_3 = G_1||(G_2||G_3).$$

2.4 Failure diagnosis

Failure is a term that defines a deviation from normal behavior operation of a system. The failure treatment, called failure diagnosis, is an important problem in engineering and consists of detecting and isolating the failure with as much detail as possible, such as the place where the failure occurred and its dimension.

In Figure 2.10, the components of a general failure diagnosis framework is described. The figure shows a controlled process system and indicates the different sources of failures. In [1] the authors classify the failures in three classes:

- Parameter changes in a model. In a complex plant, there are several processes occurring under the selected level of detail of the model. Usually, the processes which are not detailed in the model, are typically gathered in some parameters of the model. Frequently, parameter failures arise when there is a disturbance in the process from the environment.

- Structural changes. Structural changes are changes in the physical structure of the process. They occur due to failures in equipments. Structural malfunctions

result in a change of characteristics of the plant, and consequently, in the information flow of several variables. In order to handle such a failure in a diagnosis system, it is necessary to remove the current model equations and replace them with other equations which are capable to describe the current situation of the process.

- Malfunctioning sensors and actuators. Errors and malfunctioning usually occur in several devices, but, mainly in actuators and sensors. Some of the devices provide feedback signals which are very important for the monitoring and control of the plant. A failure in one of these instruments can cause, in the plant state variables, a deviation beyond safe acceptable limits, unless the failure is detected and corrected in time.

The failure diagnosis systems are very important in supervisor systems and in the failure management of processes. They are responsible for monitoring the behavior of a complex plant and for providing information about abnormal operation conditions of its components. In order to design a good failure diagnosis system, a set of desirable characteristics have to be taken into account. In the sequel, some of these characteristics are listed [1]:

- Quick detection and diagnosis. An important characteristics of a diagnosis system is its capacity to respond quickly in detecting and isolating process failures.

- Isolability. Isolability is the ability of the diagnosis system to distinguish a failure among different types of failures.

- Robustness. It is important that the diagnosis system is robust to noises and uncertainties, so that, when a disturbance occurs, the diagnoser does not confuse it with a failure and send a false alarm.

- Adaptability. Processes, in general, change and evolve due to changes in external inputs or structural changes due to the feedback loop. Process operating conditions can change not only due to disturbances but also due to change in environmental conditions such as variations in production quantities with

changing demands, changes in the quality of raw material etc. Thus the diagnosis system must be adaptable to changes.

- Modeling requirements. The amount of modeling required for the development of a diagnosis classifier is an important issue. For fast and easy deployment of real-time diagnosis classifiers, the modeling effort must be as minimal as possible.

Due to these characteristics, many algorithms have been developed to failure diagnosis problem. In the next subsection, the classification of some algorithms is shown.

2.4.1 Algorithms of failure diagnosis

A previous knowledge of the process is necessary in order to build a diagnosis system. It is provided by several characteristics and relations among observed symptoms and failures. These knowledge can be acquired with experiences in the process and is usually referred to as knowledge based models or process history [1].

The knowledge based models can be classified as qualitative or quantitative. In quantitative models, the plant is expressed in terms of mathematical functional relationships between the inputs and outputs of the system. In contrast, in qualitative model, these relationships are expressed in terms of qualitative functions in different units in a process. Other methods can be developed from the extraction of the historical system data. These extractions or abstractions of characteristics can be qualitative or quantitative. In quantitative characteristics, the abstractions can behave as statistical or not statistical [1].

In contrast to the model-based approaches, where a priori knowledge about the model (either quantitative or qualitative) of the process is assumed, in process history based methods, only the availability of large amount of historical process data is assumed. This classification can be seen in Figure 2.11.

The quantitative methods based models require two steps: *(i)* verification of inconsistencies and residual "r" (difference among values of several functions of outputs and their desirable values under/without failure conditions) between real and desirable behavior and; *(ii)* choice of decision rules for failure diagnosis. Parameters

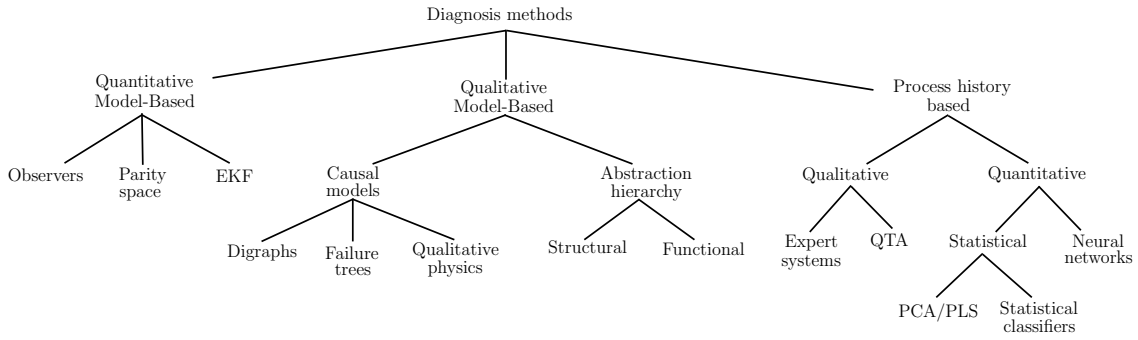


Figure 2.11: Classification of diagnosis algorithms [1].

and states estimators are used in this kind of strategy. The residuals generated are verified and decision functions are performed based on the residuals and decision rules. The quantitative based models also involve analysis of cause and effect on the system behavior. The main disadvantages of this method is the use of too many assumptions and its high computational effort; on the other hand, the fact that it tries to imitate the human analysis is a great advantage.

Methods based in process history usually require a great deal of data. They are divided in qualitative and quantitative methods. The qualitative methods are based on specialized systems and involve an explicit mapping of known symptoms. The quantitative methods, on the other hand, use neural networks and statistical methods.

Traditionally, the most widely used methods for failure diagnosis are based on process models. As presented earlier, such methods try to detect all deviations between the output of the system and the expected output model, supposing that this deviation is related to a failure. In this context, we use a qualitative method based on models (failure trees) in this thesis. The method is based on discrete event systems as presented in [3] and [59].

2.4.2 Failure diagnosis of DES

In this section we will present the main concepts of failure diagnosis systems of DESs. As seen before, when we add, in the model G, unobservable events, they can represent a failure event or not. Thus, it is possible to take into account the normal behavior of the system, described by observed and unobserved events that are not associated with failures in the system or abnormal behaviors, that is described by

unobservable events associated to failures. Then, let $\Sigma_f \subseteq \Sigma_{uo}$ be a set of events associated with failures of the system. In general, the set $\Sigma_f = \bigcup \Sigma_{f_i}, i = 1, 2, \ldots, n$, where i means the types of failure that can occur in the plant and each set Σ_{f_i} is formed by events that model failures that are correlated in some manner.

In theory of failure diagnosis in DES, we adopt the following notations.

- s_f: last event of trace s.
- $\psi(\Sigma_f) = \{s \in L : s_f \in \Sigma_f\}$: set of all traces of L that finish with event $\sigma_f \in \Sigma_f$.
- $L/s = \{t \in \Sigma^\star : st \in L\}$: language L of continuations after prefix s.
- $\Sigma_f \in s$: trace $s \in L$ contains a failure event.

We can now present the formal definition of diagnosability of a language L [3].

Definition 2.4 *Let L be a language generated by an automaton G and suppose that L is prefix closed. Thus, L is diagnosable with respect to projection $P_o : \Sigma^\star \to \Sigma_o^\star$ and $\Sigma_f = \{\sigma_f\}$ if the following condition holds true [3]:*

$$(\exists n \in \mathbb{N})(\forall s \in \psi(\Sigma_f))(\forall t \in L \setminus s)(||t|| \geq n \Rightarrow D),$$

where the diagnosis D is

$$(\nexists \omega \in L)[P_o(st) = P_o(\omega) \wedge (\Sigma_f \notin \omega)].$$

In words, the language generated by an automaton G will be diagnosable with respect to the set of observable events Σ_o, projection P_o and the set of failure events $\Sigma_f = \{\sigma_f\}$, if the occurrence of event σ_f can be detected after a finite number of transitions after the occurrence of σ_f using traces of observable events only.

2.5 Diagnoser

In order to diagnose failures based on observation of the system behavior in real time and to verify if the language generated by an automaton G is diagnosable, we can use a deterministic automaton called diagnoser. Furthermore, depending on how the information about the dynamic evolution of the system is available, the diagnosis

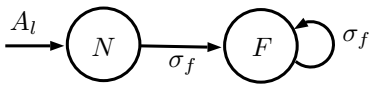


Figure 2.12: Label automaton A_l for building diagnoser

system can be divided in two different classes: centralized, that has a unique acquisition system or distributed, which is connected by a communication network, for example, manufacturing systems or electric power systems. Thus, broadly speaking diagnosis systems can be classified as follows:

- Centralized, when there is only one diagnoser that has access to all observable events of system;
- Decentralized, when the reading of the sensors are not centralized, but distributed in different modules. Each module observes the behavior of some part of the system using a subset of the observable event set of the system [30].

2.5.1 Centralized diagnosis

The centralized diagnoser of plant G, denoted by G_d, is an automaton whose set of events is formed by observable events of G and the states are formed by adding labels F and N to the states of G in order to indicate if event σ_f has occurred or not. Formally, G_d is defined as

$$G_d = (X_d, \Sigma_d, f_d, \Gamma_d, x_{o_d})$$

where $X_d \subseteq 2^{X \times \{N,Y\}}$. The diagnoser G_d can be constructed in two steps: *(i)* get the parallel composition $G||A_l$, where A_l is the label automaton with two states shown in Figure 2.12; *(ii)* compute $Obs(G||A_l)$.

It is important to observe that the automaton obtained after the parallel composition computed in step *(i)* generates the same language as automaton G. Moreover, the states of $G||A_l$ follow the form (x, F) or (x, N) if σ_f is in the trace that takes x_0 to the state or not, respectively.

A diagnoser, such as presented in Figure 2.13c is implemented in practice using a digital computer, (or programable logic controller) [59]. Its initial state is x_{0_d},

and after any occurrence of observable events, its state is updated according to the transition function f_d. When the diagnoser reaches a state whose label is equal to Y, then it is certain that the failure has occurred.

Example 2.9 *Consider a plant represented by automaton G, where $\Sigma_o = \{a, b, c\}$ and $\Sigma_{uo} = \{\sigma_f\}$, shown in Figure 2.13a. In order to obtain its diagnoser, the parallel composition $G\|A_l$ is first computed, which is shown in Figure 2.13b. After that, we obtain the observer automaton of $G\|A_l$. The diagnoser of plant G, G_d, is shown in Figure 2.13c. Notice that the initial state of G_d, $\{1N, 3F\}$, has both labels Y and N. This happens because event σ_f is unobservable and, if it occur, the diagnoser will not realize its occurrence; then the system can either be in states $1N$ or $3F$. As a consequence, the diagnoser cannot say, for sure, whether the failure event has occurred, i.e., it is uncertain with respect to the occurrence of the failure event. On the other hand, another event that can occur when the system is in the initial state is the observable event b; thus, if event b occurs, the diagnoser must indicate that the system is in state $\{2N\}$ and, thus, it will be certain that the failure event has not occurred. Supposing that from the initial state event σ_f has occurred in the plant, the diagnoser will remain in state $\{1N, 3F\}$, but, when event a occurs, then the diagnoser changes the current state, $\{1N, 3F\}$, to state $\{4F\}$, meaning that the diagnoser is now certain that event σ_f occurred.* □

Notice that, by the construction of automaton G_d, $G_d = Obs(G||A_l)$, once the diagnoser is sure of the occurrence of the failure event, all the following states will indicate the failure occurrence. However, it is possible for the diagnoser to change from a normal state to an uncertain.

2.5.2 Decentralized diagnosis

In order to solve the problems associated with the distributed nature of some systems, in [30] the decentralized structure shown in Figure 2.14 has been proposed. In this structure, the reading of sensors is decentralized, or distributed in different modules of diagnosers. Each diagnoser module observes part of the system behavior based on the information from the sensors connected to it, *i.e.*, based on the observable events of each diagnoser Σ_{o_i} where $i = \{1, 2, ..., n\}$, where n is the number

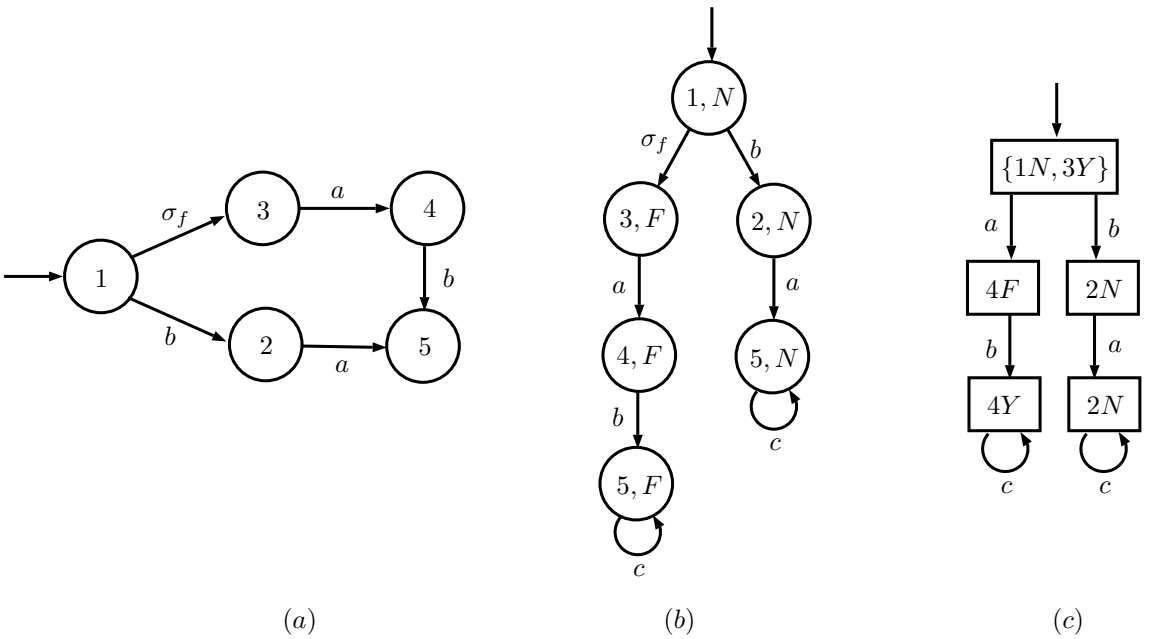


Figure 2.13: (a) Automaton G (b) Parallel composition between G and A_l, (c) $G_d = Obs(G||A_l)$

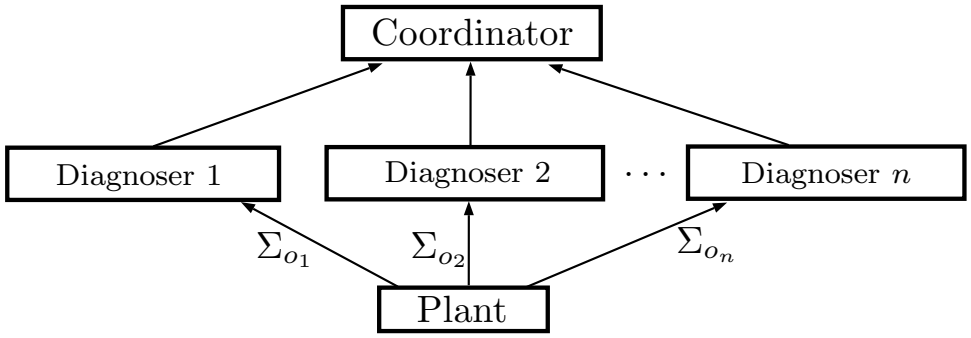


Figure 2.14: Codiagnosis structure.

of diagnosers. Each diagnoser processes the received information and communicates the result to the coordinator. The coordinator receives the information from the diagnosers and processes it according to well defined rules and makes a decision with respect to the failure occurrence.

The diagnosis of a language L depends on a set of elements called protocol, which is composed of rules used to generate local diagnosis, communication rules between the module and the coordinator, decision rules used by the coordinator to diagnose the failure, and the projections $P_{o_i} : \Sigma^\star \to \Sigma_{o_i}^\star, i = 1, 2, ..., n$, associated with each diagnoser. In protocol 3 of [30], a failure is diagnosed when, at least, one local diagnoser identifies its occurrence. Such approach is called codiagnosis. In the sequel, we present the definition of codiagnosability of a language L, but, before it, we will review the notion of normal and failure traces.

Definition 2.5 (Normal and failure traces) *Let $s \in L$, and define $L_s = \{s\}$. Then s is a failure trace if $\exists s_p \in \overline{L}_s$ such that $s_p = \tilde{s}_p \sigma_f \tilde{s}_q$ for some $\tilde{s}_p, \tilde{s}_q \in \Sigma^\star$. Otherwise, s is a normal trace.* □

According to Definition 2.5, a failure trace is a trace of events s such that σ_f is one of its events and a normal trace, on the other hand, does not contain event σ_f.

The set of all normal traces generated by the system is the prefix-closed language $L_N \subset L$. Thus, the set of all failure traces is given by $L \setminus L_N$. Let G_N be the sub-automaton of G that models the normal language of the system with respect to the failure event set Σ_f. Then, $L(G_N) = L_N$.

Definition 2.6 ***(Language codiagnosability)*** *Let L and $L_N \subset L$ be prefix closed languages generated by G and G_N, respectively, and $P_{o_i} : \Sigma^\star \to \Sigma_{o_i}^\star, i = 1, ..., n$ projection operations. Then, L is codiagnosable with respect to the projections P_{o_i} e Σ_f if*

$$(\exists z \in \mathbb{N})(\forall s \in L \setminus L_N)(\forall st \in L \setminus L_N, ||t|| \geq z) \to$$
$$(\exists i \in \{1, 2, \ldots, n\})(P_{o_i}(st) \neq P_{o_i}(\omega), \forall \omega \in L_N)$$

Remark 2.1 *Notice that, when $n = 1$, Definition 2.6 is equal to the definition of language diagnosability proposed in [3].* □

According to Definition 2.6, L is codiagnosable with respect to P_{o_i} and Σ_f if, and only if, for all failure traces $s_F = st$ of arbitrarily long length after the occurrence of the failure event, there do not exist traces $s_{N_i} \in L_N$, where s_{N_j} is not necessarily different from s_{N_k} for $j \neq k$, such that $P_{o_i}(s_{N_i}) = P_{o_i}(s_F)$, for all $i \in \{1, 2, \ldots, n\}$.

2.5.3 Codiagnosability verification

Codiagnosability verification of the language of a discrete event system is the first step to develop a failure diagnosis system for a DES. Some works in literature have addressed the problem of codiagnosability verification of a DES [34, 57]. In this thesis, we will adopt the algorithm presented in [57] as the basis for the problem of network robust diagnosability against delays and loss of observation to be considered later in this work. We use algorithm 2.1, below, originally presented in [57]

Algorithm 2.1 *Codiagnosability verification of DES*

Input: $G = (X, \Sigma, \Gamma, f, x_0)$.
Output: $V = (X_V, \Sigma_V, \Gamma_V, f_V, x_{0_V})$.

1. *Compute automaton* G_N *that models the normal behavior of* G.
   - 1.1. *Define* $\Sigma_N = \Sigma \setminus \Sigma_f$.
   - 1.2. *Build the single state automaton* $A_l^N = (\{N\}, \Sigma_N, f_l^N, x_{0,N})$, *where* $f_l^N(N, \sigma) = N$, *for all* $\sigma \in \Sigma_N$, *and* $x_{0,N} = N$.
   - 1.3. *Build the automaton* $G_N = G \times A_l^N$.
   - 1.4. *Redefine the set of events of* G_N *as* Σ_N, *i.e.,* $G_N = (X_N, \Sigma_N, f_N, (x_0, N))$.
2. *Compute automaton* G_F *that models the failure behavior of automaton* G.
   - 2.1. *Build the label automaton* $A_l = (\{N, F\}, \Sigma_f, f_l^{NF}, x_{0,NF})$ *where* $x_{0,NF} = N$, $f_l^{NF}(N, \sigma_f) = F$, *and* $f_l^{NF}(F, \sigma_f) = F$, *for all* $\sigma_f \in \Sigma_f$, *as shown in Figure 2.12.*
   - 2.2. *Compute* $G_l = G \| A_l$ *and mark all states labeled with* F.
   - 2.3. *Compute* $G_F = CoAc(G_l)$.

3. *Rename the unobservable events of G_{N_i}, as follows.*

   3.1. *Define the following set:*

$$\Sigma'_{R_i} = \{\sigma_{R_i} : \sigma \in \Sigma_{uo_i} \setminus \Sigma_f\}$$

   3.2. *Define $\Sigma_{R_i} = \Sigma_{o_i} \cup \Sigma_f \cup \Sigma'_{R_i}$.*

   3.3. *Define the following renaming function*

$$R_i : \Sigma_N \to \Sigma_{R_i}$$

   where

$$R_i(\sigma) = \begin{cases} \sigma, \ \textit{if } \sigma \in \Sigma_{o_i} \\ \sigma_{R_i}, \ \textit{if } \sigma \in \Sigma_{uo_i} \setminus \Sigma_f \end{cases} \tag{2.5}$$

4. *Compute automaton $G_{R_i} = (X_N, \Sigma_{R_i}, f_{N_i}, (x_0, N))$ obtained from G_N, by renaming its unobservable events according to equation (2.5), for $i = 1, \ldots, n$, i.e., $f_{N_i}(x_N, R_i(\sigma)) = f_N(x_N, \sigma)$, for all $x_N \in X_N$ and $\sigma \in \Sigma_N$.*

5. *Compute the verifier automaton*

$$G_V = G_{R_1} \| G_{R_2} \| \ldots \| G_{R_n} \| G_F \tag{2.6}$$

6. *Verify the existence of a cyclic path $cl = (y_V^k, \sigma_k, y_V^{k+1}, \sigma_{k+1}, \ldots, \sigma_\ell, y_V^k)$, where $\ell \geq k > 0$ in G_V, that satisfy the following condition:*

$$\begin{gathered} \exists j \in \{k, k+1, \ldots, \ell\} \textit{ such that, for some} \\ y_V^j, (y_l^j = F) \wedge (\sigma_j \in \Sigma) \end{gathered} \tag{2.7}$$

   If the answer is yes, them L is not codiagnosable with respect to P_{o_i} and Σ_f.

We present the necessary and sufficient condition for codiagnosability of DES proposed in [57] as follows.

Theorem 2.1 *[57] Let L and L_N be ($L_N \subset L$) the prefix closed languages generated by G and G_N, respectively, and let Σ_f be the set of failure events. Then, L is not diagnosable with respect to $P_{o_i} : \Sigma^\star \to \Sigma_{o_i}^\star$, $i = 1, \ldots, n$, and Σ_f if, and only if, there*

exists a cyclic path $cl := (y_V^k, \sigma_k, y_V^{k+1}, ..., y_V^l, \sigma_l, y_V^k)$*, where* $l \geq k > 0$*, in* V*, that satisfies the following condition:*

$$\exists j \in \{k, k+1, \ldots, \ell\} \text{ s.t. for some } y_V^j, (y_l^j = F) \wedge (\sigma_j \in \Sigma), \tag{2.8}$$

The construction of verifier automaton is illustrated in the next example.

