CODIAGNOSABILITY OF NETWORKED DISCRETE EVENT SYSTEMS WITH TIMING STRUCTURE

Gustavo da Silva Viana

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.

Orientador: João Carlos dos Santos Basilio

Rio de Janeiro

Fevereiro de 2018

CODIAGNOSABILITY OF NETWORKED DISCRETE EVENT SYSTEMS WITH TIMING STRUCTURE

Gustavo da Silva Viana

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. Lilian Kawakami Carvalho, D.Sc.

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

Prof. Carlos Andrey Maia, D.Sc.

Prof. José Reinaldo Silva, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2018

Viana, Gustavo da Silva

Codiagnosability of Networked Discrete Event Systems with Timing Structure/Gustavo da Silva Viana. – Rio de Janeiro: UFRJ/COPPE, 2018.

XIV, 147 p.: il.; 29, 7cm.

Orientador: João Carlos dos Santos Basilio

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2018.

Referências Bibliográficas: p. 138 – 147.

1. Discrete event systems. 2. Networked systems. 3. Fault diagnosis. I. Basilio, João Carlos dos Santos. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

"Thus far the Lord has helped us".

(1 Samuel 7:12)

Acknowledgments

First, I will give thanks to you, Lord, with all my heart; I will tell of all your wonderful deeds.

I thank my mother and grandmother for my education, patience and the support every time. I could not have achieved my D.Sc. degree without them.

I deeply thank my advisor João Carlos Basilio for your patience and dedication to teach me.

I would like to thank Prof. Lilian Kawakami Carvalho for the support and the help during my postgraduate courses.

A special thank to my friend Marcos Vinicius for our nice discussions regarding our doctoral theses.

I thank the friends of Laboratory of Control and Automation (LCA), specially, Prof. Marcos Vicente Moreira, Felipe Cabral, Carlos Eduardo, Antonio Gonzalez, Ingrid Antunes, Raphael Barcelos, Alexandre Gomes, Juliano Freire, Públio Lima, Tiago França and Wesley Silveira.

I thank the National Council for Scientific and Technological Development (CNPq) for the financial support.

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 COM ESTRUTURA TEMPORIZADA

Gustavo da Silva Viana

Fevereiro/2018

Orientador: João Carlos dos Santos Basilio

Programa: Engenharia Elétrica

Considera-se, neste trabalho, o problema da codiagnosticabilidade de sistemas discretos de eventos em rede com estrutura de temporização (NDESWTS) sujeitos a atrasos e perdas de observações de eventos entre os locais de medição e diagnosticadores locais e, para esse propósito, apresenta-se um novo modelo com temporização que representa o comportamento dinâmico da planta com base no conhecimento prévio do tempo mínimo de disparo para cada transição e dos atrasos máximos nos canais de comunicação que conectam LM e DL. Em seguida, converte-se o modelo temporizado em um não temporizado e adiciona-se possíveis perdas intermitentes de pacotes na rede de comunicação. Com base nesse modelo não temporizado, condições necessárias e suficientes para a codiagnosabilidade de NDESWTS são apresentadas e dois testes para sua verificação são propostos: um utilizando diagnosticadores e outro que utiliza verificadores. Um outro tópico de pesquisa abordado neste trabalho é o cálculo da τ-codiagnosticabilidade (tempo máximo para diagnosticar uma ocorrência de falha) e K-codiagnosticabilidade (número máximo de eventos para diagnosticar uma ocorrência de falha) também usando diagnosticadores e verificadores.

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 WITH TIMING STRUCTURE

Gustavo da Silva Viana

February/2018

Advisor: João Carlos dos Santos Basilio

Department: Electrical Engineering

We address, in this work, the problem of codiagnosability of networked discrete event systems with timing structure (NDESWTS) subject to delays and loss of observations of events between the measurement sites (MS) and local diagnosers (LD). To this end, we first introduce a new timed model that represents the dynamic behavior of the plant based on the, *a priori*, knowledge of the minimal firing time for each transition of the plant and on the maximal delays in the communication channels that connect MS and LD. We then convert this timed model into an equivalent untimed one, and add possible intermittent packet loss in the communication network. Based on this untimed model, we present necessary and sufficient conditions for NDESWTS codiagnosability and propose two tests for its verification: one that deploys diagnosers and another one that uses verifiers. Another topic addressed in this work is the computation of τ-codiagnosability (maximal time to diagnose a failure occurrence) and K-codiagnosability (maximal number of event occurrences necessary to diagnose a failure). To this end, we propose two tests: (i) one test based on a diagnoser-like that does not require usual assumptions on language liveness and nonexistence of unobservable cycles and (ii) another one based on the extended verifier that shows not only the ambiguous paths but also those paths that lead to language diagnosis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, new challenges to make production processes more efficient, autonomous and customizable have led to a new industrial revolution. A new concept of industry, called Industry 4.0 [1, 2], has emerged and is currently adopted to denominate the current trend of automation and data exchange in manufacturing technologies by creating a "smart factory" [3]. One of the fundamentals of Industry 4.0 is the Cyber-Physical systems (CPS) [4–7]. CPS can be regarded as a mechanism that is controlled or monitored by computer-based algorithms, tightly integrated with the Internet and its users. Examples of CPS include smart grid, autonomous automobile systems, process control systems, robotics systems and manufacturing systems.

An important class of Cyber-Physical systems is called Discrete Event Systems (DES) [8–12], which are event-driven dynamic systems, with discrete state spaces. Such systems arise in a variety of contexts ranging from computer operation systems to the control and monitoring of large scale complex processes. Several problems in the literature have been solved by using DES modeling tools; among them, we mention: supervisory control [8, 11, 13], opacity [14, 15], detectability [16], prognosis [17], failure diagnosis [18–22], and many other topics.

In order to supply the demand for interconnected devices, communication networks are more widely used in engineering systems [23, 24], since devices are usually positioned far away from each other in a distributed system. Although there are benefits, the use of communication networks have introduced problems such as communication delays and loss of information [25, 26]. Time delays come from com-

putation 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. In order to deal with these problems, some relevant theoretical approaches have been considered in the literature, regarding supervisory control [27–39] and failure diagnosis [40–43] of networked discrete event systems.

We address, in this thesis, the problem of failure diagnosis of DES subject to delays and losses in the transmission of observed events from measurement sites (MS) to local diagnosers (LD). Such a failure diagnosis problem is referred, in the literature to as codiagnosability of networked discrete event systems [42, 43]. In addition, we revisit the codiagnosability of discrete event system problem in order to provide some contributions as follows. The verification of diagnosability of discrete event systems by using diagnoser proposed by SAMPATH *et al.* [19] and the verification of decentralized diagnosability (codiagnosability) of discrete event systems by using diagnoser DEBOUK *et al.* [21] have some drawbacks, since the authors assume language liveness and nonexistence of unobservable cycles of states connected with unobservable events only. To overcome these limitations, we propose a new algorithm to check codiagnosability by changing the diagnoser structure so as to consider both observable and unobservable events, therefore, removing the assumptions imposed in [19, 21]. Regarding codiagnosability verification of discrete event systems by using verifiers [44–47], MOREIRA *et al.* [47] proposed a verifier whose language stops just before the language becomes diagnosable. As a consequence, the event that removes the ambiguity is not shown in the verifier. We propose here an algorithm to extend the verifier automaton proposed in [47] to show the complete paths that lead to the failure diagnosis. As an application of these algorithms, we compute by means of weighted automaton formalism [48, 49] the maximum time that the diagnosis system takes to detect the failure occurrence (τ-codiagnosability) and the maximum number of events that occur after the failure occurrence until the diagnosis system becomes sure of the failure occurrence (K-codiagnosability).

1.1 Failure Diagnosis and Computation of τ- and K-Codiagnosability

Failure detection and isolation has received a lot of attention in recent years and has become a well-established area of research [18–22,44–47,50–56]. A failure is defined to be any deviation of a system from its normal or intended behavior. Diagnosis is the process of detecting an abnormality in the system behavior and isolating the cause or the source of this abnormality. Failures in industrial systems could arise from several sources such as design errors, equipment malfunctions, operator mistakes, and so on. In the design of a failure diagnosis system for DES, the first step is to check whether the language generated by an automaton is diagnosable, *i.e.*, whether the system is able to diagnose the failure occurrence in a finite number of events occurrences.

Several approaches have been proposed in the DES literature to check this property using diagnoser or verifier. A diagnoser is an automaton whose states are sets formed with the states of the automaton that models the plant together with labels that indicate if the trace occurred so far possesses or not the fault event. An advantage of diagnosers is that they can be used for both on-line and off-line purposes. Verifiers have been proposed in [44–47], and are, widely speaking, obtained by performing a parallel composition between the failure system behavior and the system behavior without failure. The disadvantage of verifiers is that they are suitable only off-line verification purposes. It is well known that diagnosers have, in the worst case, exponential complexity in the plant state-space as opposed to verifiers that have polynomial computational complexity in the number of the states of the automaton that generates the language [44–47]. However, it has been recently conjectured in [57], based on experimental evidences, that the diagnosers built in accordance with [19] have state size $\Theta(n^{0.77 \log k + 0.63})$, on the average, where k (resp. n) is the number of events (resp. states) of the plant automaton. This result is encouraging in the sense that it makes diagnosers a viable tool for diagnosability analysis as well. However, diagnosability analysis using diagnosers still requires the search for cycles which has computation complexity worse than exponential [58], as opposed to diagnosability analysis using verifiers that requires the search for strongly connected

components, which is linear in the number of automaton transitions [59, 60].

Although the diagnosis of a failure is an important issue regarding safety of industrial automation systems, it is also important to know how long the diagnosis system takes to detect the failure occurrence (τ-codiagnosability), or, in the context of discrete event system models, how many events must occur after the occurrence of the failure event in order for the diagnosis system to be sure of its occurrence (K-codiagnosability). In order to address these concerns, we need to analyze the diagnoser characteristics. Notice that the diagnoser proposed by [19] provides information on the failure occurrence based solely on observable events, *i.e.*, those events whose occurrence can be recorded by sensors. Therefore, when diagnosers are used offline to predict the time spent to diagnose the failure, it is not possible to take into account the time interval between occurrences of observable events that have unobservable events in-between. Another aspect regarding diagnosis is that liveness and nonexistence of unobservable cycles in the plant automaton are actually assumed for both diagnosability and codiagnosability verification using SAMPATH *et al.*'s and DEBOUK et al.'s diagnosers. For this reason, K-diagnosability was defined by [19] and [22] as the number of observable events that must occur after the failure occurrence in order for the diagnoser to be sure about the failure occurrence. We remove here all of these assumptions and propose a diagnoser-based test that also takes into account unobservable events and does not require the search for cycles, but for strongly connected components.

In this thesis, we change the diagnoser structure so as to consider both observable and unobservable events. The main advantages of the approach proposed here are as follows: *(i)* diagnosability verification becomes a particular case of codiagnosability verification as opposed to [19] and [21], which requires two different tests for diagnosability and codiagnosability; *(ii)* it does not require the usual assumptions on language liveness and non-existence of cycles of states connected with unobservable events [19, 21, 54]; *(iii)* it is based on the search for strongly connected components, as opposed to cycles in the usual tests using diagnosers [19, 21, 54] ; *(iv)* we can address τ-codiagnosability by adding weights associated with transitions of the automaton, forming, therefore, the so-called weighted automaton. It is worth remarking that weighted automata have only been employed for performance measure

in the context of supervisory control of discrete event system. Thus, the approach proposed here reduces to the step counting by replacing all transition weights with unity weight, and therefore, K-codiagnosability analysis becomes a particular case of τ-codiagnosability.

We now present a brief comparison between the methods proposed here to compute τ- and K-codiagnosability and those found in the literature.

QIU and KUMAR proposed in [45] an algorithm to compute the maximum delay of codiagnosability using a verifier, also proposed in [45]. However, the proposed approach cannot be extended to τ-codiagnosability since the value of K is computed by adding 1 to the longest trace of the verifier deployed in [45] because the next event, *i.e.,* the event that removes the ambiguity is not shown in the verifier; therefore it is not possible to take into account its time in the computation of τ-codiagnosability.

TOMOLA *et al.* [61] proposed an algorithm for the computation of the delay bound for robust disjunctive decentralized diagnosis based on the algorithm for the computation of the delay bound in the non-robust case using the verifier automaton proposed in [47]. Like the strategy developed in [45], the extension to τ-diagnosability is not straightforward.

YOO and GARCIA [62] proposed an algorithm to compute K-diagnosability (referred there to as fault detection delay) using a verifier similar to F_i-verifier proposed in [46], which can have cycles, but those cycles have zero weight. As in the previous papers, it is not clear how to extend the approach proposed in [62] to K- and τ-codiagnosability.

VIANA *et al.* [63] proposed an algorithm to compute τ-diagnosability, which is a particular case using of the diagnoser developed in this work for codiagnosability, and compute the maximum time for failure diagnosis by using a max-plus matrix representation for the time-weighted automata [49]. The computational complexity of the approach proposed in [63] is, however, worse than both methods presented here.

More recently K- and τ-diagnosability were addressed by BASILE *et al.* [64] for labeled Timed Petri nets, using the centralized diagnoser proposed in [19]. As a consequence: *(i)* it is, in the worst case, exponential in the cardinality of the state space of the system model; *(ii)* it requires the usual assumptions on language liveness

and nonexistence of unobservable cycles of states connected with unobservable events only; *(iii)* it is based on the search for cycles, and; *(iv)* K-diagnosability is defined taking into account observable events only [22].

An important work on τ- and K-codiagnosability was presented by CASSEZ [65], in which the author approaches the decentralized failure diagnosis problem for discrete event systems modeled by finite automata (FA) and, timed systems modeled by timed automata (TA) [66]. For FA (resp. TA), he computes the maximum number of steps (resp. the maximum delay), both denoted as Δ, that are necessary for the detection of the failure occurrence. As far as τ-codiagnosability is concerned, the approach presented here is also different from [65, 67, 68]. In [67], Wonham's timed discrete event model [69] is deployed, and a sixth order polynomial algorithm in the size of the state space of the automaton that models the plant was proposed to verify and compute the maximum delay for failure diagnosis, whereas in [65, 68], diagnosability is defined using Alur & Dill timed automaton model [66, 70]. Here, we approach τ-codiagnosability by using weighted automata, which cannot be included in the class of timed automata, although they carry information on the maximum time between event occurrences, which makes them suitable to be used as a performance index. Therefore, in this regard, the work developed here and CASSEZ's work are incomparable. Regarding K-codiagnosability analysis, the computational complexity of the algorithm proposed by CASSEZ in [65] depends on the complexity of the verifier automaton used, and on the complexity of performing the search for the longest failure trace, which is quadratic in the number of transitions of the verifier, since binary search is used. Here, we propose a polynomial time algorithm based on an extension of the verifier automaton presented in [47], which has the smallest computational complexity among all verifiers presented in the literature [71], and whose search algorithm is linear in the number of transitions of the verifier; therefore, the algorithm proposed here to compute K-codiagnosability is more efficient than that proposed in [65]. A comparison between the computational complexity of the algorithm proposed in [65] and the algorithm proposed in this thesis is presented in Chapter 3.

1.2 Discrete Event Systems Subject to Communication Delays and Losses

Most of the works in the area of failure diagnosis of DES assume that all information is sent to the diagnoser in a seamless and immediate manner [19, 21]. However, due to the complexity of the plants, diagnosers are often implemented in a distributed way and, consequently, with the development of network technology, it has become more and more common in industry, communication system implementations by using shared communication networks [34]. In diagnosis systems based on communication networks, the intense data traffic in communication channels, or the long distance between measurement sites and diagnosers, may delay the information communicated through the channel. Thus, the diagnoser can observe events with some delay after its occurrence, and also, when multiple communication channels are deployed, in order different from their occurrence in the plant; thus, being led to make wrong decisions regarding failure occurrence. In addition, in the sending of information, losses may occur.

The problem of failure diagnosis of DES with delays in communication networks was first addressed by DEBOUK *et al.* [40] and QIU and KUMAR [41]. However, the problem addressed in this work is different from [40, 41]. First, the problem of decentralized failure diagnosis proposed in [40] is subject to communication delays between local diagnosers and the coordinator, under Protocols 1 and 2 of [21]. A key feature of Protocol 1 studied in [21] is the following: under the assumption that all communicated messages are received in the correct order by the coordinator, the coordinator is capable of tracking the state of the system as well as a centralized diagnoser, even though it does not have any knowledge of dynamics of the system. Protocol 2 studied in [21] is the following: if the system has no failure ambiguous traces (Definition 18 in [21]), and if the communicated messages are received in the correct order by the coordinator, the coordinator can identify exactly the same failure types as the centralized diagnoser even when the communication between local sites and the coordination is not continuous. In [40], it is assumed that the events received by the coordinator can be observed in a different order from the original order of occurrence; however, no delay between the measurement sites and

the diagnoser. Here we consider Protocol 3 of [21], since we deal with communication delays between the measurement sites and the local diagnosers. Finite delays in the communication between local diagnosers and coordinator are not assumed here since they do not affect the diagnosis decision. The problem proposed in this work is also different from the so-called distributed diagnosis scheme proposed in [41], where each local diagnoser can exchange information with the other local diagnosers to infer the failure event occurrence. In addition, in [41] the communication delay between two local diagnosers is considered equal, and it is assumed that there is no delay between the measurement sites and diagnosers. The problem of DES subject to unreliable observations of events was addressed in [54] and [61] (in the context of failure diagnosis) without considering communication networks. In this work, we model the loss of observation based on the technique proposed by CARVALHO *et al.* in [54]

In [42, 43], the definition of network codiagnosability of DES subject to event communication delays was introduced, where the concept of step [33, 35] was used to measure communication delays, *i.e.*, $k \in \mathbb{N}$ steps accounts for the occurrence of, at most, k events until the information of the event executed by the plant arrives at the local diagnoser. In this thesis, we propose a new approach called networked discrete event systems with timing structure (NDESWTS) by adding two parameters to the automaton that models the system behavior as follows: (i) the maximal time communication delays between the distributed measurement sites in the plant and the local diagnosers; (ii) a minimal time function that is associated with each plant automaton transition, which corresponds to the minimal time the system must remain in the state before the transition can fire. Regarding the modeling of the observation of the events by a local diagnoser subject to delay and losses, NUNES *et al.* [42, 43] carried out it in two steps as follows: in first step, an automaton that model the effects of the delays is proposed; in the second step, the automaton that models the observation of the events by a local diagnoser is computed by performing the parallel composition between the automaton obtained in the previous step and the automaton that models the plant. In this thesis, we propose an algorithm to consider the models the observation of the events by a local diagnoser subject to delay and losses without this intermediate step. To this end, we convert the

new timed model that represents the dynamic system behavior of the plant into an untimed one, and add possible intermittent packet loss in the communication network. We check codiagnosability of NDESWTS proposed here by developing two algorithms: one based on diagnosers and another based on verifiers.

We remark that the Timed Discrete Event System (TDES) model proposed by BRANDIN and WONHAM in [69], that introduces one additional event called *tick* to represent "tick of global clock", could also be used. The authors in [72–74] have addressed untimed model to deal with communication delays by introducing *tick* events in the plant to represent a clock cycle. The main limitation of that approach is when the system has far apart temporal characteristics since due to the fast system behavior, the *tick* will be associated with a small time interval, and, as consequence, the corresponding untimed model may have a large number of states to represent slow dynamics in the model. Regarding models to represent time information, timed automata [65, 66, 68, 70] could also be an option to model the time information. Timed automata provide a way to model the behavior of real-time systems over time by using state-transition graphs with timing constraints using finitely many real-valued clocks. However, the cost to obtain the behavior of real-time systems over time makes this formalism more complicated to model and analyze. Indeed, the construction of region graph to recognize untimed language of timed automata is $O(|L|\ fact(|X|)\ 2^{|X|}\ K^{|X|})$, where L is the number of locations (states), X is the number of clocks, K is the large constant used in timed automata, and $fact(|X|)$ denotes the factorial of the cardinality of X [65]. Thus, the adoption of this formalism implies unnecessary computational effort to the goal of this work.

It is important to remark that the problem of communication delays has also been addressed in the context of supervisory control of DES by [27, 30, 32, 33, 35], for the monolithic case, and by PARK and CHO [31], and [34] for the decentralized and distributed case. In the aforementioned works, it is assumed that there is only one communication channel between the plant and supervisor, and, thus, no change in the order of event observations by the supervisor occurs. Since codiagnosability is not time critical, i.e., the diagnoser can detect the fault after an arbitrarily large number of event occurrences, bounded communication delays that cannot change the order of event observation are not important in the context of failure diagnosis. We

here consider decentralized diagnosis of NDESWTS assuming that communication delays can be large enough that it can modify the order of observation of the events received by the local diagnosers. Still in the context of supervisory control, TRIPAKIS [28] and SADID *et al.* [29] assume that communication delays may change the order of event observation. One important restriction of these approaches is that the same delay upper bound is assumed for all communication channels. In addition, SADID *et al.* restricts the problem to those systems whose automaton models have no loops of communication events (events that are subject to communication delays) in the original system. None of these assumptions are made here.

1.3 Contributions of the Thesis

In this thesis, we address the problem of codiagnosability of networked discrete event systems with timing structure (NDESWTS) subject to delays and losses of observations of events between the measurement sites (MS) and local diagnosers (LD). We introduce a new timed model that represents the dynamic system behavior of the plant based on the, *a priori*, knowledge of the minimal firing time for each transition of the plant and on the maximal delays in the event observation after it is recorded in the MS. In order to avoid using the concept of step [33, 42, 43] or the TDES [69], we model the consequences of communication delays of the observations received by the local diagnoser by directly applying the time information. To this end, we convert this timed model in an untimed one, and add possible intermittent packet loss in the communication network. Based on this untimed model, we present necessary and sufficient conditions for NDESWTS codiagnosability and propose two tests for its verification: one that deploys diagnosers and another one that uses verifiers. NDESWTS model proposed here is sufficiently general to address problems of other research areas such as supervisory control of networked discrete event systems [38, 39].

In order to establish a comparison between the approach presented here and others previously presented in the literature, we shown in Figure 1.1 the main differences between the approach presented here and previous works regarding the location of the communication channels subject to delays, the number of commu-

Location of communication channels subject to delays

Communication between Plant and Diagnosers/Agents	Communication between Agents	Communication between Coordinator and Diagnosers
This work [27,30-32,33-37,38,39,42,43,72,74]	[28,29,41,73]	[40]

Number of communication channels and communication delay bounds

Several communication channels with different delay bounds	Several communication channels with the same delay bound	Single communication channel
Events may be observed in a different order of their occurrences This work [38,39,42,43]	Events may be observed in a different order of their occurrences [28,29,40,41,73]	Events are observed in the same order of their occurrences [27,30-32,33-37,72,74]

Formalism used to measure communication delays

Timing Structure	Step approach	TDES approach	Unbounded delay
This work	[29-39,42,43]	[72-74]	[27,28]

Figure 1.1: Comparison among different networked DES in the literature regarding the location of the communication channels subject to delays, the number of communication channels of communication delays (and delay effects), and the formalism used to measure communication delays.

nication channels and communication delays bounds, and the formalism used to measure communication delays.

Another research topic addressed in this work is the computation of τ-codiagnosability (maximal time to diagnose a failure occurrence) and K-codiagnosability (maximal number of events to diagnose a failure occurrence). To this end, we propose two tests: (i) one test based on a diagnoser-like automaton that does not require usual assumptions on language liveness and nonexistence of unobservable cycles and (ii) another one based on the extended verifier that shows not only the ambiguous paths but also those paths that lead to language diagnosis.

1.4 Thesis Organization

The organization of this doctoral thesis is summarized as follows. In Chapter 2, we present a review of DES theory and a brief introduction on failure diagnosis of DES. In Chapter 3, we revisit the codiagnosability of discrete event systems problem and propose two new verification algorithms: (i) the first algorithm based on a diagnoser-like automaton that does not require the usual assumptions on language liveness and nonexistence of cycles of states connected with unobservable events; (ii) and an extended verifier developed to show not only the ambiguous paths but also those paths that lead to language diagnosis. As an application, we address τ- and K-codiagnosability problems. In Chapter 4, we address the failure diagnosis problem of networked discrete event systems with timing structure (NDESWTS), and, to this end, we formally define NDESWTS and propose an equivalent untimed model. Subsequently, we present necessary and sufficient conditions for codiagnosability of NDESWTS and propose two tests to its verification: the first one based on diagnosers, and a second one, based on verifiers. Finally, in Chapter 5, we conclude the thesis and point out potential future directions.

Chapter 2

Fundamentals of Discrete Event Systems and Failure Diagnosis

In this chapter, we present the necessary background on Discrete Event Systems (DES), and on failure diagnosis of DES. The theoretical foundations of DES presented in Section 2.1 are based on [11].

The structure of the chapter is as follows. In Section 2.1, the main formalisms for DES are presented. In Section 2.2, we present a model for DES subject to intermittent loss of observations. The main concepts associated with failure diagnosis are presented in Section 2.3. Finally, we draw some conclusions in Section 2.4.

2.1 Discrete Event Systems

In recent years, the growth of computer technology has led to the propagation of a class of highly complex dynamical systems, with the distinct attribute that their behavior is determined by the asynchronous occurrence of certain events. Such systems are called Discrete Event Systems (DES).

DES are dynamical systems with discrete state-spaces and event-triggered dynamics. An event may be identified with a specific taken action, or may be viewed as a spontaneous occurrence dictated by nature or, still, the result of several conditions which are all suddenly met. Examples of events are the beginning and ending of a task, the arrival of a client to a queue or the reception of a message in a communication system. The occurrence of an event causes, in general, an internal change in

the system, which may or may not manifest itself to an external observer. In addition, a change can be caused by the occurrence of an event internal to the system itself, such as the termination of an activity or timing. In any case, these changes are characterized by being abrupt and instantaneous, *i.e.*, by perceiving an event occurrence, the system reacts immediately, accommodating itself to a new situation where it remains until a new event occurs. In this way, the simple passing of time is not enough to ensure that the system evolves; for this, it is necessary that events occur. The system behavior in the DES framework is, therefore, by sequences of events. All sequences of the events 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, we start the review of DES theory with the concept of language.

2.1.1 Languages

One formal way 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. We will assume that Σ is finite. The event set Σ is the "alphabet" and the sequences are the "words" of a language. In the literature, the sequences of a language are also called traces or strings; however, the term trace will be used throughout this thesis. The length of a trace s is the number of events it contains and will be denoted by $|s|$. The word that does not contain events is called the empty trace, and is denoted by ε, *i.e.*, $|\varepsilon| = 0$.

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

Example 2.1 *Let $\Sigma = \{a, b, c\}$ be a set of events. As an example, we may then define, over Σ, the language $L_1 = \{\varepsilon, a\}$ consists of only two traces; or the language $L_2 = \{a, bb, ac\}$ that contains three traces.*

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

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

The basic process of forming a language is concatenation. The concatenation of two traces of events results in a trace formed with the events of first one immediately followed by the events of the second one. For instance, trace abc formed from events in $\Sigma = \{a, b, c\}$, can be formed by the concatenation of trace ab with event c, and trace ab is obtained from the concatenation of events a and b. The empty trace is the identity element of concatenation, *i.e.*, $s\varepsilon = \varepsilon s = s$, for every trace s.

Before presenting the operations on languages, we need to define some terminology about traces. Let us consider a trace s arbitrarily partitioned as $s = tuv$, where $t, u, v \in \Sigma$. We say that t, u and v are subtrace of s, in particular, subtrace t is a prefix of s, whereas subtrace v is a suffix of s. Notice that, ε and s are both subtrace, prefixes and suffixes of s.

2.1.2 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. In addition, the following operations can be defined for languages: concatenation, Kleene-closure, prefix-closure, post-language and natural projection.

Concatenation

Let $L_a, L_b \subseteq \Sigma^*$, then the concatenation between two languages is the set of the concatenations of all traces in L_a with all traces in L_b. Formal definition is provided as follows.

$$L_a L_b := \{s = s_a s_b \in \Sigma^* : (s_a \in L_a) \wedge (s_b \in L_b)\}. \tag{2.1}$$

Kleene-closure

The Kleene-closure of a language is the set of all possible traces formed by the concatenation of all traces of this language. Formally, if $L \subseteq \Sigma^*$ then Kleene-closure is defined as:

$$L^* := \{\varepsilon\} \cup L \cup LL \cup LLL... \tag{2.2}$$

Prefix-closure

Another important operation is the prefix closure of a language L, which consists of all prefixes of all traces in L. A trace $t \in \Sigma^*$ is prefix of a trace $s \in \Sigma^*$ if there exists a trace $v \in \Sigma^*$ such that $tv = s$, and thus, both s and ε are prefixes of s. Formally, we can define prefix-closure as follows.

$$\bar{L} := \{s \in \Sigma^* : (\exists t \in \Sigma^*)[st \in L]\}. \tag{2.3}$$

L is said to be prefix-closed if $L = \bar{L}$. Thus language L is prefix-closed if all prefixes of every trace in L are also an element of L.

Post-language

Let $L \subseteq \Sigma^*$ and $s \in L$. Then the post-language of L after s, denoted by L/s, is the language

$$L/s := \{t \in \Sigma^* : st \in L\}. \tag{2.4}$$

By definition, $L/s = \emptyset$ if $s \notin L$.

Projection

Another type of operation performed on traces and languages is the so-called natural projection, or simply projection, denoted by P. This operation takes a trace formed from the larger event set Σ and erases the events in it that do not belong to the smaller event set Σ_s. Formally, the projection $P : \Sigma^* \to \Sigma_s^*$ can be defined as follows.

$$P(\varepsilon) := \varepsilon, \tag{2.5}$$

$$P(e) = \begin{cases} e, & \text{if } e \in \Sigma_s, \\ \varepsilon, & \text{otherwise,} \end{cases} \tag{2.6}$$

$$P(se) := P(s)P(e), \;\; s \in \Sigma^*, e \in \Sigma. \tag{2.7}$$

We will also be working with the corresponding inverse map, $P^{-1} : \Sigma_s^* \to 2^{\Sigma^*}$, defined as the following.

$$P^{-1}(t) := \{s \in \Sigma^* : P(s) = t\}. \tag{2.8}$$

The projection P and its inverse P^{-1} are extended to languages by simply applying them to all the traces in the language. For $L \subseteq \Sigma^*$,

$$P(L) = \{t \in \Sigma_s^* : (\exists s \in L)[P(s) = t]\}. \tag{2.9}$$

For $L_s \subseteq \Sigma_s^*$,

$$P^{-1}(L_s) : \{s \in \Sigma^* : (\exists t \in L_s)[P(s) = t]\}. \tag{2.10}$$

In order to illustrate the concepts of this subsection, consider the following example.

Example 2.2 *Let us consider again the set of events* $\Sigma = \{a, b, c\}$ *and languages* $L_1 = \{\varepsilon, a\}$ *and language* $L_2 = \{a, bb, ac\}$. *Since* $\overline{L_1} = \{\varepsilon, a\}$ *and* $\overline{L_2} = \{\varepsilon, a, b, bb, ac\}$, $L_1 = \overline{L_1}$ *and* $L_2 \neq \overline{L_2}$. *Consequently,* L_1 *is prefix-closed and* L_2 *is not prefix-closed. In addition, we can see that:*

$$L_1^* = \{\varepsilon, a, aa, aaa, \ldots\}$$
$$L_1 L_2 = \{a, bb, ac, aa, abb, aac\}$$
$$L_2 L_1 = \{a, bb, ac, aa, bba, aca\}$$
$$L_2/a = \{\varepsilon, c\}$$

If we define projection $P : \Sigma^* \to \Sigma_s^*$ *such that* $\Sigma_s = \{a, b\}$, *then:*

$$P(abc) = \{ab\}$$
$$P^{-1}(\varepsilon) = \{c\}^*$$
$$P^{-1}(ab) = \{c\}^*\{a\}\{c\}^*\{b\}\{c\}^*$$
$$P(L_2) = \{a, bb\}$$
$$P^{-1}(L_1) = \{\{c\}^*, \{c\}^*\{a\}\{c\}^*\}$$

2.1.3 Automata

Automata are devices that are capable of representing a language by using a state transition structure, *i.e.*, by specifying which events can occur at each state of the system. They are an intuitive and natural description of a discrete event system. The modeling formalism of automata is a framework for representing and manipulating languages and solving problems that pertain to the logical behavior of DES. Automata are a useful model for many kinds of hardware and software: *(i)* software

for designing and checking behavior of digital circuits; *(ii)* the lexical analyzer of a typical compiler; *(iii)* software for scanning large bodies of text; *(iv)* software for verifying systems such as communication protocols or protocols for secure exchange of information. In addition, in the literature, automaton model is a classical tool to deal with the problem we will address in this thesis, the so-called failure diagnosis problem. Mathematically, we can define Deterministic automaton as follows.

Definition 2.2 *A deterministic finite-state automaton G is a six-tuple*

$$G = (X, \Sigma, f, \Gamma, x_0, X_m), \tag{2.11}$$

where X is the set of states, Σ is the set of events, $f : X \times \Sigma \to X$ is the partial transition function such that $f(x, \sigma) = z$, means that there is a transition labeled by event σ that takes G from state x to state z. $\Gamma : X \to 2^{\Sigma}$ is the set of active events, that is, $\forall x \in X$, $\Gamma(x) = \{\sigma \in \Sigma : f(x, \sigma)!\}$, with $!$ meaning that $f(x, \sigma)$ is defined, x_0 is the initial state and X_m is the set of marked states.

Regarding the set of marked states, proper selection of which states to mark is a modeling issue that depends on the problem of interest. By designating certain states as marked, we may, for instance, be recording that the system, upon entering these states, has completed some operation or task. Notice that Definition 2.11 does not impose that the set of states X must be finite. However, in this thesis, we deal with finite set of states X only; thus, the term finite-state deterministic automaton will be replaced with automaton for short.