Example 2.10 *[57] Consider the automaton* G *shown in Figure 2.15. Consider a failure diagnosis system composed by two diagnosers, diagnosers* 1 *and* 2*, that observe part of the system, characterizing a decentralized diagnosis system. We need to verify the codiagnosability of* $L(G)$ *with respect to* P_{o_i}*,* $i = 1, 2$ *and* Σ_f*, where* $\Sigma = \{a, b, c, y, \sigma_f\}$*,* $\Sigma_{o_1} = \{a, b\}$*,* $\Sigma_{o_2} = \{a, c\}$*,* $\Sigma_{uo} = \{y, \sigma_f\}$*, and* $\Sigma_f = \{\sigma_f\}$*. Following the steps of Algorithm 2.1, we can obtain an automaton that verify if* $L(G)$ *is codiagnosable. First, we obtain automaton* G_N *by computing the parallel composition between automata* G *and* A_l^N*, i.e.,* $G_N = G \| A_l^N$*. Automata* A_l^N *and* G_N *are shown in Figures 2.16 and 2.17, respectively. After that, we need to obtain automaton* G_F *by performing the parallel composition between automaton* G *and label automaton* A_l*, and then mark all states labeled with* F *and take the coaccessible part of* $G_l = G \| A_l$*,* $G_F = CoAc(G_l)$*. Automaton* A_l *has already been presented in Figure 2.12 and automaton* G_F *is shown in Figure 2.18.*

After we obtain automata G_N *and* G_F*, we need to obtain automata* $G_{R,1}$ *and* $G_{R,2}$*, from* G_N*, by renaming the unobservable events in the sets* $\Sigma_{uo_1} \setminus \Sigma_f = \{c, y\}$ *and* $\Sigma_{uo_2} = \{b, y\}$*, respectively. Automata* $G_{R,1}$ *and* $G_{R,2}$ *are shown in Figures 2.19 and 2.20, respectively. Finally, we obtain the verifier by computing the parallel composition between automata* $G_{R,1}$*,* $G_{R,2}$ *and* G_F*, i.e.,* $G_V = G_{R,1} \| G_{R,2} \| G_F$*. Due to the size of* G_V*, Figure 2.21, shows only the part of* G_V *that corresponds to the trace that satisfies the Condition 2.7, therefore implying that the language generated by* G*,* $L(G)$*, is not codiagnosable with respect to* P_{o_i} *and* Σ_f*.*

2.6 Final remarks

The goal of this chapter was to present the main definitions and concepts of SED modeled by automata. Notions of languages and operations of automata such as parallel and product compositions were introduced. In addition, the main concepts

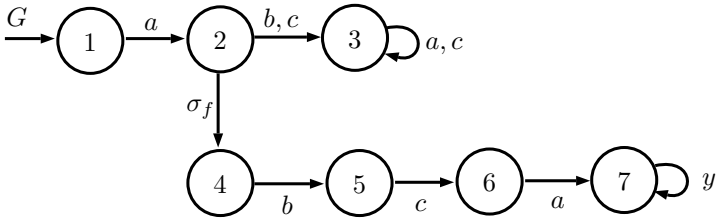


Figure 2.15: Automaton G of example 2.10

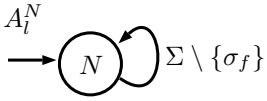


Figure 2.16: Automaton A_l^N

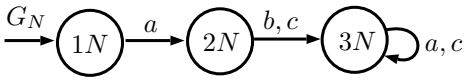


Figure 2.17: Automaton G_N

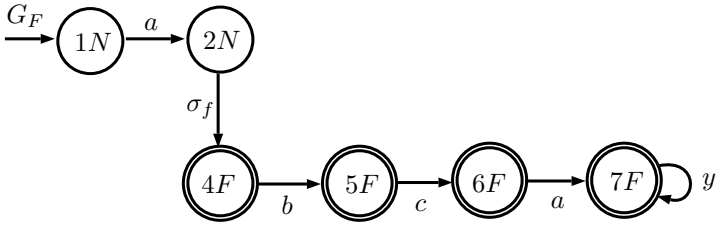


Figure 2.18: Automaton G_F

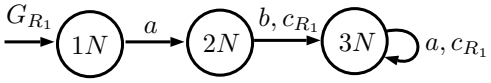


Figure 2.19: Automaton G_{R_1}

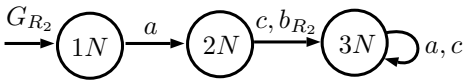


Figure 2.20: Automaton G_{R_2}

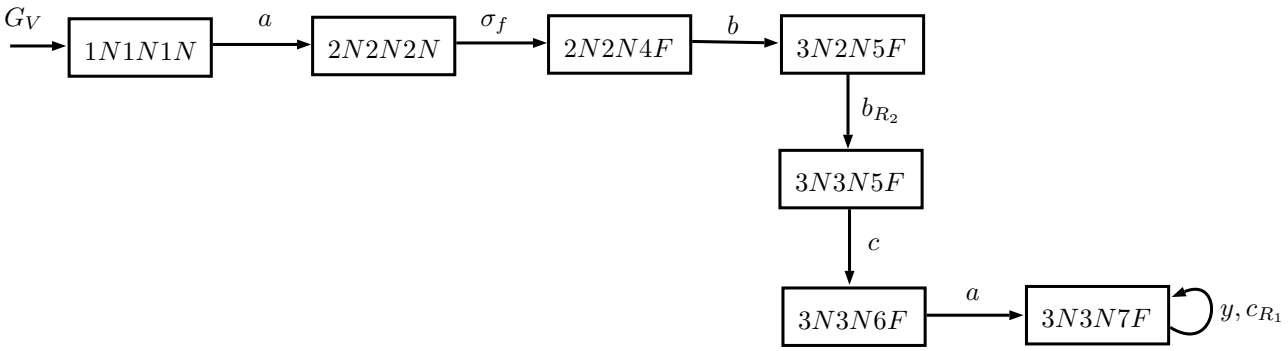


Figure 2.21: Verifier automaton G_V

of failure diagnosis of SED, as well as, an algorithm to verify the codiagnosability of a language have also been presented.

Chapter 3

Communication Networks Subject to Delays and Losses

The introduction of communication among computers had a deep influence in the organization of computer systems. The old model of a single computer serving all computational needs of a system has been replaced by the so-called networked computers, in which the tasks are performed by a large number of computers working separately but interconnected [60].

In spite of making the communication of large systems, communication networks have introduced problems such as communication delays and loss of information. Time delays come from computation time required for coding physical signals, communication processing and network traffic time, while losses of information come mainly from the limited memory in the devices, network traffic congestion in the network and drop out packets. Delays can be constant, random, bounded and depend on the network. Due to the asymmetry of delays, the performance of the system degrades, causing loss of data and out-of-order receptions [61, 62].

The problem of communication delay and losses of information in network communications is known in literature, and there are several works that address this problem [63, 64, 65], as well as the problem of out of order receptions [66, 67]. In this chapter, we present the main reasons for the existence of delays and losses. We also present a communication protocol, architectures and some specifications of communication networks.

This chapter is structured as follows. In Section 3.1, the structure of a network is

presented, and we describe the main communication protocols, the types of network and a few network devices. In Section 3.2, the main causes of delays, losses and out-of-order reception are presented. After that, in Section 3.3 we analyze some features of networks that contribute to problems such as delays and losses. Finally, in Section 3.4, we review the main concepts presented in this chapter.

3.1 Communication networks

According to [60], a computer network stands for a set of autonomous and interconnected computers. Two or more computers are interconnected if they exchange information through some physical medium, such as copper wires, optic fiber, etc. Communication networks arose from the necessity to interconnect computers, programmable logic controllers (PLC), and other industrial devices that are used independently. This network interconnection allows data sharing and provides a secure and accurate information to users.

Since the main application of communication networks is in industries, a robust communication system against typical characteristics of industrial environments, such as, hostile environments, electromagnetic interference, real-time characteristic and a safe and reliable network is necessary. Those aspects have motivated the ongoing research for new techniques and methods to establish such communications.

Usually, industrial plants are complex and distributed, and the amount of devices needed to interconnect them is large. Thus, in order to the communication among these devices to work properly, we need to use a standard model of network communication called OSI (Open System Interconnection) model. This model allows the standardization of protocols and, consequently, it improves the compatibility among devices of different systems, allowing the integration among components of the system. After we understand how the communication among devices is executed, it becomes easier to understand the main causes of delays and losses in the network communication and also the problem of out-of-order reception.

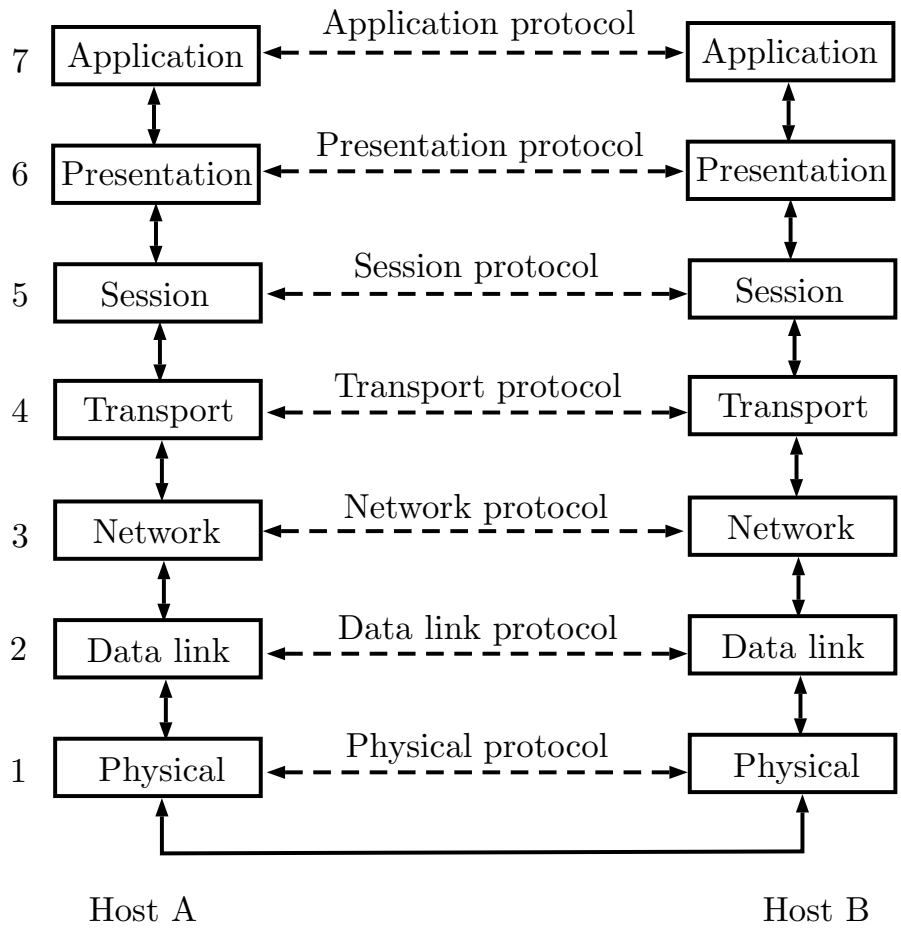


Figure 3.1: OSI model.

3.1.1 OSI model

OSI model, shown in Figure 3.1, is a multilayer architecture proposed by the International Standards Organization, where seven different layers are used to manage data communication in a network. This model is also called reference model because other communication architectures have been created based on the OSI model. Each layer has a defined goal and is described as follows, starting from the bottom layer of Figure 3.1 [60].

> **Layer 1 - Physical.** This layer describes the physical devices used in the transmission of raw signals (electrical, electromagnetic or lights). Cables (UTP, optic fiber, etc), repeaters and connectors belong to this layer. The network must ensure the correct transmission of data, *i.e.*, when a host sends bit 1, the destination host must also receive bit 1. This is ensured by the physical layer.

Layer 2 - Data link. The data link layer is responsible for the flow control (reception and transmission), for identifying the origin and destination of data through the Media Access Control address (MAC), and to correct possible errors occurred during the transmission by the physical medium. The physical devices that belong to Data link layer are: network board, bridge, access point, and switch. The protocols CSMA/CD, token passing (revisited in section 3.3), and several network technologies (Ethernet, Token ring, IEEE 802.11, etc) belong to this layer.

Layer 3 - Network. Devices and protocols for interconnecting several networks are placed in this layer. It is responsible for controlling the traffic of information in the network routing the messages from the origin to the destination.

Layer 4 - Transport. This layer provides methods and techniques that allow the communication between origin and destination, so that, if a message is sent by an origin host, then, it is received in the destination. In this layer, the messages are divided in packets (packet is a set of hundreds of bits) in order to send through the network to the destination. Each packet receive a control number in the moment of sending, and when they arrive in their destiny, the receiver can reordering the packets. The transport layer is also responsible for end-to-end communication, messages exchange between emitter and receiver, and corrections of some errors in packets.

Layer 5 - Session. The session layer allows users of different hosts to establish communication between them. A session provides several services, such as, communication control (it controls whom should transmit the message at each time), management of signals (avoiding that two devices try to execute the same operation at the same time), and synchronization.

Layer 6 - Presentation. This layer transforms messages of different formats in a single generic format in order to make possible the communication among different hosts. This kind of transformation includes procedures such as cryptography and compaction.

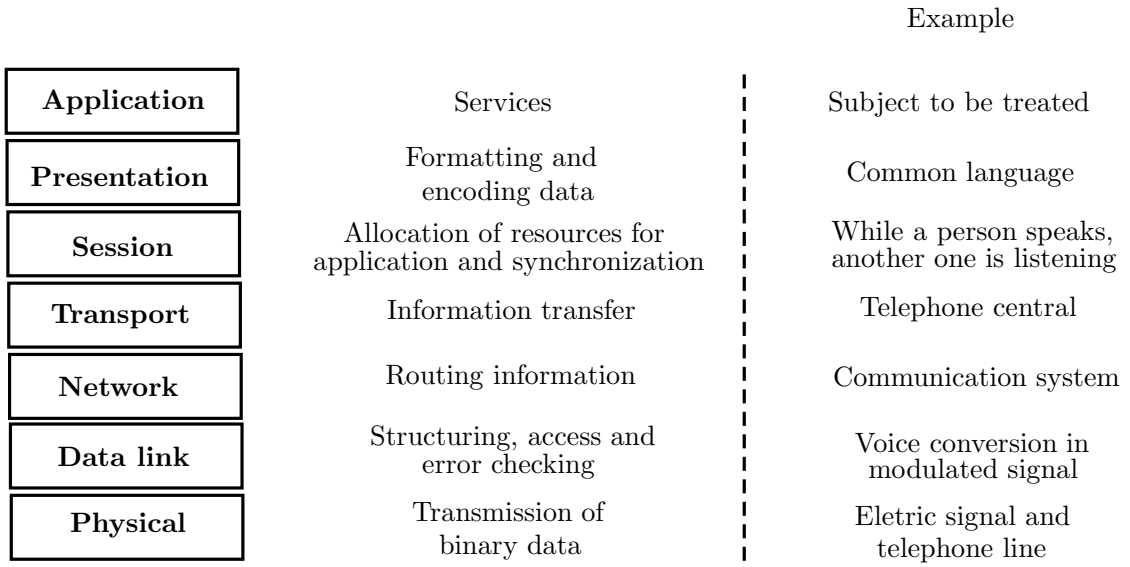


Figure 3.2: Resume and application of OSI model.

Layer 7 - Application. The application layer has protocols that execute user tasks such as transference of files. An application protocol widely used is the HTTP (HyperText Transfer Protocol), that is the base for the World Wide Web.

The description and an example of the characteristics of OSI model is shown in Figure 3.2.

In the communication network, there are two approach to transmit information: circuit switching and packet switching. The difference between them will be detailed in the next subsection.

3.1.2 Circuit switching and packet switching

In circuit switching networks, the necessary resources to provide the communication among devices (buffer, transmission rate, etc), are reserved and used during the period of transmission. This kind of network is also called deterministic network, where the information is transmitted in a given interval of time. Thus, the response time becomes known, and problems such as delays and losses can be avoided. In packet switching networks, the resources are not reserved, *i.e.*, the messages are transmitted when the resources are available. This kind of network is also called probabilistic network, where there is no knowledge about time of transmission, and

the information may even not be transmitted.

In order to better understand the difference between both approaches, consider the following analogy. Suppose two restaurants, one that requires reservations and another one that does not accept reservations. If we want go to the first restaurant, we need to make a reservation; however, in general, when we arrive there, we will be served immediately. In the other case, although, we do not need to make a reservation, when we arrive there, we may have to wait to be served or, if the restaurant is crowded, we will not be served at all. In a packet switching network, if a packet is sent and the data link is not available, it needs to wait in a buffer, and, thus, its transmission will be delayed. In a circuit switching network, when the devices need to communicate, a dedicated connection between them is formed and problems of delays and losses are avoided.

The devices of a network store and resend the packets. They need to store the whole packet before starting to resend the message to its destination. Thus, the devices present a storage and resending delay. In addition, each device has a buffer, also called queue, where packets that are ready to be resent are stored. The buffer is an important resource because if the data link is busy, the packet needs to wait there; thus, besides storage and resending delay, the packets are subject to queue delays in the buffer. Since the buffer is finite, it can be full when a packet is arriving. Thus, the packet will be discarded. Revisiting the analogy of restaurants, the queue delay is similar to time spent to be served by the waiter and the packet loss is similar to not being served because the restaurant is crowded [68].

The main devices that are in most of the communication networks and are responsible for delays and losses cited above are described in the next subsection.

3.1.3 Components of a network

In a distributed networked system, all devices must be connected in the network. In order to do so, several equipments are necessary, such as repeaters, bridges, routers and gateways.

- **Repeaters.** Devices that amplify signals in order to retransmit it to remote devices.

- **Bridges.** This kind of device connects two networks that have different electrical and protocols characteristics. For example, suppose that there is a network segment connected in bus topology (network topologies will be discussed in details in Section 3.3), and another segment connected in token ring topology. The direct connection between them is not possible because they have different rules and protocols, thus, they cannot communicate without a "interpreter". In this situation, the bridge translates the information from bus topology to token ring topology and vice-versa.

- **Routers.** Devices used to define a path in order to transmit information packet among different networks.

- **Gateway.** This device connects different networks, manages collisions, and translates protocols.

In a communication network, the information travels from the origin host to its destination crossing several devices such as those cited above. Thus, one of the main reasons of delays and losses in networks, is the time spent by the information to cross each device until it reaches its destination. In the next section we describe the main types of delays that occur in a network.

3.2 Delays and losses

In an ideal configuration, the communication network executes the information transference instantaneously and without losses. However, this condition is impossible in real networks since, in general, a network consists of several equipments, with different transmission rates, mediums of transmission, and topologies. In addition, complex plants usually have a distributed architecture, and so, their devices are placed far from each other. These characteristics contribute effectively to generate delays and losses of information in the communication among devices.

In a communication network, the information is divided into packets, and then, these packets are transmitted to the destination node. When a packet is transmitted by a node, it passes through several devices, *e.g.* routers, bridges, hubs or gateways. Along the way, the packet is subject to several types of delays and losses. The main

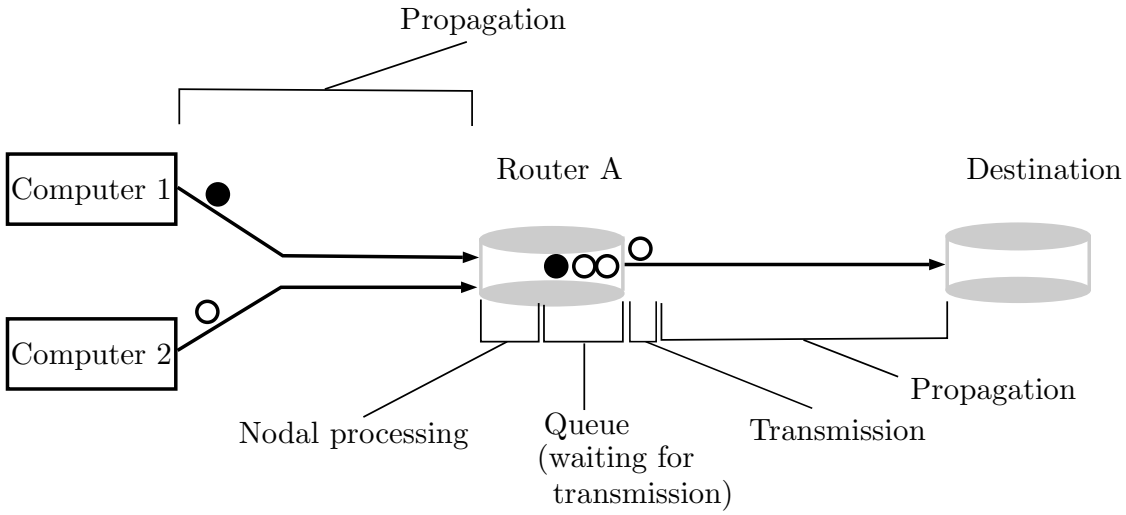


Figure 3.3: Types of delays of a network communication.

types of delays are: nodal processing delay, queue delay, transmission delay and propagation delay, and the losses can occur because of collisions in the data link or because the buffers of devices are crowded [68].

In order to facilitate the understanding of the main types of delays generated in a network, let us consider the scheme of a small network shown in Figure 3.3, which represents the communication of packets sent by computers 1 and 2 through router A, until the final destination.

In a communication network, when a packet is sent by the computer, it takes some time to arrive at the router. After that, the router identifies the packet destination and, then, the packet is sent to its destination. Notice that, during this procedure, a queue is formed at router A. The transmission of a packet is possible only if the physical medium is available, and if the packet is the first element of the queue. Otherwise, the packet that arrives in the router is added to the end of the queue. We describe, in the sequel, the possible causes of communication delays in the network.

- **Processing delay.** It is the time that the router takes to identify the packet destination and to effectively send it. The time to verify possible errors in the packets are also included in the processing delay. The processing delay in high speed devices are in the order of milliseconds.

- **Queue delay.** The queue delay occurs when a packet waits in the queue to be

transmitted, and depends on the length of the queue in the device. If the queue is empty, and there is no packet being transmitted at that moment, then the packet is transmitted immediately, and, consequently, the queue delay will be zero. Otherwise, the packet will wait in the queue, until it can be transmitted. Queue delays are in the order of milliseconds.

- **Transmission delay.** The transmission delay is the time spent by a router to send the whole packet. Suppose that a packet with K bits is sent with a transmission speed of T bits/s. Thus, the network will finish transmitting this packet after K/T seconds. Typically, the transmission delays are in the order of micro or milliseconds depending on the length of the packets.

- **Propagation delay.** After a packet is sent through a network, it propagates by the physical medium until it arrives at the destination. The packet propagation takes a certain amount of time, and it is called propagation delay. The speed of propagation is related to the physical medium where the packet propagates, and thus, the propagation delay is different according to physical medium and the distance that the devices are placed. The propagation delay is in the order of milliseconds.

The difference among the types of delays generated in a communication network can be understood as follows. The transmission delay is the amount of time that a device needs to send the whole packet. This time depends on the length of packet that will be transmitted and on the transmission rate of the network, while the propagation delay is the time that a bit of a packet takes to propagate from a device to another one. This delay depends on the distance between the devices and the rate of transmission.