To understand exactly dynamic evolution of an automaton, assume that an automaton is in state x_n when an event σ occurs. Then, automaton G moves to the state x_{n+1} instantaneously. This dynamic is characterized by the state transition function as follows: $x_{n+1} = f(x_n, \sigma)$ such that $\sigma \in \Gamma(x_n)$. It is convenient to represent graphically an automaton whose state set X is finite by means of its state transition diagram. The state transition diagram of an automaton is a directed graph whose nodes represent states and the arcs (labeled) between nodes are used to represent the transition function f : If $f(x_n, \sigma) = x_{n+1}$, then an arc labeled by σ is drawn from x_n to x_{n+1}. A special notation is used to identify the initial and marked states. The initial state is identified by an arrow pointing into it and marked states are differentiated by means of a double circle or box. From the state

transition diagram of an automaton, it is possible to infer some information about its elements, as shown in following example.

Example 2.3 *Consider the state transition diagram of automaton G_1 depicted in Figure 2.1. From this picture, it can be concluded that the set of states of G_1 is $X = \{x_0, x_1, x_2\}$, the initial state is x_0 and the set of marked states is $X_m = \{x_0, x_2\}$. Notice that the sets of active events for each state of G_1 are $\Gamma(x_0) = \{a, c\}$, $\Gamma(x_1) = \{a, b\}$ and $\Gamma(x_2) = \{a, b, c\}$. The transition function is given as follows: $f(x_0, a) = x_0$, $f(x_0, c) = x_2$, $f(x_2, b) = x_2$, $f(x_2, a) = f(x_2, c) = x_1$, $f(x_1, b) = x_1$ and $f(x_1, a) = x_0$.*

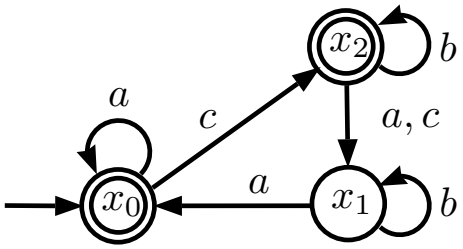


Figure 2.1: Automaton G_1 of Example 2.3.

For the sake of convenience, the transition function f of an automaton is extended from domain $X \times \Sigma$ to domain $X \times \Sigma^*$ as follows: $f(x, \varepsilon) = x$ and $f(x, s\sigma) = f(f(x, s), \sigma)$, $\forall x \in X$, $s \in \Sigma^*$ and $\sigma \in \Sigma$ such that $f(x, s) = z$ and $f(z, \sigma)$ are both defined. Thus, we can define the languages generated and marked by an automaton as follows.

Definition 2.3 *(Generated and marked languages) The language generated by automaton G is defined as $L(G) := \{s \in \Sigma^* : f(x_0, s)!\}$ and the language marked by automaton G is defined as $L_m(G) := \{s \in L(G) : f(x_0, s) \in X_m\}$.*

The language $L(G)$ contains the traces that can be followed along the state transition diagram of G, starting at the initial state; the trace corresponding to a path is the concatenation of the event labels of the transitions composing the path. The second language represented by G, $L_m(G)$, is the subset of $L(G)$ consisting only of the traces s for which $f(x_0, s) \in X_m$, that is, these traces correspond to paths

that end at a marked state in the state transition diagram. Languages $L(G)$ and $L_m(G)$ satisfy the following inclusion relation.

$$L_m(G) \subseteq \overline{L_m(G)} \subseteq L(G), \tag{2.12}$$

2.1.4 Operations on Automata

In order to examine DES modeled by automata, we need to define a set of operations that appropriately modify their state transition diagram based on well-defined criteria. The operations that modify a single automaton are called unary operations and are defined as follows.

Unary operations

- **Accessible Part**

A state $x \in X$ of an automaton G is accessible if $\exists s \in \Sigma^*$ such that $f(x_0, s) = x$. Otherwise, x is a non-accessible state. The accessible part operation removes all non-accessible states of an automaton G, and is defined as follows.

$$Ac(G) := (X_{ac}, \Sigma, f_{ac}, \Gamma_{ac}, x_0, X_{ac,m}), \tag{2.13}$$

where $X_{ac} = \{x \in X : (\exists s \in \Sigma)[f(x_0, s) = x]\}$, $X_{ac,m} = X_m \cap X_{ac}$, $f_{ac} = f|_{X_{ac} \times \Sigma \to X_{ac}}$, and $\Gamma_{ac} = \Gamma|_{X_{ac} \to X_{ac}}$, where $f_{ac} = f|_{X_{ac} \times \Sigma \to X_{ac}}$ and $\Gamma_{ac} = \Gamma|_{X_{ac} \to X_{ac}}$ means that f and Γ are restricted to X_{ac}, respectively. Notice that, the Ac operation does not change the languages generated and marked by the automaton.

- **Coaccessible Part**

A state $x \in X$ of an automaton G is coaccessible if $\exists s \in \Sigma^*$ such that $f(x, s) \in X_m$. Otherwise, x is a non-coaccessible state. The coaccessible part operation excludes all non-coaccessible states of an automaton G, and is defined as follows:

$$CoAc(G) := (X_{coac}, \Sigma, f_{coac}, \Gamma_{ac}, x_{0,coac}, X_m), \tag{2.14}$$

where $X_{coac} = \{x \in X : (\exists s \in \Sigma)[f(x, s) \in X_m]\}$, $f_{coac} = f|_{X_{coac} \times \Sigma \to X_{coac}}$, and $\Gamma_{coac} = \Gamma|_{X_{coac} \to X_{coac}}$. Notice that, the $CoAc$ operation can affect the language generated by the original automaton, but it does not change the marked language.

- **Trim operation**

An automaton that is both accessible and coaccessible is said to be *Trim*. We define the Trim operation as follows.

$$Trim(G) := Ac[CoAc(G)] = CoAc[Ac(G)] \tag{2.15}$$

Example 2.4 *Consider automaton G depicted in Figure 2.2(a). To obtain $Ac(G)$, it suffices to delete state 3 and the transition attached to it; the resulting automaton is depicted in Figure 2.2(b). In order to obtain $CoAc(G)$, we need to identify the states of G that are not coaccessible to the marked state 2, which is state 4. We, then, delete this state and the transition attached to it, to obtain $CoAc(G)$ depicted in Figure 2.2(c). Finally, $Trim(G)$ is shown in Figure 2.2(d). Notice that the order which the operations Ac and $CoAc$ are taken does not change the final result.*

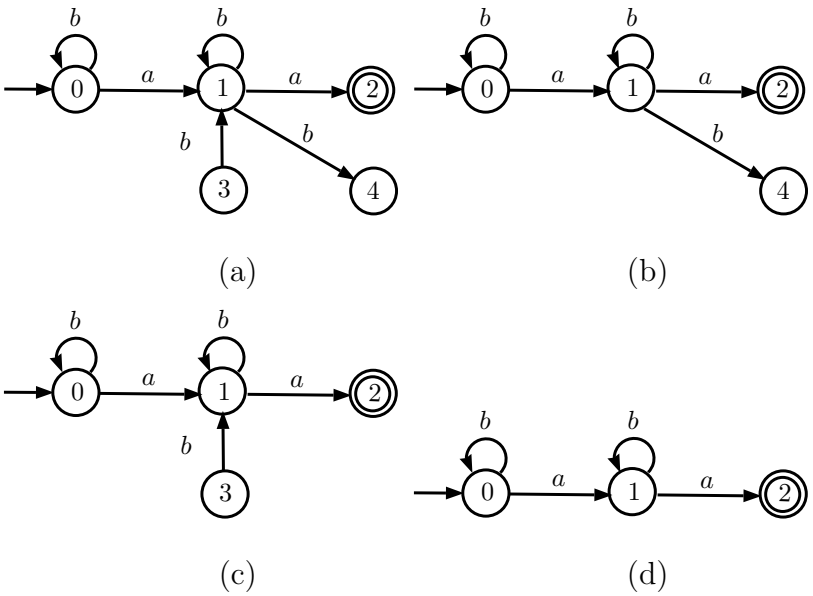


Figure 2.2: Automaton G (a); $Ac(G)$ (b); $CoAc(G)$ (c), and; $Trim(G)$ (d).

Composition operations

Discrete event models of complex dynamic systems are rarely built in a monolithic manner. Instead, a modular approach is used where models of individual components are built first, followed by the composition of these models in order to obtain the model of the overall system. The synchronization, or coupling, between components can be captured by the use of common events between system components.

Namely, if components A and B share event c, then event c should only occur if both A and B execute it. The process of composing individual automata (that model interacting system components) in a manner that captures the synchronization constraints imposed by their common events is formalized by the product and parallel composition operations as follows.

- **Product**

The product operation can be viewed as the composition that computes the intersection between languages generated (and marked) by two or more automata. Namely, an event occurs in the product composition if and only if it occurs in both automata. When there is no event in common between the active event sets of the initial states of two automata, the resulting product between these automata will be an automaton where the generated language is ε. The product of G_1 and G_2 is given by.

$$G_1 \times G_2 := Ac(X_1 \times X_2, \Sigma_1 \cup \Sigma_2, f_{1\times 2}, \Gamma_{1\times 2}, (x_{01}, x_{02}), X_{m1} \times X_{m2}), \tag{2.16}$$

where $f_{1\times 2}$ is defined as:

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

According to Equation (2.16), it can be easily verified that $L(G_1 \times G_2) = L(G_1) \cap L(G_2)$ and $L_m(G_1 \times G_2) = L_m(G_1) \cap L_m(G_2)$ [11].

- **Parallel composition**

Parallel composition is often called synchronous composition and product is sometimes called completely synchronous composition. Composition by product is restrictive as it only allows transitions on common events. In general, when modeling systems composed of interacting components, the event set of each component includes private events that pertain to its own internal behavior and common events that are shared with other automata and capture the coupling among the respective system components. The standard way of building models of entire systems from

models of individual system components is by parallel composition, which is defined as follows:

$$G_1||G_2 := Ac(X_1 \times X_2, \Sigma_1 \cup \Sigma_2, f_{1||2}, \Gamma_{1||2}, (x_{01}, x_{02}), X_{m1} \times X_{m2}), \tag{2.17}$$

where $f_{1||2}$ is defined as:

$$f((x_1, x_2), \sigma) := \begin{cases} (f_1(x_1, \sigma), f_2(x_2, \sigma)), & \text{if } \sigma \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ (f_1(x_1, \sigma), x_2), & \text{if } \sigma \in \Gamma_1(x_1) \setminus \Sigma_2 \\ (x_1, f_2(x_2, \sigma)), & \text{if } \sigma \in \Gamma_2(x_2) \setminus \Sigma_1 \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

According to Equation (2.17), we can verify that $L(G_1||G_2) = P_1^{-1}[L(G_1)] \cap P_2^{-1}[L(G_2)]$ and $L_m(G_1||G_2) = P_1^{-1}[L_m(G_1)] \cap P_2^{-1}[L_m(G_2)]$, where $P_i : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_i^*$, for $i = 1, 2$, [11].

Example 2.5 *Consider automata G_1 and G_2 depicted in Figures 2.1 and 2.3, respectively. The automata obtained by the product and parallel composition of G_1 and G_2 are shown in Figures 2.4 and 2.5, respectively. In this example, we can see that the parallel composition is an operation more generic than product composition: notice that $L(G_1 \times G_2) = a^* \subset L(G_1||G_2)$, which is due to the fact that the composition by product only allows transitions on common events while private events can be executed whenever possible in the parallel composition.*

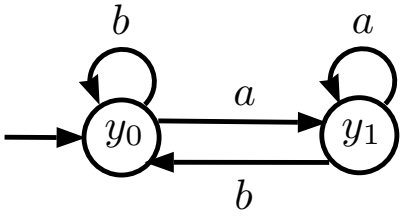


Figure 2.3: Automaton G_2 of Example 2.5.

2.1.5 Nondeterministic Automata

A nondeterministic automaton, denoted by G_{nd}, is a six-tuple $G_{nd} = (X, \Sigma_{nd}, f_{nd}, \Gamma_{nd}, X_0, X_m)$, where $\Sigma_{nd} = \Sigma \cup \{\varepsilon\}$, $f_{nd} : X \times \Sigma_{nd} \to 2^X$, that is, $f_{nd}(x, \sigma) \subseteq X$, Γ_{nd} is defined in a similar way as Γ, and the initial state may, itself, be a set of states, that is, $X_0 \subseteq X$.

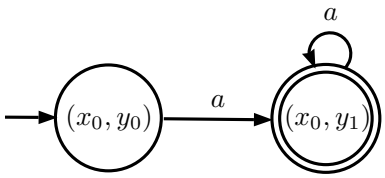


Figure 2.4: Automaton $G_1 \times G_2$ of Example 2.5.

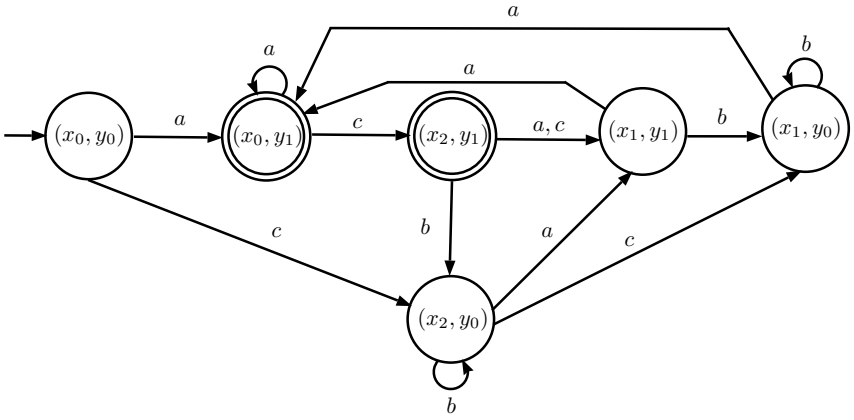


Figure 2.5: Automaton $G_1||G_2$ of Example 2.5.

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

Example 2.6 *Consider the nondeterministic automaton,* $G_{nd} = (X, \Sigma_{nd}, f_{nd}, \Gamma_{nd}, X_0, X_m)$*, shown in Figure 2.6. Notice that,* G_{nd} *has two initial states* x_{0_1} *and* x_{0_2}*, and the transition function assumes values in* 2^X*, for* $x \in X$*, for instance,* $f_{nd}(x_{0_2}, a) = \{x_{0_1}, x_1\}$*. Notice, in addition, that* $f_{nd}(x_{0_1}, \varepsilon) = x_1$*. These types of configuration suggest uncertainty in the dynamic evolution of the system, as follows:* (i) *when event* a *occurs, it is not possible to be sure if the system has moved either to state* x_{0_1} *or* x_1*, and;* (ii) *transition* ε *is silent in the sense that the system evolution from state* x_{0_1} *to* x_1 *cannot take place.*

Remark 2.1 *The nondeterministic automata to be considered in this thesis do not have transitions labeled by the empty trace* ε*, thus* $\Sigma_{nd} = \Sigma$*. Under this condition, the transition function, is extended to* Σ^* *as follows: for* $B \in 2^X$*, we first set* $f_{nd}(B, \sigma) = \cup_{x \in B : \sigma \in \Gamma_{nd}(x)} f_{nd}(x, \sigma)$*, and, then, for* $u \in \Sigma^*$ *and* $\sigma \in \Sigma$*, we set*

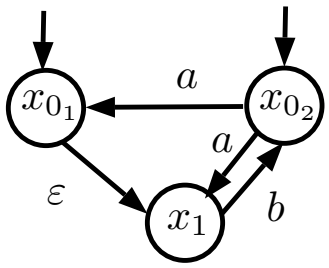


Figure 2.6: Automaton G_{nd} of Example 2.6.

$f_{nd}(B, u\sigma) = f_{nd}[f_{nd}(B, u), \sigma]$. *The languages generated and marked by a nondeterministic automaton are defined as follows:* $L(G_{nd}) = \{s \in \Sigma^* : (\exists x \in x_0)[f_{nd}(x, s)!]\}$ *and* $L_m(G_{nd}) = \{s \in \Sigma^* : (\exists x \in x_0)[f_{nd}(x, s) \cap X_m \neq \emptyset]\}$. *Throughout the paper we will denote both, deterministic and nondeterministic automata, by* G*; the context will make clear which one is being considered.*

2.1.6 Deterministic Automata With Unobservable Events

In this subsection, we will consider the case of partially-observed DES, *i.e.*, when some events cannot have their occurrences seen by an outside observer. This lack of observability can be due to the absence of a sensor to record the occurrence of the event or to the fact that the event occurs at a remote location but is not communicated to the site being modeled. In this case, some form of state estimation becomes necessary when analyzing the behavior of the system. To this end, the set of events Σ is partitioned into the set of observable events Σ_o and set of unobservable events Σ_{uo}. The corresponding automaton will be deterministic, and is referred to as deterministic automaton with unobservable events.

The dynamic behavior of a deterministic automaton with unobservable events G can be described by another deterministic automaton called observer, here denoted as $\text{Obs}(G, \Sigma_o)$. The procedure for the construction of $\text{Obs}(G, \Sigma_o)$ requires the notion of unobservable reach of a state $x \in X$ with respect to a set Σ_{uo}, which is defined as:

$$UR(x, \Sigma_{uo}) = \{y \in X : (\exists t \in \Sigma_{uo}^*)[(f(x, t) = y)]\}. \tag{2.18}$$

From Equation (2.18), it is clear that $x \in UR(x, \Sigma_{uo})$. The definition of unobservable reach can be extended to a set of states $B \subseteq X$ as follows.

$$UR(B, \Sigma_{uo}) = \cup_{x \in B} UR(x, \Sigma_{uo}). \tag{2.19}$$

Finally, we can then define observer automata as follows.

$$Obs(G, \Sigma_o) = (X_{obs}, \Sigma_o, f_{obs}, \Gamma_{obs}, x_{0_{obs}}, X_{m_{obs}}), \tag{2.20}$$

where $X_{obs} \in 2^X$, $x_{0_{obs}} = UR(x_0, \Sigma_{uo})$; for all $x_{obs} \in X_{obs}$, $\Gamma_{obs}(x_{obs}) = \bigcup_{x \in x_{obs}} \Gamma(x)$; $f_{obs}(x_{obs}, \sigma) = UR(\{y \in X : (\exists x \in x_{obs})[f(x, \sigma) = y]\}, \Sigma_{uo})$; $X_{m_{obs}} = \{x_{obs} \in X_{obs} : x_{obs} \cap X_m \neq \emptyset\}$. $Obs(G, \Sigma_o)$ can be constructed by using the following algorithm [75].

Algorithm 2.1 *Construction of automaton* $Obs(G, \Sigma_o)$

Input *Automaton* $G = (X, \Sigma, f, \Gamma, x_0, X_m)$*, and set* Σ_o*.*

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

STEP 1.
- *Set* $\Sigma_{uo} = \Sigma \setminus \Sigma_o$*, and define* $x_{0_{obs}} = UR(x_0, \Sigma_{uo})$*;*
- *Set* $X_{obs} = \{x_{0_{obs}}\}$ *and* $\tilde{X}_{obs} = X_{obs}$*;*

STEP 2. *Set* $\hat{X}_{obs} \leftarrow \tilde{X}_{obs}$ *e* $\tilde{X}_{obs} \leftarrow \emptyset$*;*

STEP 3. ***For*** $B \in \hat{X}_{obs}$*:*

STEP 3.1. $\Gamma_{obs}(B) = (\bigcup_{x \in B} \Gamma(x)) \cap \Sigma_o$*;*

STEP 3.2 ***For*** $\sigma \in \Gamma_{obs}(B)$*:*

- *Set* $f_{obs}(B, \sigma) = UR(\{x \in X : (\forall y \in B)[x = f(y, \sigma)]\})$*;*
- *Set* $\tilde{X}_{obs} \leftarrow \tilde{X}_{obs} \cup f_{obs}(B, \sigma)$*;*

STEP 4 $X_{obs} \leftarrow X_{obs} \cup \tilde{X}_{obs}$*;*

STEP 5 *Repeat Step* 2 *to* 4 *until the entire accessible part of* $Obs(G, \Sigma_o)$ *has been constructed;*

STEP 6 *:* $X_{m_{obs}} = \{B \in X_{obs} : B \cap X_m \neq \emptyset\}$.

From construction of automaton, we can conclude that the language generated by $Obs(G, \Sigma_o)$ is the natural projection over Σ_o, *i.e.*, $L(Obs(G, \Sigma_o)) = P_o[L(G)]$, such that $P_o : \Sigma \to \Sigma_o$ [11].

Example 2.7 *To illustrate the construction of observers, let us consider now automaton G depicted in Figure 2.7 and suppose that event a is unobservable, i.e., $\Sigma = \{a, b, c\}$, $\Sigma_o = \{b, c\}$ e $\Sigma_{uo} = \{a\}$.*

When automaton G starts its operation, since event a is unobservable, we do not know if the system is in initial state $x_0 = 0$ or in state $x = 1$ and, thus, the initial state of $Obs(G, \Sigma_o)$ must be $\{0, 1\}$. If event b occurs, the systems moves to state $x = 3$, but it can reach state $x = 1$ through unobservable transition a, leading $Obs(G, \Sigma_o)$ to reach state $\{1, 3\}$. If event c occurs in $\{1, 3\}$, the system can be in any state of G, i.e., $Obs(G, \Sigma_o)$ reaches state $\{1, 2, 3\}$. Finally, when the occurrence of event c is recorded again, then, since the observer is in state $\{1, 2, 3\}$, it remains there. However, if event b occurs, $Obs(G, \Sigma_o)$ returns to its the initial state.

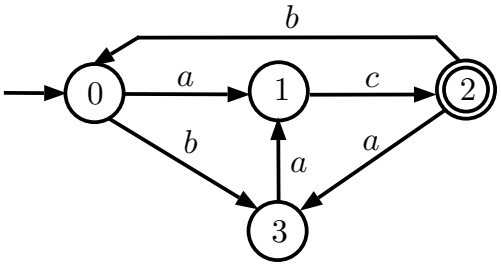


Figure 2.7: Automaton G of Example 2.7.

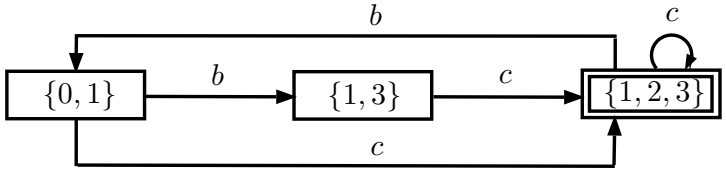


Figure 2.8: Automaton $Obs(G, \Sigma_o)$ of Example 2.7.

2.1.7 Strongly Connected Components

We now briefly touch on the concept of strongly connected components, whose definition is presented as follows.

Definition 2.4 *(Strongly connected component) [60] A strongly connected component of an automaton $G = (X, \Sigma, f, \Gamma, x_0, X_m)$ is a maximal set of states $X_{scc} \subseteq X$ such that for every pair of states $u, v \in X_{scc}$, there is a path formed by events in Σ*

from u to v and from v to u; that is, every states u and v in X_{scc} are reachable from each other, and X_{scc} is maximal.

Example 2.8 *To illustrate the concept of strongly connected components, let us consider again the automaton depicted in Figure 2.5. Notice that automaton has two strongly connected components: the first one is formed by four states $\{(x_0, y_1), (x_2, y_1), (x_1, y_1), (x_1, y_0)\}$, i.e., these states are reachable from each other; the second one is formed by unique state (x_2, y_0) which reaches itself by self-loop due to event b. The initial state (x_0, y_0) does not form a strongly connected component since it is not reached by none of the states of G. It is worth noting that the number of strongly connected components and number of cycles are not equal: states that form a strongly connected component may form several cycles, for instance, the cyclic paths $((x_0, y_1), a, (x_0, y_1))$, $((x_1, y_0), b, (x_1, y_0))$ and $((x_0, y_1), c, (x_2, y_1), a, (x_1, y_1), a, (x_0, y_1))$ are formed by states that are in the strongly connected component $\{(x_0, y_1), (x_2, y_1), (x_1, y_1), (x_1, y_0)\}$.*

Remark 2.2 *It is worth remarking that the computational complexity of the search for cycles is, in the worst case, worse than exponential, $\sum_{i=1}^{n-1}(\binom{n-i+1}{n})(n-1)!$ [58, 60], where n is the number of states, whereas the search for strongly connected components, proposed by Tarjan in [59], is linear in the number of states and transitions of an automaton, i.e., $O(|n| + |E|)$, where E is the number of transitions.*

2.2 Discrete Event Systems Subject to Loss of Observations

In this section, we present a model for the observed behavior of an automaton in the presence of intermittent loss of observations proposed by [54], which can be used to deal with loss of observations caused by either sensor malfunction or communication problems. In [54], a partition in the set of events was defined by the authors as:

$$\Sigma = \Sigma_{uo} \dot{\cup} \Sigma_{ilo} \dot{\cup} \Sigma_{nilo}, \tag{2.21}$$

where Σ_{ilo} is the set of observable events associated with intermittent loss of observations and Σ_{nilo} denotes the set of observable events that are not subject to

intermittent loss of observations. In order to represent loss of observations, let $\Sigma'_{ilo} = \{\sigma' : \sigma \in \Sigma_{ilo}\}$ and $\Sigma' = \Sigma \cup \Sigma'_{ilo}$. Therefore, the following mapping can be defined.

Definition 2.5 *(Dilation) Dilation is the mapping* $D : \Sigma^* \rightarrow \Sigma^*_{dil}$ *, recursively defined as:*

$$D(\varepsilon) := \{\varepsilon\}$$

$$D(\sigma) := \begin{cases} \{\sigma\}, & \text{if } \sigma \in \Sigma \setminus \Sigma_{ilo}, \\ \{\sigma, \sigma'\}, & \text{if } \sigma \in \Sigma_{ilo},. \end{cases}$$

$$D(s\sigma) := D(s)D(\sigma), \quad \text{if } \forall s \in \Sigma', \sigma \in \Sigma.$$

The extension of D to domain 2^{Σ^*}, *i.e.*, to languages, is defined as $D(L) = \bigcup_{s \in L} D(s)$. The idea behind the definition of dilation is to represent the loss of observation of an observable event σ_s by replacing it with σ'_s. For instance, by assuming $\Sigma = \{\sigma, \sigma_s\}$ and $\Sigma_{ilo} = \{\sigma_s\}$, the dilation of trace $s = \sigma\sigma_s$ is $D(s) = \{\sigma\sigma_s, \sigma\sigma'_s\}$ where: *(i)* trace $\sigma\sigma_s$ represents the case where no loss of observation has occurred, and; *(ii)* trace $\sigma\sigma'_s$ represents the case when the information of the occurrence of event σ_s has been lost.

With the help of Definition 2.5, we can now formally define automaton G' that models the behavior of G subject to intermittent loss of observations, as follows.

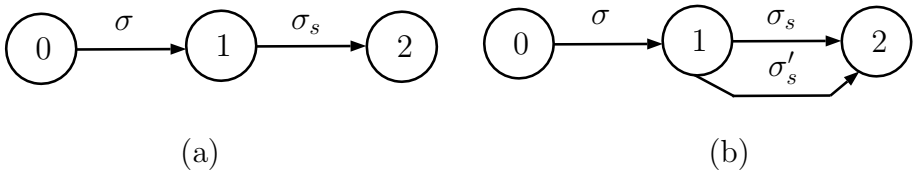
$$G = (X, \Sigma', f', \Gamma', x_0, X_m), \tag{2.22}$$

where $\Gamma'(x) = D[\Gamma(x)]$, and f' is defined as follows: $\forall \sigma' \in \Gamma'(x)$, $\sigma' \in D(\sigma)$, $f'(x, \sigma') = f(x, \sigma)$, where $\sigma \in \Gamma(x)$.

Notice that automaton G' is formed by adding to G transitions in parallel with the transitions associated with the events that are subject to intermittent loss of observations. The added transitions will be labeled with unobservable events and therefore the observable event set of G' remains Σ_o as in G.

Example 2.9 *In order to illustrate the dilation operation, let us assume that* $\Sigma = \{\sigma, \sigma_s\}$ *and* $\Sigma_{ilo} = \{\sigma_s\}$ *and consider language* $L(G) = \overline{\sigma\sigma_s}$*. Automaton* G *that generates* L *is depicted in Figure 2.9(a). Let us, initially, illustrate the application of dilation to the traces of* $L(G)$*. Notice that* $\Sigma'_{ilo} = \{\sigma'_s\}$ *and thus* $\Sigma' = \{\sigma, \sigma_s, \sigma'_s\}$*.*

Therefore, $D(\varepsilon) = \varepsilon$, $D(\sigma) = \sigma$, $D(\sigma\sigma_s) = D(\sigma)D(\sigma_s) = \sigma\{\sigma_s, \sigma'_s\} = \{\sigma\sigma_s, \sigma_s, \sigma'_s\}$. *Figure 2.9(b) shows automaton* G', *that generates* $\overline{D(L(G))}$, *from which it is not difficult to check that* $D(L(G)) = L(G')$.



(a) (b)

Figure 2.9: Automaton G of Example 2.9 (a); automaton G' of Example 2.9 (b).

2.3 Failure Diagnosis of Discrete Event Systems

A failure causes a non-desired deviation of a system or of one of its components from its normal or intended behavior. The deviation of system performance can either be tolerated or be considered as critical in the case of a failure or a breakdown. Failure diagnosis is therefore closely related to the problem of state observability, which consists in building a deterministic automaton, called the observer, whose transitions are due to the observable events of the system and whose states are estimates of the true system state, as seen in Section 2.1.

2.3.1 Centralized Diagnosis

In order to formally address the failure diagnosis problem, let $\Sigma_f = \{\sigma_f\} \subseteq \Sigma_{uo}$ denote the set of failure events of G, and assume that the occurrence of σ_f must be diagnosed. The idea behind failure diagnosis using discrete event models is that we must somehow be sure, within a bounded number of steps after the occurrence of σ_f, that the failure has actually occurred. Let $\Psi(\Sigma_f)$ denote the set of all traces of L that end with the failure event σ_f. With a slight abuse of notation, we use $\Sigma_f \in s$ to denote that $\overline{s} \cap \Psi(\Sigma_f) \neq \emptyset$, where $\overline{s} = \{u \in \Sigma^* : (\exists v \in \Sigma^*)[uv = s]\}$. Therefore, $s \in L$ is a trace that has the failure event σ_f if $\Sigma_f \in s$. Formally, failure diagnosability is defined as follows [19].

Definition 2.6 *(Diagnosability) A live and prefix-closed language* L *is diagnosable*

with respect to P_o *and* Σ_f *if*

$$(\exists n \in \mathbb{N})(\forall s \in \Psi(\Sigma_f))(\forall t \in L/s, |t| \geq n) \Rightarrow (\forall \omega \in P_o^{-1}[P_o(st)] \cap L)[\Sigma_f \in \omega],$$

where $|t|$ denotes the length of trace t.

Remark 2.3 *Let* $\Sigma_f = \Sigma_{f_1} \dot{\cup} \Sigma_{f_2} \dot{\cup} \ldots \dot{\cup} \Sigma_{f_r}$ *be a partition of the set of fault events, where* r *denotes the number of fault types, and let* Π_f *denote this partition. The diagnosability of* L *with respect to* Π_f *is equivalent to the diagnosability of* L *with respect to each fault type separately, as ensured by the following result [46]. The language* L *of the system is diagnosable with respect to projection* $P_o : \Sigma^* \to \Sigma_o^*$ *and partition* Π_f *if, and only if,* L *is diagnosable with respect to* P_o *and* Σ_{f_i}*, for each* $i \in \{1, 2, \ldots, r\}$ *[76].*

One way to verify diagnosability is by means of an automaton called diagnoser [19], which is given by

$$G_d = \text{Obs}(G \| A_\ell, \Sigma_o) = \text{Obs}(G_\ell, \Sigma_o) = (X_d, \Sigma_o, f_d, \Gamma_d, x_{0_d}), \tag{2.23}$$

where $A_\ell = (X_\ell, \Sigma_\ell, f_\ell, \Gamma_\ell, x_{0_\ell})$ is the so-called label automaton, depicted in Figure 2.10, with $X_\ell = \{N, Y\}$, $\Sigma_\ell = \{\sigma_f\}$, $f_\ell(N, \sigma_f) = f_\ell(Y, \sigma_f) = Y$ and $x_{0_\ell} = N$. From Equation (2.23), it can be seen that $L(G_d) = P_o(L(G \| A_\ell)) = P_o(L(G))$.

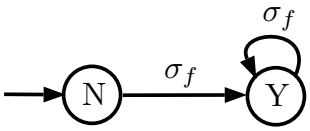


Figure 2.10: Label automaton A_ℓ.

A state $x_d \in X_d$ is called Y-certain (or faulty), if $\ell = Y$ for all $(x, \ell) \in x_d$, and normal (or non-faulty) if $\ell = N$ for all $(x, \ell) \in x_d$. If there exist $(x, \ell), (y, \tilde{\ell}) \in x_d$, x not necessarily distinct from y such that $\ell = Y$ and $\tilde{\ell} = N$, then x_d is an uncertain state of G_d. When the diagnoser is in a Y-certain (resp. normal) state, it is certain that a failure has (resp. has not) occurred. However, if the diagnoser is in an uncertain state, it is not sure if the failure event has occurred or not. As a consequence, if there exists a cycle formed with uncertain states only, where the diagnoser can remain forever, then it will never be able to diagnose the failure

occurrence; on the other hand if somehow it always leaves this cycle of uncertain states, then this cycle is not indeterminate. Therefore, it is important to distinguish between cycles of uncertain states that are indeterminate (in the sense that the diagnoser is not able to determine if the failure has occurred) and those cycles of uncertain states that are not indeterminate.

Definition 2.7 *[19] (Indeterminate observed cycles of G_d) A set of uncertain states $\{x_{d_1}, x_{d_2}, \ldots, x_{d_p}\} \subset X_d$ forms an indeterminate observed cycle if the following conditions hold true:*

IOC.1) *$x_{d_1}, x_{d_2}, \ldots, x_{d_p}$ form a cycle in G_d;*

IOC.2) *$\exists (x_l^{k_l}, Y), (\tilde{x}_l^{r_l}, N) \in x_{d_l}$, $x_l^{k_l}$ not necessarily distinct from $\tilde{x}_l^{r_l}$, $l = 1, 2, \ldots, p$, $k_l = 1, 2, \ldots, m_l$, and $r_l = 1, 2, \ldots, \tilde{m}_l$ in such a way that the sequence of states $\{x_l^{k_l}\}$, $l = 1, 2, \ldots, p$, $k_l = 1, 2, \ldots, m_l$ and $\{\tilde{x}_l^{r_l}\}$, $l = 1, 2, \ldots, p$, $r_l = 1, 2, \ldots, \tilde{m}_l$ form cycles in G;*

IOC.3) *there exist $s = s_1 s_2 \ldots s_p \in \Sigma^*$ and $\tilde{s} = \tilde{s}_1 \tilde{s}_2 \ldots \tilde{s}_p \in \Sigma^*$ such that $P_o(s) = P_o(\tilde{s}) \neq \varepsilon$, where $s_l = \sigma_{l,1}\sigma_{l,2}\ldots\sigma_{l,m_l-1}$, $f(x_l^j, \sigma_{l,j}) = x_l^{j+1}$, $j = 1, 2, \ldots, m_l - 1$, $f(x_l^{m_l}, \sigma_{l+1,0}) = x_{l+1}^1$, and $f(x_p^{m_p}, \sigma_{1,0}) = x_1^1$, and similarly for $\tilde{s}_l$.* □

In [19], the authors make the following assumptions on the system under investigation:

H1. The language L generated by G is live. This means that there is a transition defined at each state $x \in X$, *i.e.*, the system cannot reach a point at which no event is possible.

H2. There does not exist in G any cycle of unobservable events.

A necessary and sufficient condition for language diagnosability is provided by the following result.

Theorem 2.1 *[19] The language L generated by automaton G is diagnosable with respect to projection P_o and $\Sigma_f = \{\sigma_f\}$ if, and only if, its diagnoser G_d has no indeterminate observed cycles.* ■

Example 2.10 *Consider a system modeled by the automaton G shown in Figure 2.11(a), where $\Sigma_o = \{a, b, c\}$ is the set of observable events, and $\Sigma_o = \{u_o, \sigma_f\}$. In order to check the language diagnosability, we first compute the parallel composition between automaton G of Figure 2.11(a) and label automaton A_ℓ depicted in Figure 2.10, and thus, we obtain automaton G_ℓ depicted in Figure 2.11(b). Using automaton G_ℓ, we can now compute diagnoser automaton $G_d = Obs(G_\ell, \Sigma_o)$, depicted in Figure 2.11(c). If we examine G_d, we can check that its state set is formed with a normal state ($\{1N\}$); two uncertain states ($\{2N, 4Y\}$ and $\{3N, 5Y, 6Y\}$); and two Y-certain states (such as $\{5Y, 6Y\}$, $\{4Y\}$ and). Notice that Figure 2.11(c) shows an indeterminate cycle between the diagnoser states $\{2N, 4Y\}$ and $\{3N, 5Y, 6Y\}$. This cycle corresponds to the presence of two cycling traces in the automaton G: (i) a normal trace $s_N = c(ab)^p$, $p \in \mathbb{N}$, i.e., a trace without failure event; (ii) a failure trace $s_Y = c\sigma_f(ab)^q$, $q \in \mathbb{N}$, i.e., a trace that has a failure event. Since they have the same observable projection $P_o(s_Y) = P_o(s_N) = (ab)^r$, $r \in \mathbb{N}$, according to Theorem 2.1, language $L(G)$ is not diagnosable with respect to projection P_o and $\Sigma_f = \{\sigma_f\}$.*

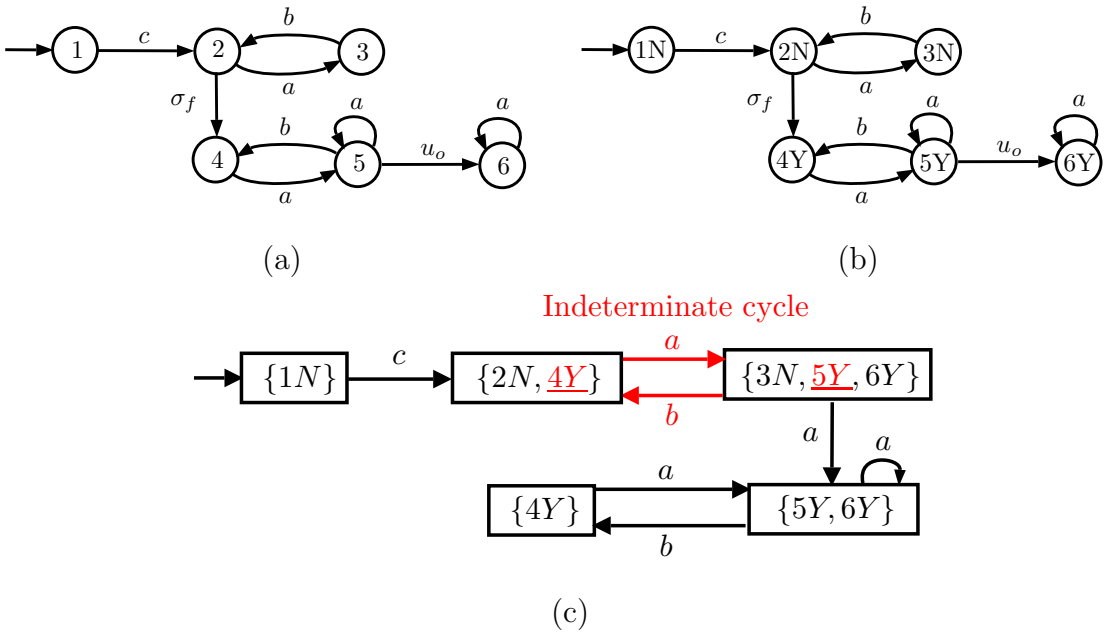


Figure 2.11: Automata G (a); G_ℓ (b) and Diagnoser automaton G_d (c) of Example 2.10.

It is worth commenting on the fact that the presence of a cycle of uncertain states in a diagnoser does not necessarily imply inability to diagnose with certainty

an occurrence of event σ_f. This becomes clear in the following example.

Example 2.11 *[11] Consider the system modeled by automaton G and its diagnoser G_d depicted in Figure 2.12(a) and 2.12(b), respectively. The only unobservable event in the system is the failure event σ_f. This diagnoser has a cycle of uncertain states. However, we cannot form a cycle in the system from entries that appear in the uncertain states in the diagnoser and have the Y label. The only system cycle that can cause the diagnoser to remain in its cycle of uncertain states is the cycle formed by states 7, 11 and 12 in G, and these states all have the N label in the corresponding diagnoser states. The cycle of uncertain states in the diagnoser is therefore not indeterminate. Due to the absence of indeterminate cycles, we can say that $L(G)$ is diagnosable. Indeed, if event σ_f occurs, the diagnoser will leave the cycle of uncertain states and eventually enters state $6Y$ upon the observation of event t. Thus, the fact that the system may cycle in states 7, 11, and 12, causing the diagnoser to cycle in its cycle of uncertain states, cannot interpreted as a lack of diagnosability, since the traces causing such cycling do not contain σ_f. Notice that the diagnoser will for sure exit its cycle of uncertain states, via event t, if event σ_f occurs in the system. This will occur in at most 6 observable transitions after the occurrence of σ_f, i.e., suffix bgdbgt. This is different from the situation in Example 2.10, when the diagnoser may cycle in an indeterminate cycle due to a trace t of arbitrarily long length after the occurrence of a failure event.*

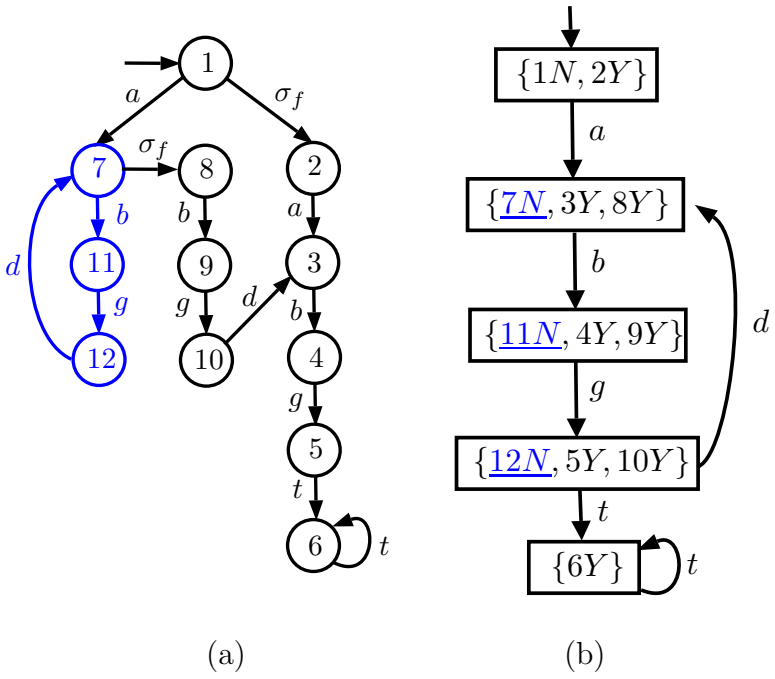


Figure 2.12: Automata G (a) and G_d (b) of Example 2.11.

The first attempt to circumvent the restrictions imposed by assumptions **H1.** and **H2.** was to modify the diagnoser proposed in [19] to include the so-called hidden cycles [54, 77, 78]. The concept of hidden cycles is explained as follows. Assume now that there exists a set of states $\{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\} \subset X$ that forms a cycle of states connected with unobservable events. Consider a trace $s = s_o(\sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_k})^n \in L$ ($n \in \mathbb{N}$), where $(\sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_k})^n \in \Sigma_{uo}^*$ and assume, without loss of generality, that the last event of s_o is observable. Let us suppose, initially, that $\Sigma_f \notin s$, and that there is no faulty or failure trace[1] s' such that $P_o(s) = P_o(s')$. In this case there will exist in G_d a state x_d^N such that $\{x_{i_1}N, x_{i_2}N, \ldots, x_{i_k}N\} \subseteq x_d^N$. On the other hand, if $\Sigma_f \in s_o$ and $f_\ell(x_{0,\ell}, s_o) = x_\ell^Y$, where f_ℓ is the transition function of $G_\ell = G \| A_\ell$, and $x_{0,\ell}$ and x_ℓ^Y are, respectively, the initial and a Y-certain state of G_ℓ, and if there does not exist any normal trace s'' such that $P_o(s) = P_o(s'')$, then, there will exist a Y-certain state x_d^Y of G_d such that $(x_\ell^Y \cup \{x_{i_1}Y, x_{i_2}Y, \ldots, x_{i_k}Y\}) \subseteq x_d^Y$. It is still possible that a normal trace s'' (bounded length or not) such that $f_\ell(x_{0,\ell}, s_o) = x_\ell^N$, where x_ℓ^N is a normal state of G_ℓ, and $P_o(s) = P_o(s'')$, exists, in which case, there will exist an uncertain state x_d^{YN} in G_d such that $(x_\ell^Y \cup \{x_{i_1}Y, x_{i_2}Y, \ldots, x_{i_k}Y\} \cup x_\ell^N) \subseteq x_d^{YN}$. In all the above cases, G_d halts when it reaches the corresponding normal, faulty

[1] A trace s is said to be faulty (normal) if $\Sigma_f \in s$ (resp. $\Sigma_f \notin s$).

and uncertain states, whether or not the plant continues evolving. We say, in this case, that there exist hidden cycles in the aforementioned states.

Definition 2.8 *[54, 77, 78] (Hidden cycles and indeterminate hidden cycles of G_d) Let $x_d = \{x_1\ell_1, x_2\ell_2, \ldots, x_n\ell_n\}$ be a state of G_d. Then, there exists a hidden cycle in x_d if for some index set $\{i_1, i_2, \ldots, i_k\} \subseteq \{1, 2, \ldots, n\}$, the following conditions hold true:*

HC.1) *$x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ form a cycle in G;*

HC.2) *$\{\sigma_{i_1}, \sigma_{i_2}, \ldots, \sigma_{i_k}\} \subseteq \Sigma_{uo}$, where $\sigma_{i_1}, \sigma_{i_2}, \ldots, \sigma_{i_k}$ are such that $f(x_{i_j}, \sigma_{i_j}) = x_{i_{j+1}}$, $j = 1, 2, \ldots, k-1$, and $f(x_{i_k}, \sigma_{i_k}) = x_{i_1}$.*

If x_d is an uncertain state of G_d and besides conditions HC.1) and HC.2), the following condition is also satisfied,

HC.3) *$\ell_{i_j} = Y$, $j = 1, 2, \ldots, k$,*

then x_d has an indeterminate hidden cycle. □

In accordance with Definition 2.8, only hidden (but not indeterminate) cycles may exist in states x_d^N and x_d^Y of G_d, on the other hand, indeterminate hidden cycles may appear only in x_d^{YN}. Notice that in the verification of language diagnosability, state x_d^Y (resp. x_d^N) ensures that the failure has (resp. has not) occurred, and so, the existence of hidden cycles in normal or certain states of G_d does not affect the language diagnosability. On the other hand, the existence of indeterminate hidden cycles implies that the language is not diagnosable since there will exist two traces, a faulty trace (unbounded length), s, and a normal trace (bounded length), s'', such that $P_o(s) = P_o(s'')$. Hidden cycles are represented in the state transition diagrams of partial diagnosers by dashed self-loops: indeterminate hidden cycles will be labeled as *ihc* and hidden cycles in normal or certain states will be labeled simply as *hc*, since, as it will be seen in the sequel, they do not interfere in the language diagnosability.

The following necessary and sufficient condition for diagnosability can be stated.

Theorem 2.2 *[19, 54] The language L generated by automaton G is diagnosable with respect to projection P_o and $\Sigma_f = \{\sigma_f\}$ if, and only if, its diagnoser G_d has no indeterminate cycles (including hidden cycles).* ■

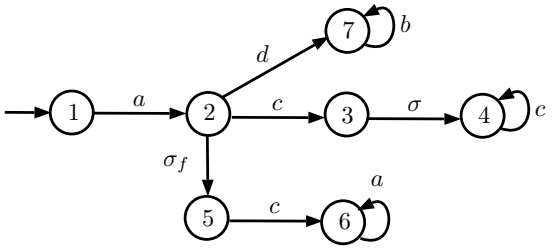


Figure 2.13: Automaton G of Example 2.12.

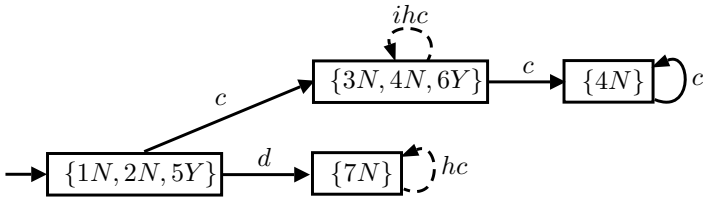


Figure 2.14: Automaton G'_d of Example 2.12.

Example 2.12 *To illustrate the results of Theorem 2.2, consider automaton G whose state transition diagram is depicted in Figure 2.13. Assume that $\Sigma = \{a, b, c, d, \sigma, \sigma_f\}$, $\Sigma_o = \{c, d\}$, $\Sigma_{uo} = \{a, b, \sigma, \sigma_f\}$, and $\Sigma_f = \{\sigma_f\}$. Diagnoser automaton G'_d which includes the hidden cycles is depicted in Figure 2.14. Notice that since G'_d has an indeterminate hidden cycle in state $\{3N, 4N, 6Y\}$, $L(G)$ is not diagnosable with respect to projection P_o and $\Sigma_f = \{\sigma_f\}$. This is so because failure trace $s_Y = a\sigma_f ca^p$, $p \in \mathbb{N}$, and normal (bounded) trace $s_N = ac$ have the same projection with respect to P_o, i.e., $P_o(s_N) = P_o(s_Y) = c$. It is important to point out that G'_d has another hidden cycle, but not indeterminate, in state $\{7N\}$ due to event b.*

2.3.2 Decentralized Diagnosis

When the information is distributed, as in the case of communication networks, manufacturing systems, and electric power systems, centralized diagnosis is no longer used, being replaced with decentralized diagnosis systems. The authors in [21] proposed a decentralized architecture, in which, sites S_i, $i = 1, 2, \ldots, N_s$, observe the system behavior based on the information provided by the sensors connected to it; therefore, forming sets Σ_{o_i}, $i = 1, 2, \ldots, N_s$, of observable events for each site, and so, all events $\sigma \in \Sigma \backslash \Sigma_{o_i}$ are considered unobservable for site S_i. In the decentralized

structure of Figure 2.15, each site processes the information received (an occurrence of an event) and can only communicate their diagnosis decision to the coordinator, which processes this information according to a predetermined rule and makes a decision regarding the failure occurrence; this process is called protocol. One of the protocols proposed in [21] has led to the concept of codiagnosability, which is a property that requires that every trace $s \in \Psi(\Sigma_f)$ be diagnosed by at least one local diagnoser.

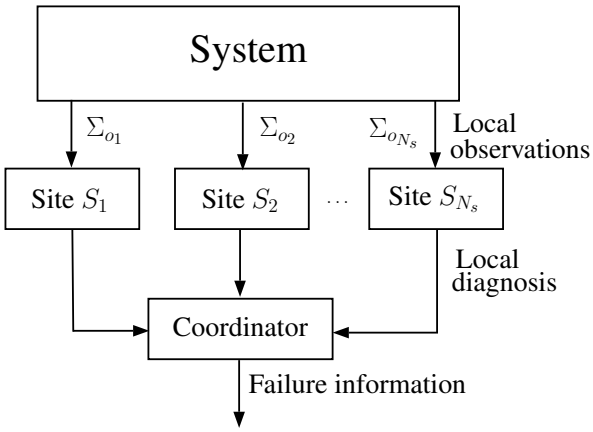


Figure 2.15: Coordinated decentralized architecture.

Definition 2.9 *(Codiagnosability) Suppose that there are N_s local sites. Then, a live and prefix-closed language L is codiagnosable with respect to projections $P_{o_i} : \Sigma^* \to \Sigma_{o_i}^*$ $(i \in I_{N_s} = \{1, \ldots, N_s\})$ and Σ_f if and only if:*

$$(\exists n \in \mathbb{N})(\forall s \in \Psi(\Sigma_f))(\forall t \in L/s, |t| \geq n) \Rightarrow (\exists i \in I_{N_s})(\forall w \in P_{o_i}^{-1}(P_{o_i}(st)) \cap L)[\Sigma_f \in w].$$

Definition 2.9 of codiagnosability generalizes Definition 2.6 of diagnosability, *i.e.*, the decentralized case reduces to the centralized one when there exists only one site. Implicit in the definition of codiagnosability is the fact that none of the sites can alone diagnose the failure occurrence; otherwise there would not be necessary to use a decentralized structure.

For codiagnosability verification [21], it is assumed that G has no cycles of unobservable events with respect to $\Sigma_{o_i}, \forall i \in I_{N_s}$. Namely, assumptions $H1.$ and $H2.$ holds true. In this regard, a test for codiagnosability verification was proposed in [21] based on a test automaton G_{test}, defined as:

$$G_{test} = (||_{i=1}^{N_s} G_{d_i})||G_d, \tag{2.24}$$

where $G_{d_i} = (X_{d_i}, \Sigma_{o_i}, f_{d_i}, \Gamma_{d_i}, x_{0_{d_i}})$ denotes the local diagnoser for site S_i, $i = 1, 2, \ldots, N_s$ and $G_d = (X_d, \Sigma_o, f_d, \Gamma_d, x_{0_d})$, where $\Sigma_o = \bigcup_{i=1}^{N_s} \Sigma_{o_i}$, denotes the centralized diagnoser. Note that states x_t of G_{test} have the following structure: $x_t = (x_{d_1}, x_{d_2}, \ldots, x_{d_{N_s}}, x_d)$, where $x_{d_i} \in X_{d_i}$ and $x_d \in X_d$. The definitions of uncertain state and indeterminate cycles have been extended to codiagnosability as follows:

Definition 2.10 *(Certain and uncertain states in G_{test}) A state x_t of G_{test} is Y-certain if x_d is Y-certain and x_{d_i} is Y-certain for some $i \in \{1, 2, \ldots, N_s\}$, and is uncertain if x_d is uncertain and x_{d_i} is uncertain for all $i \in \{1, 2, \ldots, N_s\}$.*

Definition 2.11 *(Indeterminate cycles in G_{test}) A cycle in G_{test} is indeterminate if all the corresponding cycles in G_{d_i}, $i \in \{1, 2, \ldots, N_s\}$ are indeterminate.*

The following theorem can be stated.

Theorem 2.3 *[21] A live and prefix-closed language L is codiagnosable with respect to projections $P_{o_i} : \Sigma^* \to \Sigma_{o_i}^*$, $i = 1, 2, \ldots, N_s$ and $\Sigma_f = \{\sigma_f\}$, if and only if, G_{test} has no indeterminate cycles.* ■

Example 2.13 *Let us consider the system modeled by automaton G depicted in Figure 2.16, where $\Sigma = \{a, b, c, d, \sigma_f\}$ and assume that not all observable events are available in one place. Therefore, it is necessary to rely on a decentralized diagnosis scheme. Let $\Sigma_{o_1} = \{a, c\}$, $\Sigma_{o_2} = \{b, c\}$, $\Sigma_{uo} = \{d, \sigma_f\}$. We will now verify the codiagnosability of $L(G)$ with respect to P_{o_i}, $i = 1, 2$, and Σ_f by using Theorem 2.3. First, we build the partial diagnosers G_{d_1} for $\Sigma_{o_1} = \{a, c\}$, G_{d_2} for $\Sigma_{o_2} = \{b, c\}$, and diagnoser G_d for $\Sigma_o = \{a, b, c\}$, which are depicted in Figures 2.17(a), 2.17(b) and 2.17(c), respectively. Notice that L is not diagnosable with respect to P_{o_i}, $i = 1, 2$, and Σ_f, due to the existence of indeterminate cycles in state $\{0N, 1Y, 2Y\}$ of G_{d_1} and G_{d_2}. The next step is to compute $G_{test} = G_{d_1} || G_{d_2} || G_d$. depicted in Figure 2.18. Notice that G_{test} has a cycle in uncertain state $(\{0N, 1Y, 2Y\}, \{0N, 1Y, 2Y\}, \{0N, 1Y, 2Y\})$. Since this cycle forms indeterminate cycles in G_{d_1}, G_{d_2} and G_d, they also form indeterminate cycle in G_{test}, and thus, L is not codiagnosable with respect to projections P_{o_i}, $i = 1, 2$ and $\Sigma_f = \{\sigma_f\}$. If we examine the traces of G, we can see that failure trace $s_Y = c^p \sigma_f d c^q$, $p, q \in \mathbb{N}$, is*

not detected by sites S_1 and S_2, since there exist normal traces (not necessarily equal) $s_{N_1} = s_{N_2} = c^p$ such that $P_{o_1}(s_Y) = P_{o_1}(s_{N_1}) = c^n$ and $P_{o_2}(s_Y) = P_{o_2}(s_{N_2}) = c^n$, $n \in \mathbb{N}$.

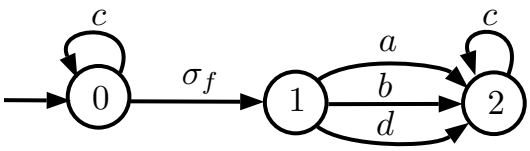


Figure 2.16: Automaton G of Example 2.13.

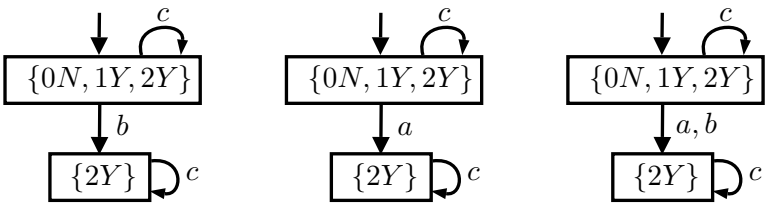


Figure 2.17: Automata G_{d_1} (a); G_{d_2} (b); and G_d (c) of Example 2.13.

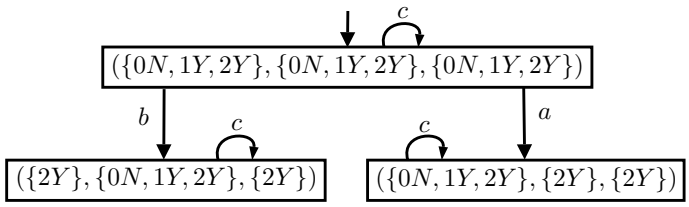


Figure 2.18: Automaton $G_{test} = G_{d_1}||G_{d_2}||G_d$ of Example 2.13.

It is important to remark that the diagnoser automaton can be used either off-line to check diagnosability or online (on-the-fly) by connecting it to the system to provide on-line diagnosis upon the occurrence of observable events, being an efficient structure because it provide a complete characterization of the diagnosis problem under the considered model since updating the diagnosis after a new observation only requires the firing of a single transition. However, the construction of the entire diagnoser may be unwieldy as in the worst case its size is exponential in the number of states of G, as well as in the number of faults if a single diagnoser is desired. The second limitation can be addressed by building separate diagnosers

for each fault type. In this case, when building a diagnoser for a given fault type, the events corresponding to the other fault types are treated as other unobservable events; thus, the total complexity is linear in the number of fault types.

The limitation for off-line purpose can be addressed by using the notion of verifier automaton [44–47], whose corresponding verification algorithm requires polynomial time in the cardinality of the state space and the event set of G as opposed to diagnosers that require exponential time on the cardinality of the state space of G. Broadly speaking, the verifier automaton is the parallel composition of the system behavior with faults and the system behavior without faults, with a synchronization on the observable events.

For the computation of the verifier automaton G_V proposed in [47], we need to define the renaming function R. In order to do so, let $\Sigma_i = \{\sigma_{R_i} : \sigma \in \Sigma_{uo_i} \setminus \Sigma_f\}$, $\Sigma_N = \Sigma \setminus \Sigma_f$ and define $\Sigma_{R_i} = \Sigma_{o_i} \cup \Sigma_i$, for $i = 1, \ldots, N_s$. Then, $R_i : \Sigma_N \to \Sigma_{R_i}$, $i = 1, 2, \ldots, m$, such that:

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

Verifier G_V is constructed according to Algorithm 2.2 [47].

Algorithm 2.2 *Construction of automaton G_V [47]*

Input Automaton G, sets Σ_f, Σ_{o_i} and Σ_{uo_i}, $i = 1, 2, \ldots, N_s$.
Output Automaton G_V.

STEP 1. Compute automaton G_N that models the normal behavior of G, as follows:

STEP 1.1 Define $\Sigma_N = \Sigma \setminus \Sigma_f$.

STEP 1.2 Build automaton A_N, depicted in Figure 2.19, composed of a single state N (also its initial state) with a self-loop labeled with all events in Σ_N,.

STEP 1.3 Construct the nonfailure automaton $G_N = G \times A_N = (X_N, \Sigma, f_N, \Gamma_N, x_{0,N})$.

STEP 1.4 Redefine the event set of G_N as Σ_N.

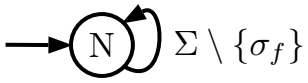


Figure 2.19: Automaton A_N.

STEP 2. Compute automaton G_F that models the failure behavior of the system, as follows:

STEP 2.1 Compute $G_\ell = G||A_\ell$, according to Equation (2.23), and mark all states of G_ℓ whose second coordinate is equal to Y.

STEP 2.2 Compute the failure automaton $G_F = CoAc(G_\ell)$.

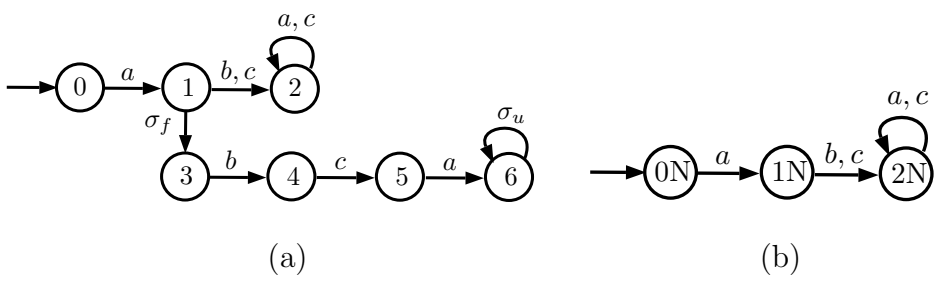
STEP 3. Construct automata $G_{N,i} = (X_N, \Sigma_{R_i}, f_{N,i}, \Gamma_{N,i}, x_{0,N})$, for $i = 1, 2, \ldots, N_s$, with $f_{N,i}(x_N, R_i(\sigma)) = f_N(x_N, \sigma)$ for all $\sigma \in \Sigma_N$.

STEP 4. Compute the verifier automaton $G_V = (||_{i=1}^{N_s} G_{N,i})||G_F = (X_V, (\bigcup_{i=1}^{N_s} \Sigma_{R_i}) \cup \Sigma, f_V, x_{0,V})$.

Codiagnosability verification can be carried out using G_V, according to the following theorem.

Theorem 2.4 *[47] L is not codiagnosable with respect to P_{o_i}, $i = 1, \ldots, N_s$, and Σ_f if and only if there exists a cycle $cl := (x_V^k, \sigma_k, x_V^{k+1}, \ldots, x_V^l, \sigma_l, x_V^k)$, where $l \geq k \geq 0$, in G_V satisfying the following conditions:*

$$\exists j \in \{k, k+1, \ldots, l\},\ \textit{s.t. for some}\ x_V^j, (x_F^j = xY) \wedge (\sigma_j \in \Sigma).$$



(a) (b)

Figure 2.20: Automata G (a) and G_N (b) of Example 2.14.

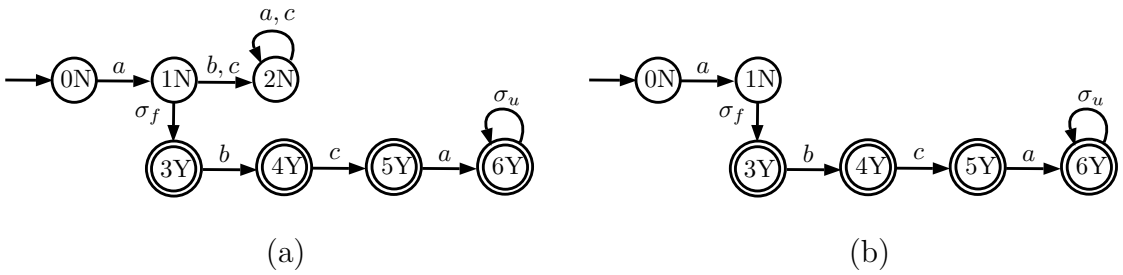


Figure 2.21: Automata G_ℓ (a) and G_F (b) of Example 2.14.

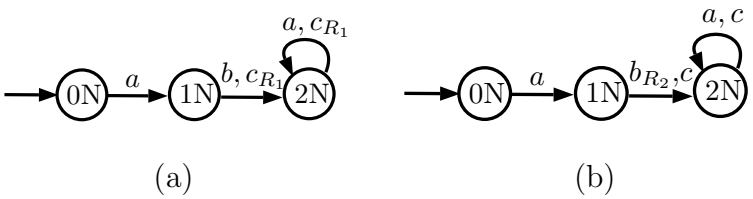


Figure 2.22: Automata $G_{N,1}$ (a) and $G_{N,2}$ (b) of Example 2.14.

Example 2.14 *[47] Consider the system modeled by automaton G depicted in Figure 2.20(a), and suppose we want to verify the codiagnosability of $L(G)$ with respect to P_{o_i}, $i = 1, 2$, and Σ_f, where $\Sigma = \{a, b, c, \sigma_u, \sigma_f\}$, $\Sigma_{o_1} = \{a, b\}$, $\Sigma_{o_2} = \{a, c\}$, $\Sigma_{uo} = \{\sigma_u, \sigma_f\}$ and $\Sigma_f = \{\sigma_f\}$. According to Algorithm 2.2, the first step is to obtain automaton A_N to compute the nonfailure automaton G_N, which are depicted in Figures 2.19 and 2.20(b), respectively. The next step is to obtain automaton G_ℓ, depicted in Figure 2.21(a), by computing the parallel composition between automaton G and A_ℓ (shown in Figure 2.10) and marking all states that has Y as the second component. We must now build the failure automaton G_F, depicted in Figure 2.21(b), by taking the coaccessible part of G_ℓ. The next step of Algorithm 2.2 is to obtain automata $G_{N,1}$ and $G_{N,2}$ (shown in Figures 2.22(a) and 2.22(b), respectively) from G_N by renaming the unobservable events in the sets $\Sigma_{uo_1} \setminus \Sigma_f = \{c, \sigma_u\}$ and $\Sigma_{uo_2} \setminus \Sigma_f = \{b, \sigma_u\}$, respectively. The final step of Algorithm 2.2 is the computation of the verifier automaton $G_V = G_{N,1}||G_{N,2}||G_F$, depicted in Figure 2.23. In order to check the codiagnosability it is necessary to find cycles of failure states in G_V formed with events in Σ. Notice that G_V has several cycles, but only one (the one formed in state $\{2N, 2N, 6Y\}$) has some event (σ_u) in Σ; all the others have events in either $\Sigma_{R_1} \setminus \Sigma_f$ or $\Sigma_{R_2} \setminus \Sigma_f$. The existence of cycle in $\{2N, 2N, 6Y\}$ formed by $\sigma_u \in \Sigma$ implies that $L(G)$ is not codiagnosable with respect to P_{o_i}, $i = 1, 2$, and Σ_f.*

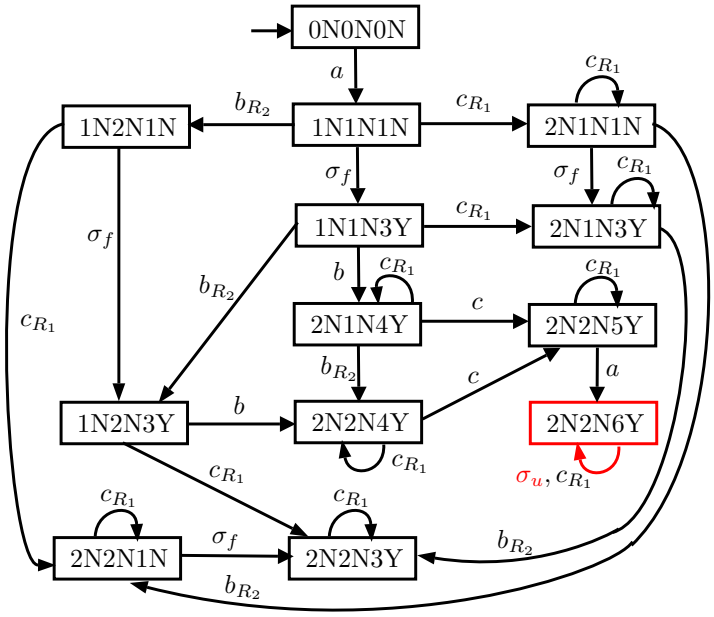


Figure 2.23: Automaton G_V of Example 2.14.