In order to understand better the difference between transmission and propagation delay, consider the following analogy. Suppose a highway with some toll booths and a convoy of cars crossing them as shown in Figure 3.4. Now, imagine that the toll booths are routers and the part of highway between toll booths is a data link. Suppose now that, the convoy of cars is a packet and a car of the convoy is a bit. Thus, when the convoy arrives at the toll booth, each car needs to cross it; thus, the convoy will spend a certain amount of time to completely cross the toll . This

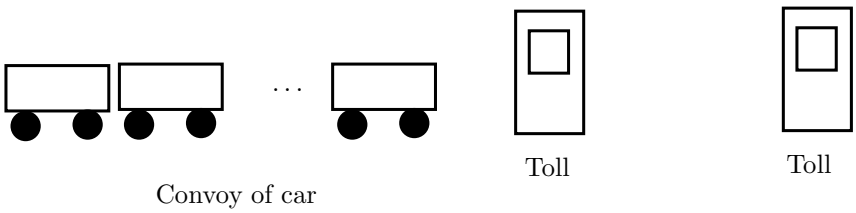


Figure 3.4: Convoy analogy.

time is analog to the transmission delay in a router. After the car crosses the toll, it goes to the next toll booth. The time spent by the car to reach the next toll booth is analog to the propagation delay.

The queue delay depends on the length of the queue. For example, if several packets arrive in an empty queue, the first packet will be transmitted without delay, while the last one will be subject to delay. The length of the queue depends on the speed, rate and nature of transmission in data link. The nature of transmission means that the packets can be sent in different ways: intermittent or continuous. Let us revisit the analogy represented in Figure 3.4. If several convoys arrive in the toll booth, they will form a queue to cross the toll booth. Thus, the queue delay of the last convoy of the queue is longer than the first one. Another factor that contributes to the queue delay is the relationship between the speed of arrivals and departures of packets in routers, so that, if the speed of arrival is greater than the speed of departure, the queue delay will increase. However, the capacity of a queue is limited, *i.e.*, if a packet arrives in a router and it finds the queue full, the router will discard it. In other words, the packet will be lost. Another reason for the loss of packets is the possibility of packets collisions in the network. Revisiting the analogy of car convoy again, suppose that two or more cars of a convoy are involved in a hit; thus, this convoy will not be able to reach its destination.

The number of lost packets in a network is related with the intensity of data traffic. Thus, a measure of the performance of a network can be obtained by using the probability of packet loss. It is important to remark that, in some protocols, a lost packet can be retransmitted in order to guarantee that all information is transferred to the destination.

The problem of communication delays and losses is an important aspect of a communication network. One of the consequences of communication delays in a

network, is that the packets can be received in an order different from the order they were originally sent. In order to understand better this situation, consider two convoys that depart from different places to the same destination. Assume that a convoy departed first, but the highway where it travels has many toll booths and congestion, and the highway of the second convoy has fewer toll booths and is free. Thus, even when the distance from origin to destination of the second convoy is larger, the second convoy may arrive at the destination first before the first convoy. Due to it, there is a lot of investment in researches and new techniques to solve these problems.

A complex communication network is, usually, composed of several subnetworks with different characteristics, such as, transmission rate, topologies, transmission medium, communication technology and access algorithms, and these specifications contribute to problems such as delays and losses of information. In the next section, these specifications are described.

3.3 Specification of a communication network

Specification of a communication network is a set of parameters used in order to the communication network work correctly. Typically, these parameters are: transmission rate, physical network topology, physical transmission medium, communication technology and network access algorithms [69]. We explain in the sequel each one of these parameters.

- **transmission rate.** The transmission rate is the amount of information that a communication channel or a device can transmit in a period of time. It is also called *throughput*, and can be measured in kilobits per second (kbps) or bits per second (bps). This parameter depends on the transmission medium and the devices of the network. Let us consider a communication between two devices and suppose that device 1 sends a message to device 2 and that the transmission rate of device 1 is greater than the transmission rate of device 2. As a consequence, device 2 will receive the information faster than it can transmit. Then, this situation can lead to data congestion in the network and the buffer of device 2 can become full, which can make the information be

delayed or lost.

- **Physical network topology.** Several systems consist of many devices interconnected. In order to connect a lot of devices in a network, the notion of topology must be introduced. Topology is a layout that determines how the devices will be interconnected. It can be described as physical or logic. Physical topology is the real connection among devices, *i.e.*, how wires and devices are connected, whereas logic topology is related to the information flow through the network. The most common topologies are: peer-to-peer, bus, ring, star, and tree. Each of these topologies is detailed below.

  Peer-to-peer. This topology provides communication between two or more devices which can be used as routers in order to improve the communication. Peer-to-peer topology is commonly used in temporary connection, for example, the communication between a computer and a PLC in order to execute some temporary activities. Peer-to-peer topology is illustrated in Figure 3.5.

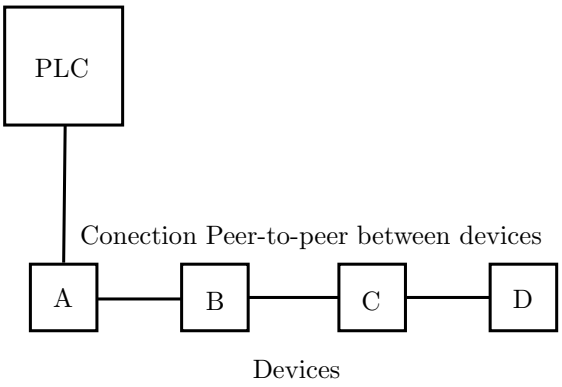


Figure 3.5: Peer-to-peer topology.

  Bus. In bus topology, the physical communication medium is shared by all peripheral devices and controllers. It is widely used in industrial plants due to its capacity of expansion. Another characteristic of bus topology is that it works by diffusion (broadcast), *i.e.*, a message sent by a computer arrives in all devices in the network. On the other hand, the disadvantage of this topology is that, if a large number of computers are connected in the network, its performance (velocity, collision, traffic) is

deteriorated. The bus topology is illustrated in Figure 3.6.

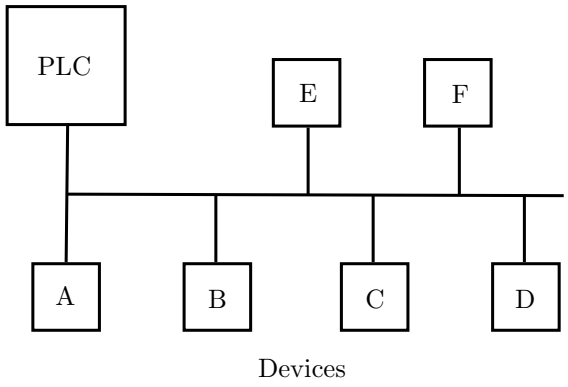


Figure 3.6: Bus topology.

Ring. The ring topology is similar to the peer-to-peer architecture where the devices are connected in a cyclic path, with the last device connected to the first one. The information passes through the devices in the ring until it reaches its destination. A disadvantage of this topology is that, if we want to introduce another device in the network or if a failure in one of the nodes occurs, the whole connection is interrupted and the communication is stopped. Although the ring topology is more reliable than the peer-to-peer topology, it has low capacity of expansion because of the increase in transmission delay. This topology is shown in Figure 3.7.

Star. In star topology, the devices are connected by a central equipment that is capable of managing the communications between them. If a failure occurs in the central node, the whole network will be affected and the communication will stop. Otherwise, if a failure occurs in some peripheral nodes of the network, it will not affect the whole network. This topology is illustrated in Figure 3.8.

Tree. The tree topology is an architecture where devices are organized in a hierarchical way. In this topology, there is only one path connecting the nodes, thus, if any connection is lost, then, the communication is interrupted. Industrial systems are examples of networks in tree structure. This topology is shown in Figure 3.9.

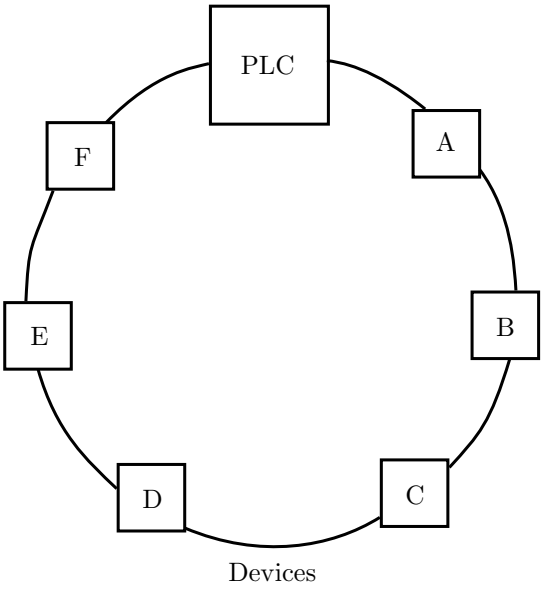


Figure 3.7: Ring topology.

- **Physical transmission medium.** There are two types of communication mediums: physical wire and wireless technology. The physical wires can be divided in two kind of wires: copper wire or optic fiber. Cooper wires can be coaxial or twisted pair cables, where the signal is transmitted by electric impulses while in optic fiber the signal is transmitted by light signals. A coaxial cable is constructed with two concentric cooper wires and a special shield that provides isolation from the environment. Thus, this wire supports electromagnetic interferences [69]. The twisted pair cables are made by four pair of copper wire twisted with each other. They are twisted in order to reduce electromagnetic interferences from similar cables around them and are widely used in several kinds of communication networks. In general, twisted pairs are cheaper than coaxial cable and have low cost of maintenance. The optic fiber is a cable that transmits light, not electrical signal; thus, it is not subject to electromagnetic interferences. However, optic fiber cables are very expansive due to their complicated manufacturing process. In order to transmit information, an optic transmitter is used in the origin host to convert the electrical signal to light one, that is sent through optic fiber until a receiver that executes the inverse process. Usually, a fiber optic cable consists of a pair of fibers: (i) a fiber that transmit the information in a way and, (ii) another one that transmits the information in opposite way.

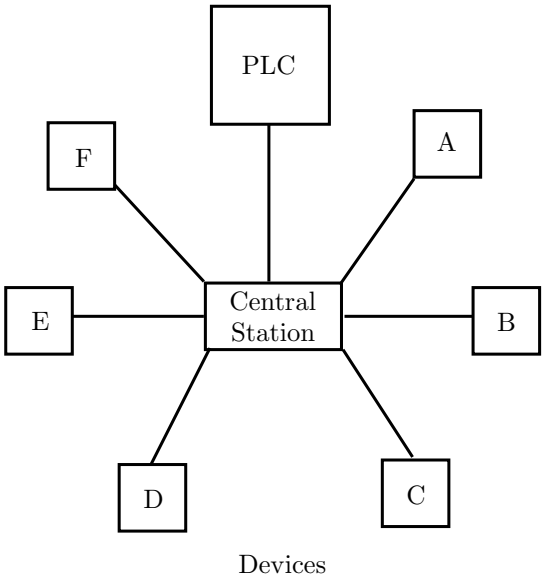


Figure 3.8: Star topology.

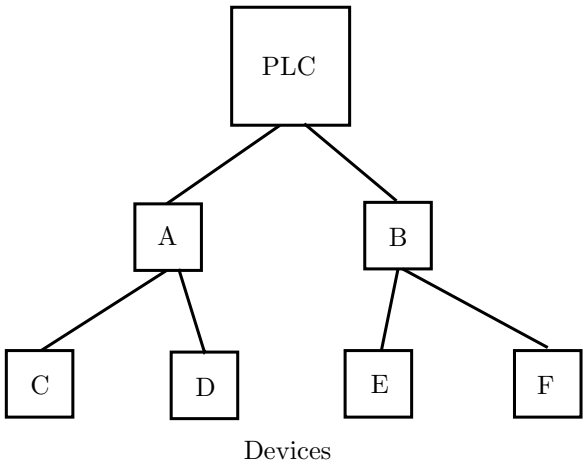


Figure 3.9: Tree topology.

Wireless communications are good alternatives in places that it is not feasible to implement copper cables or optic fibers. The transmission medium used to transmit signals using wireless network are known as infrared, microwaves and radio frequency.

The transmission medium is an important aspect in a network and must be taken into account when we want to work with delays and losses. In complex networked systems with several environments and several types of signals, it is natural that the network is composed by many types of transmission medium. Thus, it contributes for the increasing of delays and losses in a network.

The communication physical medium can be used in three different ways: simplex, half-duplex and full-duplex.

Simplex. The data link is used only by one of two directions of transmission, as shown in Figure 3.10.

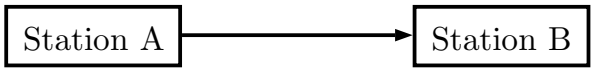


Figure 3.10: Simplex mode

Half-duplex. The data link is used in both directions, but not simultaneously. It is illustrated in Figure 3.11.

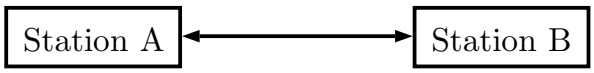


Figure 3.11: Half−duplex mode

Full-duplex. The data link is used in both directions simultaneously. It is illustrated in Figure 3.12.

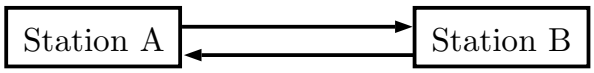


Figure 3.12: Full−duplex mode

- **Communication technology.** The communication technology is responsible for the management of the communication between devices in the network (nodes) [69].

  Master/Slave. In a master/slave system, there is a station called master that has the right of transmission. The master has the capacity of passing the transmission rights to the slave stations. Thus, in this configuration, the system becomes dependent on the central station. Slaves are simple devices (input/output devices, valves, transducers, etc.), that receive information directly from the master. A slave is connected only with its master, so that, it responds only to direct requests coming from

a determined master. In order to understand better the master/slave technology, let us consider a classroom where a teacher is explaining a determined lesson. Suppose that, at some point, a student has a question. Then, when the teacher finishes his explanation, he gives the right of speaking to the student. When the student finishes, the right of speaking returns to the teacher. In this analogy, the teacher represents a master device and the student represents the slave. The master/slave structure can be divided in single-master or multi-master.

Single-master. In this structure, there is only one master in the system. In general, a CPU or a PLC are central devices that work as masters, while slaves are connected in a decentralized way in the system.

Multi-master. There is more than one master in this structure, so that, all masters can read the information of the system. It is important to remark that a slave can only be controlled by one master.

Producer/consumer: In this technology, there is an origin and a destination identifier in the information which allows the data to reach the specified destination. In addition, all nodes of the network can be synchronized. Thus, multiple nodes (producers) can transmit data to other nodes (consumers), and a node that is a producer, in a specific moment, can assume the role of consumer in another moment.

- **Network access algorithms.** The access algorithms are used by devices in order to access the information from the network or to send some message. The most used algorithms to access the network are CSMA/CD and Token passing [69].

  CSMA/CD (Carrier Sense Multiple Access/Colision Detection). The algorithm CSMA/CD is a media access control that a device transmits when it detects an available data link. When two or more devices try to transmit at the same time, a collision can occur. When a collision occurs, the devices receive an information that the data did not arrive at its destination, and then, the transmission is interrupted. After

that, each device waits for a random time interval before resending the information. This access algorithm tries to guarantee that the message will arrive at its destination by resending when necessary; however the retransmission is aborted if there are numerous collisions and the information does not arrive to its destination. Therefore, although CSMA/CS decreases losses of packets, it does not avoid losses of information completely. In addition, the CSMA/CD increases the communication delays and the possibility of out-of-order reception due to retransmission time.

Token Passing. Token passing algorithm is used, in general, in ring topology. This algorithm indicates the direction that the token circulates and, in case a device wants to transmit an information, it has to pick up this token and replace it by the message. Then, after finishing the transmission, the device recovers the token, thus allowing another device to access the network. This access algorithm eliminates the possibility of collisions, however, a station wishing to transmit must wait for the token, increasing the communication delay. Since the token needs to pass through every device of the ring in a cyclic way, the messages can be sent out of its original order of occurrence.

3.4 Final remarks

The main goal of this chapter is to present the main reasons of delays and losses in communication networks. In order to do so, we presented the basic structure of networks used for establish the communication between several devices of a system. In addition, we made a detailed explanation of the main types of delays and losses, as well as, the network characteristics that contribute for these problems, which is the central problem addressed in this work.

Chapter 4

Codiagnosability of Networked Discrete Event Systems

After we have seen the basic concepts of discrete event systems theory and the main reasons for the occurrence of delays and losses in communication networks in the previous chapters, we will address now the problem of failure diagnosis of networked DES subject to delays and losses of event observation. An important consequence of communication delays to diagnosis systems is the possibility of reception of data by diagnosers in a different order from its original order of occurrence in the plant.

In order to analyze the network codiagnosability of discrete event systems, this chapter is divided as follows: in Section 4.1, we formulate the problem of decentralized diagnosis of networked DES subject to delays and losses of event observation. In Section 4.2, we present an algorithm to obtain the automaton that models all possible delays in the communication of events to local diagnosers, and prove the correctness of this algorithm. In Section 4.3, we review the problem of intermittent loss of observation, in particular, the Dilation operation, which is the main tool to model intermittent losses of event observation. The final model of a plant subject to delays and losses of events is presented in Section 4.4. In Section 4.5, we introduce the definition of network codiagnosability. In Section 4.6, we propose an algorithm for the verification of the network codiagnosability of a language generated by a DES and based on the verification automaton, we present a necessary and sufficient condition for network codiagnosability. The complexity of the algorithm for verification of the network codiagnosability is presented in Section 4.7. Finally, in Section

4.8, remarks are presented in order to summarize the contributions of this chapter.

4.1 Problem formulation

In general, different sensors in distributed systems do not share the same communication channel. This is so because, either the measurement sites are far away from each other, or a single communication channel may not have enough capacity to transmit all data from a measurement site to a local diagnoser. Thus, the implementation of several communication channels between measurement sites and diagnosers is, in general, necessary in network-controlled systems.

In this work, we introduce a network decentralized diagnosis scheme for a distributed plant with different measurement sites MS_j, $j = 1, \ldots, m$, where each measurement site MS_j reads the signals associated with a subset $\Sigma_{MS_j} \subset \Sigma_o$ of the observable events of the system. In this scheme, events of Σ_{MS_j} are communicated to a local diagnoser LD_i, $i = 1, 2, \ldots, n$, by an exclusive communication channel ch_{ij}, *i.e.*, only the events detected by measurement site MS_j can be communicated through channel ch_{ij} between measurement site MS_j and local diagnoser LD_i. Let us denote the set of events communicated to the local diagnoser LD_i, through communication channel ch_{ij}, as $\Sigma_{o_{ij}} \subseteq \Sigma_{MS_j}$. It is important to remark that if the communication channel ch_{yx}, between a measurement site MS_x and a local diagnoser LD_y, does not exist, then $\Sigma_{o_{yx}} = \emptyset$. Thus, the set of observable events of LD_i, Σ_{o_i}, is given by:

$$\Sigma_{o_i} = \bigcup_{j=1}^{m} \Sigma_{o_{ij}}. \tag{4.1}$$

It is important to notice that $\Sigma_o = \bigcup_{i=1}^{n} \Sigma_{o_i}$. In Figure 4.1, we show the network decentralized diagnosis scheme proposed in this thesis for a distributed plant with four measurement sites and two local diagnosers. Notice that measurement site MS_1 is capable of communicating to local diagnoser LD_1 through channel ch_{11} only the events in $\Sigma_{o_{11}} \subseteq \Sigma_{MS_1}$, and that measurement site MS_3 communicates the events in $\Sigma_{o_{13}} \subseteq \Sigma_{MS_3}$ and $\Sigma_{o_{23}} \subseteq \Sigma_{MS_3}$ to local diagnosers LD_1 and LD_2, respectively, through communication channels ch_{13} and ch_{23}. It is important to remark that in the network architecture proposed in this work, a measurement site can transmit a different set of observable events for different local diagnosers, which

implies that, in the example depicted in Figure 4.1, $\Sigma_{o_{13}}$ can be different from $\Sigma_{o_{23}}$. The communication between measurement sites and local diagnosers through

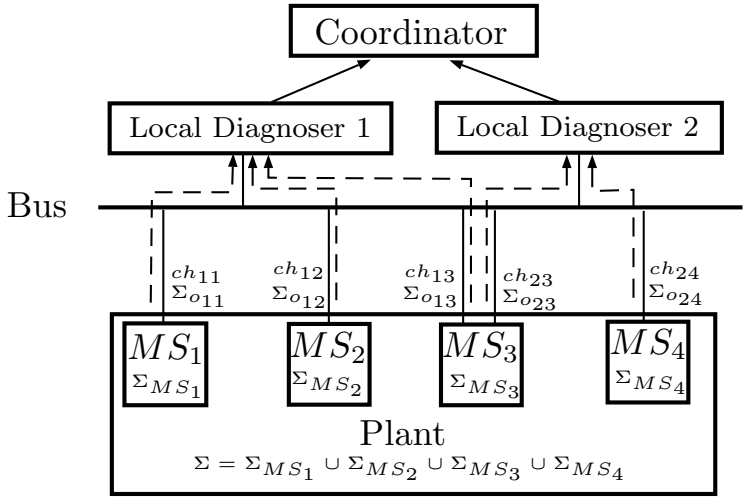


Figure 4.1: Network decentralized diagnosis architecture.

a communication network can introduce two problems for the failure diagnosis as follows: (i) delay in the communication of an event occurrence to a local diagnoser; and (ii) loss of data transmitted through communication channels. When either one of the situations above occurs, the diagnoser may send a wrong diagnosis decision to the coordinator, and then, the implemented diagnosis scheme is no longer reliable.

Regarding event communication delays, we make the following assumptions:

A1. The delay in the communication of an event $\sigma \in \Sigma_o$ is measured by steps [48], where one step is the occurrence of an event, *i.e.*, the delay is measured by the number of events that are executed by the plant after the occurrence of σ and before its observation by a local diagnoser.

A2. The event communication delays are bounded.

A3. The communication channels follow first-in first-out (FIFO) rule as far as sending and reception are concerned.

A4. There is one and only one channel ch_{ij} between measurement site MS_j and local diagnoser LD_i, and the maximum communication delay of channel ch_{ij}, denoted by k_{ij}, is previously known. If a channel ch_{yx} does not exist, then by convention, $k_{yx} = 0$.

Assumptions **A1**–**A4** can be justified with the help of Chapter 3, as follows. The information travels by the network and it is subject to different types of delays. The total delay is the sum of all delays that the information is subject. However, since the system is modeled by non timed automata, time counting becomes complicated. Thus, we define delays in the classical way in Assumption **A1**. As we saw in Chapter 3, delays are finite, thus, we assume that the delays are bounded as defined in Assumption **A2**, infinite delays are accounted for assuming by the loss of packet.

When a device sends a message, a communication protocol is used. Usually, protocols like OSI guarantee the correct order of reception between origin and destination; thus, Assumption **A3**.

Assumption **A4** considers that the network has been tested exhaustively in order to find the time of the delay. Besides that, based on the concepts of network, we consider that there is only one channel connecting a MS_j to LD_i.

A5. The event sets Σ_{MS_i} and Σ_{MS_j} are disjoint for all $i, j \in \{1, 2, \ldots, m\}$, $i \neq j$.