2.4 Concluding Remarks

The basic definitions about DES and failure diagnosis were presented in this chapter. In Chapter 3, we will revisit the problem of codiagnosability of discrete event systems, and first, we will propose a new necessary and sufficient condition to check the property of codiagnosability by using an algorithm based on a diagnoser-like automaton that overcomes the drawbacks of the approaches presented in [19, 21]. Second, assuming that the considered language is codiagnosable, we will propose an algorithm to extend the language of verifier presented in Section 2.3 to show not only the ambiguous paths but also those paths that lead to language diagnosis. Third, we will show an application of these algorithms.

Chapter 3

Codiagnosability of Discrete Event Systems Revisited: A New Necessary and Sufficient Condition and Its Applications

In Chapter 2, we presented diagnosability and codiagnosability verification by using diagnosers proposed by [19] and [21], respectively. These verifications have some drawbacks, since the authors assume language liveness and nonexistence of unobservable cycles of states connected with unobservable events only. In this chapter, we revisit the problem of codiagnosability of discrete event system and propose a new test for diagnosability verification by changing the diagnoser structure so as to consider both observable and unobservable events in order to circumvent the drawbacks of the approaches proposed by [19] and [21]. In addition, we can adapt this diagnoser-like automaton to check the codiagnosability of networked discrete event systems with timing structure which will be presented in Chapter 4.

Regarding codiagnosability verification using verifiers, we saw in Chapter 2 that when the language is not codiagnosable the verifier proposed in [47] necessarily has a cycle cl of failure states where at least one of the events in cl belongs to the set of events of the plant Σ. However, since the verifier is based on the search for ambiguous traces, for (co)diagnosable languages, the verifier language stops when the ambiguity ceases to exist. As a consequence, events that remove the ambiguity

are not shown in the verifier. Thus, we will propose an algorithm to extend the verifier automaton proposed in [47] to show the paths that lead to failure diagnosis.

As an application of the algorithms in this chapter to be proposed (the diagnoser-like automaton and extended verifier), we will address two important problems of DES: τ-codiagnosability and K-codiagnosability. In failure diagnosis, it is not only important to check if the failure event has occurred or not (language diagnosability) but also how long the diagnosis system takes to detect the failure occurrence (τ-codiagnosability), or, if no time information is available, how many events occur after the failure occurs before the diagnosis system becomes sure of its occurrence (K-codiagnosability). One way to verify τ-codiagnosability is by adding weights associated with transitions of the plant automaton, which can be accomplished by using weighted automata. The use of weighted automata allows the maximum delay τ to detect a failure to be defined as a performance measure. This approach has another advantage that it also allows the verification of K-codiagnosability by replacing all transition weights with unity weight, making, therefore, K-codiagnosability a particular case of τ-codiagnosability. A version of the results obtained in this chapter was published in [63] and submitted for publication in [79, 80].

In Section 3.1, we propose a diagnoser-like automaton, and based on this automaton, we generalize the necessary and sufficient condition for diagnosability to codiagnosability, and present a diagnosability verification algorithm which relies solely on the search for strongly connected components. In Section 3.2, we propose an extended verifier, developed to show not only the ambiguous paths but also those paths that lead to language diagnosis. In Section 3.3, we apply the diagnoser-like automaton and extended verifier to compute τ- and K-codiagnosability. Finally, in Section 3.4, we draw some conclusions.

3.1 A New Diagnoser-Based Test for Codiagnosability Verification

We address, in this section, the problem of language codiagnosability. Encouraged by a recent conjecture [57] that diagnosers have state size $\Theta(n^{0.77 \log k + 0.63})$, on the average, where k (resp. n) is the number of events (resp. states) of the plant

automaton, we present a new necessary and sufficient condition for a language co-diagnosability of DES and, based on this condition, we propose a new test for its verification that is based on a diagnoser-like automaton. Since this automaton test has in its event set both observable and unobservable events of the plant, the usual assumptions on language liveness and nonexistence of unobservable cycles of states connected with unobservable events only are no longer required here. An important advantage of the test proposed here is that the search for indeterminate cycles is replaced with the search for strongly connected components; the former has computational complexity that is worse than exponential whereas the latter is linear in the state size. Other advantages of the proposed test is that diagnosability verification becomes a particular case of codiagnosability verification. The motivation for proposing a new test is as follows:

- diagnosers G_d and G_{test} do not carry enough information to determine if an observed cycle of uncertain states is an indeterminate cycle, since it is necessary to also perform a search for cycles in G. Namely, in order to check diagnosability (resp. codiagnosability) in accordance with the test proposed in [19] (resp. [21]), we need to find a set of uncertain states in G_d (resp. G_{test} associated to uncertain states in G_{d_1}, G_{d_2}, ..., $G_{d_{N_s}}$) and verify if these states lead to two cycles in G_ℓ: one cycle associated with states labeled by N and another cycle formed with states labeled by Y. Thus, the centralized diagnoser (resp. decentralized diagnoser), by itself, does not have enough information to determine if a language is diagnosable or not;

- the same is true in G_d and G_{test} as far as hidden cycles are concerned, *i.e.*, it is also necessary to search for cycles of states connected with unobservable events in G;

- the search for cycles, as pointed out in [58], is worse than exponential in the number of states;

- although the approaches proposed to diagnosability and codiagnosability verification are both based on diagnosers, automaton G_{test} proposed in [21] for codiagnosability verification is not a generalization of G_d proposed in [19] — notice that, for $N_s = 1$, $G_{test} = G_d||G_d$ which, although $L(G_d||G_d) = L(G_d)$,

an unnecessary parallel composition must be carried out in order to obtain the G_{test};

- We will remove assumptions of language liveness and nonexistence of unobservable cycles of states connected with unobservable events only;
- We will check the codiagnosability of networked discrete event systems with timing structure by using a slight variation of the decentralized diagnoser presented in this chapter.

Let us introduce the following test automaton:

$$G_{scc} = G_d||G_\ell, \tag{3.1}$$

where $G_\ell = G||A_\ell$, A_ℓ is the label automaton and G_d is computed according to Equation (2.23). We may state the following result.

Fact 3.1 *$L(G_{scc}) = L(G_\ell) = L(G)$.*

Proof. The proof is straightforward and comes from the fact that $L(G_\ell) = L$ and $L(G_d) = P_o(L)$ since $G_d = Obs(G_\ell, \Sigma_o)$. ■

Notice that, since automaton G_{scc} is obtained by performing a parallel composition between G_d and G_ℓ, its states are of the form (x_d, x_ℓ), which leads to the following inclusion relationship between x_ℓ and x_d.

Fact 3.2 *For every state (x_d, x_ℓ) of G_{scc}, $x_\ell \subseteq x_d$.*

Proof. The proof is straightforward and comes from the fact that automaton $G_{scc} = G_d||G_\ell = \text{Obs}(G_\ell, \Sigma_o)||G_\ell$ is a state partition automaton [81]. ■

We will now present a necessary and sufficient condition for language diagnosability that replaces the search for cycles with the search for strongly connected components, which, as shown in [60] and [59], is linear in the number of transitions.

Theorem 3.1 *The language L generated by automaton G is diagnosable with respect to projection P_o and $\Sigma_f = \{\sigma_f\}$ if, and only if, G_{scc} does not have strongly connected components formed with states (x_d, x_ℓ), such that x_d is uncertain and x_ℓ is an Y-labeled state, i.e., $x_\ell = (x, Y)$, where $x \in X$.*

Proof. ($\Rightarrow$) Assume that there exists a strongly connected component formed with states (x_{d_1}, x_{ℓ_1}), (x_{d_2}, x_{ℓ_2}), $\ldots, (x_{d_n}, x_{\ell_n})$ such that x_{d_i}, x_{ℓ_i} $i = 1, \ldots, n$, are, respectively, uncertain and Y-alabeled states.

Two possibilities arise:

(i) $x_{d_1} = x_{d_2} = x_{d_3} = \ldots = x_{d_n} = x_d$. This means that states (x_d, x_{ℓ_1}), (x_d, x_{ℓ_2}), $\ldots, (x_d, x_{\ell_n})$ are connected by unobservable events since these events are private events of G_ℓ. In addition, due to Fact 3.2, $x_{\ell_i} \in x_d$, which, together with the fact that x_{ℓ_i} are Y-labeled states, then, in accordance with Definition 2.8, we can conclude that there exists an indeterminate hidden cycle in x_d; therefore according to Theorem 2.2, L is not diagnosable.

(ii) There exists $\{i_1, i_2, \ldots, i_p\} \subseteq \{1, \ldots, n\}$ such that $x_{d_{i_k}} \neq x_{d_{i_l}}$, $k \neq l$, $k, l \in \{1, 2, \ldots, p\}$. Since x_{ℓ_i}, $i = 1, 2, \ldots, n$ are Y-labeled states, and $L(G_{scc}) = L$, then, there exists an unbounded trace $s_Y = st \in L$ such that $s \in \Psi(\Sigma_f)$ and $|t| \geq n$, for all $n \in \mathbb{N}$. In addition, since $x_{d_{i_k}}$ $j = 1, 2, \ldots, p$, are uncertain states, then, as proved in [19], there exists a trace $s_N \in L$ such that $P_o(s_Y) = P_o(s_N)$, which implies that L is not diagnosable with respect to P_o and Σ_f.

($\Leftarrow$) Assume, now, that L is not diagnosable with respect to P_o and Σ_f. Thus, there exist two traces: an unbounded trace $s_Y = st$, $s \in \Psi(\Sigma_f)$, and $|t| > n$ for all $n \in \mathbb{N}$ and a not necessarily unbounded trace s_N, such that $\Sigma_f \notin s_N$, which satisfies $P_o(s_Y) = P_o(s_N)$. This implies that $\exists x_d \in X_d$, $s_d \in L(G_d)$: $f_d(x_{0,d}, s_d) = x_d$, x_d uncertain and $P_o(s_Y) = P_o(s_N) = s_d$. In addition, since $L(G_\ell) = L(G)$, $\exists x_{\ell_1} \in X_\ell$: $f_\ell(x_{0,\ell}, s) = x_{\ell_1}$, where x_{ℓ_1} an Y-labeled state. Two possibilities arise:

(i) If $P_o(s_N)$ is bounded, then $P_o(s_N) = P_o(s_Y) = s_d$ will be bounded. Therefore s_Y can be written as: $s_Y = st_1t_2$, where $P_o(st_1) = s_d$ and $P_o(t_2) = \varepsilon$. Thus, $\exists p \leq |X_\ell|$: $t_2 = (\sigma_1, \sigma_2, \ldots, \sigma_p)^m$, $m \geq 1$, such that σ_i is unobservable, $\forall i = 1, 2, \ldots, p$. This implies that there exists an indeterminate hidden cycle in x_d formed with Y-labeled states $x_{\ell_1}, x_{\ell_2}, \ldots, x_{\ell_p}$ related as follows:

$$f(x_{\ell_i}, \sigma_i) = \begin{cases} x_{\ell_{i+1}}, & \text{if } i = 1, \ldots, p-1 \\ x_{\ell_1}, & \text{if } i = p. \end{cases} \tag{3.2}$$

Since $\sigma_i, i = 1, \ldots, p$ are private events of G_ℓ, and $x_{\ell_i} \subseteq x_d$, according to Fact 3.2, we have that the following cyclic path is formed in G_{scc}: $((x_d, x_{\ell_1}), \sigma_1, (x_d, x_{\ell_2}), \sigma_2, \ldots, (x_d, x_{\ell_p}), \sigma_p, (x_d, x_{\ell_1}))$, which defines the following cycle in G_{scc}: $\{(x_d, x_{\ell_1}), (x_d, x_{\ell_2}), \ldots, (x_d, x_{\ell_p}), (x_d, x_{\ell_1})\}$. Since, cycles are strongly connected components, the result is proved.

(ii) If $P_o(s_N)$ is unbounded, then $P_o(s_N) = P_o(s_Y) = s_d$ will be unbounded. Consequently, since n can be arbitrarily large, then $P_o(t)$ is unbounded. Thus, $\exists p \leq |X_\ell| : t = (\sigma_1, \sigma_2, \ldots, \sigma_p)^m$, $m \geq 1$, $\{i_1, i_2, \ldots, i_k\} \subseteq \{1, \ldots, p\}$ such that $\sigma_{i_1}, \sigma_{i_2}, \ldots, \sigma_{i_k}$ are observable. Without loss of generality and, for the sake of simplicity, let us assume that there exist only two observable events σ_k, σ_ℓ, $k, \ell \in \{1, 2, \ldots, p\}$ and let $x_{\ell_i}, i = 1, \ldots, p$ be defined according to Equation (3.2). Therefore, the following cyclic path is defined in G_d: $(x_{d_1}, \sigma_\ell, x_{d_k}, \sigma_k, x_{d_1})$, which implies that following cyclic path is defined in G_{scc}: $((x_{d_1}, x_{\ell_1}), \sigma_1, \ldots, \sigma_{k-1}, (x_{d_1}, x_{\ell_{k-1}}), \sigma_k, (x_{d_k}, x_{\ell_k}),\ \sigma_{k+1}, (x_{d_k}, x_{\ell_k+1}), \sigma_{k+2}, \ldots, (x_{d_{\ell-1}}, x_\ell),\ \sigma_\ell, (x_{d_1}, x_\ell),\ \sigma_{\ell+1}, \ldots, \sigma_{p-1}, (x_{d_1}, x_{\ell,p-1}), \sigma_p, (x_{d_1}, x_{\ell_1}))$, whose first components of the states of G_{scc} are uncertain states of G_d and the second are Y-labeled states; therefore concluding the proof. ■

Therefore, according to Theorem 3.1, diagnosability verification can be performed according to Algorithm 3.1.

Algorithm 3.1 *Diagnosability verification using automaton G_{scc}*

Input: *Plant G, P_o and $\Sigma_f = \{\sigma_f\}$.*

Output: *Diagnosability decision: Yes or No.*

STEP 1. *Compute automaton $G_{scc} = G_d || G_\ell$.*

STEP 2. *Find all strongly connected components of G_{scc}.*

STEP 3. *Verify if there exists at least one strongly connected component formed with states (x_d, x_ℓ) such that x_d is uncertain and x_ℓ is an Y-labeled state, i.e., $x_\ell = (x, Y)$, where $x \in X$.*

STEP 4. ***If** the answer is yes, **then** L is not diagnosable with respect to projection P_o and $\Sigma_f = \{\sigma_f\}$. Otherwise, L is diagnosable.*

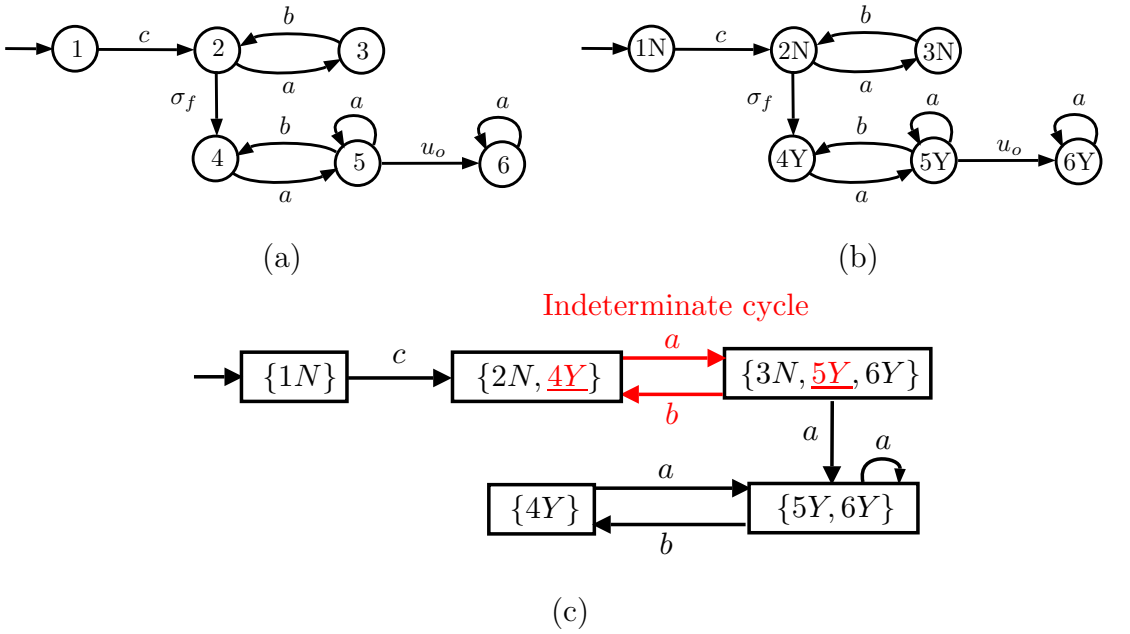


Figure 3.1: Automata G (a); G_ℓ (b) and Diagnoser automaton G_d (c) of Example 2.10, which are considered again in Example 3.1.

Example 3.1 *Consider again the system of Example 2.10, modeled by automaton G shown again in Figure 3.1(a), where $\Sigma_o = \{a, b, c\}$ is the set of observable events, and $\Sigma_o = \{u_o, \sigma_f\}$. In order to check the language diagnosability according to Algorithm 3.1, we first compute G_{scc}, depicted in Figure 3.2, by performing a parallel composition between automaton G_ℓ of Figure 3.1(b) and diagnoser automaton $G_d = Obs(G_\ell, \Sigma_o)$, depicted in Figure 3.1(c). If we examine G_{scc}, we can observe two properties of G_{scc}: $L(G_{scc}) = L(G_\ell) = L(G)$ (Fact 3.1) and its states are of the form (x_d, x_ℓ), being such that $x_\ell \subseteq x_d$ (Fact 3.2). Since there exists a strongly connected component formed by states $(\{2N, 4Y\}, 4Y)$ and $(\{3N, 5Y, 6Y\}, 5Y)$ whose first component x_d is an uncertain state and x_ℓ is an Y-labeled state, we can conclude according to Algorithm 3.1, that language $L(G)$ is not diagnosable with respect to projection P_o and $\Sigma_f = \{\sigma_f\}$. The same result was obtained in Example 2.10. However, notice that here is not necessary to find two cycling traces in automaton G that have same projection to determine the diagnosability decision.*

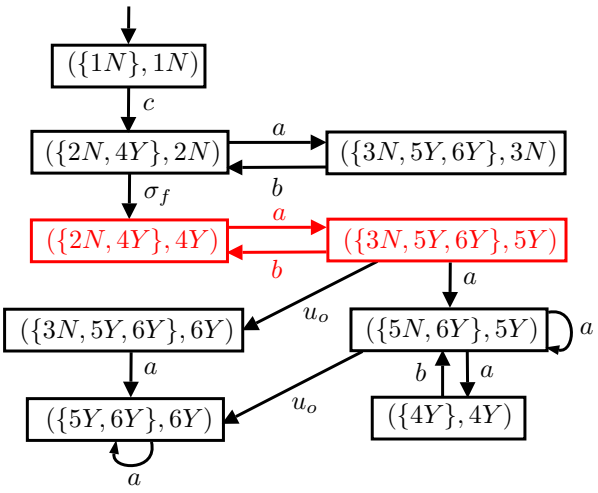
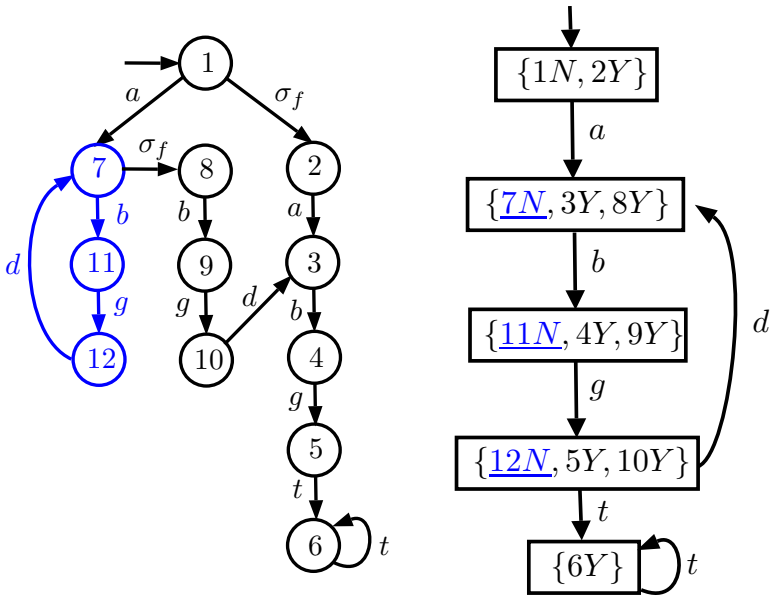


Figure 3.2: Automaton G_{scc} of Example 3.1.



(a) (b)

Figure 3.3: Automata G (a) and G_d (b) of Example 2.11, which are considered again in Example 3.2.

Example 3.2 *Consider again the system of Example 2.11 modeled by automaton G and its diagnoser G_d represented again in Figure 3.3(a) and 3.3(b), respectively. We can compute G_{scc} through Equation* (3.1)*, and the resulting automaton is depicted in Figure 3.4. Due to the absence of strongly connected components formed by states (x_d, x_ℓ) such that x_d is uncertain and x_ℓ is an Y-labeled state, we can say that $L(G)$ is diagnosable. Notice that, to obtain the diag-*

nosability decision by using Algorithm 3.1 it is not necessary to identify if uncertain cycles of G_d are indeterminate cycles. For instance, the cycle formed by normal states appear in G_{scc} as strongly connected components formed by states $(\{12N, 5Y, 10Y\}, 12N), (\{11N, 4Y, 9Y\}, 11N), (\{7N, 3Y, 8Y\}, 7N)$, where the first component of x_d is uncertain state whereas the second component is a normal state.

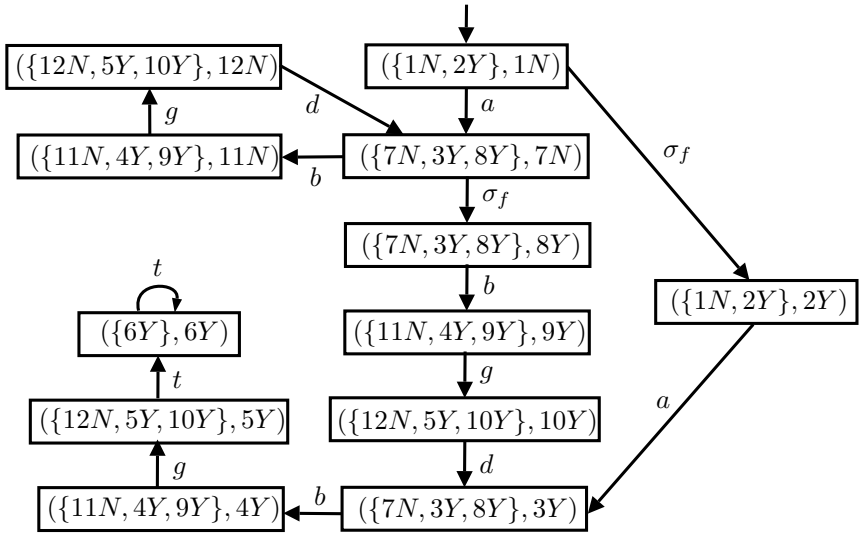


Figure 3.4: Automaton G_{scc} of Example 3.2.

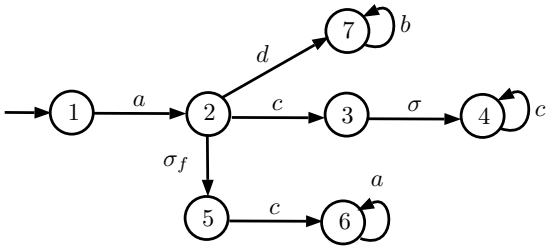


Figure 3.5: Automaton G of Example 2.12 considered again in Example 3.3.

Example 3.3 *Let us consider again automaton G whose state transition diagram is depicted again in Figure 3.5. Remember that $\Sigma = \{a, b, c, d, \sigma, \sigma_f\}$, $\Sigma_o = \{c, d\}$, $\Sigma_{uo} = \{a, b, \sigma, \sigma_f\}$, and $\Sigma_f = \{\sigma_f\}$. Diagnoser automaton G_d, which does not include the hidden cycles, is depicted in Figure 3.6. We then obtain G_{scc} by following Equation* (3.1)*, being depicted in Figure 3.7. We can, then, conclude that Language $L(G)$ is not diagnosable with respect to projection P_o and $\Sigma_f = \{\sigma_f\}$ since there exists a strongly connected component formed by state $(\{3N, 4N, 6Y\}, 6Y)$, which*

such that x_d is uncertain and x_ℓ is an Y-labeled state. Notice that it is not necessary to search for hidden cycles as in G'_d of Example 2.12, since G_{scc} "shows" the cycles in $(\{3N, 4N, 6Y\}, 6Y)$ (corresponding to ihc in G_d of Example 2.12) and $(\{7N\}, 7N)$(corresponding to hc in G_d of Example 2.12).

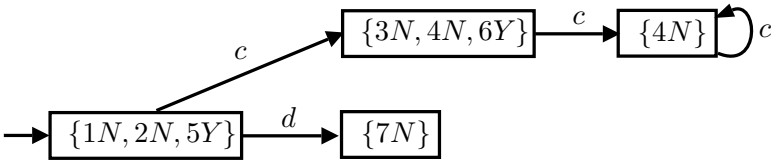


Figure 3.6: Automaton G_d of Example 2.12 (without hidden cycles).

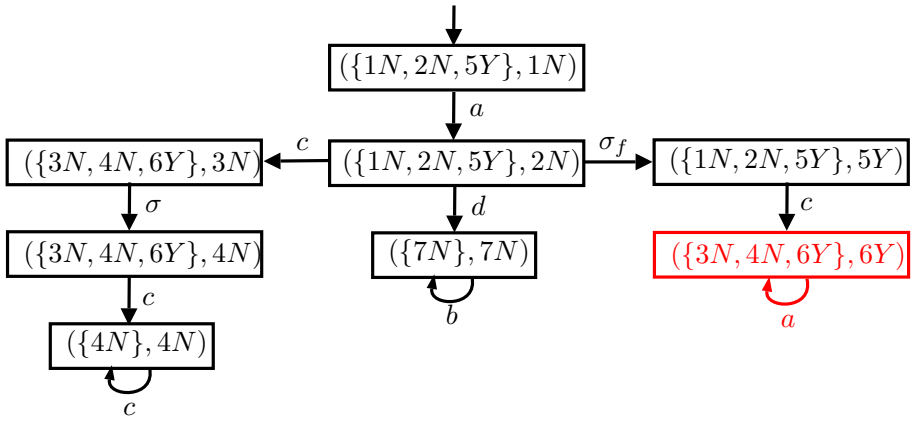


Figure 3.7: Automaton G_{scc}.

We will now extend the diagnosability verification test to codiagnosability, and, to this end, we introduce automaton $G_{scc}^{N_s}$, which is defined as follows:

$$G_{scc}^{N_s} = (||_{i=1}^{N_s} G_{d_i})||G_\ell, \tag{3.3}$$

where N_s is number of sites. We may state the following result.

Fact 3.3 $L(G_{scc}^{N_s}) = L(G_\ell) = L(G)$.

Proof. The proof is straightforward from Fact 3.1. ■

Notice that, since automaton $G_{scc}^{N_s}$ is obtained by performing a parallel composition between G_{d_i}, $i = \{1, 2, \ldots, N_s\}$ and G_ℓ, its states are of the form $(x_{d_1}, x_{d_2}, \ldots, x_{d_{N_s}}, x_\ell)$. Therefore the following inclusion relationship between x_ℓ and x_{d_i} holds true.

Fact 3.4 *For all states* $(x_{d_1}, x_{d_2}, \ldots, x_{d_{N_s}}, x_\ell)$ *of* $G_{scc}^{N_s}$, $x_\ell \subseteq x_{d_i}$, $i = 1, 2, \ldots, N_s$.

Proof. The proof is straightforward from Fact 3.2. ∎

Let us now define $I_m = \{1, 2, \ldots, m\}$, $m \in \mathbb{N}$. We may state the following result.

Theorem 3.2 *The language* L *generated by automaton* G *is codiagnosable with respect to projections* $P_{o_i} : \Sigma^* \to \Sigma_{o_i}^*$, $i = 1, 2, \ldots, N_s$ *and* $\Sigma_f = \{\sigma_f\}$, *if, and only if,* $G_{scc}^{N_s}$ *has no strongly connected components formed with states* $(x_{d_1}^1, x_{d_2}^1, \ldots, x_{d_{N_s}}^1, x_\ell^1)$, $(x_{d_1}^2, x_{d_2}^2, \ldots, x_{d_{N_s}}^2, x_\ell^2)$, …, $(x_{d_1}^m, x_{d_2}^m, \ldots, x_{d_{N_s}}^m, x_\ell^m)$, *such that, for all* $j \in I_m$, $x_{d_i}^j$, $i = 1, 2, \ldots, N_s$, *is uncertain, and* x_ℓ^j *is an Y-labeled state.*

Proof. ($\Rightarrow$) Let us assume that there exists a strongly connected component formed with states $(x_{d_1}^1, x_{d_2}^1, \ldots, x_{d_{N_s}}^1, x_\ell^1)$, $(x_{d_1}^2, x_{d_2}^2, \ldots, x_{d_{N_s}}^2, x_\ell^2)$,…, $(x_{d_1}^m, x_{d_2}^m, \ldots, x_{d_{N_s}}^m, x_\ell^m)$ in $G_{scc}^{N_s}$, such that, for all $j \in I_m$, $x_{d_i}^j$, $i = 1, 2, \ldots, N_s$, is uncertain, and x_ℓ^j is an Y-labeled state. Since $L(G_\ell) = L(G_\ell || G_\ell)$, we can rewrite $G_{scc}^{N_s}$ as follows:

$$G_{scc}^{N_s} = G_{scc_1} || G_{scc_2} || \ldots || G_{scc_{N_s}} = (G_{d_1} || G_\ell) || (G_{d_2} || G_\ell) || \ldots || (G_{d_{N_s}} || G_\ell).$$

Thus, there is a strongly connected component in $G_{scc}^{N_s}$ formed with states $((x_{d_1}^1, x_\ell^1), (x_{d_2}^1, x_\ell^1), \ldots, (x_{d_{N_s}}^1, x_\ell^1)), ((x_{d_1}^2, x_\ell^2), (x_{d_2}^2, x_\ell^2), \ldots, (x_{d_{N_s}}^2, x_\ell^2))$, …, $((x_{d_1}^m, x_\ell^m), (x_{d_2}^m, x_\ell^m), \ldots, (x_{d_{N_s}}^m, x_\ell^m))$, such that, for all $j \in I_m$, $x_{d_i}^j$, $i = 1, 2, \ldots, N_s$, is uncertain, and x_ℓ^j is an Y-labeled state. By construction, we can see that each G_{scc_i} $i = 1, 2, \ldots, N_s$ has a strongly connected component formed with states $(x_{d_1}^j, x_\ell^j)$, $(x_{d_2}^j, x_\ell^j)$, $\ldots, (x_{d_{N_s}}^j, x_\ell^j)$ such that x_{d_i}, x_ℓ^j are, respectively, uncertain and Y-labeled states. Therefore, according to Theorem 3.1, L is not diagnosable with respect to P_{o_i} and Σ_f.

($\Leftarrow$) Assume, now, that L is not codiagnosable with respect to projections $P_{o_i} : \Sigma^* \to \Sigma_{o_i}^*$, $i = 1, 2, \ldots, N_s$ and $\Sigma_f = \{\sigma_f\}$. Thus, according to Definition 2.9, there exists a fully-ambiguous trace[1] $s \in L(G)$, which, implies, due to Definition 2.6, that L is not diagnosable with respect to P_{o_i} and Σ_f, $i = 1, 2, \ldots, N_s$. Thus, according to Theorem 3.1, there exists a strongly connected component in G_{scc_i}, $(x_{d_i}^1, x_\ell^1)$, $(x_{d_i}^2, x_\ell^2)$, …, $(x_{d_i}^m, x_\ell^m)$, such that x_{d_i}, $i = 1, 2, \ldots, N_s$, are

[1] An unbounded trace $s \in L(G)$ is said to be fully-ambiguous with respect to projections P_{o_i}, $i = 1, 2, \ldots, N_s$ and Σ_f if there exist traces, $s_1, s_2, \ldots, s_{N_s} \in L(G)$ not necessarily unbounded and not necessarily distinct, such that: $P_{o_i}(s) = P_{o_i}(s_i)$, $\Sigma_f \in s$ but $\Sigma_f \notin s_i$, $i = 1, 2, \ldots, N_s$.

uncertain states and x_ℓ^j are Y-labeled states. Since $L(G_{scc}^{N_s}) = L(G_\ell) = L(G_{scc_i})$, if we compute $G_{scc_1}||G_{scc_2}||\ldots||G_{scc_{N_s}} = (G_{d_1}||G_\ell)||(G_{d_2}||G_\ell)||\ldots||(G_{d_{N_s}}||G_\ell)$, we obtain an automaton that has a strongly connected component $((x_{d_1}^1, x_\ell^1), (x_{d_2}^1, x_\ell^1), \ldots, (x_{d_{N_s}}^1, x_\ell^1), x_\ell^1)((x_{d_1}^2, x_\ell^2), (x_{d_2}^2, x_\ell^2), \ldots, (x_{d_{N_s}}^2, x_\ell^2), x_\ell^2), \ldots, ((x_{d_1}^m, x_\ell^m), ((x_{d_2}^m, x_\ell^m), \ldots, ((x_{d_{N_s}}^m, x_\ell^m), x_\ell^m)$ in $G_{scc}^{N_s}$, such that, for all $j \in I_m$, $x_{d_i}^j$, $i = 1, 2, \ldots, N_s$, is uncertain, and x_ℓ^j is an Y-labeled state. Since $L(G_\ell||G_\ell) = L(G_\ell)$, this strongly connected component also appears in $G_{scc}^{N_s}$, which completes the proof. ■

Remark 3.1 *Notice that, when $N_s = 1$, $G_{scc}^1 = G_{d_1}||G_\ell = G_d||G_\ell = G_{scc}$, which shows that the diagnosability test is a particular case of the codiagnosability test developed here as opposed to that proposed in [21].*

Therefore, according to Theorem 3.2, we now present Algorithm 3.2 to be used for codiagnosability verification based on automaton $G_{scc}^{N_s}$.

Algorithm 3.2 *Codiagnosability verification using automaton $G_{scc}^{N_s}$*

Input: *Plant G, P_{o_i}, $i = 1, 2, \ldots, N_s$, and $\Sigma_f = \{\sigma_f\}$.*

Output: *Codiagnosability decision: Yes or No.*

STEP 1. *Compute automaton $G_{scc}^{N_s} = (||_{i=1}^{N_s} G_{d_i})||G_\ell$.*

STEP 2. *Find all strongly connected components of $G_{scc}^{N_s}$.*

STEP 3. *Verify if there exists at least one strongly connected component formed with states $(x_{d_1}^1, x_{d_2}^1, \ldots, x_{d_{N_s}}^1, x_\ell^1)$, $(x_{d_1}^2, x_{d_2}^2, \ldots, x_{d_{N_s}}^2, x_\ell^2)$, $\ldots$, $(x_{d_1}^m, x_{d_2}^m, \ldots, x_{d_{N_s}}^m, x_\ell^m)$, such that, for all $j \in I_m$, $x_{d_i}^j$, $i = 1, 2, \ldots, N_s$, is uncertain, and x_ℓ^j is an Y-labeled state.*

STEP 4. ***If** the answer is yes, **then** L is not codiagnosable with respect to projections $P_{o_i} : \Sigma^* \to \Sigma_{o_i}^*$, $i = 1, 2, \ldots, N_s$ and $\Sigma_f = \{\sigma_f\}$. Otherwise, L is codiagnosable.*

Remark 3.2 *Notice that, for the codiagnosability test proposed here, none of the assumptions made in [19], [21] and [54] are necessary. Therefore, besides being more efficient, as far as the search for cycles is concerned, the test proposed here can be applied to a larger class of DES modeled by automata.*

Example 3.4 *In order to illustrate the results of this section, let us consider automaton G, shown in Figure 3.8(a), where $\Sigma = \{a, b, c, \sigma_f\}$ and $\Sigma_f = \{\sigma_f\}$. Let $\Sigma_{o_1} = \{a, c\}$ and $\Sigma_{o_2} = \{a, b\}$, and consider projections $P_{o_1} : \Sigma^* \to \Sigma_{o_1}^*$ and $P_{o_2} : \Sigma^* \to \Sigma_{o_2}^*$. In order to verify if L is codiagnosable, with respect to projections P_{o_1} and Σ_f, and P_{o_2} and Σ_f, by applying Algorithm 3.1, we must, firstly, compute $G_{d_1} = Obs(G_\ell, \Sigma_{o_1})$ and $G_{d_2} = Obs(G_\ell, \Sigma_{o_2})$, depicted in Figures 3.8(a) and (b), respectively.*

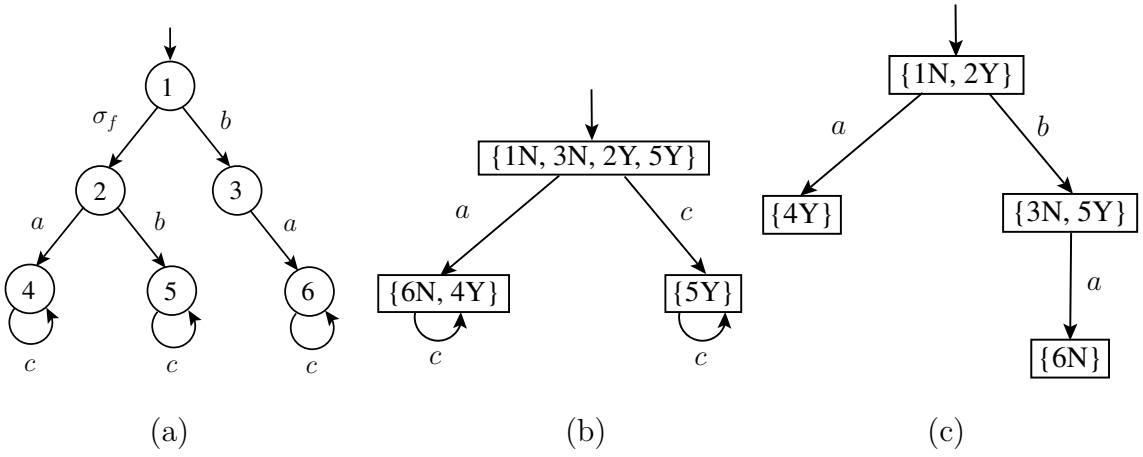


Figure 3.8: Plant G (a); diagnoser G_{d_1}, for $\Sigma_{o_1} = \{a, c\}$ (b) and diagnoser automaton G_{d_2}, for $\Sigma_{o_2} = \{a, b\}$ (c) of Example 3.4.

Secondly, we compute $G_{scc_1} = G_{d_1} || G_\ell$, depicted in Figure and Figure 3.9. According to Theorem 3.1, since G_{scc_1} has a strongly connected component formed with state $(\{6N, 4Y\}, \{4Y\})$, the language L generated by automaton G is not diagnosable with respect to projections P_{o_1} and Σ_f. The same conclusion we draw when we compute $G_{scc_2} = G_{d_2} || G_\ell$, which is shown in Figures 3.10, since G_{scc_2} has a strongly connected component formed with state $(\{3N, 5Y\}, \{5Y\})$, L is not diagnosable with respect to P_{o_2} and Σ_f.

Finally, let us verify if L is codiagnosable with respect to P_{o_1}, P_{o_2} and $\Sigma_f = \{\sigma_f\}$, by applying Algorithm 3.2. Automaton $G_{scc}^2 = G_{d_1} || G_{d_2} || G_\ell$ is depicted in Figure 3.11. Notice that since G_{scc}^2 has no strongly connected components formed with states $(x_{d_1}, x_{d_2}, x_\ell)$, where x_{d_i}, $i = 1, 2$, are both uncertain and x_ℓ is an Y-labeled state, we can conclude that language L, generated by automaton G, is codiagnosable with respect to projections P_{o_1}, P_{o_2} and $\Sigma_f = \{\sigma_f\}$.

An observation is worth making in this example is the following. When

the system executes trace $s'_Y = \sigma_f bc$*, automaton* G^2_{scc} *reaches state* $x'_Y = (\{5Y\}, \{3N, 5Y\}, \{5Y\})$*. Notice that the Y-certain state* $\{5Y\}$ *in the first component of* x'_Y *indicates that the failure occurrence is detected by site* S_1*. On the other hand, when the system executes trace* $s''_Y = \sigma_f a$*, automaton* G^2_{scc} *reaches state* $x''_Y = (\{6N, 4Y\}, \{4Y\}, \{4Y\})$*, and, in this case, the Y-certain state* $\{4Y\}$ *in the second component of* x''_Y *indicates that the failure occurrence is detected by site* S_2*. Therefore, the computation of* G^2_{scc} *also allows the identification of the local site that generates the diagnostic information, for every failure trace in* $L(G)$*.*

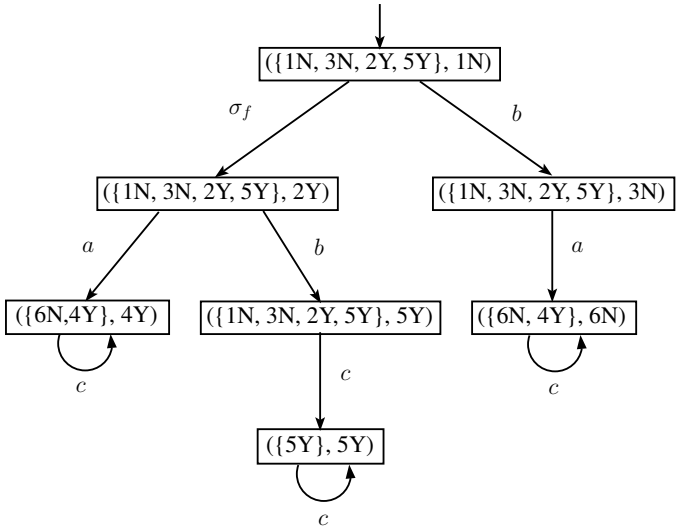


Figure 3.9: Diagnoser automata $G_{scc_1} = G_{d_1} || G_\ell$ of Example 3.4.

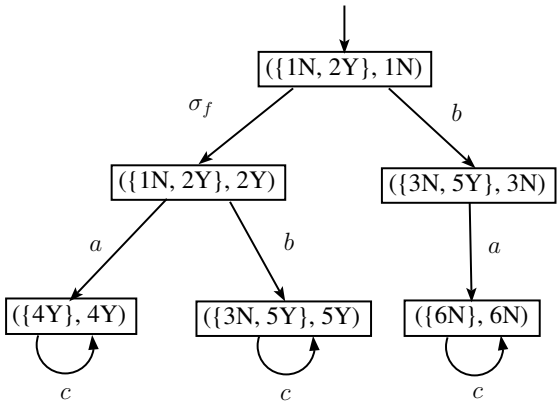


Figure 3.10: Diagnoser automata $G_{scc_2} = G_{d_2} || G_\ell$ of Example 3.4.

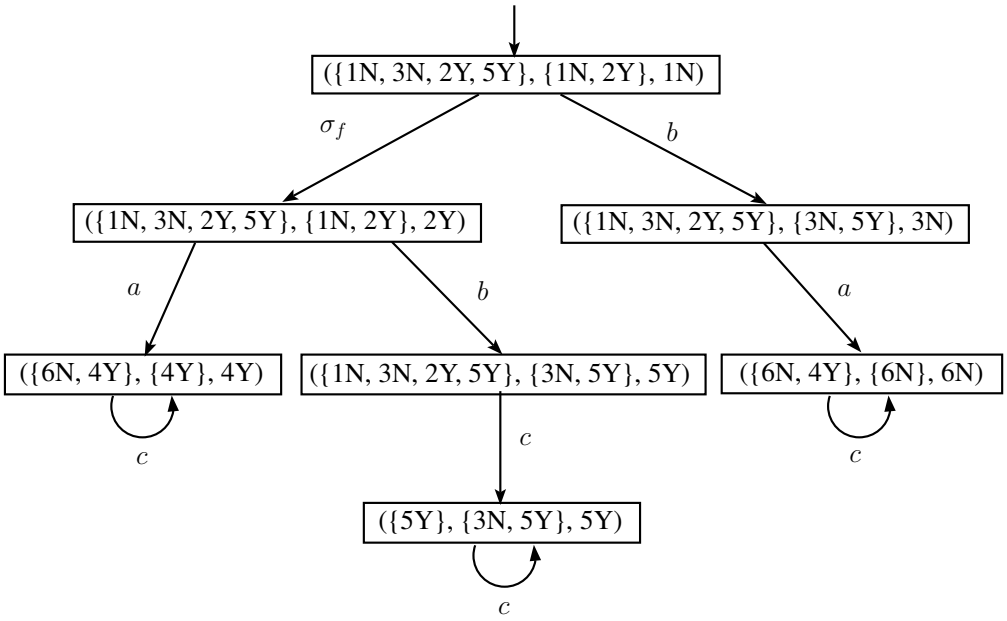


Figure 3.11: Diagnoser automaton $G^2_{scc} = G_{d_1}||G_{d_2}||G_\ell$ of Example 3.4.

3.2 Extending Verifiers to Show All Diagnosis Paths

Since verifier G_V, constructed in Algorithm 2.2, only provides the paths where there is ambiguity between faulty and normal traces, which is not enough for the problem we deal with here, we will now present a new verifier automaton G_{VT} that shows the diagnosis paths necessary to ensure codiagnosability. We start by defining the following projections:

$$P_{R_i} : (\Sigma \cup \Sigma_{R_1} \cup \Sigma_{R_2} \ldots \Sigma_{R_{N_s}})^* \to \Sigma^*_{R_i};$$
$$P_F : (\Sigma \cup \Sigma_{R_1} \cup \Sigma_{R_2} \ldots \Sigma_{R_{N_s}})^* \to \Sigma^*;$$
$$P_{o_i} : \Sigma^* \to \Sigma^*_{o_i},\ i \in I_{N_s},$$

where, as defined in Chapter 2, $\Sigma_{R_i} = \Sigma_{o_i} \cup \Sigma_i$, with $\Sigma_i = \{\sigma_{R_i} : \sigma \in \Sigma_{uo_i} \setminus \Sigma_f\}$. We can state the following result, which is a generalization of Lemma 1 of [56].

Lemma 3.1 *Let $G_V = (||_{i=1}^{N_s} G_{N,i})||G_F$. Then, for every $s_V \in L(G_V)$, there exist traces $s_N \in L(G_N)$ and $s_Y \in L(G_F)$, such that $P_{o_i}(s_Y) = P_{o_i}(s_N)$, $i = 1, 2, \ldots, N_s$, and conversely.*

Proof. ($\Rightarrow$) Let us consider a trace $s_V \in L(G_V)$, and let $s_{R_i} = P_{R_i}(s_V)$, $i \in I_{N_s}$ and $s_Y = P_F(s_V)$. Because $G_V = (||_{i=1}^{N_s} G_{N,i})||G_F$, then $L(G_V) = (\cap_{i=1}^{N_s} P_{R_i}^{-1}(L(G_{N,i}))) \cap P_F^{-1}(G_F)$, and thus, $s_{R_i} \in L(G_{N,i})$, $i \in I_{N_s}$, and $s_Y \in L(G_F)$.

Since $G_{N,i}$ is obtained from G_N by renaming the unobservable events of Σ_N, there always exists a trace s_N such that $s_{R_i} = R_i(s_N)$. Notice that $P_{o_i}(s_N) = P_F(s_{R_i})$ and $P_{o_i}(s_Y) = P_{R_i}(s_Y)$. Since $s_{R_i} = P_{R_i}(s_V)$ and $s_Y = P_F(s_V)$, we can conclude that $P_{o_i}(s_N) = P_F(P_{R_i}(s_V))$ and $P_{o_i}(s_F) = P_{R_i}(P_F(s_V))$. Finally, since $P_F(P_{R_i}(s_V)) = P_{R_i}(P_F(s_V))$, it is not difficult to check that $P_{o_i}(s_Y) = P_{o_i}(s_N)$, $i \in I_{N_s}$.

($\Leftarrow$) Since $G_{N,i}$ is obtained from G_N by renaming its unobservable events, then the renamed unobservable events of Σ_{R_i}, and the unobservable events of Σ, become private events of $G_{N,i}$ and G_F, respectively, in the parallel composition $G_V = (||_{i=1}^{N_s} G_{N,i})||G_F$. Let $s_N \in L(G_N)$ and $s_Y \in L(G_F)$ be such that $P_{o_i}(s_N) = P_{o_i}(s_F)$, $\forall i \in I_{N_s}$, and define $s_{R_i} = R_i(s_N)$. Then, by construction of the verifier automaton, there exists a trace $s_V \in L(G_V)$ associated with s_{R_i} and s_F. ■

We will now extend G_V to show all traces that ensure language codiagnosability. The idea is to augment G_V by creating a new state F (marked) and transitions labeled with events in Σ_V from the states of G_V to F, in such a way that for every trace $s_{VT} = s_V\sigma \in L_m(G_{VT})$, there exists $s_{YT} = P_F(s_{VT}) \in L(G_F)$, $\sigma_f \in s_{YT}$, such that $\exists i \in I_{N_s}$, $P_{o_i}(s_{YT}) \neq P_{o_i}(s_{NT})$, for all $s_{NT} \in L(G_N)$.

Algorithm 3.3 *Construction of automaton G_{VT}*

Input *Automaton G, N_s (number of sites), Σ_f, Σ_{o_i}, and Σ_{uo_i}, $i = 1, 2, \ldots, N_s$.*
Output *Extended verifier automaton G_{VT}.*

STEP 1. *Compute $G_{N,i}$, $i = 1, 2, \ldots, N_s$ and $G_V = (X_V, \Sigma_V, f_V, x_{0,V})$ according to Algorithm 2.2.*

STEP 2. ***For*** $i = 1, 2, \ldots, N_s$*:*

STEP 2.1 *Form* $X_i = \{x_{0_{N,i}}\} \cup \{x \in X_{N,i} : (\exists(\tilde{x}, \sigma) \in X_{N,i} \times \Sigma_{o_i})[f_{N,i}(\tilde{x}, \sigma) = x]\}$.

STEP 2.2 *Compute* $UR(X_i, \Sigma_{uo_i})$=$\{UR(x, \Sigma_{uo_i})$:$x \in X_i\}$.

STEP 2.3 ***If*** $\exists \tilde{U}, \hat{U} \in UR(X_i, \Sigma_{uo_i}) : \tilde{U} \subseteq \hat{U}$, ***then*** *set* $UR(X_i, \Sigma_{uo_i}) = UR(X_i, \Sigma_{uo_i}) \setminus \{\tilde{U}\}$.

STEP 3. *Compute* $G_{VT} = (X_{VT}, \Sigma_V, f_{VT}, x_{0,V}, X_{m,VT})$ *from* G_V *as follows:*

STEP 3.1 *Set* $X_{VT} = X_V \cup \{F\}$, $X_{m,VT} = \{F\}$, $\Gamma_{VT}(F) = \emptyset$.

STEP 3.2 *Set* f_{VT} *as follows:*

1. $f_{VT}(x, \sigma) = f_V(x, \sigma)$, $\forall x \in X_V$, $\sigma \in \Gamma(X_V)$.
2. ***For*** *each* $x_V = (x_{N,1}, x_{N,2}, \ldots, x_{N,N_s}, x_F) \in X_V$ *such that* $x_F = (x, Y), x \in X$*:*
   (a) *Define* $\Sigma'_o = (\Sigma_o \cap \Gamma_F(x_F)) \setminus \Gamma_V(x_V)$
   (b) ***For*** *each* $\sigma \in \Sigma'_o$*:*
   – ***For*** *each component* $x_{N,i}$, $i = 1, 2, \ldots, N_s$, *of* x_V, *compute*
   i. $U_s(x_{N,i}) := \{U \in UR(X_i, \Sigma_{uo_i})$*:* $x_{N,i} \in U\}$.
   ii. $U(x_{N,i}) = \bigcup_{U \in U_s(x_{N,i})} U$.
   iii. $\Gamma_{N,i}(U(x_{N,i})) :=$
   $\bigcup_{x_i \in \Gamma_{N,i}(U(x_{N,i}))} [\Gamma_{N,i}(x_i) \cap \Sigma_{o_i}]$.
   iv. ***If*** $\sigma \in \Sigma_{o_i} \setminus \Gamma_{N,i}(U(x_{N,i}))$, ***then*** *Set* $f_{VT}(x_V, \sigma) = \{F\}$.

In Algorithm 3.3, we create automaton G_{VT} as follows. We first compute automaton G_V in Step 1. In Step 2, we calculate the unobservable reaches of the initial state and of all states of $G_{N,i}$, $i = 1, 2, \ldots, N_s$, reached by observable events. In the last step, we add a new (marked) state F that receives transitions labeled by an event $\sigma \in \Sigma_V$ from a state $x_V = (x_{N,1}, x_{N,2}, \ldots, x_{N,N_s}, x_F) \in X_V$ providing the following conditions hold true:

C1. x_F has an Y-label, meaning that, the failure has occurred;

C2. $\sigma \in \Sigma_o$;

C3. $\sigma \in \Gamma_F(x_F)$, but $\sigma \notin \Gamma_V(x_V)$;

C4. $\exists i \in I_{N_s}$: $\sigma \in \Sigma_{o_i}$ and $\nexists x'_V = (x'_{N,1}, x'_{N,2}, \ldots, x'_{N,N_s}, x'_F) \in X'_V$ that has a component $x'_{N,i}$ such that $\{x_{N,i}, x'_{N,i}\} \subseteq X_i$, and $\sigma \in \Gamma_{N,i}(x'_{N,i})$, *i.e.*, if $x_{N,i}$ (resp. $x'_{N,i}$) is in the unobservable reach of $x'_{N,i}$ (resp. $x_{N,i}$) then σ is not in the active event set of $x'_{N,i}$.

The following theorem shows that the completion of G_V according to Algorithm 3.3, removes all trace ambiguities of G_V.

Theorem 3.3 *Let G_{VT} be constructed according to Algorithm 3.3. Then, every trace $s_{VT} \in L_m(G_{VT})$ is of the following form: $s_{VT} = s_V\sigma$, where $s_V \in L(G_V)$ and $\sigma \in \Sigma_o$. Moreover, for every trace $s_{VT} \in L_m(G_{VT})$, there exists $s_{YT} = P_F(s_{VT}) \in L(G_F)$ such that $\forall s_{NT} \in L(G_N)$, $P_{o_i}(s_{NT}) \neq P_{o_i}(s_{YT})$, for some $i \in I_{N_s}$ and conversely, for all traces $s_N, s_Y \in L(G)$ such that $\sigma_f \notin s_N$, $s_Y = st$, $s \in \Psi(\Sigma_f)$ and $|t| \geq 1$, $P_{o_i}(s_N) \neq P_{o_i}(s_Y)$ for some $i \in I_{N_s}$, there exists $s_{YT} \in \overline{s_Y}$ such that $\sigma_f \in s_{YT}$ and $s_{YT} = P_F(s_{VT})$ for some $s_{VT} \in L_m(G_{VT})$.*

Proof. *($\Rightarrow$) According to Lemma 3.1, for every trace $s_V \in L(G_V)$ there exist traces $s_{N_i} \in L(G_N)$ and $s_Y \in L(G_F)$ such that $P_{o_i}(s_{N_i}) = P_{o_i}(s_Y)$, $\forall i \in I_{N_s}$. We will now show that conditions* **C1**–**C3** *are necessary for the existence of an event σ that removes this ambiguity: (i) by Definition 2.9 and from the construction of the states of G_V, an state $x_V = (x_{N_1}, x_{N_2}, \ldots, x_{N_{N_s}}, x_F) \in X_V$, where $x_F = xY$ only appears in G_V if σ_f has occurred, and, thus, only those states of G_V whose last components have label Y can have transitions to state F; (ii) all events in Σ_{uo} and Σ_{R_i}, $i = 1, \ldots, N_s$, are particular events of G_F and $G_{N,i}$, respectively, being already in $\Gamma_V(x_V)$, which implies that, $\sigma \notin \Sigma_{uo} \cup \Sigma_{R_i}$ or equivalently, $\sigma \in \Sigma_o$; (iii) $\sigma \in \Gamma_F(x_F)$, since if $\sigma \notin \Gamma_F(x_F)$ then $s_{YT} = s_Y\sigma = P_F(s_V)\sigma \notin L(G_F)$, and, consequently $s_{YT} \notin L(G)$. However, from Lemma 3.1, such a σ must not be in the active event of G_V, i.e., $\sigma \notin \Gamma_V(x_V)$. Therefore, $\sigma \in \Sigma'_o$, where $\Sigma'_o = (\Sigma_o \cap \Gamma_F(x_F)) \setminus \Gamma_V(x_V)$. Let us consider an event $\sigma \in \Sigma'_o$ and assume that $\sigma \in \Sigma_{o_k}$ for some $k \in I_{N_s}$. Moreover, let $x_V = (x_{N,1}, \ldots, x_{N,k}, \ldots, x_{N,N_s}, x_F)$ and $f_V(x_{0_V}, s_V) = x_V$ such that $s_V \in L(G_V)$. Define $s_{R_k} = P_{R_k}(s_V) \in L(G_{N,k})$ where $f_{N,k}(x_{0_N}, s_{R_k}) = x_{N,k}$. Let an event $\sigma' \in \Sigma_{o_k}$. If $\exists x'_{N,k} \in U(x_{N,k})$: $\sigma' \in \Gamma_{N,k}(x'_{N,k})$ then $\sigma' \in \Gamma(U(x_{N,k}))$. By construction, for all $x'_{N,k} \in U(x_{N,k})$, $\exists s'_{R_k} \in \Sigma_k^*$ such that either $f_{N,k}(x_{N,k}, s'_{R_k}) = x'_{N,k}$ or $f_{N,k}(x'_{N,k}, s'_{R_k}) = x_{N,k}$. Without loss of generality let us consider the first possibility only. Consequently, $\exists s'_V = P_{R_k}(s'_{R_k})$ such that $f_V(x_V, s'_V) = x'_V = (x'_{N,1}, \ldots, x'_{N,k}, \ldots, x'_{N,N_s}, x'_F)$. Indeed, since $s'_{R_k} \in \Sigma_k^*$, there exist $s_{N_k} = R^{-1}(s_{R_k}) \in L(G_N)$ and $s'_{N_k} = R^{-1}(s'_{R_k}) \in L(G_N)/s_{N_k}$ such that $s'_{N_k} \in \Sigma_{uo_k}^*$, then $P_{o_k}(s'_{N_k}) = \varepsilon$. If we define $s_{NT} = s_{N_k}s'_{N_k}\sigma'$, then: $P_{o_k}(s_{NT}) = P_{o_k}(s_{N_k})P_{o_k}(s'_{N_k})P_{o_k}(\sigma') = P_{o_k}(s_{N_k})\sigma'$.*

Define now $s_{YT} = s_Y\sigma$ such that $s_Y = P_F(s_V)$. Thus, $P_{o_k}(s_{YT}) = P_{o_k}(s_Y)\sigma$, and since $s_Y = P_F(s_V)$ and $s_{N_k} = R^{-1}(P_{R_k}(s_V))$, by Lemma 3.1, we can conclude that

$P_{o_k}(s_Y) = P_{o_k}(s_{N_k})$. *Therefore, if* $\sigma' = \sigma$ *then* $P_{o_k}(s_{NT}) = P_{o_k}(s_{YT})$. *But if* $\sigma' \neq \sigma$ *then* $P_{o_k}(s_{NT}) \neq P_{o_k}(s_{YT})$, *for some* $k \in I_{N_s}$.

(⇐) Suppose that there exists a trace $s_Y = st \in L(G)$, $s \in \Psi(\Sigma_f)$ *and* $|t| \geq 1$ *such that, for some* $k \in I_{N_s}$, $P_{o_k}(s_N) \neq P_{o_k}(s_Y)$ *for all traces* $s_N \in L(G)$ *for which* $\sigma_f \notin s_N$. *Thus, there exist* $s'_Y \in L(G)$, *and* $\sigma \in \Sigma_o$, *such that* $\Sigma_f \in s'_Y$, $s'_Y \in \overline{s_Y}$ *and* $\Sigma_f \notin s'_N$, $s'_N \in \overline{s_N}$ *such that* $P_{o_k}(s'_Y) = P_{o_k}(s'_N)$, $\forall k \in I_{N_s}$, *and, for some* $k \in I_{N_s}$, $P_{o_k}(s'_Y\sigma) \neq P_{o_k}(s'_N s''_N)$ *and for all traces* $s''_N \in L(G)/s'_N$ *such that* $\sigma_f \notin s''_N$.

Thus, $\exists s'_V \in L(G_V)$ *such that* $P_F(s'_V) = s'_Y$ *and* $P_{o_k}(s'_V) = s'_N$, *for some* $k \in I_{N_s}$. *In addition,* $s_{VT} = s'_V\sigma \notin L(G_V)$ *and satisfies conditions* **C1**–**C4**, *which completes the proof.* ■

Traces $s_{YT} \in P_F(L_m(G_{VT}))$ have an important property, as follows.

Definition 3.1 *(Minimum length codiagnosable traces) A trace* $s_Y \in L(G_F)$, *where* $\Sigma_f \in s_Y$, *is a minimum length codiagnosable trace with respect to projections* $P_{o_i} : \Sigma^* \to \Sigma^*_{o_i}$, $i = 1, 2, \ldots, N_s$, *and* Σ_f, *if: (i)* $\exists i \in I_{N_s} : \forall s_N \in L(G_N)$, $P_{o_i}(s_Y) \neq P_{o_i}(s_N)$, *and; (ii)* $\forall s'_Y \in \overline{s_Y} \setminus \{s_Y\}$, $\Sigma_f \in s'_Y$, $\exists s'_{N,i} \in L(G_N) : \forall i \in I_{N_s}, P_{o_i}(s'_Y) = P_{o_i}(s'_{N,i})$.

Let $\Theta(L(G_F))$ denote the set of all minimum traces of G_F, *i.e.*, $\Theta(L(G_F)) = \{s_Y \in L(G_F) : s_Y \text{ is a minimum length codiagnosable trace}\}$. Comparing Definition 3.1, Lemma 3.1 and Theorem 3.3 , we can state the following fact.

Fact 3.5 $\Theta(L(G_F)) = P_F\left(L_m(G_{VT})\right)$.

The following example illustrates the results of this section.

Example 3.5 *Let us consider again automaton* G *of Example 3.4, shown again in Figure 3.12, where* $\Sigma = \{a, b, c, \sigma_f\}$ *and* $\Sigma_f = \{\sigma_f\}$. *Let* $\Sigma_{o_1} = \{a, c\}$ *and* $\Sigma_{o_2} = \{a, b\}$, *and consider projections* $P_{o_1} : \Sigma^* \to \Sigma^*_{o_1}$ *and* $P_{o_2} : \Sigma^* \to \Sigma^*_{o_2}$.

We will first illustrate the construction of automaton G_{VT} *using Algorithm 3.3. The first step is to compute* $G_{N,1}$ *and* $G_{N,2}$, *shown in Figures 3.13(a) and 3.13(b), respectively. Next, we obtain the failure automaton* G_F, *depicted in Figure 3.13(c), and compute* $G_V = G_{N,1}||G_{N,2}||G_F$, *according to Algorithm 2.2, which is shown in Figure 3.14(a). According to Theorem 2.4,* $L(G)$ *is codiagnosable with respect*

to P_{o_1}, P_{o_2} and Σ_f. The next step is to construct state sets X_i, $i = 1, 2$, formed with the initial state of $G_{N,i}$ and the states of $G_{N,i}$ reached through some observable event. Thus, according to Step 2 of Algorithm 3.3, $X_1 = \{\{1N\}, \{6N\}\}$ and $X_2 = \{\{1N\}, \{3N\}, \{6N\}\}$. Using these two sets, it is possible to compute $UR(X_i, \Sigma_{uo_i})$, for $i = 1, 2$, which are given by $UR(X_1, \Sigma_{uo_1}) = \{\{1N, 3N\}, \{6N\}\}$ and $UR(X_2, \Sigma_{uo_2}) = \{\{1N\}, \{3N\}, \{6N\}\}$. The final step is to complete G_V with transitions to state F according to Step 3. In order to do so, we will consider every state of G_V. Notice that, since in states $(1N, 1N, 1N)$ and $(3N, 1N, 1N)$, component x_F does not have an Y-label, no transitions from these states to F can be introduced. Consider now state $(1N, 1N, 2Y)$. Since $\Sigma_o = \{a, b, c\}$, $\Gamma_F(2Y) = \{a, b\}$ and $\Gamma_V((1N, 1N, 2Y)) = \{b, b_{R_1}\}$, we can see that $\Sigma'_o = (\Sigma_o \cap \Gamma_F(2Y)) \setminus \Gamma_V((1N, 1N, 2Y)) = \{a\}$. Notice, for state $(1N, 1N, 2Y)$, that $x_{N,1} = \{1N\}$ and $x_{N,2} = 1N$, and, thus, according to Step 3.2.2).b), $U(x_{N,1}) = \{1N, 3N\}$ and $\Gamma_{N,1}(U(1N)) = \{a\}$, and since $a \notin \Sigma_{o_1} \setminus \Gamma_{N,1}(U(1N)) = \{c\}$, no transition labeled by event a from state $(1N, 1N, 2Y)$ to F must be initially introduced using the set of observable events Σ_{o_1} of Site 1. On the other hand, $U(x_{N,2}) = U(1N) = \{1N\}$ and $\Gamma_{N,2}(U(1N)) = \{b\}$, and therefore $a \in \Sigma_{o_2} \setminus \Gamma_{N,2}(U(1N)) = \{a\}$. Thus, according to the last step of Algorithm 3.3, $f_{VT}((1N, 1N, 2Y), a) = \{F\}$. Similar analysis can be carried out for state $(3N, 1N, 2Y)$, $(1N, 3N, 5Y)$, and $(3N, 3N, 5Y)$, leading to $f_{VT}((3N, 1N, 2Y), a) = f_{VT}((1N, 3N, 5Y), c) = f_{VT}((3N, 3N, 5Y), c) = \{F\}$.

To conclude the example, notice that after the occurrence of trace $\sigma_f bc$ (resp. $\sigma_f a$), Site 1 (resp. 2) detects the occurrence of σ_f. Therefore, $\Theta(L(G_F)) = \{\sigma_f a, \sigma_f bc\}$. Comparing with Figure 3.14(b), it is not difficult to conclude that $P_F(L_m(G_{VT})) = \Theta(L(G_F))$.

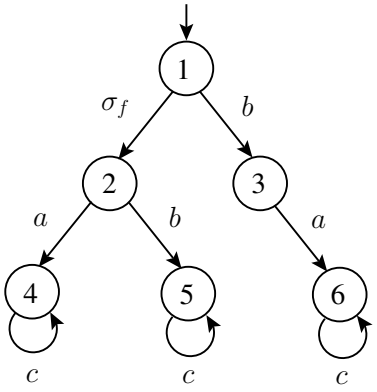


Figure 3.12: Plant G of Example 3.4, which is considered again in Example 3.5.