Regarding loss of data in communication channels, we make the following assumption:

A6. The loss of observation of events occur in the communication channels that connect measurement sites and local diagnosers.

Therefore, according to assumption **A6**, the loss of observation of an event does not change the plant behavior, but only the observation.

4.2 Model of the plant subject to communication delays

In the system structure shown in Figure 4.1, data transmitted by a communication channel, ch_{ip}, can delay with respect to another communication channel ch_{iq}, where $p \neq q$ and $p, q \in \{1, 2, \ldots, j\}$. As a consequence, in the communication of the events to the local diagnoser LD_i, they can be observed in an order different from their actual occurrence in the system. Thus, in order to address the problem of failure diagnosis in networked DES with communication delays, it is necessary to construct

automata G_i, $i = 1, 2, \ldots, n$, that represent all possible ordering of observation by the local diagnosers LD_i of the traces executed by the plant.

To distinguish an event $\sigma \in \Sigma_{o_{ij}}$, that is transmitted from measurement site MS_j to the local diagnoser LD_i through communication channel ch_{ij}, from its observation by the local diagnoser LD_i, we create an event σ_{s_i}, *i.e.*, the transition labeled with σ represents the occurrence of event σ in the plant, while that, transitions labeled with σ_{s_i} represent the success of observation of event σ by local diagnoser i. In this regard, let

$$\Sigma^s_{o_{ij}} = \{\sigma_{s_i} : \sigma \in \Sigma_{o_{ij}}\}, \tag{4.2}$$

denote the set of events that are observable by a local diagnoser LD_i and correspond to those observable events whose occurrence are recorded at MS_j and are communicated through channel ch_{ij}, and let

$$\Sigma^s_{o_i} = \bigcup_{j=1}^{m} \Sigma^s_{o_{ij}}, \tag{4.3}$$

denote the set of observable events that are successfully communicated to local diagnoser LD_i. Then, the following bijective function can be defined:

$$\begin{aligned} \phi_i : \quad & \Sigma^s_{o_i} \to \Sigma_{o_i}, \\ & \sigma_{s_i} \mapsto \phi_i(\sigma_{s_i}) = \sigma \end{aligned} \tag{4.4}$$

The definition of ϕ_i can be extended to sets of events as

$$\phi_i(\Sigma^s_{o_i}) = \bigcup_{\sigma_{s_i} \in \Sigma^s_{o_i}} \phi_i(\sigma_{s_i}). \tag{4.5}$$

In order to obtain an algorithm for the computation of the automaton models G_i, $i = 1, \ldots, n$, it is necessary to obtain automata D_i, $i = 1, \ldots, n$, that model all possible delays in the communication of the events to local diagnoser LD_i, according to the delay bound of each communication channel that transmits the occurrence of events to LD_i.

Before we present the algorithm for the computation of D_i, it is important to present the following definition.

Definition 4.1 *Let $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$. Define $\Sigma_{o\nu} = \Sigma_o \cup \{\nu\}$ and the set of states Q, where each state $q \in Q$ is labeled with a trace $s \in \Sigma^\star_{o\nu}$. Then, the following functions can be defined:*

a. The replacement function rep is defined as:

$$rep : Q \times \mathbb{N} \rightarrow Q$$

where for all $q = q_1 q_2 ... q_\ell \in Q$,

$$rep(q, i) = \begin{cases} q_1 q_2 ... q_{i-1} \nu q_{i+1} ... q_\ell, \text{ if } i \leq \ell \\ \text{undefined, otherwise.} \end{cases}$$

b. The elimination function cut is defined as:

$$cut : Q \rightarrow Q$$

where for all $q = q_1 q_2 ... q_\ell \in Q$,

$$cut(q) = \begin{cases} q_i q_{i+1} ... q_\ell, \text{ if } (\exists i \leq \ell)[(q_i \neq \nu) \wedge (q_k = \nu, \forall k \in \{1, 2, ..., i-1\})] \\ \nu, \text{ if } q_k = \nu, \forall k \in \{1, 2, \ldots, \ell\}. \end{cases}$$

c. The measurement site index function ms is defined as:

$$ms : \Sigma_{o\nu} \rightarrow \{1, 2, ..., m\}$$

where for all $\sigma \in \Sigma_{o\nu}$,

$$ms(\sigma) = \begin{cases} j : \text{ if } \sigma \in \Sigma_{o_{ij}} \text{ for some } i \in \{1, 2, ..., n\} \\ \text{undefined, otherwise.} \end{cases}$$

□

According to Definition 4.1, function $rep(q, i)$ replaces the *i-th* element of state q with element ν. Function $cut(q)$ eliminates the largest prefix of state q formed only by elements ν, and function $ms(\sigma)$ returns the index j which corresponds to the measurement site (MS_j) that detects the occurrence of event σ.

Algorithm 4.1 describes the construction of automaton D_i, associated with local diagnoser LD_i, that models all possible delays in the communication of events to LD_i, from measurement site MS_j, $j = 1, 2, \ldots, m$. Automaton D_i will be referred to as the communication delay model.

Notice that, Algorithm 4.1 can be divided in three parts: (i) initialization of automaton D_i, Steps $1 - 4.2(b)$, where we define the initial state and the associated transition functions; (ii) checking of how many events can occur in the plant, with

respect to communication delay k_{ij}, before one of them is observed, Steps 4.2(c) and 4.2(e), and; (iii) modeling of observation of the events by LD_i, Steps 4.2(g) and 4.2(h). The correctness of Algorithm 4.1 will be ensured by Lemma 4.2.

Algorithm 4.1 *Construction of automaton D_i (Communication delay model)*

Input: m, n, $\Sigma_{o_{ij}}$, k_{ij}, *for* $i = 1, \ldots, n$, $j = 1, \ldots, m$.

Output: $D_i = (Q_i, \Sigma_i, \delta_i, \Lambda_i, q_{0_i})$, $i = 1, \ldots, n$.

For $i = 1, 2, \ldots, n$

1: Define $u_0 = \nu$ *and* $Q_i = \emptyset$.

2: Construct $\Sigma^s_{o_i}$ *according to Equations* (4.2) *and* (4.3), *and define* $\Sigma_i = \Sigma \cup \Sigma^s_{o_i}$.

*3: Create a FIFO (*first in-first out*) queue* F *and add* u_0 *to* F.

4: While $F \neq \emptyset$ *do*

4.1: $u \leftarrow F_1$, *where* F_1 *is the first element of the queue.*

4.2: If $u = u_0$

- *For all* $\sigma \in \Sigma$, *compute:*

  (a) $\tilde{q} = \delta_i(u, \sigma) = \begin{cases} \sigma, & \text{if } \sigma \in \Sigma_{o_i}; \\ u, & \text{if } \sigma \in \Sigma_{uo_i}. \end{cases}$

  (b) If $\tilde{q} \neq u$, *add* $\tilde{q}$ *to the end of the queue* F
- $Q_i \leftarrow Q_i \cup \{u\}$
- *Define* $q_{0_i} = \nu$
- *Remove element* F_1 *from the queue* F

Else

- *Set* $\ell = \|u\|$ *and form set* $I_\ell = \{1, 2, \ldots, \ell\}$
- *Denote* $u = \sigma_1 \sigma_2 \ldots \sigma_\ell$ *and compute* $I_\nu = \{y \in I_\ell : (\exists \sigma_y \in u)[\sigma_y = \nu]\}$
- *Compute* $I_{\ell \setminus \nu} = I_\ell \setminus I_\nu$
- *For each* $\sigma \in \Sigma_{o_i}$:

  (c) $\tilde{q} = \delta_i(u, \sigma) = \begin{cases} u\sigma, & \text{if } \|\sigma_y \sigma_{y+1} \ldots \sigma_\ell\| \leq k_{i,ms(\sigma_y)}, \forall y \in I_{\ell \setminus \nu} \\ \text{undefined, otherwise} \end{cases}$

(d) Add $\tilde{q}$ to the end of the queue F if $\tilde{q}$ is defined

- *For each $\sigma \in \Sigma_{uo_i}$:*

  (e) $\tilde{q} = \delta_i(u, \sigma) = \begin{cases} u\nu, \text{ if } \|\sigma_y \sigma_{y+1} ... \sigma_\ell\| \leq k_{i,ms(\sigma_y)}, \forall y \in I_{\ell \setminus \nu} \\ \text{undefined, otherwise} \end{cases}$

  (f) If $\tilde{q} \notin F$, add $\tilde{q}$ to the end of the queue F

- *For each $\Sigma^s_{o_{ij}}$, where $j = 1, 2, \ldots, m$:*

  (g) Form set $Y = \{y : (\sigma_y \in u) \wedge (\sigma_y \in \phi_i(\Sigma^s_{o_{ij}})\}$

  (h) If $Y \neq \emptyset$ - Compute $\hat{y} = min(Y)$ - $\tilde{q} = \delta_i(u, \phi_i^{-1}(\sigma_{\hat{y}})) = cut(rep(u, \hat{y}))$

  (i) If $(\tilde{q} \notin Q_i) \wedge (\tilde{q} \notin F)$, add $\tilde{q}$ to the end of the queue F.

- *Set $Q_i \leftarrow Q_i \cup \{u\}$.*

- *Remove element F_1 from the queue F.*

5: For each $q_i \in Q_i$, $\Lambda_i(q_i) = \{\sigma \in \Sigma_i : \delta_i(q_i, \sigma)!\}$

Before we present the correctness proof of Algorithm 4.1, we will first give an example to illustrate the construction of automaton D_i according to Algorithm 4.1.

Example 4.1 *Let G, depicted in Figure 4.2, be the automaton model of a distributed system where $\Sigma = \{a, b, c, e, \sigma_f\}$, and consider the network decentralized diagnosis scheme depicted in Figure 4.3, which consists of two local diagnosers, LD_1 and LD_2, and three measurement sites, MS_1, MS_2 and MS_3. Let $\Sigma_{MS_1} = \{a\}$, $\Sigma_{MS_2} = \{c\}$ and $\Sigma_{MS_3} = \{b, e\}$, be the sets of events that the measurement sites MS_1, MS_2 and MS_3, respectively, record. Assume that the set of observable events of the local diagnoser LD_1 is $\Sigma_{o_1} = \{a, c\}$. Thus, the occurrences of the events in Σ_{o_1} are transmitted through communication channels ch_{11} and ch_{12}, which implies that, $\Sigma_{o_{11}} = \{a\}$ and $\Sigma_{o_{12}} = \{c\}$. Assume now that the set of observable events of LD_2 is $\Sigma_{o_2} = \{b, c, e\}$. Thus, the occurrences of the events in Σ_{o_2} are communicated through channels ch_{22} and ch_{23}, and thus, $\Sigma_{o_{22}} = \{c\}$ and $\Sigma_{o_{23}} = \{b, e\}$. Let σ_f be the failure event, and assume that the delay bounds of the communication channels are $k_{11} = k_{23} = 1$ and $k_{12} = k_{22} = 0$.*

Notice that automaton G generates failure traces, $s_{F_1} = \sigma_f bc^n$ and $s_{F_2} = \sigma_f abec^n$, and normal trace, $s_N = bac^n$, where $n \in \mathbb{N}$. Since the sets of observable events of

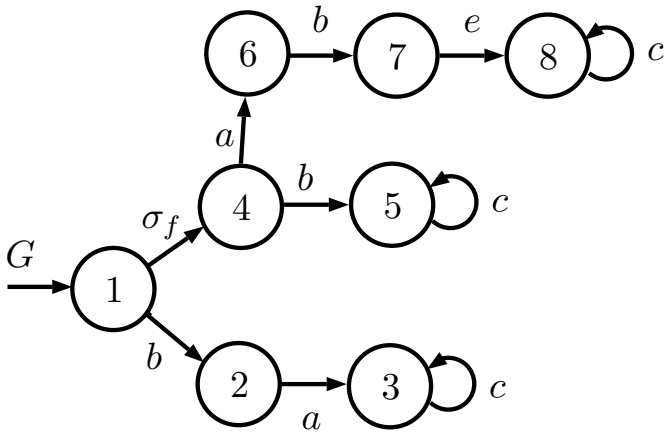


Figure 4.2: Automaton G

*LD*1 *and LD*2 *are* $\Sigma_{o_1} = \{a, c\}$ *and* $\Sigma_{o_2} = \{b, c, e\}$*, respectively, and assuming that the system works perfectly, i.e, there is neither observation delays nor losses of events, the traces observed by* LD_1 *are* $P_{o_1}(s_N) = P_{o_1}(s_{F_2}) = ac^n$ *and* $P_{o_1}(s_{F_1}) = c^n$ *and the traces observed by* LD_2 *are* $P_{o_2}(s_N) = P_{o_2}(s_{F_1}) = bc^n$ *and* $P_{o_2}(s_{F_2}) = bec^n$*. This implies that none of the local diagnosers can diagnose* $L(G)$ *alone. However, since* LD_1 *diagnosis trace* s_{F_1}*, and* LD_2 *diagnosis trace* s_{F_2}*, we conclude that the system is codiagnosable.*

Assume now that the system is subject to communication delays. Thus, in order to consider the observation delays associated with LD_1 *we will construct automaton* D_1 *by following Algorithm 4.1 step-by-step. In the first step, variable* u_0 *is defined as* ν *and the set of states* Q_1 *is defined as the empty set. In the second step, sets* $\Sigma^s_{o_1} = \{a_{s_1}, c_{s_1}\}$ *and* $\Sigma_1 = \{a, b, c, e, \sigma_f, a_{s_1}, c_{s_1}\}$ *are formed. In Step* 3*, queue* F *is created and state* $u_0 = \nu$ *is added to* F*. While queue* F *is not empty, the first element of* F *is assigned to variable* u *according to Step* 4.1*, and, in Step* 4.2 *since* $u = \nu$*, transitions from* ν *will be defined for all* $\sigma \in \Sigma$*, as follows:* $\delta_1(\nu, a) = a, \delta_1(\nu, c) = c$ *and* $\delta_1(\nu, \sigma_f) = \delta_1(\nu, b) = \delta_1(\nu, e) = \nu$ *as shown in Figure 4.4(a). Then, state* ν *is defined as initial state, as illustrated in Figure 4.4(b). Next, states* a *and* c *are added to the end of queue* F*, that is* $F = (\nu, a, c)$ *and state* ν *is added to set* Q_1*, i.e,* $Q_1 = \{\nu\}$*. In the second iteration, the first element of* F *is removed, and the queue becomes* $F = (a, c)$*. The first element of the queue is then assigned to variable* u*, i.e,* $u = a$*, and since* u *is different from* ν *in Step* 4.2*, the length of* u *is computed and assigned to variable* ℓ*; thus set* $I_\ell = \{1\}$ *is formed. Then, sets* $I_\nu = \emptyset$ *and*

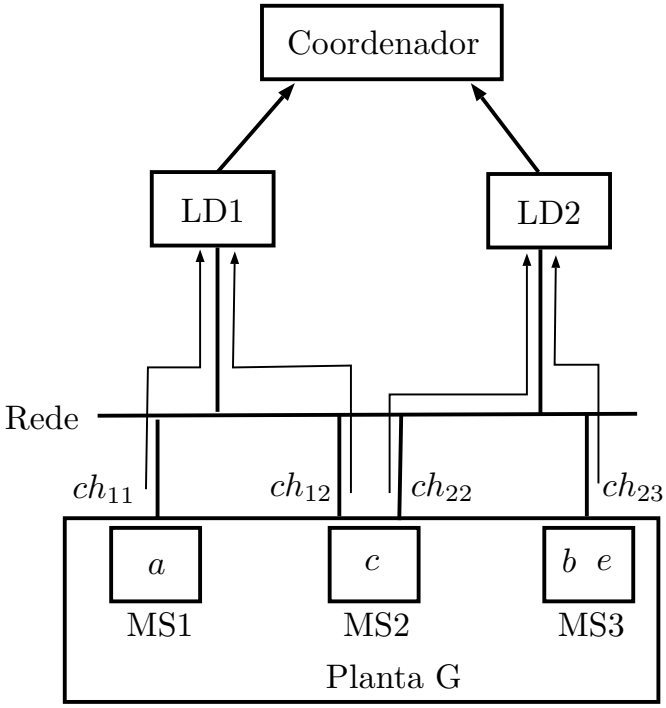


Figure 4.3: Network codiagnosis scheme of example 4.1.

$I_{\ell\setminus\nu} = I_\ell$ are computed. Notice that, the condition in Step 4.2(c) *and* 4.2(d) *check if the length of the suffixes of $u = \sigma_1\sigma_2\ldots\sigma_\ell$, is less than or equal to the delay of the communication channel that transmits event σ_y for all y in $I_{\ell\setminus\nu}$. Since for state a, the condition in Steps* 4.2(c) *and* 4.2(d) *holds true, in Step* 4.2(c)*, transitions from state a are defined for all $\sigma \in \Sigma_{o_1}$ as follows: $\delta_1(a,a) = aa$ and $\delta_1(a,c) = ac$, as shown in Figure 4.4(c), and in Step* 4.2(d)*, states aa and ac are added to the end of the queue F, i.e., $F = (a, c, aa, ac)$. After this, transitions from state a are defined for all $\sigma \in \Sigma_{uo_1} = \{b, e, \sigma_f\}$, following the condition of Step* 4.2(e)*, as follows: $\delta_1(a,b) = \delta_1(a,e) = \delta_1(a,\sigma_f) = a\nu$, as illustrated in Figure 4.4(d), and state $a\nu$ is added to the end of F in Step* 4.2(f) *which implies that $F = (a, c, aa, ac, a\nu)$. To finish this iteration, according to Steps* 4.2(g) *and* 4.2(h)*, a transition from state a labeled with a_{s_1} is created, i.e., $\delta_1(a, a_{s_1}) = \nu$, as illustrated in Figure 4.4(e). Notice that, $\delta_1(a, c_{s_1})$ is undefined because c has not occurred yet. Then, state a is added to set Q_1, i.e., $Q_1 = \{\nu, a\}$ and is removed from queue F, which becomes $F = (c, aa, ac, a\nu)$.*

Step 4 will be repeated for all elements of queue F until it becomes empty. Thus, according to Steps 4.2(g) *and* 4.2(h)*, the unique transition that can be defined from*

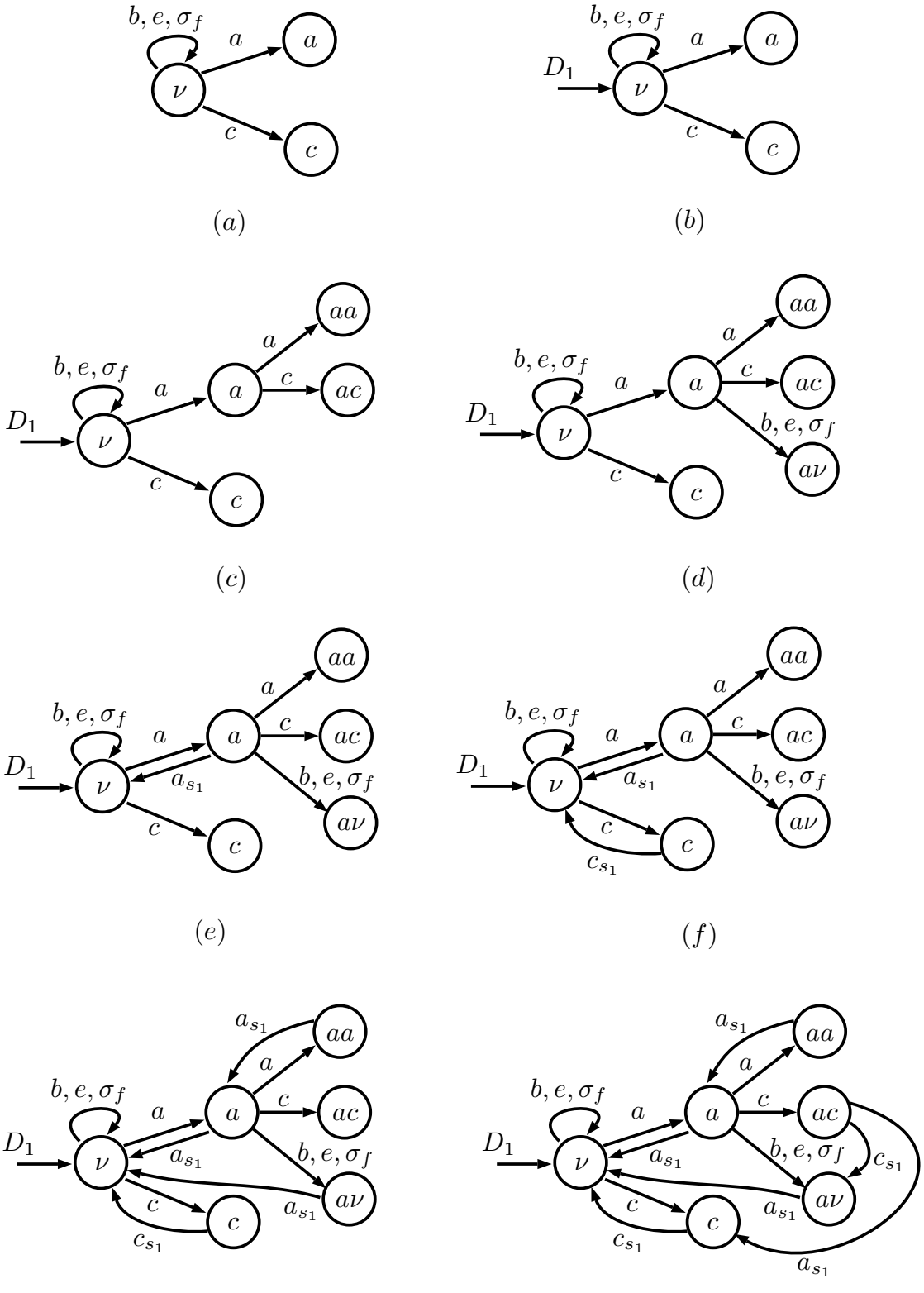
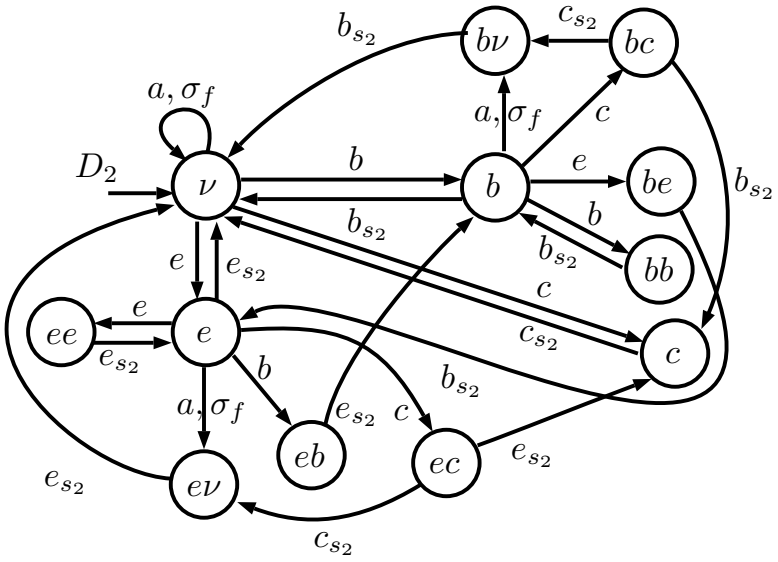


Figure 4.4: Construction of automaton D_1 step-by-step.



(b)

Figure 4.5: Automaton D_2.

state c is $\delta_1(c, c_{s_1}) = \nu$*, as shown in Figure 4.4(f), since the delay in communication channel* ch_{12} *is zero. Notice that a is the first element of the other states of F, namely aa, ac, and* $a\nu$*. Thus, there is no feasible transition labeled with* $\sigma \in \Sigma_i$ *according to Steps* 4.2(*c*) *and* 4.2(*e*)*, Since the maximum delay of communication channel* ch_{11} *that transmits event a is equal to one. Moreover, according to Steps* 4.2(*g*) *and* 4.2(*h*)*, the only feasible event for states aa and* $a\nu$ *is* a_{s_1}*, which implies that the new transitions from aa and* $a\nu$ *that will be defined, are, respectively,* $\delta_1(aa, a_{s_1}) = a$ *and* $\delta_1(a\nu, a_{s_1}) = \nu$*, as illustrated in Figure 4.4(g). Since event a belongs to* Σ_{MS_1}*, and event c belong to* Σ_{MS_2}*, the feasible events of state ac are* a_{s_1} *and* c_{s_1}*. Thus, transitions* $\delta_1(ac, a_{s_1}) = c$ *and* $\delta_1(ac, c_{s_1}) = a\nu$ *are defined, as shown in Figure 4.4(h). Since no other state is created, queue F becomes empty and the algorithm execution is ended.*

In order to model observation delays for LD_2*, we need to construct automaton* D_2 *shown in Figure 4.5, which is constructed in a similar way as* D_1*.*

□

From algorithm 4.1, Lemmas 4.1 and 4.2 can be stated. However, before we state the Lemmas, the projection P_i is defined as follows.

Notice that, from Algorithm 4.1, $\Sigma_i = \Sigma \cup \Sigma^s_{o_i}$. Then, let us now define the

following projection:

$$P_i : \Sigma_i^* \rightarrow \Sigma^*. \tag{4.6}$$

Now, the following results can be stated.

Lemma 4.1 *$P_i(L(D_i)) = \Sigma^\star$.*

Proof: According to steps 4.2(g) and 4.2(h) there is always a transition from state q_{z_i} to state q_{p_i} labeled by an event $\sigma_{s_i} \in \Sigma^s_{o_i}$, where q_{p_i} is formed by replacing the first appearance element $\phi_i(\sigma_{s_i})$ in q_{z_i} with ν (using the rep function), and by eliminating the largest prefix formed only with ν (using the cut function). Notice that, if the observed event $\phi_i(\sigma_{s_i})$ is not the first element of the trace registered in state q_{z_i}, then $\|q_{p_i}\| = \|q_{z_i}\|$; otherwise, $\|q_{p_i}\| \leq \|q_{z_i}\|$. In addition, when, after $\phi_i(\sigma_{s_i})$, where $\phi_i(\sigma_{s_i})$ is the first element in q_{z_i}, there are either only ν elements or $\|q_{z_i}\| = 1$, then $q_{p_i} = \nu$ (initial state). Thus, from state q_{p_i} it is possible to reach another state q_{x_i} by a transition labeled by an event σ_{s_i}, where either $\sigma_{s_i} \in \Sigma^s_{o_i}$ and $\phi(\sigma_{s_i}) \in q_{p_i}$, where (i) $\|q_{x_i}\| = \|q_{p_i}\|$; (ii) $\|q_{x_i}\| \leq \|q_{p_i}\|$, or (iii) $q_{x_i} = \nu$. Thus, from state q_{z_i}, it is always possible to reach the initial state ν after at most $\|q_{z_i}\|$ occurrences of events $\sigma_{s_i} \in \phi_i^{-1}(\Sigma_{o_i})$, which implies that there is a trace $t \in \Sigma^{s\star}_{o_i}$, such that, $\delta_i(q_{z_i}, t) = \nu$. This implies that, all states of $Obs(D_i, \Sigma)$ have the initial state ν as one of its component. Since $\Gamma_i(\nu) = \Sigma$, all states of $Obs(D_i, \Sigma)$ have Σ as their feasible event set, which implies that $L(Obs(D_i, \Sigma) = P_i(L(D_i)) = \Sigma^\star$. □

Lemma 4.2 *The language of automaton D_i, $L(D_i)$, models all possible orderings of observation of a trace $s \in \Sigma^*$ with respect to k_{ij}, for $j = 1, \ldots, m$, and Σ_{o_i}.*

Proof: The event set of D_i is obtained by enlarging the event set of G so as to make a distinction between event occurrences (plant event set Σ) and those actually observed by the local diagnosers ($\Sigma^s_{o_i}$). In the first step of Algorithm 4.1, $u_0 = \nu$ is defined, which later will be assigned as the initial state of D_i, and the set of states Q_i of automaton D_i is defined as the empty set. In Step 2, set $\Sigma^s_{o_i}$ is constructed in order to define the event set $\Sigma_i = \Sigma \cup \Sigma^s_{o_i}$ of automaton D_i. In Step 3, a FIFO queue F is generated to store new states created along the algorithm and to ensure that all states will be processed. The first state to be added to the queue is ν. The next step, Step 4, will be executed until queue F becomes empty, meaning that all states of D_i have been created and visited. In Step 4.1, the first element of F is assigned to

variable u, in order to be processed. In the first run, $u = u_0$, and so, only transitions labeled with events in $\Sigma = \Sigma_{o_i} \cup \Sigma_{uo_i}$ can be created: transitions labeled with $\sigma \in \Sigma_{o_i}$ are created from state ν to state σ, or back to state ν if $\sigma \in \Sigma_{uo_i}$; the former models the possibility of occurrence of observable events for LD_i and the state label informs the sequence of observable events that has occurred, whereas the latter accounts for the occurrence of unobservable events for LD_i. After that, the created states are added to the end of queue F to be processed later, element u is added to set Q_i, the initial state $q_{0_i} = \nu$ is defined, and ν is removed from queue F.

Let us consider the creation of states in D_i according to Steps 4.2(c) *and* 4.2(e). *Let $u = \sigma_y \sigma_{y+1} \ldots \sigma_{y+n-2}$ be the current state of D_i, and let n_y denotes the delay of the channel that communicates the occurrence of event σ_y to diagnoser LD_i. Let us assume that $u \in \Sigma^*_{o\nu}$, and consider the problem of evaluating the possibility of occurrence of an event $\sigma_{y+n-1} \in \Sigma$ before the observation of one of the events that form u. According to Steps* 4.2(c) *and* 4.2(e), *this evaluation is made in a recursive way through the suffixes of u. Thus, the condition holds true if, for all suffixes of u whose first element is not ν, the delay n_k of the first element σ_k is bigger than the length of the suffix, and when this condition holds true, two new states are created, namely, $u'_{new_n} = \sigma_y \sigma_{y+1} \ldots \sigma_{y+n-2} \sigma_{y+n-1}$, if $\sigma_{y+n-1} \in \Sigma_{o_i}$, and $u''_{new_n} = \sigma_y \sigma_{y+1} \ldots \sigma_{y+n-2} \nu$, if $\sigma_{y+n-1} \in \Sigma_{uo_i}$. If the conditions of Steps* 4.2(c) *and* 4.2(e) *are not verified, σ_{y+n-1} cannot occur before one of the events of u is observed. Consequently, no other new state is created. Steps* 4.2(c) *and* 4.2(e) *guarantee that the maximum delay of the communication channels will be respected in the creation of the states of D_i.*

Let us now consider the creation of the transitions in D_i labeled with events in $\Sigma^s_{o_i}$, according to Steps 4.2(g) *and* 4.2(h). *Assume, initially, that the elements different from ν of $u = \sigma_y \sigma_{y+1} \ldots \sigma_{y+n-2}$ belong to different measurement sites. Thus, each event $\phi_i^{-1}(\sigma_k) \in \Sigma^s_{o_i}$, where $\sigma_k \in \Sigma_{o_i}$, for $k \in \{y, y+1, \ldots, y+n-2\}$, is processed separately. According to Step* 4.2(g), *set Y is formed for each set $\Sigma^s_{o_{ij}}$. In this case, $Y = \{k\}$, for $k \in \{y, y+1, \ldots, y+n-2\}$. Then, in Step* 4.2($h$), *the lowest value of set Y is taken and the next state is computed by replacing the event on this position in the label of u by element ν, and then, by applying the function cut, leading to states $u_y = \sigma_{y+1} \ldots \sigma_{y+n-2}$, $u_{y+1} = \sigma_y \nu \ldots \sigma_{y+n-2}$, $\ldots$, $u_{y+n-2} = \sigma_y \sigma_{y+1} \ldots \nu$. Notice*

that, from state $u = \sigma_y\sigma_{y+1}\ldots\sigma_{y+n-2}$*, any event of* u *can be observed, and the observation of this event lead to one of the states* $u_y, u_{y+1}\ldots, u_{y+n-2}$*. Moreover, from any of these states, another event can also be observed, leading to other states until the initial state* ν *is reached. Thus, trace* $\sigma_y\sigma_{y+1}\ldots\sigma_{y+n-2}$ *can be observed in all possible order of observation with respect to* k_{ij}*, for* $j = 1, 2, \ldots, m$*.*

Assume now that some of the elements of state u *belong to the same measurement site* $\Sigma_{o_{ij}}$*. For instance, without loss of generality, let* $\sigma_{y+k}, \sigma_{y+\ell}, \sigma_{y+p} \in \Sigma_{o_{ij}}$*,* $(k < \ell < p)$*. This implies that* $\phi_i^{-1}(\{\sigma_{y+k}, \sigma_{y+\ell}, \sigma_{y+p}\}) \subseteq \Sigma^s_{o_{ij}}$*. Then, in Step* 4.2$(g)$*, set* $Y = \{y+k, y+\ell, y+p\}$*, associated with* $\Sigma^s_{o_{ij}}$ *is formed. Thus, according to Step* 4.2(h)*, the smallest element of* Y*,* $y+k$*, is taken, and the next state is computed by replacing* σ_{y+k} *in* u *with element* ν*, and then, by applying the cut function. As a result, the unique event of* $\Sigma^s_{o_{ij}}$ *that is feasible in* u *is* $\phi_i^{-1}(\sigma_{y+k})$*, and its occurrence leads to state* $u_{y+k} = \sigma_y\ldots\sigma_{y+k-1}\nu\sigma_{y+k+1}\ldots\sigma_{y+n-2}$*. The procedure is repeated for other sets* $\Sigma^s_{o_{ij}}$ *in a similar way. Thus, Steps* 4.2(g) *and* 4.2(h) *guarantee that all possible observations of a trace are represented in* D_i*, respecting the fact that the communication channels are FIFO.*

The procedure described above is repeated until no new state is created and queue F *becomes empty. Since, according to Step 4.2, the delayed observations are obtained based on the suffixes of the trace executed by the system, respecting the maximum delays of the communication channels, and all states* $q \in Q_i$*, where* $\Sigma \cap \Lambda_i(q) \neq \emptyset$*, are such that* $\Lambda_i(q) = \Sigma$*, then automaton* D_i *models all possible ordering of observation of all traces* $s \in \Sigma^*$ *with respect to* k_{ij}*, for* $j = 1, \ldots, m$*, and* Σ_{o_i}*.*

□

It is important to remark that Steps 4.2(g) and 4.2(h) are important when there are more than one event to be processed. Notice that, when more than one event of u belong to set $\Sigma^s_{o_{ij}}$, only the first occurred event is feasible to reach the new state defined in Step 4.2(h). This condition is a direct consequence of Assumption $A3$, which establishes that each communication channel follows FIFO rules, *i.e.*, there is no change in the order among events transmitted in the same communication channel.

Remark 4.1 *In [54], a nondeterministic model is proposed to represent the effects of communication delays between local diagnosers in a distributed diagnosis archi-*

tecture, assuming that there exists a unique delay bound k for all communication channels between diagnosers. This model was called k-delaying&masking model. It is worth remarking that, differently from [54], we address here the problem of decentralized diagnosis using Protocol 3 of [30], assuming that each communication channel between a measurement site and a local diagnoser can have different delay bounds k_{ij}. The effects of these communication delays are captured by automaton D_i, computed according to Algorithm 4.1. It is also important to remark that, differently from the k-delaying&masking model, the communication delay model D_i proposed here is deterministic. □

After the computation of automata D_i, $i = 1, \ldots, n$, we can obtain automata G_i, $i = 1, 2, \ldots, n$, that model all possible ordering of observation of the traces of L by local diagnoser LD_i due to communication delays, by performing the parallel composition of automata G and D_i, *i.e.*:

$$G_i = G \| D_i = (X_i, \Sigma_i, f_i, \Gamma_i, x_{0_i}, \emptyset). \tag{4.7}$$

Notice that the observable event set of G_i is $\Sigma_{i_o} = \Sigma^s_{o_i}$ and not Σ_{o_i}, and its unobservable event set is $\Sigma_{i_{uo}} = \Sigma$, *i.e.*, the occurrence of an event $\sigma_{s_i} \in \Sigma^s_{o_i}$ represents the successful observation of event $\sigma \in \Sigma_{o_i}$ by the local diagnoser LD_i.

Since $G_i = G \| D_i$, then the language generated by G_i is given by:

$$L(G_i) = {P_i}^{-1}(L(G)) \cap L(D_i), \tag{4.8}$$

where P_i is the projection defined in Equation (4.6) and $L(D_i)$ denotes the language generated by automaton D_i.

Based on Algorithm 4.1, Equation (4.7), and Lemmas 4.1 and 4.2, we can state the following theorem related to the observation of the language generated by G_i.

Theorem 4.1 *The language generated by G_i, $L(G_i)$, models all possible orderings of observation of the traces of $L(G)$ with respect to k_{ij}, for $j = 1, \ldots, m$, and Σ_{o_i}.*

Proof: Notice that, since $L(G_i) = {P_i}^{-1}(L(G)) \cap L(D_i)$, then

$$P_i(L(G_i)) = P_i[{P_i}^{-1}(L(G)) \cap L(D_i)] \subseteq L(G) \cap P_i(L(D_i)). \tag{4.9}$$

Now, we will prove that

$$L(G) \cap P_i(L(D_i)) \subseteq P_i[{P_i}^{-1}(L(G)) \cap L(D_i)]. \tag{4.10}$$

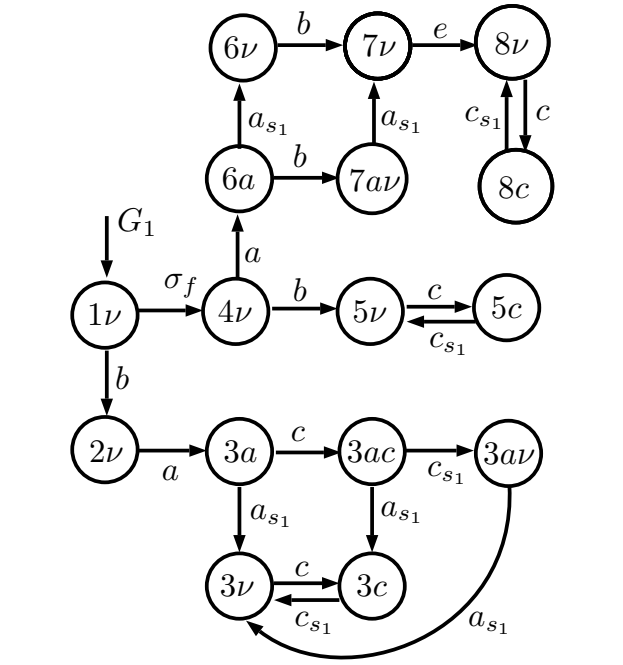
In order to do so, let $\mu \in L(G) \cap P_i(L(D_i))$. Then, there exist languages $K_1 \subseteq P_i^{-1}(L(G))$ and $K_2 \subseteq L(D_i)$ such that $P_i(s_1) = P_i(s_2) = \mu$ for any trace $s_1 \in K_1$ and $s_2 \in K_2$. Notice that the events in $\Sigma_{o_i}^s$ are private events of D_i, and thus, any trace $s_2 \in K_2$ must also belong to K_1, which implies that $K_2 \subseteq K_1$. Thus, since $\mu \in P_i(K_2)$ and $K_2 \subseteq K_1$, $\mu \in P_i(K_1 \cap K_2) = P_i[P_i^{-1}(L(G)) \cap L(D_i)]$, which shows that $L(G) \cap P_i(L(D_i)) \subseteq P_i[P_i^{-1}(L(G)) \cap L(D_i)]$. Since both inclusions (4.9) and (4.10) hold true, then $P_i(L(G_i)) = P_i[P_i^{-1}(L(G)) \cap L(D_i)] = L(G) \cap P_i(L(D_i))$.

According to Lemma 4.1, $P_i(L(D_i)) = \Sigma^*$, which implies that $P_i(L(G_i)) = L(G)$. This shows that automaton G_i is not capable of representing the delayed observations of a trace that does not belong to $L(G)$. Moreover, since all events of $\Sigma_{o_i}^s$ are private events of D_i, the occurrence of these events is allowed whenever they are feasible by the states of automaton D_i in the parallel composition $G \| D_i$. Therefore, since, according to Lemma 4.2, D_i models all possible ordering of observation of all traces in Σ^*, G_i will model all possible ordering of observation of the traces in $L(G)$. □

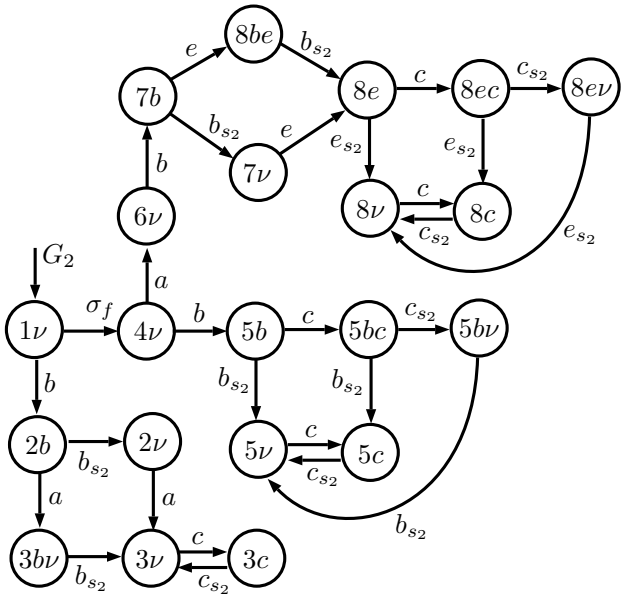
Example 4.2 *Consider the same plant and decentralized diagnosis architecture presented in Example 4.1. The automata G_1 and G_2, depicted in Figure 4.6(a) and 4.6(b), respectively, are computed according to Equation (4.7) as $G_i = G \| D_i$, for $i = 1, 2$. The sets of observable and unobservable events of G_1 are $\Sigma_{1_o} = \{a_{s_1}, c_{s_1}\}$, and $\Sigma_{1_{uo}} = \{a, b, c, e, \sigma_f\}$, respectively, and the sets of observable and unobservable events of G_2 are $\Sigma_{2_o} = \{b_{s_2}, c_{s_2}, e_{s_2}\}$, and $\Sigma_{2_{uo}} = \{a, b, c, e, \sigma_f\}$, respectively.*

Notice that, languages $L(G_1)$ and $L(G_2)$ represent all possible ordering of observation of traces $s \in L(G)$ with respect to $\Sigma_{o_{ij}}$ and k_{ij}, for $j \in \{1, 2, 3\}$. For instance, let us consider trace $s = bac^z \in L(G)$, where $z \in \mathbb{N}$. Since $\Sigma_{o_{11}} = \{a\}$ and $\Sigma_{o_{12}} = \{c\}$, and $k_{11} = 1$ and $k_{12} = 0$, then the possible observations of s for local diagnoser LD_1 are ac^z and cac^{z-1}. Notice that the following traces are generated in $L(G_1)$ associated with trace s: $s_1 = baa_{s_1}(cc_{s_1})^z$, $s_2 = baca_{s_1}c_{s_1}(cc_{s_1})^{z-1}$ and $s_3 = bacc_{s_1}a_{s_1}(cc_{s_1})^{z-1}$ whose projections $P_{is_i} : \Sigma_i^* \rightarrow \Sigma_{o_i}^{s^*}$ are, respectively, $P_{1s_1}(s_1) = P_{1s_1}(s_2) = a_{s_1}c_{s_1}^z$ and $P_{1s_1}(s_3) = c_{s_1}a_{s_1}c_{s_1}^{z-1}$. □

Before we present the modeling of a plant subject to communication delays, let us introduce the approach of intermittent loss of observation proposed in [26] in the next section. This approach will be used as base of modeling of a plant subject to communication delays and intermittent loss of observations proposed in this work.



(a)



(b)

Figure 4.6: (a) Automaton G_1; (b) Automaton G_2.

4.3 Modeling of intermittent loss of events

Modeling of physical systems using discrete event models assumes that a set of sensors always reports event occurrences correctly. However, in a complex plant with network communication, bad sensor operation that results from bad electrical linkage, defective components, large flow of information and collisions in the communication channels, may lead to observation loss of the events.

One of the most important problem of failure diagnosis is when events are lost and, consequently, the diagnoser can not observe them. Some works have addressed this problem in the literature, such as [70], [71] and [26]. In [70] and [71] permanent loss of events is assumed, *i.e.*, once the system lose an event, it never recovers, and in [26], the problem of intermittent losses of events is addressed. Since the approach presented in [26] is more general than permanent loss of observation, in this thesis we will adopt the second approach as the basis for the network codiagnosability proposed in this thesis.

Modeling of intermittent loss of observation is made by Dilation operation, defined as follows.

Let $\Sigma_o = \Sigma_{ilo} \dot{\cup} \Sigma_{nilo}$ be a partition of Σ_o, where Σ_{ilo} is set of observable events subject to intermittent loss of observations and Σ_{nilo} is set of observable events that are not subject to intermittent loss of observation. In addition, let $\Sigma'_{ilo} = \{\sigma' : \sigma \in \Sigma_{ilo}\}$ be a set of unobservable events, and $\Sigma_{dil} = \Sigma \cup \Sigma'_{ilo}$, *i.e.*, $\Sigma_{dil} = \Sigma_o \cup \Sigma_{uo} \cup \Sigma'_{ilo}$.