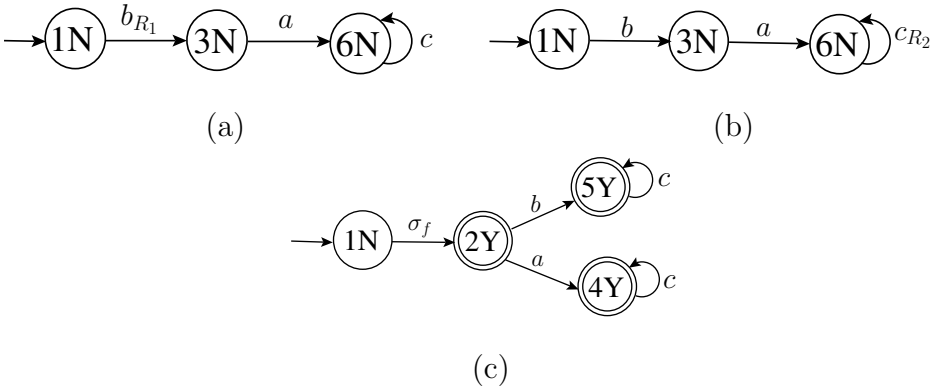


Figure 3.13: Automata $G_{N,1}$ (a); $G_{N,2}$ (b) and G_F (c); of Example 3.5.

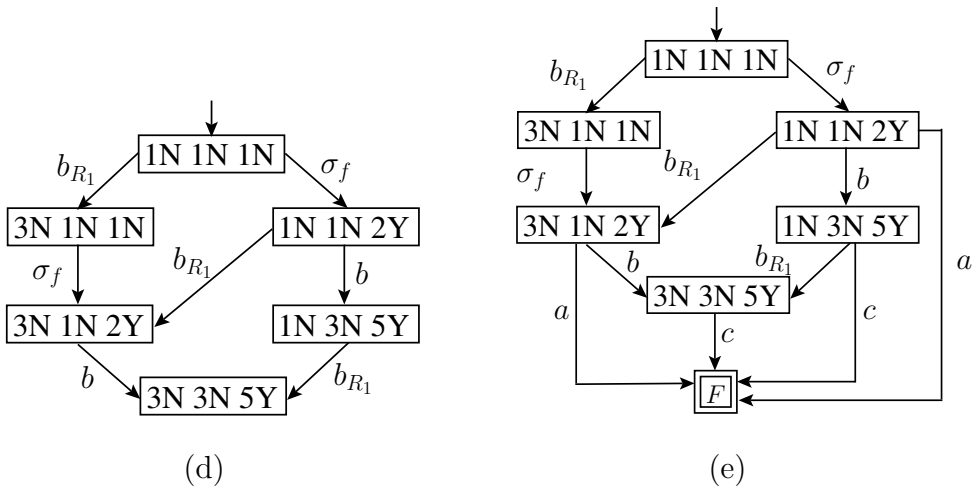


Figure 3.14: Automata G_V (a); and G_{VT} (b) of Example 3.5.

3.3 Application of the Proposed Approach to τ- and K-Codiagnosability Analysis

In a decentralized architecture, codiagnosability ensures that at least one local site detects and diagnoses the failure occurrence within a bounded number of event occurrences. However, just being sure that the failure has occurred may be not enough; for example, components may burn or parts may misalign before the failure occurrence is detected. So, it is important to also incorporate either the time elapsed or the number of event occurrences since the failure occurrence as a performance index for the decentralized diagnosis system; possibly based on this parameter, we may, for instance, establish further safety measures. This suggests that, besides codiagnosability analysis, two performance indices should also be considered, as

follows.

- τ-codiagnosability, which ensures that the failure occurrence is detected in at most τ time units after the occurrence of the failure event;
- K-codiagnosability, which ensures that the failure occurrence is detected in at most K events after the failure occurrence.

With those objectives in mind, we will propose in this section a new methodology to compute performance indices τ and K based on automata $G_{scc}^{N_s}$ and G_{VT}, proposed in this thesis. In order to approach τ-codiagnosability we need to add to the untimed model used in the previous section, some information regarding the time that the system takes to complete the transitions. Such an information can be provided by using weighted automaton.

3.3.1 Brief Review on Weighted Automaton

A finite-weighted automaton [49] is a two-tuple (G, w), where $G = (X, \Sigma, f, \Gamma, X_0, X_m)$[2] and $w : X \times \Sigma \to \mathbb{R}^+$ is the weight function that assigns a nonnegative weight to each transition of G and is defined over a pair $(x, \sigma) \in X \times \Sigma$ if, and only if, transition $f(x, \sigma)$ is defined. The weights denote the duration required for the corresponding transition to be completed. We will denote weighted automata as $\mathcal{G} = (G, w)$.

The parallel composition of weighted automata $\mathcal{G}_1 = (G_1, w_1)$ and $\mathcal{G}_2 = (G_2, w_2)$ is denoted as $\mathcal{G} = (G, w) = \mathcal{G}_1||\mathcal{G}_2 = (G_1, w_1)||(G_2, w_2)$, where $G = G_1||G_2$, and $w : X_1 \times X_2 \times (\Sigma_1 \cup \Sigma_2) \to \mathbb{R}^+$ is defined as follows.

$$w((x_1, x_2), \sigma) = \begin{cases} w_1(x_1, \sigma), & \text{if } \sigma \in \Gamma_1(x_1) \setminus \Sigma_2 \\ w_2(x_2, \sigma), & \text{if } \sigma \in \Gamma_2(x_2) \setminus \Sigma_1 \\ \max\{w_1(x_1, \sigma), w_2(x_2, \sigma)\}, & \text{if } \sigma \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ \text{undefined}, & \text{otherwise.} \end{cases} \tag{3.4}$$

Since $\mathcal{G} = (G, w)$ only adds weights to G, the languages generated by G and $\mathcal{G}$ are the same; thus, $L(\mathcal{G})$ and $L(G)$ will be used indistinctly throughout the

[2]Notice that from this point onwards, automaton G will be assumed to be nondeterministic. The reason for that will become clear later on the text.

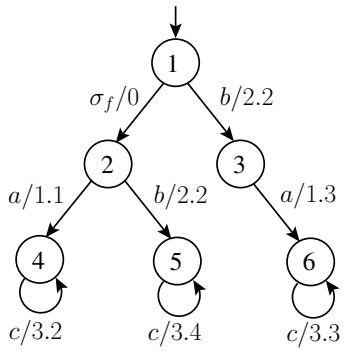


Figure 3.15: Weighted automaton $\mathcal{G}$.

text. With some abuse of notation, we denote $Trim(\mathcal{G}) = (Trim(G), Trim(w))$, where $Trim(w)$ consists of removing all weight functions of w associated with those transitions of G removed by $Trim(G)$.

Figure 3.15 depicts weighted automaton $\mathcal{G} = (G, w)$, where $\mathcal{G}$ was obtained from automaton G of Figure 3.8(a) by adding the following weight functions: $w(1, \sigma_f) = 0$, $w(1, b) = w(2, b) = 2.2$, $w(2, a) = 1.1$, $w(3, a) = 1.3$, $w(4, c) = 3.2$, $w(5, c) = 3.4$ and $w(6, c) = 3.3$. Notice that label "a/1.1" over transition $f(2, a) = 4$ means that $w(2, a) = 1.1$; the remaining labels are interpreted similarly.

The introduction of weighted automata allows the formulation of an important problem, namely the computation of the maximum weight of all paths between states in weighted automata. When automaton G is acyclic, this problem can be solved by following the steps of Algorithm 3.4.

Algorithm 3.4 *Computation of the maximum weight of all paths between states x_p and x_q in an acyclic weighted automaton*

Input: *Acyclic weighted automaton $\mathcal{G} = (G, w)$, source state x_p, and sink state x_q.*
Output: *τ, the maximum weight between states x_p and x_q.*

STEP 1. *Create a topological ordering of all states in $\mathcal{G}$, i.e., $TO(X) = (y_1, y_2, \ldots, y_n)$ = Topological Sort(G) [60], where $y_i \in X$, for $i \in I_n$, such that $I_n = \{1, 2, \ldots, n\}$ and $n = |X|$.*

STEP 2. *Map $x_p \to y_k$ and $x_q \to y_\ell$.*

STEP 3. *Form a list $L = (y_k, y_{k+1}, \ldots, y_{\ell-1}) \subseteq TO(X)$.*

STEP 4.

For $i = 0, 1, \ldots, \ell - k$

If $i = 0$ ***then***

Set $d[y_{k+j}] = 0$, $j = 0, 1, \ldots, \ell - k$

Otherwise

For $j = i, i+1, \ldots, \ell - k$

If $f(y_{k+i-1}, y_{k+j})!$

Set $d = d[y_{k+i-1}] + w(y_{k+i-1}, y_{k+j})$

If $d[y_{k+j}] < d$, ***then*** *set* $d[y_{k+j}] = d$.

STEP 5. $\tau = d[y_\ell]$.

Algorithm 3.4 works as follows. In the first step, a topological sort is carried out which returns the linked list of the states of G, such that if G has a transition from state u to v, then u appears before v in the ordering. As seen in [60], topological sort is linear in the number of transitions of an acyclic automaton and thus, $O(n^2)$ in number of states of the automaton. In Step 4, we travel over all transitions from x_p to x_q following the ordering of list L and store the weights in a variable d for every state in L. Thus, when state x_q (that is mapped to y_ℓ) is reached, we obtain $\tau = d[y_\ell]$.

Example 3.6 *Consider the acyclic weighted automaton* $\mathcal{G}_1$ *of Figure 3.16. We will illustrate the computation of the maximum weight of all paths between source state* x_0 *(initial state of* $\mathcal{G}_1$*) and sink state* x_3 *(marked state of* $\mathcal{G}_1$*) according to Algorithm 3.4. First, according to Step 1, we create a topological ordering of* $\mathcal{G}_1$*, as shown in Figure 3.17, and map state* x_0 *(resp.* x_3*) into* y_1 *(resp.* y_4*) according to Step 2. Table I shows the execution of Step* 4 *of Algorithm 3.4, and from it, it is clear that the maximum weight is* $\tau = 7$.

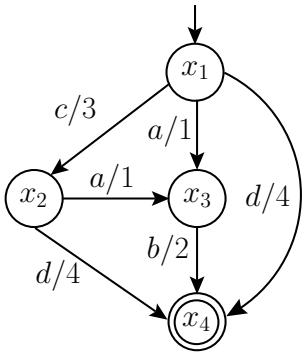


Figure 3.16: Weighted automaton $\mathcal{G}_1$.

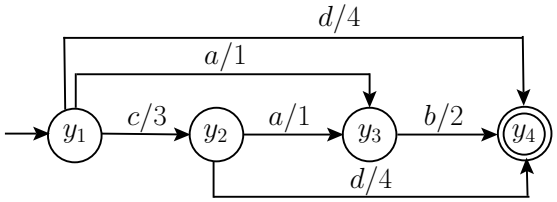


Figure 3.17: Weighted automaton $\mathcal{G}_1$ after topological ordering.

Table 3.1: Step 4 of Algorithm 3.4 to Example 3.6

	j = 0	j = 1	j = 2	j = 3
Iteration	$d[y_1]$	$d[y_2]$	$d[y_3]$	$d[y_4]$
i = 0	0	0	0	0
i = 1	0	3	1	4
i = 2	0	3	4	7
i = 3	0	3	4	7

Weighted automata that have all cycles of states connected with zero-weight transitions are called zero-weight cycle automata. In this case, it is possible to exploit the weighting structure of zero-weight cycle automata to find the strongly connected components in order to obtain an equivalent acyclic component automaton by shrinking each of the strongly connected components to a single state (and consequently, it will be possible to apply Algorithm 3.4). This can be done according to Algorithm 3.5.

Algorithm 3.5 *Transformation of zero-weight cycle automata into acyclic weighted automata*

Input: *Zero-weight cycle automaton* $\mathcal{G} = (G = (X, \Sigma, f, \Gamma, X_0, X_m), w)$.

Output: *Acyclic weighted automaton* $\mathcal{G}' = (((G' = (X', \Sigma, f', \Gamma', X_0', X_m), w))$.

STEP 1. *Find all strongly connected components* $X_{scc_1}, X_{scc_2}, \ldots, X_{scc_{n_c}}$ *in* $\mathcal{G}$ *and shrink each strongly connected component* $X_{scc_i} = \{x_{i_1}, x_{i_2}, \ldots, x_{i_{m_i}}\}$, *in single states called "super states"* x_{sup_i}, $i \in I_{n_c}$, *where* $I_{n_c} = \{1, \ldots, n_c\}$, *by removing all transitions between the states of each strongly connected component.*

STEP 2. *Form* $X_{scc} = X_{scc_1} \cup X_{scc_2} \cup \ldots \cup X_{scc_{n_c}}$ *and* $X_{sup} = \{x_{sup_1}, x_{sup_2}, \ldots, x_{sup_{n_c}}\}$.

STEP 3. *Set* $X' = (X \setminus X_{scc}) \cup X_{sup}$.

STEP 4. *Set*

$$f'(x,\sigma) = \begin{cases} f(x,\sigma), & \text{if } x \in X \setminus X_{scc} \text{ and } (\nexists y \in X_{scc})[f(x,\sigma) = y] \\ x_{sup_i}, & \text{if } x \in X \setminus X_{scc} \text{ and } ((\exists i \in I_{n_c}) \wedge \\ & (\exists y \in X_{scc_i}))[f(x,\sigma) = y] \\ f(\tilde{x},\sigma), & \text{if } (x = x_{sup_i}, \text{for some } i \in I_{n_c}) \text{and} \\ & (\exists \tilde{x} \in X_{scc_i},)[f(\tilde{x},\sigma) \in X \setminus X_{scc}] \\ x_{sup_j}, & \text{if } (x = x_{sup_i}, \text{for some } i \in I_{n_c}) \text{and} \\ & (\exists (j,\tilde{x}) \in I_{n_c} \setminus \{i\} \times X_{scc_i})[f(\tilde{x},\sigma) \in X_{scc_i}] \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

STEP 5. *Set* $\Gamma'(x) = \{x \in X' : (\exists \sigma \in \Sigma)[f(x,\sigma)!]\}$, $\forall x \in X'$.

STEP 6. *Set* $X_0' = X_0$.

For each $i \in I_{n_c}$

Set $X_{aux} = X_0 \cap X_{scc_i}$;

If $X_{aux} \neq \emptyset$, *set* $X_0' = (X_0 \setminus X_{aux}) \cup \{x_{sup_i}\}$.

Since, in this section, the marked states will represent the states in which the failure occurrence is detected, they will not have output transitions. Thus, they cannot belong to any strongly connected component. Therefore, X_m is the set of marked states of both the input and output automata of Algorithm 3.5.

Example 3.7 *We will illustrate here how we obtain an weighted automaton $\mathcal{G}_3$ that has no zero-weight cycles equivalent to a zero-weight cycle automaton. To this end, consider weighted automaton $\mathcal{G}_2$ illustrated in Figure 3.18(a). According to Algorithm 3.5, after the first step, we find all strongly connected components of $\mathcal{G}_2$, which are given by $X_{scc_1} = \{x_0, x_2, x_3\}$ and $X_{scc_2} = \{x_4, x_5\}$, and shrink into single states x_{sup_1} and x_{sup_2}, respectively. In accordance with Step 2, we form sets $X_{scc} = \{x_0, x_2, x_3, x_4, x_5\}$ and $X_{sup} = \{x_{sup_1}, x_{sup_2}\}$. Finally, Steps 3 to 7 of Algorithm 3.5 are to effectively build the corresponding acyclic time-weighted automaton $\mathcal{G}_3$, which is illustrated in Figure 3.18(b).*

It is important to remark that since $\mathcal{G}_3$ is an acyclic time-weighted automaton, we can apply Algorithm 3.4 to compute the maximum weight between any two states of $\mathcal{G}_3$.

Remark 3.3 *Note, as illustrated in Example 3.7, that the resulting output of Algorithm 3.5 can be a nondeterministic automaton that has multiple transitions leaving a state with the same event label. However, since we are only interested in the computation of the maximum weight between states, it is not necessary to compute the observer automaton if we apply Algorithm 3.4 to the new automaton. In addition, in the context of this application, such transitions may only appears as a result of the suppression of zero-weight cycles.*

3.3.2 τ-Codiagnosability Analysis

We will now address the problem of finding the maximum time a diagnosis system takes to diagnose a failure occurrence. To this end, we introduce the definition of τ-codiagnosability, as follows.

Definition 3.2 *(τ-codiagnosability) A language $L(G)$ is τ-codiagnosable with respect to P_{o_i}, $i = 1, 2, \ldots, N_s$ and $\Sigma_f = \{\sigma_f\}$ if it is possible to detect the occurrence of σ_f in at most τ time units after the occurrence of the failure event.*

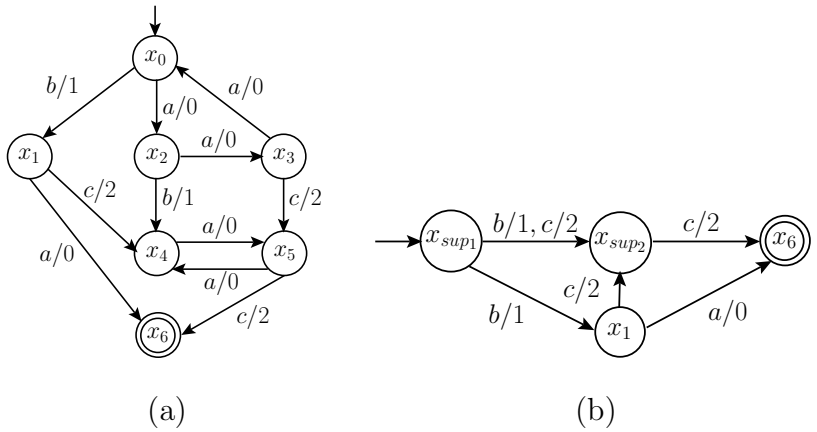


Figure 3.18: Zero-weight cycle automaton $\mathcal{G}_2$ (a) and acyclic time-weighted automaton $\mathcal{G}_3$ (b).

We make the following assumption.

AT1. Language L is codiagnosable with respect to projections $P_{o_i} : \Sigma^* \to \Sigma_{o_i}^*$, $i = 1, 2, \ldots, N_s$, and $\Sigma_f = \{\sigma_f\}$.

Assumption **AT1** is necessary since the problem addressed here would not make sense if L were not codiagnosable. Let $\Theta(L(G))$ denote the set of all minimum length codiagnosable traces of $L(G)$, according to Definition 3.1.

We may state the following result.

Fact 3.6 *Let $st \in \Theta(L(G))$, and assume that $s \in \Psi(\Sigma_f)$. Write s and t as $s = \tilde{s}\sigma_f$ and $t = \sigma_1\sigma_2 \ldots \sigma_n$ and assume that $x_f = f(x_0, \tilde{s})$, for some $x_0 \in X_0$, $x_1 = f(x_f, \sigma_f)$ and $x_{j+1} = f(x_j, \sigma_j)$, $j = 1, \ldots, n$. Define*

$$\tau(st) = \sum_{j=1}^{n} w(x_j, \sigma_j). \tag{3.5}$$

Then, $L(G)$ is τ-codiagnosable for

$$\tau := \max_{st \in \Theta(L(G)): s \in \Psi(\Sigma_f)} \tau(st). \tag{3.6}$$

We will now present two methods for computing τ: the first one using the diagnoser-like automaton, proposed in Section 3.1 and the second one using the extended verifier automaton proposed in Section 3.2.

(1) Computation of τ using diagnoser

In order to compute τ using the diagnoser-like weighted automaton $\mathcal{G}_{scc}^{N_s}$, we must compute weighted automata $\mathcal{G}_{d_i} = (G_{d_i}, w_{d_i}), i = 1, \ldots, N_s$. In order to do so, let

$\mathcal{G} = (G, w)$ denote the weighted automaton that models the plant. The observer of $\mathcal{G} = (G, w)$ with respect to a set Σ_o of observable events is defined as follows:

$$Obs(\mathcal{G}, \Sigma_o) = Obs(Obs(G, \Sigma_o), w_o), \tag{3.7a}$$

where

$$w_o(x_{obs}, \sigma) \quad = \quad \min_{x \in x_{obs}} w(x, \sigma). \tag{3.7b}$$

The choice for the minimum weight has been made by technical reasons and will become clear in Example 3.8.

Let $\mathcal{A}_\ell$ denote the weighted label automaton, which is obtained from A_ℓ by adding the weight functions $w(N, \sigma_f)$ and $w(Y, \sigma_f)$ to the corresponding transitions. Thus, we can define the following weighted automata:

$$\mathcal{G}_\ell = G_\ell || A_\ell \tag{3.8}$$

and

$$\mathcal{G}_{d_i} = Obs(\mathcal{G}_\ell, \Sigma_{o_i}), i = 1, \ldots, N_s \tag{3.9}$$

Thus, we may define weighted automaton $\mathcal{G}_{scc}^{N_s}$, as follows.

$$\mathcal{G}_{scc}^{N_s} = (||_{i=1}^{N_s} \mathcal{G}_{d_i}) || \mathcal{G}_\ell \tag{3.10}$$

The following example illustrates the computation of $\mathcal{G}_{scc}^{N_s}$ for $N_s = 2$.

Example 3.8 *Let us consider weighted automaton $\mathcal{G}$ of Figure 3.15, and, assume that $P_{o_1} : \Sigma^* \to \Sigma_{o_1}^*$ and $P_{o_2} : \Sigma^* \to \Sigma_{o_2}^*$, where $\Sigma_{o_1} = \{a, c\}$ and $\Sigma_{o_2} = \{a, b\}$. Firstly, to construct diagnoser $\mathcal{G}_{scc}^2$, we must compute, according to Equation* (3.8)*, weighted automaton $\mathcal{G}_\ell = (G||A_\ell, w_\ell)$, which is shown in Figure 3.19(a). Weighted automata $\mathcal{G}_{d_1} = Obs(\mathcal{G}_\ell, \Sigma_{o_1}^*)$ and $\mathcal{G}_{d_2} = Obs(\mathcal{G}_\ell, \Sigma_{o_2}^*)$ are obtained according to Equation* (3.8) *whose corresponding transition diagrams are depicted in Figures 3.19(b) and 3.19(c), respectively. After obtaining all weighted automata necessary to compute $\mathcal{G}_{scc}^2$, we can obtain weighted automaton $\mathcal{G}_{scc}^2 = \mathcal{G}_{d_1} \| \mathcal{G}_{d_2} \| \mathcal{G}_\ell$, according to Equation* (3.10)*, which is shown in Figure 3.19(d). Notice that not only the languages of $\mathcal{G}_{scc}^2$ and $\mathcal{G}_\ell$ are equal but also their respective transition weights.*

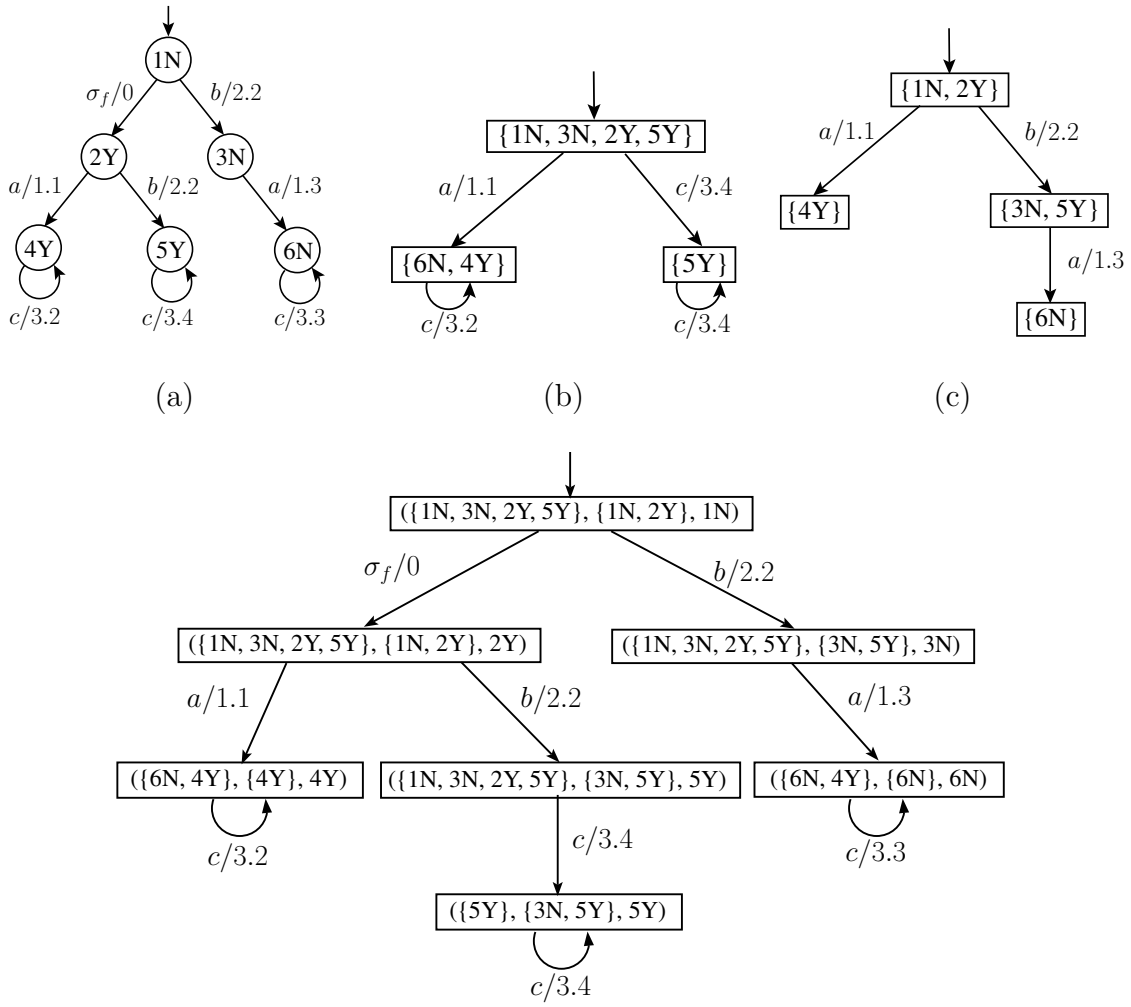


Figure 3.19: Weighted automata: $\mathcal{G}_{d_1}$ (a); $\mathcal{G}_{d_2}$ (b); $\mathcal{G}_\ell$ (c) and $\mathcal{G}^2_{scc}$ (d).

We will now clarify the point behind the definition of the weight function of observers, i.e., $w_o(x_{obs}, \sigma) = \min_{x \in x_{obs}} w(x, \sigma)$ *in Equation* (3.7b)*. To this end, consider states* $4Y$ *and* $6N$ *of* $\mathcal{G}_\ell$ *shown in Figure 3.19(c). Notice that* $w_\ell(4Y, c) = 3.2$ *and* $w_\ell(6N, c) = 3.3$ *being therefore different, but associated with the same event* c*. When we compute diagnoser* $\mathcal{G}_{d_1}$*, they form a unique state* $(\{6N, 4Y\})$*, and, according to Equation* (3.7b)*,* $w_{d_1}(\{6N, 4Y\}, c) = min(w_\ell(4Y, c), w_\ell(6N, c)) = 3.2$*. In addition, since event* c *is unobservable for site* S_2*, states* $\{6N\}$ *and* $\{4Y\}$ *have no transitions in* $\mathcal{G}_{d_2}$*, and thus,* $w_{d_2}(\{6N\}, c)$ *and* $w_{d_2}(\{4Y\}, c)$ *are not defined. When we compute* $\mathcal{G}^2_{scc} = \mathcal{G}_{d_1} \| \mathcal{G}_{d_2} \| \mathcal{G}_\ell$ *through a parallel composition, weight function of* $\mathcal{G}^2_{scc}$ *is defined according to Equation* (3.4)*, and thus,* $w^2_{scc}((\{6N, 4Y\}, \{4Y\}, \{4Y\}), c) = max(w_{d_1}(\{6N, 4Y\}, c), w_\ell(4Y, c)) = 3.2$ *and* $w^2_{scc}((\{6N, 4Y\}, \{6N\}, \{6N\}), c) =$

max $(w_{d_1}(\{6N, 4Y\}, c)\ , w_\ell(6N, c)) = 3.3$*, which are equal to the values obtained by* $w_\ell(4Y, c)$ *and* $w_\ell(6N, c)$*, respectively. This would not occur if* $w_o(x_{obs}, \sigma)$ *had been defined as* $\max_{x \in x_{obs}} w(x, \sigma)$*. In this case,* $w^2_{scc}((\{6N, 4Y\}, \{4Y\}, \{4Y\}), c) = w^2_{scc}((\{6N, 4Y\}, \{6N\}, \{6N\}), c) = 3.3$*, which is not correct.*

We will now present a methodology for computing τ based on weighted automaton $G^{N_s}_{scc}$. We start by calculating a new weighted automaton $\mathcal{G}_{d_f}$, according to the following algorithm.

Algorithm 3.6 *Computation of nondeterministic weighted automaton* $\mathcal{G}_{d_f}$

Input *Weighted automaton* $\mathcal{G}$*,* $\mathcal{A}_\ell$*,* Σ_f *and* Σ_{o_i} $i = 1, 2, \ldots, N_s$*.*

Output *Nondeterministic weighted automaton* $\mathcal{G}_{d_f}$*.*

STEP 1. *Compute* $\mathcal{G}_\ell = \mathcal{G} \| \mathcal{A}_\ell$ *and* $\mathcal{G}_{d_i} = Obs(\mathcal{G}_\ell, \Sigma_{o_i})$ *with respect to* P_{o_i}*,* $i = 1, 2, \ldots, N_s$*, according to Equations* (3.8) *and* (3.9)*, respectively.*

STEP 2. *Compute* $\mathcal{G}^{N_s}_{scc} = (||^{N_s}_{i=1} \mathcal{G}_{d_i}) || \mathcal{G}_\ell$*, according to Equation* (3.10)*.*

STEP 3. *Compute* $\mathcal{G}^{N_s,m}_{scc}$ *from* $\mathcal{G}^{N_s}_{scc}$ *by marking all states* $(x_{d_1}, x_{d_2}, \ldots, x_{d_{N_s}}, x_\ell)$ *from* $\mathcal{G}^{N_s}_{scc}$ *such that* $\exists x_{d_i}$*, for some* $i = 1, 2, \ldots, N_s$*,* x_{d_i} *Y-certain, and* x_ℓ *an Y-labeled state.*

STEP 4. ***For*** *all marked states* x_m *of* $\mathcal{G}^{N_s,m}_{scc}$*, set* $\Gamma(x_m) = \emptyset$*. Compute* $\mathcal{G}^{N_s,m}_{scc,t} = Trim(\mathcal{G}^{N_s,m}_{scc}) = (X^{N_s,m}_{scc,t}, \Sigma^{N_s,m}_{scc,t}, f^{N_s,m}_{scc,t}, \Gamma^{N_s,m}_{scc,t}, x^{N_s,m}_{0_{scc,t}}, X^{N_s,m}_{m_{scc,t}})$*.*

STEP 5. *Compute automaton* $\mathcal{G}_{df}$ *as follows.*

STEP 5.1 *Form the set* $X_{\sigma_f} = \{x \in \mathcal{G}^{N_s,m}_{scc,t} : (\exists x' \in \mathcal{G}^{N_s,m}_{scc,t})[f^{N_s,m}_{scc,t}(x', \sigma_f) = x]\}$*.*

STEP 5.2 *Construct automaton* $\mathcal{G}^{N_s,f}_{scc} = (X^{N_s,f}_{scc}, \Sigma^{N_s,f}_{scc}, f^{N_s,f}_{scc}, \Gamma^{N_s,f}_{scc}, X^{N_s,f}_{0_{scc}}, X^{N_s,f}_{m_{scc}})$*, such that* $X^{N_s,f}_{scc} = X^{N_s,m}_{scc,t}$*,* $\Sigma^{N_s,f}_{scc} = \Sigma^{N_s,m}_{scc,t}$*,* $f^{N_s,f}_{scc} = f^{N_s,m}_{scc,t}$*,* $\Gamma^{N_s,f}_{scc} = \Gamma^{N_s,m}_{scc,t}$*,* $X^{N_s,f}_{0_{scc}} = X_{\sigma_f}$ *and* $X^{N_s,f}_{m_{scc}} = X^{N_s,m}_{m_{scc,t}}$*.*

STEP 5.3 $\mathcal{G}_{d_f} = Trim(\mathcal{G}^{N_s,f}_{scc})$*.*

We explain the key steps of Algorithm 3.6. In Step 1 and 2, we build weighted automaton $\mathcal{G}^{N_s}_{scc}$. In Step 3, we build $\mathcal{G}^{N_s,m}_{scc}$ by marking the states of $\mathcal{G}^{N_s}_{scc}$ that

have at least one component x_{d_i} and x_ℓ are both, $i \in 1, 2, \ldots, N_s$, Y-certain, since this indicates that there exists at least one local diagnoser that is certain that the failure has occurred. In Step 4, we remove all transitions from these states and compute $\mathcal{G}_{scc,t}^{N_s,m}$ by $Trim$ operation since we are only interested in determining the first reached Y-labeled state. In Step 5.2, we form weighted automaton $\mathcal{G}_{scc}^{N_s,f}$ whose unique difference with respect to $\mathcal{G}_{scc,t}^{N_s,m}$ is the new initial state set whose components are all states of $\mathcal{G}_{scc,t}^{N_s,m}$ reached by transitions labeled by failure event σ_f (X_{σ_f} determined in Step 5.1). After computing the $Trim$ operation, we then obtain nondeterministic weighted automaton $\mathcal{G}_{d_f}$. It is worth remarking that $\mathcal{G}_{d_f}$ is assumed to be nondeterministic only because it may have more than one initial states.

The main idea behind Algorithm 3.6 is to obtain an automaton that marks the language after the failure whose marked states are certain states. Thus, we can may state the following result regarding language of $\mathcal{G}_{d_f}$.