Definition 4.2 *(Dilation [26]) Let $\Sigma_{ilo} \dot{\cup} \Sigma_{nilo} \dot{\cup} \Sigma_{uo}$ be a partition of Σ, where Σ_{ilo} is a set of observable events associated with intermittent loss of observations and Σ_{nilo} denotes set of observable events that are not subject to intermittent loss of observations. Let Σ'_{ilo} and Σ_{dil} sets already defined earlier. The Dilation operation, D, is a mapping as follows*

$$\begin{aligned} D: \quad & \Sigma^* \rightarrow 2^{(\Sigma^*_{dil})} \\ & s \mapsto D(s) \end{aligned}$$

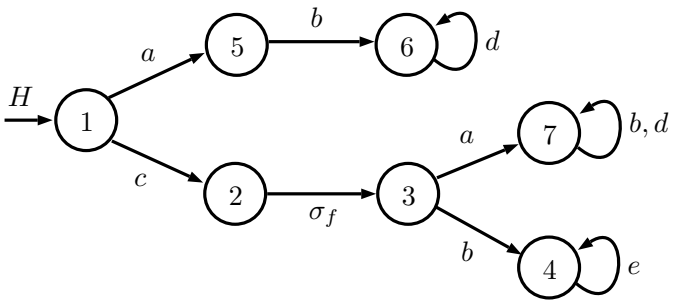


Figure 4.7: Automaton H of Example 4.3.

where

$$D(\epsilon) = \{\epsilon\}$$
$$D(\sigma) = \begin{cases} \{\sigma\}, \ \textit{if } \sigma \in \Sigma \setminus \Sigma_{ilo} \\ \{\sigma, \sigma'\}, \ \textit{if } \sigma \in \Sigma_{ilo} \end{cases}$$
$$D(s\sigma) = D(s)D(\sigma), s \in \Sigma^*, \sigma \in \Sigma$$

The dilation operation D can be extended from traces to languages by applying it to all traces in the language, that is,

$$D(L) = \bigcup_{s \in L} D(s)$$

□

In order to illustrate the dilation operation consider the example 4.3.

Example 4.3 *[26] Consider the automaton H shown in Figure 4.7, where $\Sigma = \{a, b, c, d, e, \sigma_f\}$, $\Sigma_{ilo} = \{c\}$ and language $L(H) = \{ab\}\{d\}^* \cup \{c\sigma_f\}(\{c\}\{e\}^* \cup \{a\}\{b, d\}^*)$. Thus, $\Sigma_{dil} = \{a, b, c, c', d, e, \sigma_f\}$. Suppose the application of dilation in trace $s_1 = abd$, then, $D(s_1) = \{abd\}$ since $a, b, d \notin \Sigma_{ilo}$. For trace $s_2 = c\sigma_f be$, then, $D(s_2) = \{c, c'\}\{\sigma_f\}\{b\}\{e\} = \{c\sigma_f be, c'\sigma_f be\}$. If we apply dilation function to $L(H)$, it is easy see that $D[L(H)] = L_{H,dil}$ which is generated by automaton of Figure 4.8*

□

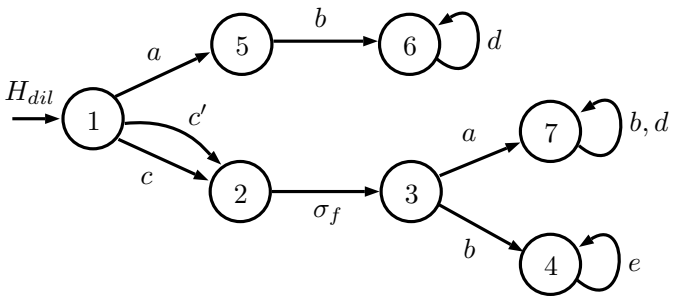


Figure 4.8: Automaton H_{dil}.

4.4 Model of the plant subject to communication delays and intermittent loss of observations

After the computation of automata G_i, for $i = 1, 2, \ldots, n$, that represent all possible observations by local diagnosers LD_i, $i = 1, 2, \ldots, n$, of the language generated by G due to communication delays of events, we will now model the intermittent loss of observation of events in the communication channels. In order to do so, we will use the dilation function introduced in [26].

Consider the partition of the set of observable events associated with diagnoser LD_i, $\Sigma_{i_o} = \Sigma_{i,ilo} \dot{\cup} \Sigma_{i,nilo}$, where $\Sigma_{i,ilo}$ and $\Sigma_{i,nilo}$ denote, respectively, the set of events that are subject to intermittent loss of observation, and the set of events that are not subject to intermittent loss of observation. Let $\Sigma^s_{i,ilo} = \phi^{-1}(\Sigma_{i,ilo})$ and $\Sigma^s_{i,nilo} = \phi^{-1}(\Sigma_{i,nilo})$. Then, since the observable event set of G_i is given by $\Sigma_{i_o} = \Sigma^s_{o_i}$, we can make the following partition of the observable event set of G_i:

$$\Sigma^s_{o_i} = \Sigma^s_{i,ilo} \dot{\cup} \Sigma^s_{i,nilo}, \tag{4.11}$$

where the events of $\Sigma^s_{i,ilo}$ and $\Sigma^s_{i,nilo}$ denote the successful transmission to diagnoser L_{D_i} of the events of $\Sigma_{i,ilo}$ and $\Sigma_{i,nilo}$, respectively.

Let us define now the set of unobservable events that models the intermittent loss of observation of events $\sigma \in \Sigma^s_{i,ilo}$ as $\Sigma^{s'}_{i,ilo} = \{\sigma' : \sigma \in \Sigma^s_{i,ilo}\}$ and set $\Sigma'_i = \Sigma_i \cup \Sigma^{s'}_{i,ilo}$. Then, the dilation function $D_{s_i} : \Sigma_i^* \to 2^{(\Sigma'_i)^*}$ is defined in a recursive way as:

$$D_{s_i}(\epsilon) = \{\epsilon\},$$

$$D_{s_i}(\sigma) = \begin{cases} \{\sigma\}, \text{ if } \sigma \in \Sigma_i \setminus \Sigma^s_{i,ilo} \\ \{\sigma, \sigma'\}, \text{ if } \sigma \in \Sigma^s_{i,ilo} \end{cases}$$

$$D_{s_i}(s_i\sigma) = D_{s_i}(s_i)D_{s_i}(\sigma),\ \forall s_i \in \Sigma_i^\star,\ \forall \sigma \in \Sigma_i.$$

The dilation operation D_{s_i} is extended to languages in a straightforward way as $D_{s_i}(L) = \bigcup_{s\in L} D_{s_i}(s)$.

We can now obtain automaton G_i' that generates language $D_{s_i}[L(G_i)]$, and that models both, all possible ordering of observation of events $\sigma \in \Sigma_o$ due to communication delays and the intermittent loss of observation of events $\sigma \in \Sigma_{i,ilo}$. This automaton will be defined as follows:

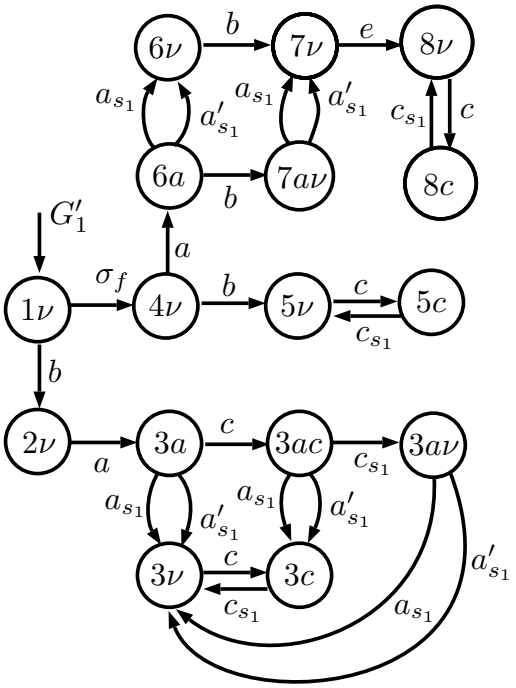
$$G_i' = (X_i, \Sigma_i', f_i', \Gamma_i', x_{0_i}, \emptyset),$$

where $\Gamma_i'(x_i) = D_{s_i}[\Gamma_i(x_i)]$, for all $x_i \in X_i$, and $f_i'(x_i, \sigma') = f_i(x_i, \sigma)$, if $\sigma' \in \Sigma_{i,ilo}^{s'}$, and $f_i'(x_i, \sigma) = f_i(x_i, \sigma)$, if $\sigma \in \Sigma_i \setminus \Sigma_{i,ilo}^{s'}$. Notice that, if $\Sigma_{i,ilo} = \emptyset$, $G_i' = G_i$, which implies that $D_{s_i}[L(G_i)] = L(G_i)$.

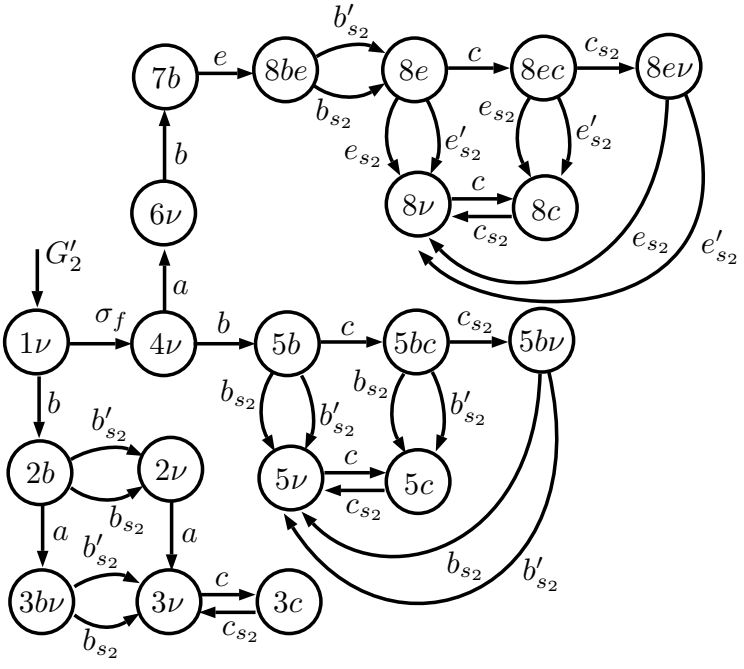
The following examples illustrate the construction of G_i' considering different conditions of losses of observation. Later, in Section 4.6, Example 4.4 is used to illustrate a noncodiagnosable system and Example 4.5 is used to illustrate a codiagnosable system.

Example 4.4 *Let us consider the problem considered in Example 4.2, and assume that automata G_1 and G_2 have been calculated. In addition, suppose that event a is subject to intermittent loss of observation only by local diagnoser LD_1, and events b and e are subject to intermittent loss of observation only by local diagnoser LD_2. Thus, for local diagnoser LD_1, $\Sigma_{1,ilo} = \{a\}$ and $\Sigma_{1,nilo} = \{c\}$, and for local diagnoser LD_2, $\Sigma_{2,ilo} = \{b, e\}$ and $\Sigma_{2,nilo} = \{c\}$. Automata G_1' and G_2' that model the communication delay and intermittent loss of observations of the events in $\Sigma_{1,ilo}$ and $\Sigma_{2,ilo}$, respectively, are shown in Figures 4.9(a) and 4.9(b). Notice that, as expected, $L(G_1') = D_{s_1}[L(G_1)]$ and $L(G_2') = D_{s_2}[L(G_2)]$.* □

Example 4.5 *Let us consider, once again, the problem of Example 4.2 and assume that G_1 and G_2 have been calculated. Now, suppose that there is no event subject to intermittent loss of observation by local diagnoser LD_1, and only event b is subject to intermittent loss of observation by local diagnoser LD_2. Thus, for LD_1, $\bar{\Sigma}_{1,ilo} = \emptyset$ and $\bar{\Sigma}_{1,nilo} = \{a, c\}$, and for LD_2, $\bar{\Sigma}_{2,ilo} = \{b\}$ and $\bar{\Sigma}_{2,nilo} = \{c, e\}$. Automata $\bar{G}_1'$*



(a)



(b)

Figure 4.9: (a) automaton G'_1; (b)automaton G'_2.

and $\bar{G}'_2$ that model the communication delay and intermittent loss of observations of events in $\bar{\Sigma}_{1,ilo}$ and $\bar{\Sigma}_{2,ilo}$, are shown in Figures 4.10(a) and 4.10(b).

4.5 Definition of network codiagnosability of discrete-event systems

In order to obtain all traces of G_i whose projections $P_{is_i} : \Sigma_i^* \to \Sigma_{o_i}^{s^*}$ provide all possible ordering of observations of a trace $s \in L(G)$ due to communication delays, we will introduce function

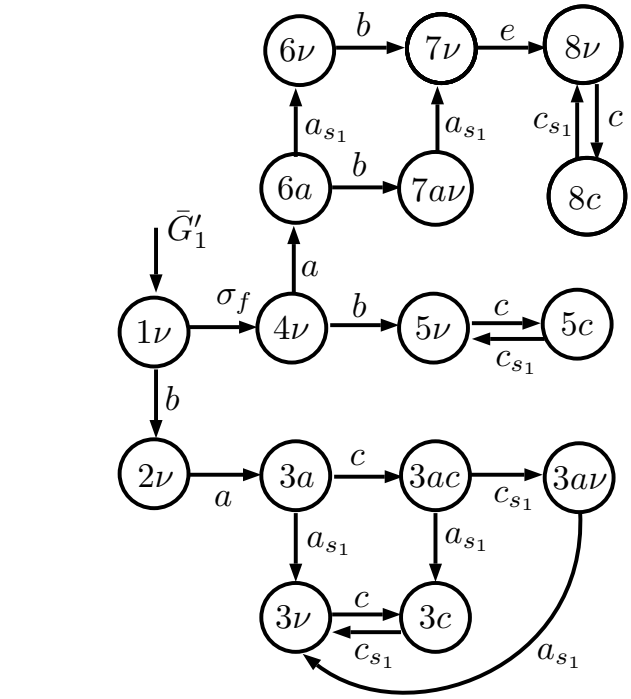
$$\begin{aligned} \chi_i : \quad & \Sigma^* \to 2^{\Sigma_i^*}, \\ & s \mapsto \chi_i(s) = P_i^{-1}(s) \cap L(D_i). \end{aligned} \tag{4.12}$$

Notice that $L(G_i) = \bigcup_{s \in L} \chi_i(s)$, *i.e.*, if the function χ_i is applied to a trace $s \in L(G)$, it generates all possible ordering of observation of s due to communication delays. In order to also take into account the intermittent loss of observation in the codiagnosability verification of networked systems, it is necessary to dilate the traces in $L(G_i)$, using function D_{s_i}. We can define the network codiagnosability of the language generated by a DES as follows.

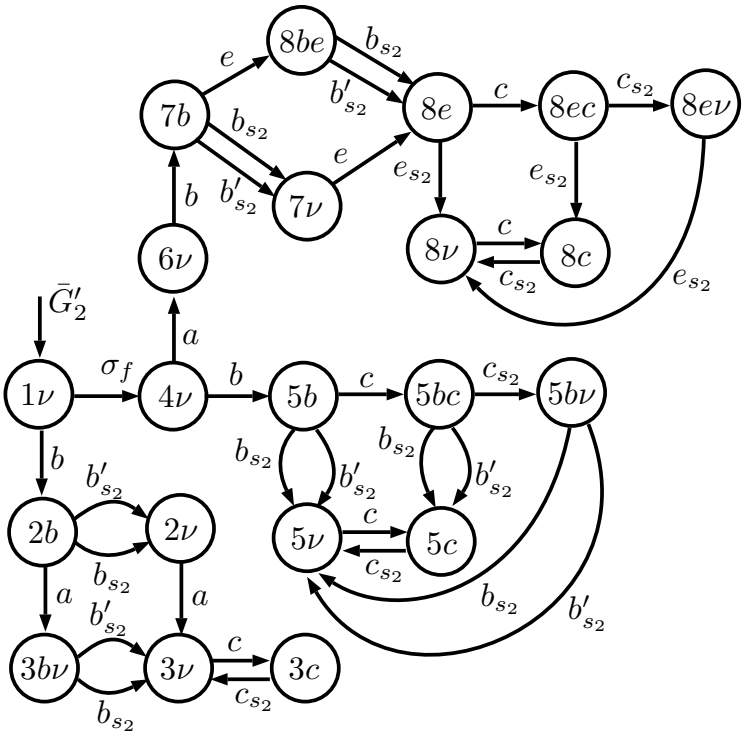
Definition 4.3 *Let L and $L_N \subset L$ be the prefix-closed languages generated by G and G_N, respectively. Then, L is said to be network codiagnosable with respect to $\chi_i : \Sigma^* \to 2^{\Sigma_i^*}$, D_{s_i}, projection $P'_{s_i} : \Sigma_i'^* \to \Sigma_{o_i}^{s^*}$, for $i = 1, \ldots, n$, and Σ_f if*

$$\begin{aligned} & (\exists z \in \mathbb{N})(\forall s \in L \setminus L_N)(\forall st \in L \setminus L_N, ||t|| \geq z) \Rightarrow \\ & (\exists i \in \{1, \ldots, n\})[P'_{s_i}[D_{s_i}(\chi_i(st))] \cap P'_{s_i}[D_{s_i}(\chi_i(\omega_i))] = \emptyset, \forall \omega_i \in L_N]. \end{aligned}$$

According to Definition 4.3, language L is not network codiagnosable if there exist a failure trace s and an arbitrarily long length trace t, such that there exist traces $s_i t_i \in D_{s_i}(\chi_i(st))$, $i = 1, 2, \ldots, n$, where $s_i t_i$ is not necessarily different from $s_j t_j$ for $i, j \in \{1, 2, \ldots, n\}$ and $s_{i_N} \in D_{s_i}(\chi_i(\omega_i))$, with $\omega_i \in L_N$, satisfying $P'_{s_i}(s_i t_i) = P'_{s_i}(s_{i_N})$, for all $i \in \{1, \ldots, n\}$. In words, a language L is not network codiagnosable if there exist a failure trace st, with arbitrarily long length after the occurrence



(a)



(b)

Figure 4.10: (a) automaton $\bar{G}'_1$; (b)automaton $\bar{G}'_2$.

of the failure event, and there exist normal traces ω_i, for $i = 1, \ldots, n$, such that, the change in the order of observation and the loss of observation of events create ambiguous observations in all local diagnosers.

4.6 Verification of network codiagnosability of discrete-event systems

We present in the sequel an algorithm for the verification of network codiagnosability of DES based on the verification algorithm proposed in [57]. In order to do so, we first present the definition of the one-to-one event renaming function

$$\begin{aligned} \rho_i : \quad & \Sigma'_{i_N} \to \Sigma'_{i_\rho}, \\ & \sigma \mapsto \rho_i(\sigma) = \begin{cases} \sigma_{\rho_i}, \text{ if } \sigma \in (\Sigma \cup \Sigma^{s'}_{i,ilo}) \setminus \Sigma_f \\ \sigma, \text{ if } \sigma \in \Sigma^{s}_{o_i}. \end{cases} \end{aligned} \tag{4.13}$$

where $\Sigma'_{i_N} = \Sigma'_i \setminus \Sigma_f$, for $i = 1, \ldots, n$. The domain of function ρ_i can be extended to $\Sigma'^{\star}_{i_N}$ as usual, *i.e.*, $\rho_i(s\sigma) = \rho_i(s)\rho_i(\sigma)$, for all $s \in \Sigma'^{\star}_{i_N}$ and $\sigma \in \Sigma'_{i_N}$. Function ρ_i can also be applied to a language K as $\rho_i(K) = \cup_{s \in K} \rho_i(s)$.

Algorithm 4.2 *Network codiagnosability verification of DES*

Input: $G'_i = (X_i, \Sigma'_i, f'_i, \Gamma'_i, x_{0_i}, \emptyset)$, *for* $i = 1, \ldots, n$.

Output: $G_V = (X_V, \Sigma_V, f_V, \Gamma_V, x_{0,V}, X_{V_m})$.

1: *Compute automata* $G'_{i,N} = (X'_{i_N}, \Sigma'_{i_N}, f'_{i_N}, \Gamma'_{i_N}, (x_{0_i}, N), \emptyset)$, *where* $\Sigma'_{i_N} = \Sigma'_i \setminus \Sigma_f$, *for* $i = 1, \ldots, n$, *that model the normal behavior of* G'_i *as presented in [57].*

2: *Compute automata* $G'_{i,F} = (X'_{i_F}, \Sigma'_i, f'_{i_F}, \Gamma'_{i_F}, (x_{0_i}, N), \emptyset)$, *for* $i = 1, \ldots, n$, *that model the failure behavior of* G'_i *as presented in [57].*

3: *Construct automata* $G'_{i,\rho} = (X'_{i_N}, \Sigma'_{i_\rho}, f'_{i_\rho}, \Gamma'_{i_\rho}, (x_{0_i}, N), \emptyset)$, *for* $i = 1, \ldots, n$, *where* $\Sigma'_{i_\rho} = \rho_i(\Sigma'_{i_N})$, *and* $f'_{i_\rho}(x_{i_N}, \sigma_{\rho_i}) = f'_{i_N}(x_{i_N}, \sigma)$ *with* $\sigma_{\rho_i} = \rho_i(\sigma)$, *for all* $\sigma \in \Sigma'_{i_N}$ *and* $x_{i_N} \in X'_{i_N}$.

4: *Compute automata* $\bar{V}_i = G'_{i,\rho} \| G'_{i,F} = (Y_{V_i}, \Sigma_{V_i}, f_{V_i}, \Gamma_{V_i}, y_{V_i,0}, \emptyset)$, *for* $i = 1, \ldots, n$, *where* $\Sigma_{V_i} = \Sigma'_{i_\rho} \cup \Sigma'_i$.

5: Find all cyclic paths $cl_i = (y^k_{V_i}, \sigma_k, y^{k+1}_{V_i}, \sigma_{k+1}, \ldots, \sigma_\ell, y^k_{V_i})$*, where* $\ell \geq k > 0$ *in* $\bar{V}_i$ *that satisfy the following condition:*

$$\begin{aligned} &\exists j \in \{k, k+1, \ldots, \ell\} \text{ such that, for some} \\ &y^j_{V_i} = (x^j_i, N, y^j_i, F) \wedge (\sigma_j \in \Sigma'_i) \end{aligned} \tag{4.14}$$

where $x^j_i, y^j_i \in X_i$.

6: Compute automata $V_i = (Y_{V_i}, \Sigma_{V_i}, f_{V_i}, \Gamma_{V_i}, y_{V_i,0}, Y_{V_i,m})$*, where* $Y_{V_i,m}$ *is formed by the states of* $\bar{V}_i$ *that belong to the strongly connected components that contain cyclic paths* cl_i *satisfying condition (4.14).*

7: Compute the verifier automaton $G_V = V_1 \| \ldots \| V_n = (X_V, \Sigma_V, f_V, \Gamma_V, x_{V,0}, X_{V_m})$*, where* $\Sigma_V = \bigcup_{i=1}^n \Sigma_{V_i}$.

8: Verify the existence of a cyclic path $cl = (x^k_V, \sigma_k, x^{k+1}_V, \sigma_{k+1}, \ldots, \sigma_\ell, x^k_V)$ *in* G_V*,* $\ell \geq k > 0$*, that satisfies the following condition:*

$$\begin{aligned} &x^q_V \in X_{V_m}, \forall q \in \{k, k+1, \ldots, \ell\}, \text{ and for some} \\ &q \in \{k, k+1, \ldots, \ell\}, \sigma_q \in \Sigma. \end{aligned} \tag{4.15}$$

If the answer is yes, then L *is not network codiagnosable with respect to* χ_i*,* D_{s_i}*,* P'_{s_i}*, for* $i = 1, \ldots, n$*, and* Σ_f*. Otherwise,* L *is network codiagnosable.*

Remark 4.2 *Notice that the renamed events of verifier* V_p *are different from the renamed events of a verifier* V_q*, where* $p \neq q$. □

Lemma 4.3 *Let* $G'_{i,N}$ *and* $G'_{i,F}$ *be computed according to Steps 1 and 2 of Algortihm 4.2, respectively. Then,* $L(G'_{i,F}) = \bigcup_{s \in \overline{L \setminus L_N}} D_{s_i}(\chi_i(s))$*, and* $L(G'_{i,N}) = \bigcup_{\omega \in L_N} D_{s_i}(\chi_i(\omega))$.

Proof: The proof is straightforward from the construction of G'_i*,* $G'_{i,N}$ *and* $G'_{i,F}$. □

Theorem 4.2 *Language* L *is network codiagnosable with respect to* χ_i*,* D_{s_i}*,* P'_{s_i}*, for* $i = 1, \ldots, n$*, and* Σ_f *if, and only if, there does not exist a cyclic path*

$cl = (x_V^k, \sigma_k, x_V^{k+1}, \sigma_{k+1}, \ldots, x_V^\ell, \sigma_\ell, x_V^k)$, $\ell \geq k > 0$ *in* G_V *satisfying the following condition:*

$$x_V^q \in X_{V_m}, \forall q \in \{k, k+1, \ldots, \ell\}, \text{ and for some}$$
$$q \in \{k, k+1, \ldots, \ell\}, \sigma_q \in \Sigma. \tag{4.16}$$

Proof: ($\Rightarrow$) *Suppose that language* L *is not network codiagnosable with respect to* χ_i, D_{s_i}, P'_{s_i}, *for* $i = 1, \ldots, n$, *and* Σ_f. *Thus, according to Definition 4.3, there exists at least one arbitrarily long length trace* $st \in L \backslash L_N$ *and traces* $\omega_i \in L_N$, $i = 1, \ldots, n$, *where* ω_i *is not necessarily distinct from* ω_j, *for* $j = 1, \ldots, n$ *and* $i \neq j$, *such that* $P'_{s_i}[D_{s_i}(\chi_i(st))] \cap P'_{s_i}[D_{s_i}(\chi_i(\omega_i))] \neq \emptyset$ *for all* $i \in \{1, 2, \ldots, n\}$. *Thus, according to Lemma 4.3, if* L *is not network codiagnosable, there exist traces* $s_i t_i \in L(G'_{i,F})$ *and* $s_{i_N} \in L(G'_{i,N})$ *such that,* $P'_{s_i}(s_i t_i) = P'_{s_i}(s_{i_N})$ *for all* $i \in \{1, 2, \ldots, n\}$. *As shown in [57], the existence of traces* $s_i t_i$ *and* s_{i_N} *such that* $P'_{s_i}(s_i t_i) = P'_{s_i}(s_{i_N})$ *for all* $i \in \{1, 2, \ldots, n\}$, *implies in the existence of a path* p_i *in* V_i, *that ends with a cyclic path* cl_i *that satisfies condition (4.14), whose associated trace* $v_i \in L(V_i)$ *satisfies* $P_{V_i i}(v_i) = s_i t_i$ *and* $P_{V_i \rho}(v_i) = s_{i_{N_\rho}}$, *where* $s_{i_{N_\rho}} = \rho_i(s_{i_N})$, $P_{V_i i} : \Sigma_{V_i}^* \to \Sigma_i^*$ *and* $P_{V_i \rho} : \Sigma_{V_i}^* \to \Sigma_{i_\rho}^*$.

Notice that, if the states of the cyclic path cl_i *are marked, then* $v_i \in L_m(V_i)$, *where* $L_m(V_i)$ *denotes the marked language of* V_i. *Since* $G_V = ||_{i=1}^n V_i$, *then* $L_m(G_V) = \bigcap_{i=1}^n P_{VV_i}^{-1}[L_m(V_i)]$, *where* $P_{VV_i} : \Sigma_V^* \to \Sigma_{V_i}^*$. *Thus,* $\bigcap_{i=1}^n P_{VV_i}^{-1}(v_i) \subseteq L_m(G_V)$. *Let* $v \in \bigcap_{i=1}^n P_{VV_i}^{-1}(v_i)$. *Since* $v_i \in L_m(V_i)$, $P_{V_i i}(v_i) = s_i t_i$ *and* $P_i(s_i t_i) = st$, *for all* $i \in \{1, \ldots, n\}$, *and the common events that synchronize the traces* v_i, *for* $i = 1, \ldots, n$, *in* $\bigcap_{i=1}^n P_{VV_i}^{-1}(v_i)$ *are in* Σ, *then there will be a cyclic path in* G_V, *associated with* v *with all states marked, with at least one transition labeled with an event* $\sigma \in \Sigma$.

($\Leftarrow$) *Suppose that there exists a path* p *in* G_V *that ends with a cyclic path* cl *that satisfies condition (4.16), and let* $v \in L_m(G_V)$ *be the trace associated with* p. *Notice that, since* $G_V = \|_{i=1}^n V_i$, *then* $L_m(G_V) = \bigcap_{i=1}^n {P_{VV_i}}^{-1}[L_m(V_i)]$, *and* $P_{VV_i}(v) = v_i \in L_m(V_i)$, *for* $i = 1, 2, ..., n$. *Notice also that, the common events of traces* $v_i \in L_m(V_i)$, *for* $i = 1, 2, ..., n$, *are events* $\sigma \in \Sigma$. *Thus, since condition (4.16) is verified, then at least one event in the cyclic path* cl *belongs to* Σ, *which implies that all traces* v_i *are associated with a path* p_i *that ends with a cyclic path* cl_i, *formed with marked states, that has an event in* Σ. *According to Algorithm 4.2, the states of a cyclic path* cl_i *in* V_i *are marked only if the failure has occurred. Thus, associated with the cyclic path* cl

of G_V there exists one cyclic path cl_i in each verifier V_i, for $i = 1, \ldots, n$, that satisfies condition (4.14), i.e., there exists a failure trace $s_i t_i \in L(G_i)$, with arbitrarily long length, and a normal trace $s_{i_N} \in L(G_i)$, such that $P'_{s_i}(s_i t_i) = P'_{s_i}(s_{i_N})$, for all $i \in \{1, \ldots, n\}$. In order to show that L is not network codiagnosable, notice that, since condition (4.16) is verified, then there exists an arbitrarily long length failure trace $st \in \Sigma^*$, such that $P_V(v) = st$, where $P_V : \Sigma_V^* \to \Sigma^*$. Since the events in Σ are common events of all verifiers V_i and $G_V = \|_{i=1}^n V_i$, then $P_{V_i}(v_i) = st$, where $P_{V_i} : \Sigma_{V_i}^* \to \Sigma^*$, which shows that there exists an arbitrarily long length failure trace st such that $s_i t_i \in D_{s_i}(\chi_i(st))$ for $i \in \{1, \ldots, n\}$. Thus, according to Definition 4.3, L is not network codiagnosable with respect to χ_i, D_{s_i}, P'_{s_i}, for $i = 1, \ldots, n$, and Σ_f.
□

Example 4.6 *Let us verify the network codiagnosability of the system presented in Example 4.4. Following Steps 1, 2, and 3 of Algorithm 4.2, automata $G'_{1,\rho}$ and $G'_{1,F}$, shown in Figures 4.11(a) and 4.11(b), respectively and automata $G'_{2,\rho}$ and $G'_{2,F}$, shown in Figures 4.12(a) and 4.12(b), respectively, are computed. Continuing Algorithm 4.2, verifiers V_1 and V_2 are computed. Due to the size of these automata, we show in Figures 4.13(a) and 4.13(b), only the paths of V_1 and V_2 that contain the cyclic paths cl_1 and cl_2 that satisfy condition (4.14). After the computation of V_1 and V_2, automaton $G_V = V_1 || V_2$ can be computed, in accordance with Step 7. We show in Figure 4.14 only the path of G_V that contains a cyclic path cl associated with the cyclic paths cl_1, $(cc_{s_1}c_{\rho_1})^n$, and cl_2, $(cc_{s_2}c_{\rho_2})^n$. Notice that cl is formed by marked states and contains an event $c \in \Sigma$. Thus, according to condition (4.15), language L is not network codiagnosable with respect to $\chi_i : \Sigma^\star \to 2^{\Sigma_i^\star}$, D_{s_i}, $P'_{s_i} : \Sigma_i'^\star \to \Sigma_{o_i}^{s\star}$, for $i = 1, 2$, and Σ_f.*

Example 4.7 *Let us verify the network codiagnosability of the system considered in Example 4.5. The new verifier $\bar{G}_V$ for this example was computed using the computational tool called DESLAB [72]. This verifier has 344 states, 963 transitions and was found that $\bar{G}_V$ has no cyclic paths that satisfy the condition (4.14). Therefore, the system is network codiagnosable with respect to communication delays and loss of observations. Because of the size of the verifier, we will consider only one trace to illustrate the verification procedure. Following the same procedure as Example 4.6, automata $\bar{G}_{1,\rho}, \bar{G}_{1,F}, \bar{G}_{2,\rho}$ and $\bar{G}_{2,F}$ are computed and shown in Figures 4.15(a),*

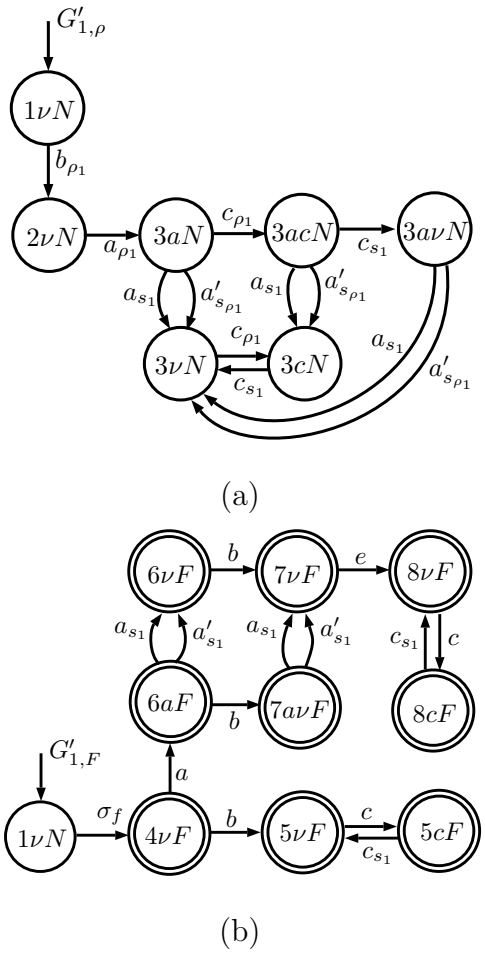


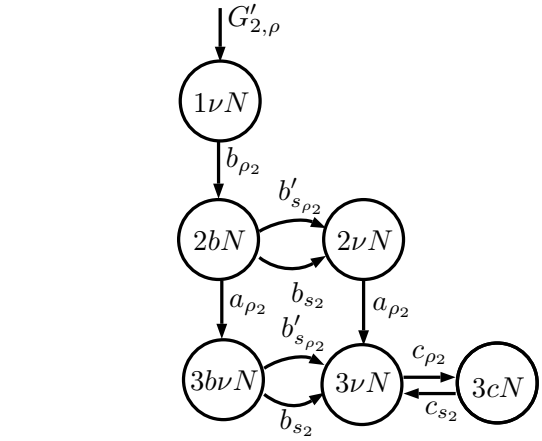
Figure 4.11: (a) Automaton $G'_{1,\rho}$, (b) Automaton $G'_{1,F}$.

4.15(b), 4.16(a) and 4.16(b), respectively. Continuing Algorithm 4.2, verifiers $\bar{V}_1$ *and* $\bar{V}_2$ *are computed. Due to the size of these automata, we show in Figures 4.17(a) and 4.17(b), only the paths of* $\bar{V}_1$ *and* $\bar{V}_2$ *containing the cyclic paths* cl_1 *and* cl_2 *that satisfy condition (4.14). After that, automaton* $\bar{G}_V = \bar{V}_1 \| \bar{V}_2$ *is computed and shown in Figure 4.18. Notice that, as expected, there is no cyclic path in automaton* $\bar{G}_V$ *that satisfies condition (4.14).*

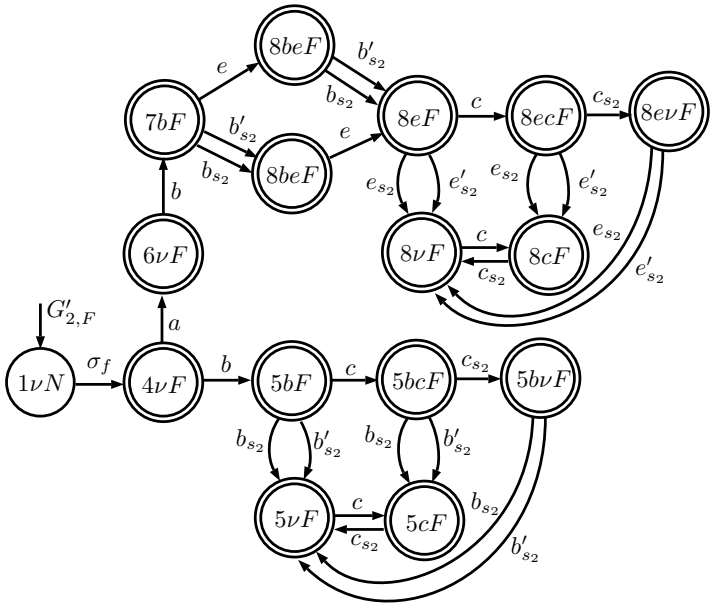
4.7 Complexity analysis of Algorithm 4.2

The computational complexity in the construction of verifier G_V, according to Algorithm 4.2, depends on the complexity of the computation of automata D_i, G_i, G'_i, and V_i, for $i = 1, \ldots, n$.

In the first step for the construction of automaton D_i according to Algorithm



(a)



(b)

Figure 4.12: (a) Automaton $G'_{2,\rho}$, (b) Automaton $G'_{2,F}$.

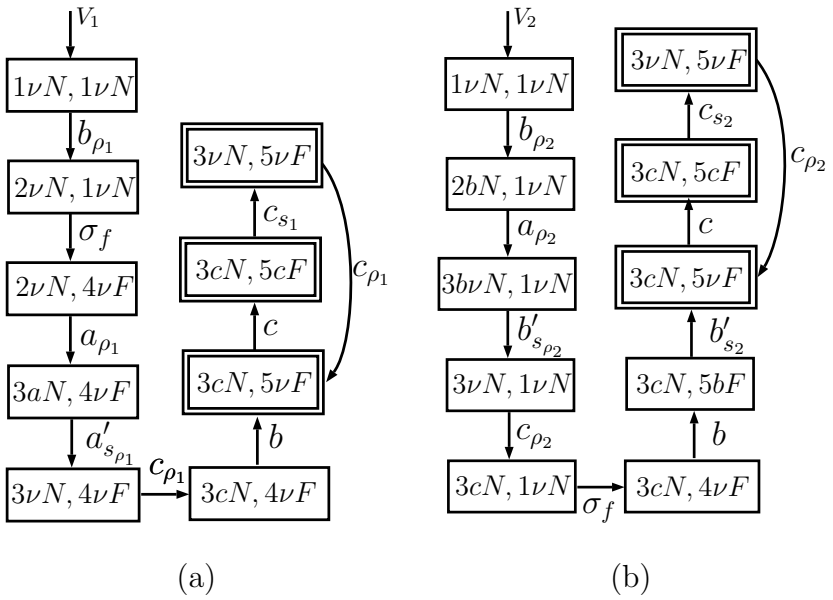


Figure 4.13: (a) Path of V_1 with cyclic path cl_1 embedded, (b) path of V_2 with cyclic path cl_2 embedded.

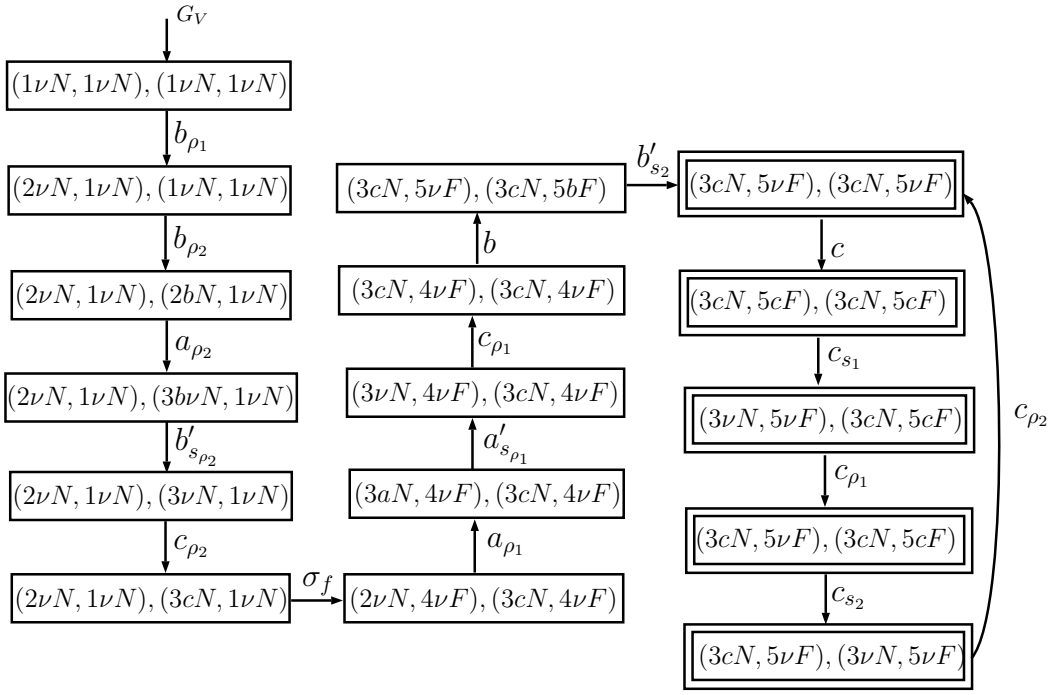
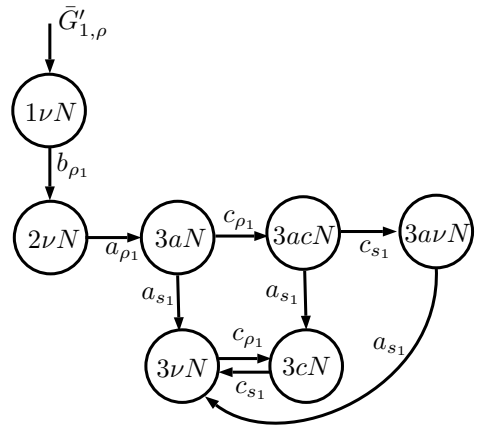
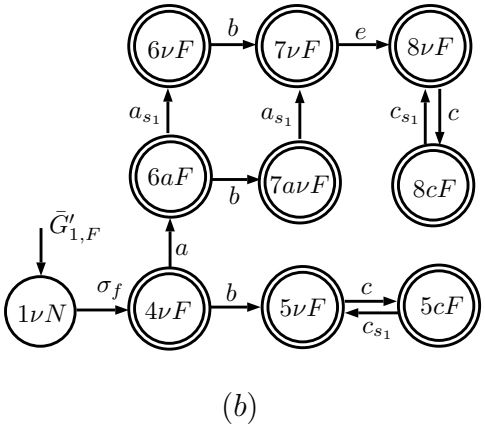


Figure 4.14: Path of G_V with an embedded cyclic path cl that violates the network codiagnosability of L.

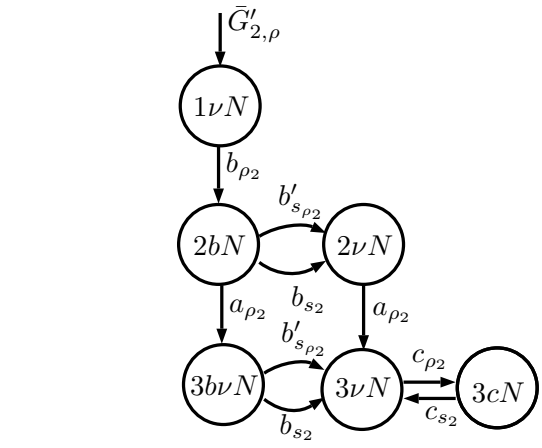


(a)

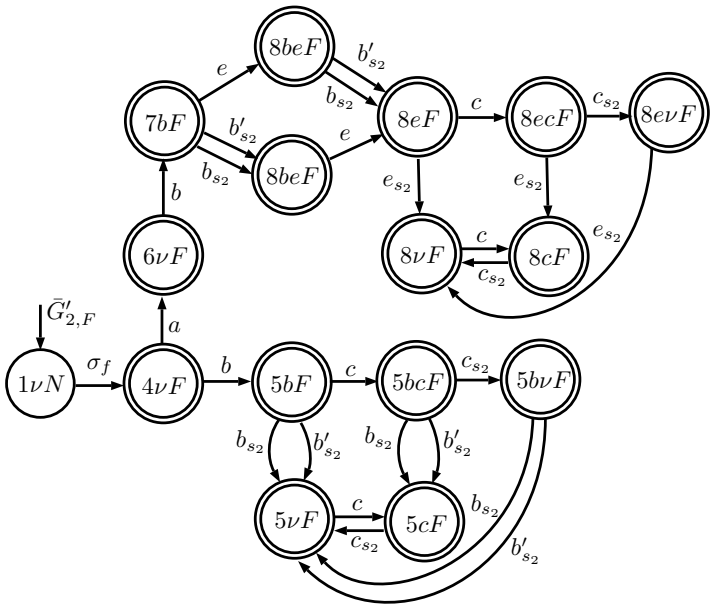


(b)

Figure 4.15: (a) Automaton $\bar{G}'_{1,\rho}$, (b) Automaton $\bar{G}'_{1,F}$.



(a)



(b)

Figure 4.16: (a) Automaton $\bar{G}'_{2,\rho}$, (b) Automaton $\bar{G}'_{2,F}$.

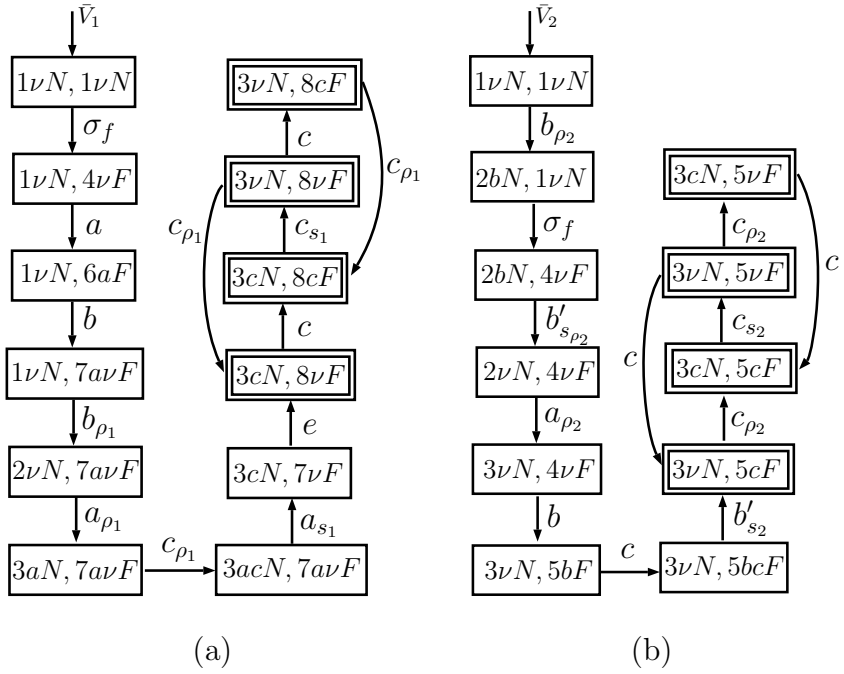


Figure 4.17: (a) Path of $\bar{V}_1$ with cyclic path cl_1 embedded, (b) path of $\bar{V}_2$ with cyclic path cl_2 embedded.