Lemma 3.2 *$L_m(\mathcal{G}_{d_f}) = \Theta(L(G))/s$, where $s \in \Psi(\Sigma_f)$.*

Proof. By construction, according to Algorithm 3.6, the set of all minimum codiagnosable traces $\Theta(L(G)) = L_m(\mathcal{G}_{scc,t}^{N_s,m})$, where $\mathcal{G}_{scc,t}^{N_s,m}$ is computed in Step 4. From automaton $\mathcal{G}_{scc,t}^{N_s,m}$, we construct $\mathcal{G}_{d_f}$ whose marked language is $L_m(\mathcal{G}_{d_f}) = L_m(\mathcal{G}_{scc,t}^{N_s,m})/s$, where $s \in \Psi(\Sigma_f)$ such that $\Psi(\Sigma_f)$ denotes the set of all traces of $L_m(\mathcal{G}_{scc,t}^{N_s,m})$ that end with the failure event σ_f. ■

Lemma 3.3 *Weighted automaton $\mathcal{G}_{d_f}$ does not have strongly connected components.*

Proof. Since the language generated by $\mathcal{G}$ is codiagnosable with respect to the set of projections $P_{o_i} : \Sigma^* \to \Sigma_{o_i}^*$, $i = 1, 2, \ldots, N_s$ and $\Sigma_f = \{\sigma_f\}$, we may conclude that, according to Theorem 3.1, $\mathcal{G}_{scc}^{N_s,m}$, and consequently, $\mathcal{G}_{d_f}$ do not have strongly connected components formed with states $(x_{d_1}, x_{d_2}, \ldots, x_{d_{N_s}}, x_\ell)$, such that all x_{d_i}, $i = 1, 2, \ldots, N_s$, are uncertain and x_ℓ is an Y-labeled state. However, we still need to consider the possibility of G_{d_f} to have strongly connected components formed as follows: (i) with states $(x_{d_1}, x_{d_2}, \ldots, x_{d_{N_s}}, x_\ell)$ such, for some $i \in I_{N_s}$, both x_{d_i} and x_ℓ are Y-certain states, (ii) with states $(x_{d_1}, x_{d_2}, \ldots, x_{d_{N_s}}, x_\ell)$ such that for some $i \in I_{N_s}$, both x_{d_i} and x_ℓ are normal states, (iii) with states $(x_{d_1}, x_{d_2}, \ldots, x_{d_{N_s}}, x_\ell)$ where

all components x_{d_i} and x_ℓ are normal. A strongly connected component formed with all x_{d_i} Y-certain and x_ℓ an Y-labeled state is not possible, since, by the construction, all transitions leaving marked states are removed. Also by construction, we can exclude possibilities (ii) and (iii), since the initial state of $\mathcal{G}_{d_f}$ is always a state after the occurrence of the failure event. ■

Notice that, since $L(\mathcal{G}_{scc}^{N_s}) = L(\mathcal{G})$ and $\mathcal{G}_{d_f}$ is constructed from $\mathcal{G}_{scc}^{N_s}$ by considering only those traces after the failure occurrence, and also, according to Lemma 3.3, $\mathcal{G}_{d_f}$ has no strongly connected components, the maximum time τ necessary to diagnose the failure occurrence can be calculated according to the following algorithm.

Algorithm 3.7 *Computation of τ using diagnoser*

Input *Nondeterministic weighted automaton $\mathcal{G}_{d_f}$.*

Output *The maximum time to diagnose the failure occurrence, τ.*

STEP 1. *Using $\mathcal{G}_{d_f}$ as input, apply Algorithm 3.4 for all $x_{0k} \in X_0$ and $x_m \in X_m$.*

STEP 2. *Let $t_1, t_2, \ldots, t_n$ be all outputs of Step 1. Set $\tau = max(t_1, t_2, \ldots, t_n)$.*

Theorem 3.4 *Let τ be computed in accordance with Algorithm 3.7. Then, L is τ-codiagnosable.*

Proof. The proof is straightforward from Fact 3.6, Lemma 3.2 and Lemma 3.3. ■

Remark 3.4 *(Computational complexity analysis of Algorithm 3.6) Table 3.4 shows the maximum number of states and transitions of all automata that must be computed in order to obtain $\mathcal{G}_{d_f}$, according to Algorithm 3.6, assuming N_s local diagnosers. Notice that, in the worst case, the size of $\mathcal{G}_{d_f}$ is equal to the size of $\mathcal{G}_{scc}^{N_s}$, and thus the computational complexity of Algorithm 3.6 is $O(|X||X_d|^{N_s}|\Sigma|)$. Therefore, as expected, the proposed algorithm requires exponential time in the number of states of $\mathcal{G}$ and local sites N_s, and is linear in the number of events of $\mathcal{G}$.*

Example 3.9 *We will illustrate the computation of τ with weighted automaton $\mathcal{G}$ depicted in Figure 3.15, and, to this end, assume the same projections as Example 3.8. According to Steps 1 and 2 of Algorithm 3.6, we compute automaton $\mathcal{G}_{scc}^2$ shown in Figure 3.19(d). In Step 3, we build $\mathcal{G}_{scc}^{2,m}$ by marking states*

Table 3.2: Computational complexity of Algorithm 3.6

	No. of states	No. of transitions
$\mathcal{G}$	$\|X\|$	$\|X\|\|\Sigma\|$
$\mathcal{A}_\ell$	2	2
$\mathcal{G}_\ell$	$2\|X\|$	$2\|X\|\|\Sigma\|$
$\mathcal{G}_{d_i}$	$\|X_d\| = 2^{2\|X\|}$	$2^{2\|X\|}\|\Sigma_o\|$
$\|\|_{i=1}^{N_s}\mathcal{G}_{d_i}$	$\|X_d\|^{N_s}$	$\|X_d\|^{N_s}\|\Sigma_o\|$
$\mathcal{G}_{scc}^{N_s}$	$2\|X\|\|X_d\|^{N_s}$	$2\|X\|\|X_d\|^{N_s}\|\Sigma\|$
Complexity		$O(\|X\|\|X_d\|^{N_s}\|\Sigma\|)$

$(\{6N, 4Y\}, \{4Y\}, \{4Y\})$ *and* $(\{5Y\}, \{3N, 5Y\}, \{5Y\})$ *of* $\mathcal{G}_{scc}^2$*. In Step 4, we remove all transitions from these states and compute* $\mathcal{G}_{scc,t}^{2,m}$ *by* $Trim$ *operation. In Step 5.1, we form a new initial state set whose components are all states of* $\mathcal{G}_{scc,t}^{2,m}$ *reached by transitions labeled by the failure event* σ_f*. Notice, in this case, that there exists only one initial state* $(\{1N, 3N, 2Y, 5Y\}, \{1N, 2Y\}, \{2Y\})$*. We then obtain, in accordance with Step 5.2 of Algorithm 3.6, weighted automaton* $\mathcal{G}_{d_f}$*, which is shown in Figure 3.20. Finally, applying Algorithm 3.7, we obtain* $\tau = 5.6$*, which implies that* L *is* 5.6*-codiagnosable.*

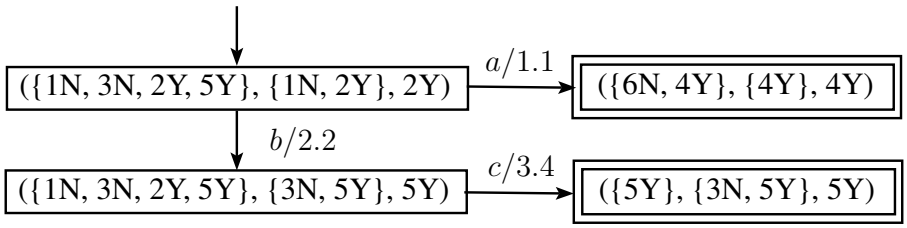


Figure 3.20: Weighted automaton $\mathcal{G}_{d_f}$ of Example 3.9.

(2) Computation of τ using verifier

Let us define the following weighted automata:

$$\mathcal{G}_{N,i} = (G_{N,i}, w_{N,i}), i = 1, \ldots, N_s, \tag{3.11a}$$

and

$$\mathcal{G}_F = (G_F, w_F), \tag{3.11b}$$

where

$$w_{N,i}(xN, \sigma) = \begin{cases} w(x, \sigma), \text{ if } \sigma \in \Sigma \\ 0, \text{ if } \sigma \in \Sigma_{R,i} \end{cases} \tag{3.11c}$$

$$w_F(x\ell, \sigma) = w(x, \sigma), \tag{3.11d}$$

where $x \in X$ and $\ell \in \{N, Y\}$. We will now address the problem of computing τ and, to do so, we propose Algorithm 3.8 to construct $\mathcal{G}_{V_f}$.

Algorithm 3.8 *Computation of automaton* $\mathcal{G}_{V_f}$

Input *Weighted automaton* $\mathcal{G} = (G, w)$*, sets* Σ_f *and* Σ_{o_i}*,* $i = 1, 2, \ldots, N_s$*.*

Output *Weighted automaton* $\mathcal{G}_{V_f} = (G_{V_f}, w_{V_f})$*.*

STEP 1. *Compute automata* $\mathcal{G}_{N,i}$*,* $i = 1, \ldots, N_s$ *and* $\mathcal{G}_F$ *according to Equations* (3.11a)–(3.11d)*.*

STEP 2. *Compute* $\mathcal{G}_V = (\|_{i=1}^{N_s} \mathcal{G}_{N,i}) \| \mathcal{G}_F$*.*

STEP 3. *Compute* $\mathcal{G}_{VT}$ *by adding marked state* F *to* $\mathcal{G}_V$*, according to Step* 3 *of Algorithm 3.3 with* $\mathcal{G}_V$ *in place of* G_V *and set* $w_{VT}(x_{VT}, \sigma) = w_F(x_F, \sigma)$*,* $\forall x_{VT}$ *:* $f_{VT}(x_{VT}, \sigma) = F$*.*

STEP 4. *Form set* $X_{V_{\sigma_f}} = \{x \in X_{VT} : (\exists x' \in \mathcal{G}_{VT})[f_{VT}(x', \sigma_f) = x]\}$*.*

1. *Construct* $\mathcal{G}'_{VT} = ((X'_{VT}, \Sigma'_{VT}, f'_{VT}, \Gamma'_{VT}, X'_{VT,0}, X'_{m,VT}), w_{VT})$*, where* $X'_{VT} = X_{VT}$*,* $\Sigma'_{VT} = \Sigma_{VT}$*,* $f'_{VT} = f_{VT}$*,* $\Gamma'_{VT} = \Gamma_{VT}$*,* $X'_{VT,0} = X_{V_{\sigma_f}}$ *and* $X'_{VT} = X_{VT}$*.*
2. $\mathcal{G}_{V_f} = Trim(\mathcal{G}'_{VT})$*.*

The idea behind Algorithm 3.8 is to obtain a weighted automaton that marks the language after the failure occurrence of $\mathcal{G}_{VT}$. Thus, nondeterministic weighted automaton $\mathcal{G}_{V_f}$, constructed according to Algorithm 3.8, has the following properties.

Lemma 3.4 *All strongly connected components of weighted automaton $\mathcal{G}_{V_f}$, constructed according to Algorithm 3.8, if they exist, will be formed with states $x_{V_f} \in X_{V_f}$ connected with events $\sigma \in \Sigma_{VT}$ such that $w_{V_f}(x_{V_f}, \sigma) = 0$.*

Proof. According to Theorem 2.4, since, by assumption, $L(G)$ is codiagnosable, the verifier constructed according to Algorithm 2.2 only has cycles formed with renamed events, which, according to Equations (3.11a)–(3.11d) have zero weights. Consequently, $\mathcal{G}_{VT}$, constructed according to Algorithm 3.3, does not have such cycles either. Since $\mathcal{G}_{V_f}$ is constructed from $\mathcal{G}_{VT}$ by choosing as its initial state, some particular states of $\mathcal{G}_{VT}$, and performing the $Trim$ operation over the new defined automaton, we may conclude that weighted automaton $\mathcal{G}_{V_f}$ can only have strongly connected components whose states $x_{V_f} \in X_{V_f}$ are connected with events $\sigma \in \Sigma_{VT}$ such that $w_{V_f}(x_{V_f}, \sigma) = 0$. ■

Lemma 3.5 *$P_F\left(L_m(\mathcal{G}_{V_f})\right) = \Theta(L(G_F))/s$, where $s \in \Psi(\Sigma_f)$.*

Proof. ($\Rightarrow$) Let us consider a trace $s_V \in L(G_V)$. Since $G_V = (||_{i=1}^{N_s} G_{N,i})||G_F$, it is clear that $s_V \in P_F^{-1}(L(G_F))$. Then, $\exists s_Y = P_F(s_V) \in L(G_F)$. Form trace $s_{VT} = s_V\sigma \in L_m(G_{VT})$, and notice that, by construction, $s_V\sigma \in P_F^{-1}(L(G_F))$. Thus, $P_F(s_V\sigma) = s_Y\sigma \in L(G_F)$. Write $s_{VT} = s't'\sigma$, where $s' = s''\sigma_f$ and $\Sigma_f \notin s''$. By construction of $\mathcal{G}_{V_f}$ in Algorithm 3.8, $t'\sigma \in L_m(\mathcal{G}_{V_f})$, which implies that $P_F(t'\sigma) = t \in P_F(Lm(\mathcal{G}_{V_f}))$. In addition, $P_F(s_{VT}) = P_F(s')P_F(t'\sigma) = st$, and, by the construction of Algorithm 3.8, $t \in L(G_F)/s$ for every $s \in \Psi(\Sigma_f)$, and, from Definition 3.1, we can conclude that $t \in \Theta(L(G_F))/s$. Conversely, let $t \in \Theta(L(G_F))/s$, where $s \in \Psi(\Sigma_f)$. From Fact 3.5, $t \in P_F\left(L_m(\mathcal{G}_{VT})\right)/s$, and since, by construction, $\mathcal{G}_{V_f}$ is formed by states of $\mathcal{G}_{VT}$ after the failure occurrence, $t \in P_F\left(L_m(\mathcal{G}_{V_f})\right)$. ■

Based on Lemmas 3.4 and 3.5, we propose Algorithm 3.9 to compute τ.

Algorithm 3.9 *Computation of maximum delay τ using verifier*

Input *Nondeterministic weighted automaton $\mathcal{G}_{V_f} = (G_{V_f}, w_{V_f})$.*

Output *The maximum delay to diagnose the failure occurrence, τ.*

STEP 1. *If $\mathcal{G}_{V_f}$ has zero-weight cycles then transform it into an acyclic weighted automaton [60].*

STEP 2. *Using $\mathcal{G}_{V_f}$ as input, apply Algorithm 3.4 for all $x_{0k} \in X_0$, and for state $F \in X_m$. Let $t_1, t_2, \ldots, t_n$ be all outputs. Set $\tau = max(t_1, t_2, \ldots, t_n)$.*

Theorem 3.5 *Let τ be computed in accordance with Algorithm 3.9. Then, L is τ-codiagnosable.*

Proof. The proof is straightforward from Lemma 3.4, Lemma 3.5, Fact 3.6, and Equation (3.6). ■

Example 3.10 *Let us consider the weighted automaton $\mathcal{G} = (G, w)$ obtained from automaton G of Example 3.5 by adding the following weight function: $w(1, \sigma_f) = 0$, $w(1, b) = w(2, b) = 2.2$, $w(2, a) = w(3, a) = 1.1$, $w(4, c) = w(5, c) = w(6, c) = 3.4$. In addition, consider the same projections as in Example 3.5, $P_{o_1} : \Sigma^* \to \Sigma_{o_1}^*$ and $P_{o_2} : \Sigma^* \to \Sigma_{o_2}^*$, where $\Sigma_{o_1} = \{a, c\}$ and $\Sigma_{o_2} = \{a, b\}$. Following Algorithm 3.3, we obtain the weighted extended verifier automaton $\mathcal{G}_{VT}$ which is similar to the extended verifier shown in Figure 3.14(b), with the following arc weights: $w((1N, 1N, 1N), b_{R_1}) = 0, w((1N, 1N, 1N), \sigma_f) = 0, w(\{3N, 1N, 1N\}, \sigma_f) = 0, w((1N, 1N, 2Y), b_{R_1}) = 0, w((1N, 1N, 2Y), b) = 2.2, w((1N, 1N, 2Y), a) = 1.1, w((3N, 1N, 2Y), a) = 1.1, w((3N, 1N, 2Y), b) = 2.2, w((1N, 3N, 5Y), b_{R_1}) = 0, w((1N, 3N, 5Y), c) = 3.4, w((3N, 3N, 5Y), c) = 3.4$. After computing $\mathcal{G}_{VT}$, the next step of Algorithm 3.8 is to compute automaton $\mathcal{G}_{V_f}$, which is depicted in Figure 3.21. Finally, applying Algorithm 3.9, we obtain $\tau = max\{1.1, 5.6\} = 5.6$, which implies that L is 5.6-codiagnosable, which is the same result as that obtained by using diagnosers, as expected.*

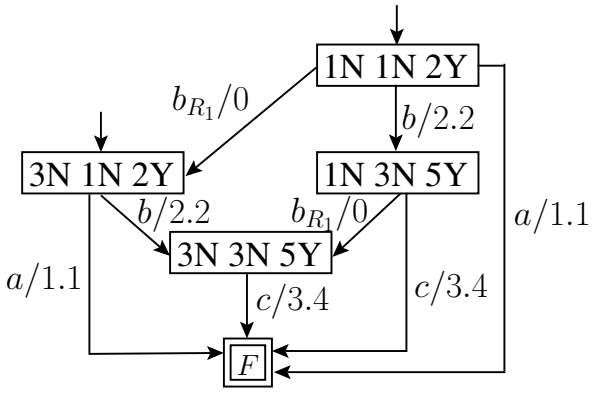


Figure 3.21: Weighted automaton $\mathcal{G}_{V_f}$.

Table 3.3: Computational complexity of Algorithm 3.8

	No. of states	No. of transitions
$\mathcal{G}_V$	$\lvert X_V\rvert = 2\lvert X\rvert^{N_s+1}$	$\lvert X_V\rvert(\lvert\Sigma\rvert + N_s\lvert\Sigma_{nf}\rvert)$
$\mathcal{G}_{VT}$	$\lvert X_{VT}\rvert = \lvert X_V\rvert + 1$	$\lvert X_V\rvert(\lvert\Sigma\rvert + \lvert\Sigma_o\rvert + N_s\lvert\Sigma_{nf}\rvert))$
	Complexity	$O(N_s\lvert X\rvert^{(N_s+1)}\lvert\Sigma\rvert)$

Remark 3.5 *(Computational complexity analysis of Algorithm 3.8) Table 3.5 shows the maximum number of states and transitions of all weighted automata that must be computed in order to obtain $\mathcal{G}_{V_f}$, according to Algorithm 3.8, where $\Sigma_{nf} = \Sigma \setminus \Sigma_f$ and N_s is the number of local diagnosers. In the worst case, the size of $\mathcal{G}_{V_f}$ is equal to the size of $\mathcal{G}_{VT}$ and, thus, the computational complexity of Algorithm 3.8 is $O(N_s\lvert X\rvert^{(N_s+1)}\lvert\Sigma\rvert)$. As expected, the proposed algorithm requires polynomial time in the number of states of $\mathcal{G}$, and is linear in the number of events of $\mathcal{G}$. In addition, notice that the Algorithm 3.9 requires the computation of τ by topological ordering and transforming the verifier obtained in Algorithm 3.8 into acyclic weighted automaton, which, as seen in [60], are both linear in the number of transitions of the automaton. Therefore, the computational complexity of Algorithm 3.9 is, still $O(N_s\lvert X\rvert^{(N_s+1)}\lvert\Sigma\rvert)$, which is significantly smaller than that of the algorithm proposed in [65], which can be checked to be $O(\lvert X\rvert^{(2(N_s+1))}\lvert\Sigma\rvert^{(2(N_s+1))})$.*

3.3.3 K-Codiagnosability Analysis

In this section, we will address the problem of finding the maximum number of events a codiagnosis system takes to diagnose a failure occurrence (K-codiagnosability). We start with the following definition.

Definition 3.3 *(K-codiagnosability) Given a system modeled as an automaton G, we say that $L(G)$ is K-codiagnosable with respect to $\Sigma_f = \{\sigma_f\}$ and P_{o_i}, $i = 1, \ldots, N_s$ if $L(G)$ is codiagnosable with respect to Σ_f and P_{o_i}, $i = 1, \ldots, N_s$ and the maximum number of event occurrences necessary for the codiagnosis system to diagnose the occurrence of σ_f is K.*

Let us form from G the following weighted automaton: $\mathcal{G}_K = (G, w_K)$, where for

all $x \in X$ and $\sigma \in \Gamma(x)$, $w_K(x, \sigma) = 1$. The following proposition establishes the relationship between K-codiagnosability and τ-codiagnosability.

Proposition 3.1 *Let $\mathcal{G}_K = (G, w_K)$ be a weighted automaton obtained from G such that $w_K(x, \sigma) = 1$, for all $x \in X$ and $\sigma \in \Gamma(x)$, and let τ denote the maximum time to diagnose the failure occurrence, computed according to Algorithm 3.7. Then, G is K-codiagnosable if $\mathcal{G}_K$ is τ-codiagnosable for $T = K$.*

Proof. According to Equation (3.6), $L(G)$ is τ-codiagnosable for $\tau := \max_{st \in \Theta(L(G)): s \in \Psi(\Sigma_f)} (T(st))$. Writing $t = \sigma_1 \sigma_2 \ldots \sigma_{n_{st}}$ then:

$$\tau(st) = \sum_{j=1}^{n_{st}} w(x_j, \sigma_j), = n_{st},$$

and since $w_K(x, \sigma) = 1$, we can conclude that $\tau(st) = n_{st}$, where n_{st} is the number of events necessary to diagnose the failure occurrence for trace st. Therefore:

$$\tau = \max \{n_{st} : (st \in \Theta(L(G))) \wedge (s \in \Psi(\Sigma_f))\} = K,$$

where the last equality comes from Definition 3.1. ■

According to Proposition 3.1, it is clear that there are two ways to compute the maximum number of events K necessary to diagnose a failure occurrence:

(i) by first applying Algorithm 3.6 to $\mathcal{G}_K$ and, after that by using Algorithm 3.7 (using diagnoser-like approach);

(ii) by first applying Algorithm 3.8 to $\mathcal{G}_K$ and, after that by using Algorithm 3.9 (using verifier approach).

The following example illustrates the computation of K.

Example 3.11 *Consider once again weighted automaton $\mathcal{G}$ of Figure 3.15.*

In order to compute K by using diagnoser-like method we first construct the weighted automaton $\mathcal{G}_K = (G, w_K)$, where for all $x \in X$ and $\sigma \in \Gamma(x)$, $w_K(x, \sigma) = 1$. Let us initially apply Algorithm 3.6 choosing $\mathcal{G}_K$ as input. The resulting weighted automaton $\mathcal{G}_{d_f}$ is shown in Figure 3.22, from where we can see that this automaton is the same as that of Figure 3.20 except that all transitions have unit weight. Therefore, applying Algorithm 3.7, we obtain $K = 2$.

In order to compute K by using the verifier method, we apply Algorithm 3.8, choosing $\mathcal{G}_K$ as input, we obtain automaton $\mathcal{G}_{V_f}$, depicted in Figure 3.23, which is the same as that of Figure 3.21 except that transitions $a/1.1$, $b/2.2$ and $c/3.4$ are replaced with $a/1$, $b/1$ and $c/1$, respectively. Finally, applying Algorithm 3.9, we obtain $K = 2$, which is the same result as that obtained by diagnoser.

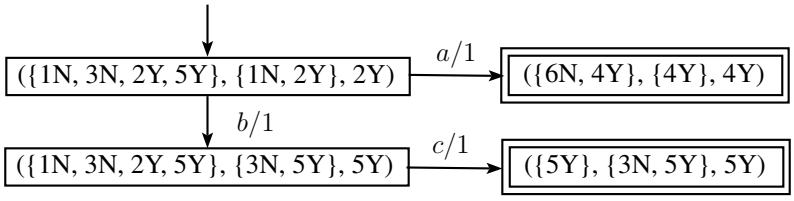


Figure 3.22: Weighted automaton $\mathcal{G}_{d_f}$ of Example 3.11.

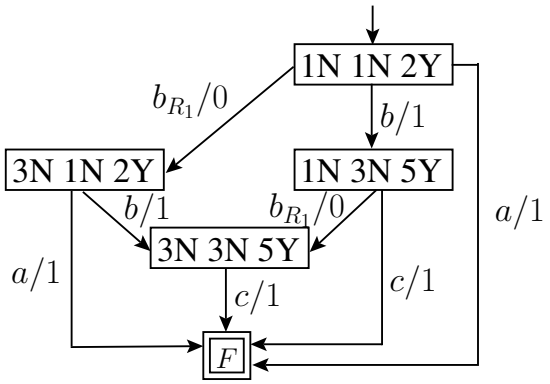


Figure 3.23: Weighted automaton $\mathcal{G}_{V_f}$ of Example 3.11.

3.4 Concluding Remarks

In this chapter, we first proposed a new test for the verification of language codiagnosability based on diagnosers that also takes into account the time interval due to the occurrence of unobservable events and does not require the usual assumptions on language liveness and prevention of cycles of states connected with unobservable events. In addition, the proposed test is based on the search for strongly connected components, as opposed to cycles [19, 21, 54]. Regarding codiagnosability verification using verifiers, we extended the verifier automaton proposed in [47] to also show those paths that lead to language diagnosis. We proposed two algorithms for the computation of the maximum time τ for failure diagnosis: one algorithm that uses

diagnosers and another one that uses verifiers. We solved the K-codiagnosability problem by converting it into an equivalent τ-codiagnosability one; therefore making K-codiagnosability a particular case of τ-codiagnosability. The resulting algorithms have the lowest worst case computational complexity among those proposed in the literature for τ-codiagnosability [63, 64] of weighted discrete event systems and for K-codiagnosability verification of finite automata [44, 62, 65]. In next chapter, we will propose another problem in the contex of failure diagnosis: codiagnosability of networked discrete event systems with timing structure. In order to check this property we will propose a slight variation of the diagnoser-like automaton introduced in this chapter and on the verifier proposed in [47].

Chapter 4

Codiagnosability of Networked Discrete Event Systems With Timing Structure

In the standard failure diagnosis problem described in Section 2.3, it is assumed that diagnosers observe the occurrence of an event immediately after it has been executed by the plant and without loss in the communication channels. When the plant and diagnosers are either far from each other or a more complex network is used to connect them, communication delays are unavoidable and must be taken into account. Such a diagnosis problem is referred, in the literature, to as codiagnosability of networked discrete event systems [42]. In this chapter, we consider the codiagnosability problem of Networked Discrete Event Systems With Timing Structure (NDESWTS) subject to bounded communication delays and intermittent loss of observations. We assume that the communication between the plant and the local diagnosers is carried out through a network that can have several channels, so that communication delays can cause changes in the order of the observations. Preliminary versions of the results presented in this chapter have been published in [82, 83]. In addition, the model for NDESWTS proposed here has also been used to address the supervisory control of NDESWTS subject to event communication delays and loss of observations in [38, 39].

This chapter is organized as follows. In Section 4.1, we formally define NDESWTS, and present a motivating example. In Section 4.2, we propose the

model of the plant subject to communication delays and, in the sequel, we include intermittent loss of observations. In Section 4.3, we present necessary and sufficient conditions for codiagnosability of NDESWTS and propose two tests to its verification: the first one based on diagnosers, being slight variation of the diagnoser-like automaton introduced in Chapter 3, and a second one, based on verifiers. We present some final comments in Section 4.4.

4.1 Problem Formulation

In this thesis, we consider the networked architecture for a distributed plant introduced in [42,43], formed with m measurement sites MS_j, $j = 1, \ldots, m$, and N_s local diagnosers LD_i, for $i = 1, \ldots, N_s$. Each measurement site MS_j can detect a subset $\Sigma_{MS_j} \subset \Sigma_o$ of the observable event set of the system. In this configuration, only the events detected by measurement site MS_j can be communicated through channel ch_{ij} to local diagnoser LD_i.

Differently from the approach adopted in [42], where communication delays were represented by steps, we consider that each channel ch_{ij} has a maximal delay $T_{ij} \in \mathbb{R}^*_+$, where $\mathbb{R}^*_+$ denotes the set of positive real numbers [1]. We denote the set of events communicated to local diagnoser LD_i, through communication channel ch_{ij}, as $\Sigma_{o_{ij}} \subseteq \Sigma_{MS_j}$. If the communication channel ch_{kl}, between a measurement site MS_l and a local diagnoser LD_k, does not exist, then $\Sigma_{o_{kl}} = \emptyset$. Thus, the set of observable events of LD_i, Σ_{o_i}, is given by:

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

and, the set of observable events of the whole system is $\Sigma_o = \bigcup_{i=1}^{N_s} \Sigma_{o_i}$.

Figure 4.1 shows the NDESWTS architecture proposed in this work for a system with three measurement sites and two local diagnosers. Measurement site MS_1 communicates to local diagnoser LD_1 through channel ch_{11}, those events in $\Sigma_{o_{11}} \subseteq \Sigma_{MS_1}$, and has a maximal delay equal T_{11}. Measurement site MS_2 communicates the events in $\Sigma_{o_{12}} \subseteq \Sigma_{MS_2}$ and $\Sigma_{o_{22}} \subseteq \Sigma_{MS_2}$ to the local diagnosers LD_1 and LD_2,

[1] This means that $T_{ij} \neq 0, i = 1, \ldots, N_s, j = 1, 2, \ldots, m$ (no instantaneous transmission) but can be made arbitrarily small.

respectively, through communication channels ch_{12} and ch_{22} with maximal delays T_{12} and T_{22}. Finally, measurement site MS_3 communicates to local diagnoser LD_2 through channel ch_{23}, those events in $\Sigma_{o_{23}} \subseteq \Sigma_{MS_3}$ with maximal delay T_{23}.

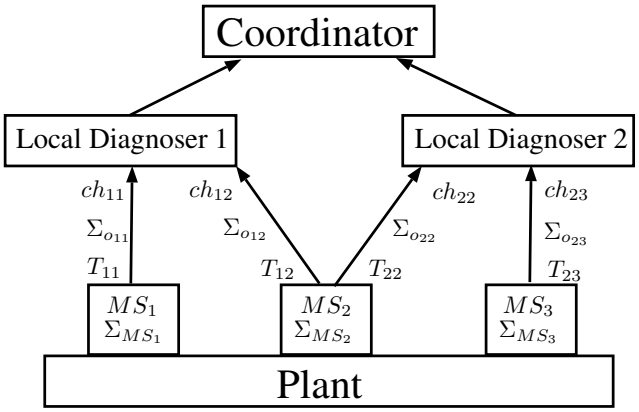


Figure 4.1: NDESWTS architecture.

We make the following assumptions.

A1. $L(G)$ is codiagnosable with respect to P_{o_i}, $i = 1, \ldots, N_s$ and Σ_f, where G is the automaton that models the plant.

A2. There is only one communication channel ch_{ij} between measurement site MS_j and local diagnoser LD_i, communicating the events in $\Sigma_{o_{ij}}$.

A3. Each channel ch_{ij}, is modeled by a first-in first-out (FIFO) queue, and it is subject to a previously known maximal communication delay $T_{ij} \in \mathbb{R}^*_+$, T_{ij} finite;

A4. $\Sigma_{MS_i} \cap \Sigma_{MS_j} = \emptyset$, $i, j \in I_m = \{1, 2, \ldots, m\}$, $i \neq j$, *i.e.*, different measurement sites have no common events.

A5. $\Sigma_o = \Sigma_{ilo} \dot{\cup} \Sigma_{nilo}$, where Σ_{ilo} (resp. Σ_{nilo}) denotes the set of events subject (resp. not subject) to communication losses.

Assumption **A1** ensures that the only cases of interest are those when the language is codiagnosable, *a priori*, *i.e.*, when both no delay and no loss in the transmission of the signal that corresponds to event observation is assumed. In accordance with Assumption **A2**, an observable event occurrence is transmitted through a unique channel to a local diagnoser; however, there may exist two channels to

transmit an event occurrence to two different local diagnosers. Assumption **A3** imposes that the maximal communication delays are strictly different from zero for all channels. In addition, since each communication channel is modeled by a FIFO queue, there is no change of order of observation among events transmitted through the same channel. According to Assumption **A4**, two different measurement sites cannot record the occurrence of the same event, which is justified by the fact that, in practice, different measurement sites are required when the system is either decentralized in nature (for example, computer and communication networks, manufacturing, process control and power systems, etc.) or is informationally decentralized. Finally, according to Assumption **A5**, the loss of observation of an event does not change the plant behavior, but only the observation.

The consequences of transmission delays in the diagnosis system depend on the dynamic behavior of the plant, since if the dynamic of the plant is sufficiently slow, transmission delays may generate no adverse consequences. Therefore to better consider the effects of the dynamic behavior in real systems, it is reasonable to assume that a state transition cannot occur immediately after a previous one, that is, the system needs to remain for some time in a given state before a new transition occurs. Thus, we need to define the partial function $t_{min} : X \times \Sigma \to \mathbb{R}_+^*$, where $t_{min}(x, \sigma) = \tau$, for $\sigma \in \Gamma(x)$, means that event σ can only occur at state x if the time elapsed since the last transition occurrence is greater than (but not equal to) τ. Namely, t_{min} assigns a minimal time to the firing of each transition of G and is defined, over a pair $(x, \sigma) \in X \times \Sigma$ if, and only if, transition $f(x, \sigma)$ is defined. In addition, we assume that, $\forall x \in X$ and $\sigma \in \Gamma(x)$, $t_{min}(x, \sigma) > 0$. The need for excluding $\tau = 0$ is imposed by a technical constraint necessary to allow the NDESWTS to be converted into an untimed finite-state automaton. With a slight abuse of language, we will refer to $t_{min}(s, \sigma)$ as the minimal activation time. Partial function t_{min} is extended to domain $X \times \Sigma^*$ in the following recursive manner: for all $x \in X$, $t_{min}(x, \varepsilon) = 0$, and $t_{min}(x, s\sigma) = t_{min}(x, s) + t_{min}(f(x, s), \sigma)$ for $s \in \Sigma^*$ and $\sigma \in \Sigma^*$. We can now introduce the concept of NDESWTS.