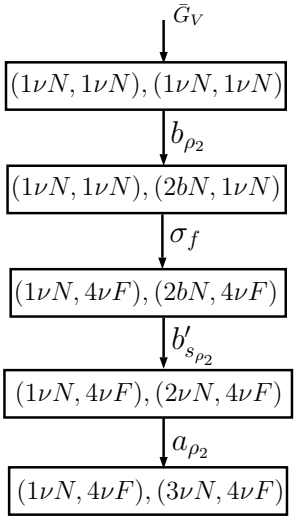


Figure 4.18: Part of verifier $\bar{G}_V$.

4.1, only one initial state is created. Then, from the initial state, $|\Sigma_o|$ states can be reached in the worst case, and for each one of these states, $|\Sigma_o| + 1$ states can be reached. The number of states created at each step of the construction of D_i depends on the delays of the communication channels. Thus, assuming that the maximum delay for all communication channels is k, then, in the worst case, the number of states of automaton D_i is $1 + \sum_{j=0}^{k}(|\Sigma_o|+1)^j \times |\Sigma_o|$. Since D_i is deterministic, then the maximum number of transitions of D_i is $\left(1 + \sum_{j=0}^{k}(|\Sigma_o| + 1)^j \times |\Sigma_o|\right) \times (|\Sigma| + |\Sigma_o|)$.

Automaton G_i is computed by the parallel composition of automata G and D_i. Let $|X_{D_i}|$ be the number of states of automaton D_i. Since $|X|$ is the number of states of G, then the number of states and transitions of G_i are, respectively, $|X| \times |X_{D_i}|$ and $|X| \times |X_{D_i}| \times |\Sigma_i|$.

Since automaton G_i' is computed by introducing a transition labeled with an event $\sigma' \in \Sigma_{i,ilo}^{s\prime}$ in parallel with the transitions of G_i labeled with $\sigma \in \Sigma_{i,ilo}^{s}$, the number of states and transitions of G_i' are, in the worst case, $|X| \times |X_{D_i}|$ and $|X| \times |X_{D_i}| \times |\Sigma_i'|$, respectively. Since $\Sigma_i' = \Sigma \cup \Sigma_{o_i}^{s} \cup \Sigma_{i,ilo}^{s\prime}$, the maximum number of events in Σ_i' is $|\Sigma| + 2 \times |\Sigma_o|$. Thus, the number of transitions in G_i' is, in the worst case, $|X| \times |X_{D_i}| \times (|\Sigma| + 2 \times |\Sigma_o|)$.

Following the complexity analysis presented in [57], each verifier V_i has, in the worst case, $2 \times (|X| \times |X_{D_i}|)^2$ states and $2 \times (|X| \times |X_{D_i}|)^2 \times (2 \times |\Sigma| + 2 \times |\Sigma_o| - |\Sigma_f|)$ transitions. Thus, since $G_V = \|_{i=1}^{n} V_i$, the maximum number of states and transitions of G_V are, respectively, $2^n \times |X|^{2n} \times \Pi_{i=1}^{n}|X_{D_i}|^2$ and $2^n \times |X|^{2n} \times \Pi_{i=1}^{n}|X_{D_i}|^2 \times [|\Sigma| + n(2 \times |\Sigma| + 2 \times |\Sigma_o| - |\Sigma_f|)]$. Therefore, the complexity of Algorithm 4.2 is $O(n \times 2^n \times |X|^{2n} \times \Pi_{i=1}^{n}|X_{D_i}|^2 \times |\Sigma|)$. In other words, the Algorithm 4.2 has exponential complexity with respect to maximum communication delay and number of local diagnosers.

4.8 Concluding Remarks

The goal of this chapter was, based on the possible occurrence of delays and event observation losses in communication networks, to consider its consequence on diagnosis systems modeled using discrete event systems theory. To this end, we propose an algorithm to construct an automaton model that describes the event delays and

all possible changing of order of event observations by the local diagnosers. Moreover, we proposed an automaton model that describes, both all possible order of observation and intermittent loss of observation of events. We introduce the definition of network codiagnosability and present a necessary and sufficient condition for network codiagnosability based on a finite deterministic automaton. Finally, we create an algorithm for the verification of network codiagnosability.

Chapter 5

Conclusion and Future Works

The goal of this work was to analyze the codiagnasobility of networked discrete event systems subject to delays and losses of observation. In this sense, the main contributions of this work are:

1. The introduction of a network codiagnosis architecture based on protocol 3 of [30] for a distributed plant with different measurement sites.

2. A systematic way to model communication delay by appropriately modifying the plant automaton into another one whose language considers all possible change in the order of observation of events due to possible delays and loss of observation.

3. The definition of network codiagnosability.

4. A verification algorithm for network codiagnosability, and based on a verifier automaton, a necessary and sufficient condition for network codiagnosability.

The automaton model of a networked discrete event system subject to communication delay and losses of events proposed in this work is general enough to allow its application in other research topics in DES such as supervisory control of networked discrete-event systems and to deal with malicious attacks/intrusions to cyberphysical systems.

Possible topics of research that can continue this work are listed below:

(i) Supervisory control of networked discrete event system. In this topic, the supervisor can observe events in an order different from the original event

occurrence in the plant. Thus, we need to construct a supervisor that, in spite of observing traces in different order from that occurred in the plant, it makes appropriate control decisions.

(ii) The decentralized diagnosis system proposed in [47] considers the delays between local diagnosers and the coordinator over protocols $1D$ and $2D$, while in this work we consider delays between sensors and diagnosers over protocol 3 of [30]. Thus, an interesting topic of research is the combination of both approaches in order to construct a general model of delays and loss of observation.

(iii) The development of a platform to implement a networked discrete event system and to test it considering the delays and losses of observation. In addition, it is important to research industrial network communication technologies, such as, FieldBus and ProfiBus and to evaluate the efficiency of the decentralized diagnosis scheme proposed in this work.

(iv) To improve the communication delay model, D_i, in order to decrease, if possible, its complexity from exponential to polynomial.

From this thesis, some works have been either published 0r submitted for publication.

(i) Codiagnostcabilidade robusta a atrasos na comunicação da ocorrência de eventos em sistemas a eventos discretos [56].

(ii) Network codiagnosability of Discrete-Event Systems subject to event communication delays [55]. This work was finalist for best student paper in WODES 2016.

(iii) Codiagnosticabilidade em rede de sistemas a eventos discretos sujeita a atrasos e perdas de observação de eventos [73].

(iv) Codiagnosability of Networked Discrete Event Systems subject to communication delays and intermittent loss of observation. This work has been submitted for a special issue of Journal of Discrete Event Dynamic Systems.

Bibliographic References

[1] VENKATASUBRAMANIAN, V., RENGASWAMY, R., YIN, K., et al., "A review of process fault detection and diagnosis: Part I: Quantitative model-based methods", *Computers & Chemical Engineering*, v. 27, n. 3, pp. 293 – 311, 2003.

[2] ZAYTOON, J., LAFORTUNE, S., "Overview of fault diagnosis methods for Discrete Event Systems", *Annual Reviews in Control*, v. 37, n. 2, pp. 308–320, 2013.

[3] SAMPATH, M., SENGUPTA, R., LAFORTUNE, S., et al., "Diagnosability of discrete-event systems", *IEEE Transactions on Automatic Control*, v. 40, n. 9, pp. 1555–1575, 1995.

[4] SAMPATH, M., SENGUPTA, R., LAFORTUNE, S., et al., "Failure diagnosis using discrete event models", *IEEE Transactions on Control Systems Technology*, v. 4, n. 2, pp. 105–124, 1996.

[5] LIN, F., "Diagnosability of discrete event systems and its applications", *Journal of Discrete Event Dynamic Systems*, v. 4, n. 2, pp. 197–212, 1994.

[6] GENC, S., LAFORTUNE, S., "Predictability in discrete-event under partial observation", *IFAC Symposium on Fault Detection, Supervision on Safety of Technical Process, Beijing, China*, 2006.

[7] KUMAR, R., TAKAI, S., "Decentralized Prognosis of Failures in Discrete Event System", *IEEE Transaction on Automatic Control*, v. 55, pp. 48–59, 2010.

[8] CHELIDZE, D., CUSUMANO, J. P., "A dynamical systems approach to failure prognosis", *Journal of Vibration and Acoustics*, v. 126, n. 1, pp. 2–8, 2004.

[9] JERON, T., MARCHAND, H., GENC, S., et al., "Predictability of sequence patterns in Discrete Event Systems", *17th World Congress The International Federation of Automatic Control*, v. 41, n. 2, pp. 537–543, 2008.

[10] TAKAI, S., KUMAR, R., "Inference-Based Decentralized Prognosis in Discrete Event Systems", *IEEE Transactions on Automatic Control*, v. 56, n. 1, pp. 165–171, 2011.

[11] YIN, X., LI, Z., "Reliable Decentralized Fault Prognosis of Discrete-Event Systems", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, v. PP, n. 99, pp. 1–6, 2015.

[12] KHOUMSI, A., CHAKIB, H., "Conjunctive and Disjunctive Architectures for Decentralized Prognosis of Failures in Discrete-Event Systems", *IEEE Transactions on Automation Science and Engineering*, v. 9, n. 2, pp. 412–417, 2012.

[13] KHOUMSI, A., CHAKIB, H., "Multi-decision decentralized prognosis of failures in discrete event systems". In: *2009 American Control Conference*, pp. 4974–4981, 2009.

[14] DEBOUK, R., LAFORTUNE, S., TENEKETZIS, D., "On an Optimization Problem in Sensor Selection", *Discrete Event Dynamic Systems*, v. 12, n. 4, pp. 417–445, 2002.

[15] CASSEZ, F., TRIPAKIS, S., "Fault diagnosis with static and dynamic observers", *Fundamenta Informaticae*, v. 88, n. 4, pp. 497–540, 2008.

[16] CASSEZ, F., TRIPAKIS, S., ALTISEN, K., "Synthesis of optimal-cost dynamic observers for fault diagnosis of discrete-event systems". In: *First Joint IEEE/IFIP Symposium on Theoretical Aspects of Software Engineering (TASE'07)*, pp. 316–325, 2007.

[17] THORSLEY, D., TENEKETZIS, D., "Active acquisition of information for diagnosis and supervisory control of discrete event systems", *Discrete Event Dynamic Systems*, v. 17, n. 4, pp. 531–583, 2007.

[18] SANTORO, L. P., MOREIRA, M. V., BASILIO, J. C., et al., "Computation of minimal diagnosis bases of Discrete-Event Systems using verifiers: Method of the ambiguous cyclic paths", *12th International Workshop on Discrete Event Systems*, v. 47, n. 2, pp. 440–445, 2014.

[19] BASILIO, J. C., LIMA, S. T. S., LAFORTUNE, S., et al., "Computation of minimal event bases that ensure diagnosability", *Discrete Event Dynamic Systems*, v. 22, n. 3, pp. 249–292, 2012.

[20] JIANG, S., KUMAR, R., GARCIA, H. E., "Optimal sensor selection for discrete-event systems with partial observation", *IEEE Transactions on Automatic Control*, v. 48, n. 3, pp. 369–381, 2003.

[21] ATHANASOPOULOU, E., LI, L., HADJICOSTIS, C., "Probabilistic failure diagnosis in finite state machines under unreliable observations", *8th International Workhop on Discrete Event Systems, Ann Arbor, Michigan, USA*, pp. 301–306, 2006.

[22] THORSLEY, D., YOO, T. S., GARCIA, H. E., "Diagnosability of stochastic discrete event systems under unreliable observations", *American Control Conference, Seattle, Washington, USA*, pp. 1158–1165, 2008.

[23] BASILIO, J. C., LAFORTUNE, S., "Robust codiagnosability of discrete event systems", *American Control Conference, St. Louis, MO, USA*, pp. 2202–2209, 2009.

[24] TAKAI, S., "Robust failure diagnosis of partially observed discrete event systems". In: *10th Word Congress International Federation of Automatic Control*, v. 43, n. 12, pp. 205–210, 2010.

[25] CARVALHO, L. K., MOREIRA, M. V., BASILIO, J. C., "Generalized robust diagnosability of discrete event systems". In: *18th World Congress International Federation of Automatic Control*, pp. 8737–8742, 2011.

[26] CARVALHO, L. K., BASILIO, J. C., MOREIRA, M. V., "Robust diagnosis of discrete event systems against intermittent loss of observations", *Automatica*, v. 48, n. 9, pp. 2068–2078, 2012.

[27] CARVALHO, L. K., MOREIRA, M. V., BASILIO, J. C., et al., "Robust diagnosis of discrete-event systems against permanent loss of observations", *Automatica*, v. 49, n. 1, pp. 223–231, 2013.

[28] LIMA, S. S., BASILIO, J. C., LAFORTUNE, S., et al., "Robust diagnosis of discrete-event systems subject to permanent sensor failures", *IFAC Proceedings Volumes*, v. 43, n. 12, pp. 90–97, 2010.

[29] TAKAI, S., "Verification of robust diagnosability for partially observed discrete event systems", *Automatica*, v. 48, n. 8, pp. 1913–1919, 2012.

[30] DEBOUK, R., LAFORTUNE, S., TENEKETZIS, D., "Coordinated decentralized protocols for failure diagnosis of discrete event systems", *Discrete Event Dynamic Systems: Theory and Applications*, v. 10, n. 1-2, pp. 33–86, 2000.

[31] PATTON, R. J., FRANK, M., P., et al., *Issue of fault diagnosis for dynamic systems*. Springer Science & Business Media, 2013.

[32] ZAD, S. H., KWONG, R. H., WONHAM, W. M., "Fault diagnosis in discrete-event systems: framework and model reduction", *IEEE Transactions on Automatic Control*, v. 48, n. 7, pp. 1199–1212, 2003.

[33] ZAD, S. H., KWONG, R., WONHAM, W., "Fault diagnosis in discrete-event systems: incorporating timing information", *IEEE Transactions on Automatic Control*, v. 50, n. 7, pp. 1010–1015, 2005.

[34] QIU, W., KUMAR, R., "Decentralized failure diagnosis of discrete event systems", *IEEE Transactions on Systems, Man and Cybernetics, Part A*, v. 36, n. 2, pp. 384–395, 2006.

[35] WANG, Y., YOO, T. S., LAFORTUNE, S., "Diagnosis of discrete event systems using decentralized architectures", *Discrete Event Dynamic Systems: Theory And Applications*, v. 17, n. 2, pp. 233–263, 2007.

[36] DOTOLI, M., FANTI, M. P., MANGINI, A. M., et al., "On-line fault detection in discrete event systems by Petri nets and integer linear programming", *Automatica*, v. 45, n. 11, pp. 2665–2672, 2009.

[37] SAMPATH, M., "A hybrid approach to failure diagnosis of industrial systems". In: *American Control Conference, Arlington, VA.*, v. 3, pp. 2077–2082, 2001.

[38] SIMSEK, H. T., SENGUPTA, R., YOVINE, S., et al., *Fault Diagnosis for Intra-platoon Communications*, Tech. rep., California Partners for Advanced Transit and Highways (PATH), 1999.

[39] KOFMAN, E., "Discrete event simulation of hybrid systems", *SIAM Journal on Scientific Computing*, v. 25, n. 5, pp. 1771–1797, 2004.

[40] CABASINO, M. P., GIUA, A., POCCI, M., et al., "Discrete event diagnosis using labeled Petri nets. An application to manufacturing systems", *Control Engineering Practice*, v. 19, n. 9, pp. 989–1001, 2011.

[41] CHEN, Z., LIN, F., WANG, C., et al., "Active Diagnosability of Discrete Event Systems and its Application to Battery Fault Diagnosis", *IEEE Transactions on Control Systems Technology*, v. 22, n. 5, pp. 1892–1898, Sept 2014.

[42] NUNES, C. E. V., BASILIO, J. C., SOTOMAYOR, O. A. Z., "Diagnóstico de falhas em uma unidade de separação água-óleo-gás usando um modelo a eventos discretos". In: *XIX Congresso Brasileiro de Automática*, 2012.

[43] LUGLI, A. B., SANTOS, M. M. D., *Redes industriais características, padrões e aplicações*. 1st ed. Saraiva, 2014.

[44] SHU, S., LIN, F., "Supervisor Synthesis for Networked Discrete Event Systems With Communication Delays", *IEEE Transactions on Automatic Control*, v. 60, n. 8, pp. 2183–2188, 2015.

[45] HUO, Z., FANG, H., MA, C., "Networked control System: State of the art". In: *Proceedings of the* 5^{th} *World Congress on Intelligent Control and Automation*, pp. 1319–1322, Hangzhou, P.R. China, 2004.

[46] ATHANASOPOULOU, E., LINGXI, L., HADJICOSTIS, C., "Maximum Likelihood Failure Diagnosis in Finite State Machines Under Unreliable Ob-

servations", *IEEE Transactions on Automatic Control*, v. 55, n. 3, pp. 579 –593, 2010.

[47] DEBOUK, R., LAFORTUNE, S., TENEKETZIS, D., "On the effect of communication delays in failure diagnosis of decentralized discrete event systems", *Discrete Event Dynamic Systems*, v. 13, n. 3, pp. 263–289, 2003.

[48] TRIPAKIS, S., "Decentralized control of discrete-event systems with bounded or unbounded delay communication", *IEEE Transactions on Automatic Control*, v. 49, n. 9, pp. 1489–1501, 2004.

[49] PARK, SEONG-JIN, CHO, et al., "Supervisory control of discrete event systems with communication delays and partial observations", *Systems & Control letters*, v. 56, n. 2, pp. 106–112, 2007.

[50] PARK, S.-J., CHO, K.-H., "Decentralized supervisory control of discrete event systems with communication delays based on conjunctive and permissive decision structures", *Automatica*, v. 43, n. 4, pp. 738–743, 2007.

[51] LIN, F., "Control of networked discrete event systems". In: *Control and Decision Conference (CCDC), China*, pp. 51–56, May 2012.

[52] SHU, S., LIN, F., "Supervisor synthesis for networked discrete event systems with communication delays". In: *Control Conference (CCC), Xian, China*, pp. 2078–2084, July 2013.

[53] SHU, S., LIN, F., "Decentralized control of networked discrete event systems with communication delays", *Automatica*, v. 50, n. 8, pp. 2108 – 2112, 2014.

[54] QIU, W., KUMAR, R., "Distributed diagnosis under bounded delay communication of immediately forwarded local observations", *IEEE Transactions on Systems, Man and Cybernetics, Part A*, v. 38, n. 3, pp. 628–642, 2008.

[55] NUNES, C. E. V., MOREIRA, M. V., ALVES, M. V. S., et al., "Network co-diagnosability of Discrete-Event Systems subject to event communication delays". In: *2016 13th International Workshop on Discrete Event Systems (WODES)*, pp. 217–223, 2016.

[56] NUNES, C. E. V., MOREIRA, M. V., BASILIO, J. C., "Codiagnosticabilidade robusta a atrasos na comunicação da ocorrência de eventos em sistemas a eventos discretos". In: *XII Simpósio Brasileiro de Automação Inteligente*, 2015.

[57] MOREIRA, M. V., JESUS, T. C., BASILIO, J. C., "Polynomial Time Verification of Decentralized Diagnosability of Discrete Event Systems", *IEEE Transactions on Automatic Control*, v. 56, n. 7, pp. 1679–1684, 2011.

[58] CASSANDRAS, C. G., LAFORTUNE, S., *Introduction to Discrete Event Systems*. 2nd ed. Springer: New York, 2008.

[59] BASILIO, J. C., CARVALHO, L. K., MOREIRA, M. V., "Diagnose de falhas em sistemas a eventos discretos modelados por autômatos finitos", *Revista Controle & Automaçao*, v. 21, n. 5, pp. 510–533, 2010.

[60] TANENBAUM, A. S., *Network computer*. 4th ed. Prentice Hall, 2002.

[61] CHAN, M. C., TSENG, C. C., YEN, L. H., "Jitter-aware packet scheduler for concurrent multipath transmission in heterogeneous wireless networks". In: *2016 IEEE Wireless Communications and Networking Conference*, pp. 1–7, April 2016.

[62] YANG, T. C., "Networked control system: a brief survey", *IEEE Proceedings Control Theory and Applications*, v. 153, n. 4, pp. 403, 2006.

[63] GODOY, E. P., LOPES, W. C., SOUSA, R. V., et al., "Modelagem e simulação de redes de comunicação baseadas no protocolo CAN-Controller Area Network", *Sba: Controle & Automação Sociedade Brasileira de Automática*, v. 21, n. 4, pp. 425–438, 2010.

[64] LIAN, F., MOYNE, J. R., TILBURY, D. M., "Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet", *IEEE Control Systems*, v. 21, n. 1, pp. 66–83, 2001.

[65] VARGAS-RODRIGUEZ, R., MORALES-MENENDEZ, R., "Network-Induced Delay Models for CAN-based Networked Control Systems", *IFAC Proceedings Volumes*, v. 40, n. 22, pp. 85–92, 2007.

[66] RAICIU, C., PAASCH, C., BARRE, S., et al., "How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP". In: *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, pp. 29–29, USENIX Association: Berkeley, CA, USA, 2012.

[67] ADHARI, H., DREIBHOLZ, T., BECKE, M., et al., "Evaluation of Concurrent Multipath Transfer over Dissimilar Paths". In: *2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications (WAINA)*, pp. 708–714, March 2011.

[68] KUROSE, J. F., ROSS, K. W., *Redes de computadores e a internet: uma abordagem top-down*. 5th ed. Pearson, 2010.

[69] MORAES, C. C., CASTRUCCI, P. L., *Engenharia de automação industrial*. 2nd ed. LTC, 2015.

[70] ROHLOFF, K. R., "Sensor failure tolerant supervisory control". In: *44th IEEE Conference on Decision and Control,and the European Control Conference 2005, Seville, Spain*, pp. 3493–3498, 2005.

[71] SÁNCHEZ, A. M., MONTOYA, F., "Safe supervisory control under observability failure", *Discrete Event Dynamic Systems*, v. 16, n. 4, pp. 493–525, 2006.

[72] BERMEO, L. E., BASILIO, J. C., CARVALHO, L. K., "DESLAB: a scientific computing program for analysis and synthesis of discrete-event systems." In: *11th International Workshop on Discrete Event Systems*, v. 45, n. 29, pp. 349–355, 2012.

[73] NUNES, C. E. V., MOREIRA, M. V., ALVES, M. V. S., et al., "Codiagnosticabilidade em rede de sistemas a eventos discretos sujeita a atrasos e perdas de observação de eventos". In: *XXI Congresso Brasileiro de Automática*, 2016.

[74] KEROGLOU, C., HADJICOSTIS, C. N., "Distributed diagnosis using predetermined synchronization strategies in the presence of communication con-

straints". In: *2015 IEEE International Conference on Automation, Science and Engineering (CASE), Gothenburg, Sweden*, pp. 831–836, 2015.