Definition 4.1 *(NDESWTS) A networked discrete event system with timing structure NDESWTS is a 3-tuple* $NDESWTS = (G, t_{min}, T)$, *where* $G = (X, \Sigma, f, \Gamma, x_0, X_m)$ *is a finite automaton,* $t_{min} : X \times \Sigma \to \mathbb{R}_+^*$ *is the minimal time*

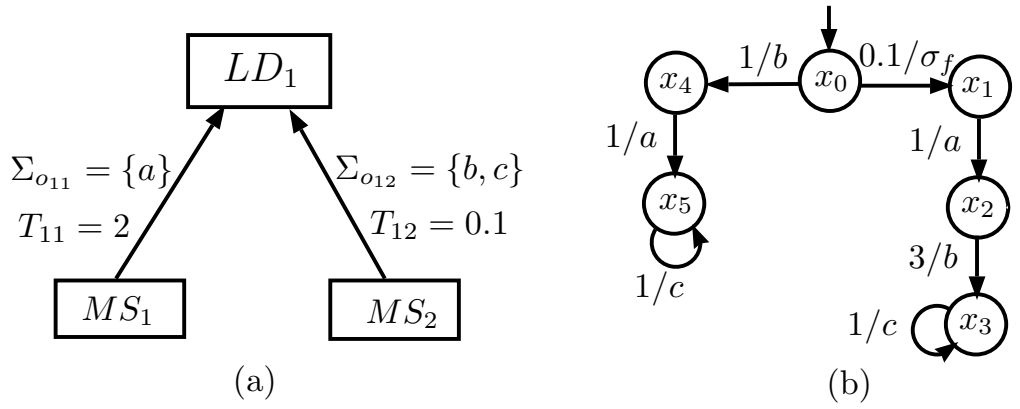


Figure 4.2: *NDESWTS* $= (G, t_{min}, T)$ for Example 4.1: communication delay structure (a) and automaton G with minimal time function t_{min} (b).

function and $T : n \times m$ *is the maximal matrix delay, where each element* T_{ij} *represents the maximal delay of each channel* ch_{ij} *such that* $T_{ij} \in \mathbb{R}^*_+$*, if there exists a communication channel between measurement site* MS_j *and local diagnoser* LD_i*, and* $T_{ij} = \infty$*, otherwise.* □

It is worth remarking that a state transition (x, σ, y), where $f(x, \sigma) = y$, of *NDESWTS* only occurs when the time elapsed since the occurrence of the last state transition is greater than a known minimal activation time $t_{min}(x, \sigma) \in \mathbb{R}^*_+$. Notice that this modeling is not restrictive since the minimal firing time $t_{min}(x, \sigma)$ can be as small as possible. The following example illustrates a networked discrete event system with timing structure.

Example 4.1 *Consider the networked discrete event system with timing structure* $NDESWTS = (G, t_{min}, T)$ *shown in Figure 4.2(a), where the communication delay structure of the NDESWTS architecture is presented and Figure 4.2(b), where automaton* G *together with the minimal time function* t_{min} *is depicted. Notice that the* $NDESWTS$ *communication delay structure consists of two measurement sites and only one local diagnoser; measurement site* MS_1 *communicates only event* a *to local diagnoser* LD_1 *through channel* ch_{11}*, with maximal delay* $T_{11} = 2$ *time units (t.u.), whereas measurement site* MS_2 *communicates events* b *and* c *to local diagnoser* LD_1 *with maximal delay* $T_{12} = 0.1$ *t.u. Therefore* $T = [T_{11}\ T_{12}] = [2\ 0.1]$*. Regarding the minimal time function* t_{min}*, according to Figure 4.2(b)* t_{min} *is given as:* $t_{min}(x_0, \sigma_f) = 0.1$*,* $t_{min}(x_0, b) = t_{min}(x_1, a) = t_{min}(x_3, c) = t_{min}(x_4, a) =$

$t_{min}(x_5, c) = 1$ *and* $t_{min}(x_2, b) = 3$. *Notice that label over transition* $f(x_4, a) = x_5$ *means that* $t_{min}(x_4, a) = 1$, *which means that the event* a *occurs at least* 1 *t.u. after* G *enters in state* x_4; *the remaining labels are similarly interpreted.*

In order to distinguish between the occurrences of events a, b *and* c *and their observations by diagnoser* LD_1, *let* a_s, b_s *and* c_s *denote the successful observation of events* a, b *and* c, *respectively. Notice that when the system generates trace* $s_1 = \sigma_f abc^p$, *where* $p \in \mathbb{N}$, *event* a *will always be observed by the local diagnoser* LD_1 *before the occurrence of event* b, *since the observation of event* a *can be delayed by at most* $T_{11} = 2$ *t.u. and event* b *occurs at least* 3 *t.u. after the occurrence of event* a, *as show in time line depicted in Figure 4.3. In addition, event* c *is always observed by* LD_1 *after the observation of* b *since they are transmitted through the same communication channel. Therefore, after the occurrence of trace* s_1, *the diagnoser observes* $s_{1s} = a_s b_s c_s^p$. *Notice that, in this case, the system dynamics and the communication channel delays do not interfere in the order of trace observation of events.*

On the other hand, when the system executes trace $s_2 = bac^q$, *where* $q \in \mathbb{N}$, *events* b *and* a *will not be observed in a different order of occurrence since* $t_{min}(x_4, a) > T_{12}$. *However, it is possible that the diagnoser observes either* $s_{2s} = b_s a_s c_s^q$ *or* $s'_{2s} = b_s c_s a_s c_s^{q-1}$ *as shown in the time line, depicted in Figure 4.4. Notice that since* $t_{min}(x_5, c) = 1 < T_{11} = 2$, *the first observation of* c (c_s) *may take place before the observation of event* a (a_s). *However, the subsequent observation of* c *will be made after* a_s, *since* $t_{min}(x_5, cc) = t_{min}(x_5, c) + t_{min}(x_5, c) = 2 = T_{11}$; *notice that in Figure 4.4, even though* a_s *and the second event* c *occur at the same time,* c_s *is always observed after* a_s, *since* c_s *must occur after event* c, *i.e., between* $]2; 2.1]$ *time units after occurrence of event* a.

Remark 4.1 *If, in the networked system of Example 4.1, we adopt, instead of the NDESWTS structure proposed here, the the communication delay approach using steps, proposed in [33,42,43], and assuming that the delays of channels* ch_{11} *and* ch_{12} *are at most* 1 *and* 0 *steps, respectively, then a change in the order of the observation of events* a *and* b *after trace* $s_1 = \sigma_f abc^p$ *must also be taken into account, i.e., two observation traces must be considered;* $s_{1s} = a_s b_s c_s^p$ *and* $s'_{1s} = b_s a_s c_s^p$. *Regarding trace* $s_2 = bac^q$, *like in the structure adopted here ,two observation are possible,*

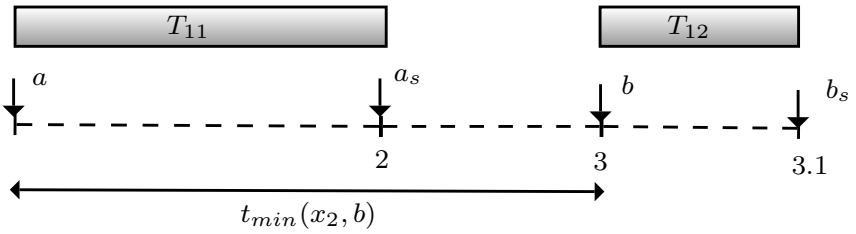


Figure 4.3: Time line after occurrence of events a and b, in trace $s_1 = \sigma_f abc^p$, where $p \in \mathbb{N}$.

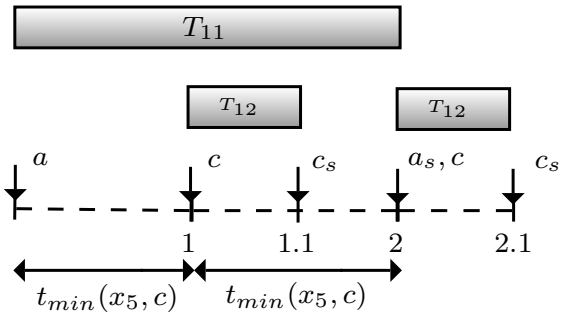


Figure 4.4: Time line after occurrence of event a and two events c, in trace $s_2 = bac^q$, where $q \in \mathbb{N}$.

$s_{2s} = b_s a_s c_s^q$ *or* $s'_{2s} = b_s c_s a_s c_s^{q-1}$. *Notice that* $s'_{1s} = s_{2s}$, *which implies that, in this case, the language is not networked diagnosable [42,43], which is incorrect since, as we saw in Example 4.1, the observed trace* $s'_{1s} = b_s a_s c_s^p$ *cannot occur.*

Motivated by this example, we propose, in this thesis, an alternative approach for failure codiagnosability that considers heterogeneous temporal behavior of the plants.

4.2 Modeling of NDESWTS

4.2.1 An Equivalent Untimed Model of NDESWTS Subject to Communication Delays Only

In order to propose a model that takes into account possible changes in the order of observation of the events when compared with actual order of occurrence in the system, we need to distinguish an event $\sigma \in \Sigma_{o_{ij}}$, that is being 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. To this end, we create an event σ_{s_i} that models the successful observation of σ by LD_i and form the set of observable events that are successfully communicated to local diagnoser LD_i, as follows.

$$\Sigma^s_{o_i} = \bigcup_{j=1}^{m} \Sigma^s_{o_{ij}}, \tag{4.2}$$

where $\Sigma^s_{o_{ij}} = \{\sigma_{s_i} : \sigma \in \Sigma_{o_{ij}}\}$.

Let us now define function ψ_i, for $i = 1, \ldots, N_s$, that returns the successful observation event σ_{s_i} of event σ generated by the plant as follows.

$$\psi_i : \Sigma^*_{o_i} \to \Sigma^{s*}_{o_i}, \tag{4.3}$$

where $\psi_i(\varepsilon) = \varepsilon$, $\psi_i(\sigma) = \sigma_{s_i}$, for all $\sigma \in \Sigma_{o_i}$ and $\psi_i(s\sigma) = \psi_i(s)\psi_i(\sigma)$, $\forall s \in \Sigma^*_{o_i}$ and $\sigma \in \Sigma_{o_i}$. Function ψ_i is extended to languages by applying it to all strings in the language. In this regard, for the set of events Σ_{o_i}, $\psi_i(\Sigma_{o_i}) = \bigcup_{\sigma \in \Sigma_{o_i}} \psi_i(\sigma)$. In addition, we can define ψ_i^{-1} its inversion function as $\psi_i^{-1} : \Sigma^{s*}_{o_i} \to \Sigma^*_{o_i}$, where $\psi_i^{-1}(\varepsilon) = \varepsilon$, $\psi_i^{-1}(\sigma_{s_i}) = \sigma$, for all $\sigma_{s_i} \in \Sigma^s_{o_i}$ and $\psi_i^{-1}(s\sigma_{s_i}) = \psi_i^{-1}(s)\psi_i^{-1}(\sigma_{s_i})$, $\forall s \in \Sigma^{s*}_{o_i}$ and $\sigma_{s_i} \in \Sigma^s_{o_i}$. The system behavior in the presence of delays will, therefore, be modeled by a language defined over the following extended set of events:

$$\Sigma_i := \Sigma \cup \Sigma^s_{o_i}. \tag{4.4}$$

In order to obtain all possible observations of a trace $s \in L(G)$ by a local diagnoser LD_i, we introduce a function that inserts events belonging to $\Sigma^s_{o_i}$ based on the maximal communication delay bound T_{ij}, minimal time function t_{min} and event sets $\Sigma_{o_{ij}}$. Let us first define the following projections:

$$P_i : \Sigma^*_i \to \Sigma^*, \tag{4.5}$$

$$P_{i,o_{ij}} : \Sigma^*_i \to \Sigma^*_{o_{ij}}, \tag{4.6}$$

$$P_{i,s_{ij}} : \Sigma^*_i \to \Sigma^{s*}_{o_{ij}}, \tag{4.7}$$

$$P_{is} : \Sigma^*_i \to \Sigma^{s*}_{o_i}. \tag{4.8}$$

In addition, let $w_{\sigma^{(l)}}$ denote the prefix of a trace $w \in \Sigma^*_i$ whose last event is the l-th occurrence of σ, and let $w_{\sigma^{(l)}_{s_i}}$ be the prefix of w whose last event is the l-th occurrence of σ_{s_i}, if $\sigma^{(l)}_{s_i} \in w$, or w, if $\sigma^{(l)}_{s_i} \notin w$. For instance, let $\Sigma_1 = \{a, b, c, \sigma_f, a_{s_1}, c_{s_1}\}$ and

$w = aba_{s_1}acc_{s_1}a_{s_1}c$. Then, $w_{a^{(2)}} = aba_{s_1}a$, $w_{c_{s_1}^{(1)}} = aba_{s_1}acc_{s_1}$ and $w_{c_{s_1}^{(2)}} = w$. The idea of defining $w_{\sigma^{(l)}}$ and $w_{\sigma_{s_i}^{(l)}}$ is to establish a comparison between the event occurrence and its observation in a given trace. We can now introduce the concept of insertion function.

Definition 4.2 *(Insertion function) An insertion function associated with local diagnoser LD_i and observable events in $\Sigma_{o_{ij}}$, transmitted through communication channels ch_{ij} that have maximal communication delays bounds T_{ij}, is a mapping:*

$$\chi_i : L(G) \rightarrow 2^{\Sigma_i^*}$$
$$s \mapsto \chi_i(s),$$

where $w \in \chi_i(s)$ if it satisfies the following conditions:

1. *$P_i(w) = s$;*
2. *For all $\sigma \in \Sigma_{o_{ij}}$, and $\sigma^{(l)} \in w$:*

$$t_{min}(x_0, P_i(w_{\sigma_{s_i}^{(l)}})) - t_{min}(x_0, P_i(w_{\sigma^{(l)}})) < T_{ij} \tag{4.9}$$

3. *For all $\sigma_{s_i} \in \Sigma_{o_{ij}}^s$, and $\sigma_{s_i}^{(l)} \in w$:*

$$\sigma^{(l)} \in w_{\sigma_{s_i}^{(l)}}, \tag{4.10}$$

and

$$|P_{i,o_{ij}}(w_{\sigma^{(l)}})| = |P_{i,s_{ij}}(w_{\sigma_{s_i}^{(l)}})|, \tag{4.11}$$

where $|.|$ denotes the length of the trace.

The extension of χ_i to the domain $2^{L(G)}$ is defined as $\chi_i(L(G)) = \bigcup_{t \in L(G)} \chi_i(t)$.

Condition 1 ensures no event in Σ_{o_i} can be inserted to s in order to form w, *i.e.*, only w must be obtained from s by inserting events in $\Sigma_{o_i}^s$. Condition 2 ensures that the delay between the occurrence of event $\sigma \in \Sigma_{o_{ij}}$, and its observation $\sigma_{s_i} \in \Sigma_{o_{ij}}^s$ is not larger than the maximum delay bound T_{ij} (Equation (4.9)). Finally, Condition 3 ensures that the observation σ_{s_i} of an event σ only occurs after event σ has occurred in trace w (Equation (4.10)), and that the observation of events transmitted through the same communication channel is in the same order of their occurrence in trace s (Equation (4.11)). The following example illustrates the usefulness of insertion function χ_i.

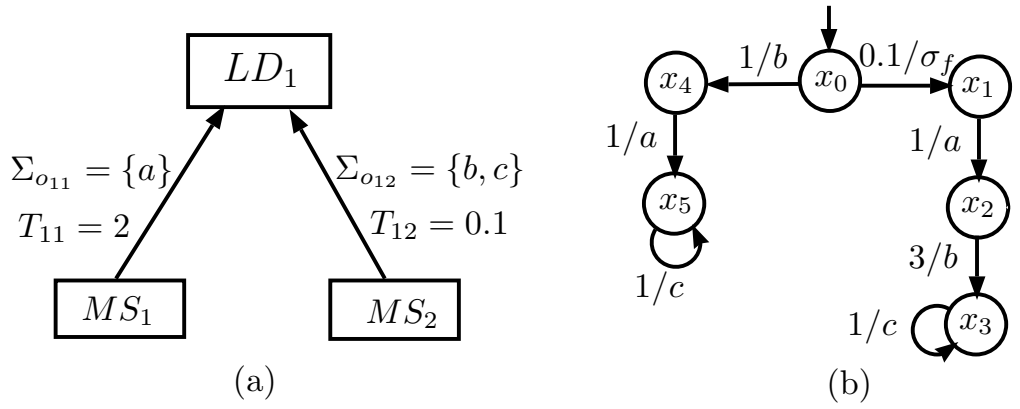


Figure 4.5: $NDESWTS = (G, t_{min}, T)$ for Example 4.1, where communication delay structure (a) and automaton G with minimal time function t_{min} (b), which is considered again in Example 4.2.

Example 4.2 *Consider the networked discrete event system with timing structure* $NDESWTS = (G, t_{min}, T)$ *shown again in Figures 4.5(a) and 4.5(b), and assume that trace* $s_1 = \sigma_f abc^p$, $p \in \mathbb{N}$ *has been executed by the system. Let us consider traces*[2] $w_1, w_2, w_3, w_4 \in \Sigma_1^*$, *where* $w_1 = \sigma_f aabb_s a_s (cc_s)^p$, $w_2 = \sigma_f aba_s b_s (cc_s)^p$, $w_3 = \sigma_f aa_s b_s b (cc_s)^p$ *and* $w_4 = \sigma_f aa_s bcc_s b_s (cc_s)^{p-1}$. *Notice that none of these traces belongs to* $\chi_1(s_1)$, *since (i)* $P_1(w_1) = \sigma_f aabc^p \neq s_1$ *(w_1 violates Condition 1 of Definition 4.2); (ii)* $t_{min}(x_0, P_1(\sigma_f aba_s) - t_{min}(x_0, P_i(\sigma_f a)) = t_{min}(x_0, \sigma_f ab) - t_{min}(x_0, \sigma_f a) = (0.1 + 1 + 3) - (0.1 + 1) = 3 > T_{11} = 2$ *(w_2 violates Condition 2 of Definition 4.2); (iii)* $b_s^{(1)} \in w_3$, *but* $b^{(1)} \notin w_{3_{b_s^{(1)}}}$ *(w_3 violates Condition 3, Equation (4.10), of Definition 4.2); (iv) one event c is observed before b in trace* w_4, *however, these events are transmitted through the same communication channel; this is recognized by Equation (4.11), as follows:* $|P_{1,o_{12}}(\sigma_f aa_s b)| = |b| = 1 \neq |P_{1,s_{12}}(\sigma_f aa_s bcc_s b_s)| = |c_s b_s| = 2$ *(w_4 violates Condition 4 of Definition 4.2). As we are going later on in this section, the set of traces in* Σ_1^* *associated with all possible observations of trace s by local diagnoser* LD_1 *due to communication delays will be given by:*

$$\chi_1(s_1) = \{\sigma_f aa_s bb_s (cc_s)^p\}.$$

Notice that the projection in $\Sigma_{o_1}^s$ *of the trace in* $\chi_1(s_1)$ *is* $P_{1s}(\chi_1(s_1)) = s_{1s} = a_s b_s c_s^p$, *as expected.*

[2]Since, in this example, there exists one local diagnoser only, we omit the subscript associated with the local diagnoser number, *i.e.*, replace σ_{s_1} with σ_s.

To characterize the behavior of NDESWTS in the presence of delays of observations, we will present an algorithm for the computation of untimed finite state automaton $G_i = (X_i, \Sigma_i, f_i, \Gamma_i, x_{0_i}, \emptyset)$, formed from $NDESWTS = (G, t_{min}, T)$. In order to capture the observable event occurrences and their possible observations, the states of G_i will have two components, as follows: *(i)* the first component accounts for the corresponding state of G, and *(ii)* the second component accounts for the observable events that were generated by G in order to reach state x and whose observations are being transmitted to local diagnoser LD_i together with the minimal time elapsed between the occurrences of these observable events. For instance, let us consider a hypothetical state $(x, a\ 0.5\ b\ 0.2\ c)$ of G_i, depicted in Figure 4.6, which corresponds to the case when some plant G has reached state x after the execution of a trace $s \in L(G)$ that contains, in that order, the observable events a, b and c, whose observations are still being transmitted to the local diagnoser, and the time elapsed between the occurrence of event a and b (resp. b and c) is, at least, equal to 0.5 (resp. 0.2) t.u. It is worth remarking that, trace s can have other observable events whose observation finished before state $(x, a\ 0.5\ b\ 0.2\ c)$ was reached.

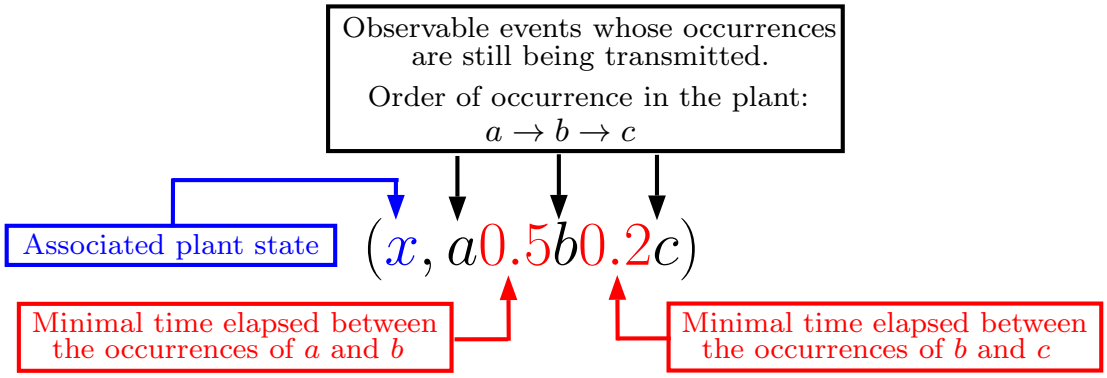


Figure 4.6: Example of a state of automaton G_i.

The construction of G_i is carried out by manipulating traces composed by events and real numbers that form the second components of the states of G_i. To this end, let $I_p = \{1, 2, \ldots, p\}$, where $p \in \mathbb{N}$, with $\mathbb{N}$ denoting the set of positive integers, and define set $\mathcal{Q}_i := \{q = q_1 q_2 \cdots q_l : \forall k \in I_\ell, (q_k \in \Sigma_{o_i}) \vee (q_k \in \mathbb{R}_+)\}$, where $\mathbb{R}_+$ denotes the set of non-negative real numbers and $i \in I_{N_s}$. With slight abuse of notation, we say that $q_p \in q$, if there exist $q', q'' \in \mathcal{Q}_i$ such that one of the following conditions in satisfied: (i) $q = q' q_p q''$; (ii) $q = q' q_p$; (iii) $q = q_p q''$; (iv) $q = q_p$. Notice that, the

elements of $\mathcal{Q}_i$ are traces formed by observable events in Σ_{o_i} and numbers in $\mathbb{R}_+$. Let us define the following operations.

Definition 4.3 *(a) The link operation is the mapping* $link : \mathcal{Q}_i \times \mathcal{Q}_i \to \mathcal{Q}_i$ *where, for every* $q = q_1 \cdots q_l$ *and* $p = p_1 \cdots p_k$ *belonging to* $\mathcal{Q}_i$,

$$link(q,p) = \begin{cases} q_1 \cdots q_{l-1}\,(q_l + p_1)\,p_2 \cdots p_k, \text{ if } q_l, p_1 \in \mathbb{R}_+ \\ q_1 \cdots q_l\, p_1 \cdots p_k, \text{ otherwise.} \end{cases} \tag{4.12}$$

(b) The cut operation is the mapping $cut : \mathcal{Q}_i \to \mathcal{Q}_i$ *where,* $\forall q = q_1 q_2 \cdots q_l \in \mathcal{Q}_i$,

$$cut(q) = \begin{cases} q_w\, q_{w+1} \cdots q_l, \text{ if } (\exists w \le l)[(q_w \in \Sigma_{o_i}) \wedge (q_j \in \mathbb{R}_+, \forall j \in \{1, ..., w-1\})] \\ 0, \text{ if } q_j \in \mathbb{R}_+, \forall j \in \{1, 2, ..., l\}. \end{cases} \tag{4.13}$$

(c) The addition operation is the mapping $add : \mathcal{Q}_i \times X \times \Sigma \to \mathcal{Q}_i$ *where, for all* $q = q_1 q_2 \cdots q_l \in \mathcal{Q}_i$, $x \in X$ *and* $\sigma \in \Sigma$,

$$add(q,x,\sigma) = \begin{cases} cut(link(q, t_{min}(x,\sigma)\sigma)), \text{ if } (\sigma \in \Sigma_{o_i}) \wedge (f(x,\sigma)!) \\ cut(link(q, t_{min}(x,\sigma))), \text{ if } (\sigma \in \Sigma_{uo_i}) \wedge (f(x,\sigma)!) \\ \text{undefined, otherwise.} \end{cases} \tag{4.14}$$

(d) The removal operation is the mapping

$$rem : \mathcal{Q}_i \times \mathbb{N} \to \mathcal{Q}_i$$

where, for all $q = q_1 q_2 \cdots q_l \in \mathcal{Q}_i$,

$$rem(q,k) = \begin{cases} cut(q_2 \cdots q_l), \text{ if } (k = 1) \\ link(q_1 \cdots q_{k-1}, q_{k+1} \cdots q_l), \text{ if } (1 < k < l) \\ cut(q_1 \cdots q_{l-1}), \text{ if } (k = l) \\ \text{undefined, otherwise.} \end{cases} \tag{4.15}$$

(e) The measurement site index function ms is the mapping

$$ms : \Sigma_{o_i} \to \{1, 2, \ldots, m\}$$

where, for all $\sigma \in \Sigma_{o_i}$,

$$ms(\sigma) = \begin{cases} j\text{: if } \sigma \in \Sigma_{o_{ij}} \text{ for some } i \in \{1, \ldots, N_s\} \\ \text{undefined, otherwise.} \end{cases} \tag{4.16}$$

(f) *The observable event index function* I_{o_i} *is the mapping*

$$I_{o_i} : \mathcal{Q}_i \times 2^{\Sigma_o} \to 2^{\mathbb{N}}$$

where, for all $q = q_1 \cdots q_l$ *and all* $\Sigma_{o_i} \subseteq \Sigma_o$,

$$I_{o_i}(q, \Sigma_{o_i}) = \{k \in \{1, \ldots, l\} : q_k \in \Sigma_{o_i}\} \tag{4.17}$$

According to Definition 4.3, function $link(q, p)$ simply concatenates two traces q and p that form the second component of the states of G_i in the usual way except when the last element of q and the first element of p are numbers, in which case they will sum up. Function $cut(q)$ either eliminates the prefix of trace q formed only by numbers before the first observable event or set q to 0 if q is formed only by numbers. Function $add(q, x, \sigma)$ adds elements to q that depends on if σ is observable or not in order to form the second component of the new state of G_i after the occurrence of σ as follows: (i) if σ is observable, it is added together with the minimal activation time; (ii) if σ is unobservable, only the minimal activation time is added. Function $rem(q, k)$ removes from q its k-th element and function $ms(\sigma)$ returns the index j which corresponds to measurement site MS_j that detects the occurrence of event σ. Finally, the observable event index function of trace q is the set of indices of all events in trace q that belong to Σ_{o_i}.

We now present Algorithm 4.1 to construct G_i. The idea behind Algorithm 4.1 is to model all changes in the order of observation of the events by local diagnoser LD_i, caused by delay in the communication channels ch_{ij}, for $j \in I_m$.

Algorithm 4.1 *Construction of automaton* G_i

Input *Networked system* $NDESWTS = (G, t_{min}, T)$, $\Sigma_{o_{ij}}$, *for all* $j \in I_m$.
Output *Automaton* $G_i = (X_i, \Sigma_i, f_i, \Gamma_i, x_{0_i}, \emptyset)$.

STEP 1. *Define the initial state as* $x_{0_i} = (x_0, 0)$ *and* $X_i = \emptyset$;

STEP 2. *Form sets* Σ_{o_i}, $\Sigma^s_{o_{ij}}$, *for all* $j \in I_m$, $\Sigma^s_{o_i}$ *and* Σ_i *according to Equations* (4.1), (4.2) *and* (4.4), *respectively;*

STEP 3. *Create a FIFO queue* F *and add* x_{0_i} *to* F;

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

STEP 4.1 $(x, q) \leftarrow Head(F)$ *and* $Dequeue(F)$*;*

STEP 4.2 $X_i \leftarrow X_i \cup \{(x, q)\}$*;*

STEP 4.3 *Let* $q = q_1\, q_2 \cdots q_l$ *and* $I_\ell = \{1, \ldots, l\}$*. Form the following set of indices:*

(a) $I_{o_i} = I_{o_i}(q, \Sigma_{o_i})$*, in accordance with Equation* (4.17)*;*

(b) $I_{t,r} = I_\ell \setminus I_{o_i}$*;*

STEP 4.4 ***For*** $\sigma \in \Gamma(x)$*:*

STEP 4.4.1 *Set* $FLAG = TRUE$

STEP 4.4.2 ***If*** $I_{o_i} \neq \emptyset$ ***then***

While $(k \in I_{o_i}) \wedge (FLAG = TRUE)$ *:*

(a) Compute

$$minet(q_k) = \begin{cases} \sum\limits_{p \in I_{t,r} \setminus I_k} q_p, \ k < \ell \\ 0, \ \textit{if } k = \ell. \end{cases}$$

(b) Compute $\rho = ms(q_k)$

(c) ***If*** $minet(q_k) + t_{min}(x, \sigma) \geq T_{i\rho}$ ***then*** *Set* $FLAG = FALSE$

STEP 4.4.3 ***If*** $FLAG = TRUE$ ***then***

Set $\tilde{x}_i = f_i((x, q), \sigma) = (f(x, \sigma), add(q, x, \sigma))$*;*

Else $\tilde{x}_i$ *not defined*

STEP 4.4.4 ***If*** $(\tilde{x}_i \notin X_i) \wedge (\tilde{x}_i \notin F) \wedge \tilde{x}_i!$, ***then*** $Enqueue(F, \tilde{x}_i)$*;*

STEP 4.5 ***For*** $j \in I_m$*:*

STEP 4.5.1 ***For*** $\sigma \in \Sigma_{o_{ij}}$*:*

(a) Form set $Y_j = \{k \in I_{o_i} : q_k \in \Sigma_{o_{ij}}\}$*.*

(b) ***If*** $Y_j \neq \emptyset$ ***then****:*

- *Compute* $\upsilon = min(Y_j)$*;*
- *Compute* $\sigma_\upsilon^s = \psi(q_\upsilon)$*, according to Equation* (4.3)*;*

- *Set* $\hat{x}_i = f_i((x,q),\sigma_v^s) = (x, rem(q,v))$*;*
- ***If*** $(\hat{x}_i \notin X_i) \wedge (\hat{x}_i \notin F)$*,* ***then*** *do enqueue*$(F,\hat{x}_i)$*;*

STEP 5. *Define* $\Gamma_i(x_i) = \{\sigma \in \Sigma_i : f_i(x_i,\sigma)$ *is defined*$\}$*,* $\forall x_i \in X_i$*;*

In Step 1 of Algorithm 1, we define the initial state of automaton G_i as $x_{0_i} = (x_0, 0)$, where x_0 is the initial state of plant G, and the second component is set as zero to determine that no observation is being transmitted to the local diagnoser LD_i at state x_{0_i}. In Step 2, we form the set of successfully observed events $\Sigma_{o_i}^s$ that are communicated to local diagnoser LD_i, and the extended set of events Σ_i. In Step 3, in order to define the other states and the transition function of G_i, we create a queue of states F, which is initially equal to $F = [x_{0_i}]$. In Step 4.1, we set state (x,q) as the first state in F and remove this state from F. In Step 4.2, we add it to set X_i and, in Step 4.3, we form sets I_{o_i} which stores the indices of the positions of the observable events in q, and $I_{t,r}$, which stores the indices of the minimal times on the right of the observable events whose indices are stored in I_{o_i}. This is important since the sum of those times will determine if a new event σ can be added to q, when the sum is smaller than the maximal delay of all events in q, or that one of the events in q must be observed by the local diagnoser. These are the ideas behind Steps 4.4 and 4.5, which are detailed as follows.

In Steps 4.4 and 4.5, we define the transitions from (x,q) and add to F only the new states reached by these transitions. In order to compute all accessible part of G_i, we repeat Step 4 until F becomes empty. At each iteration of Step 4, the new transitions, from state (x,q) are defined as follows. In Step 4.4, we define transitions from state (x,q) that correspond to new occurrences of events in the plant, and thus, they are labeled by events that are active at state x of G. To this end, in Step 4.4.1, we set a boolean flag equal to TRUE. In Step 4.4.2, if I_{o_i} is not empty then, for each index in I_{o_i}, we compute: (a) the sum of real numbers (minimal times) which are on the right of event q_k in q and (b) the measurement site index of event q_k, in accordance with Equation (4.16). In Step 4.4.2(c), if the sum of the minimal elapsed time of event q_k and the minimal time of σ is greater than the maximal communication delay of q_k then we set FLAG = FALSE. In Step 4.4.3, if FLAG is TRUE then the transition corresponding to event σ can be defined

in state (x, q), since either *(i)* there is no event in q, namely, $I_{o_i} = \emptyset$ or *(ii)* there is no observable event q_k inside q whose transmission of its observation must be finished before the occurrence of σ, *i.e.*, $\forall q_k, minet(q_k) + t_{min}(x, \sigma) < T_{i\rho}$. We then obtain the state $\tilde{x_i}$ reached by the new transitions by means of $f(x, q)$ and *add* operation. In Step 4.4.4, if $\tilde{x_i}$ does not belong to the set of events X_i and queue F, then we add $\tilde{x_i}$ to the end of queue F. In Step 4.5, we search for the events whose observation transmission can be successfully observed at state (x, q). Notice that, for each communication channel, only the first event inside q whose observation is transmitted through this channel can be successfully observed at state (x, q), since the channels are modeled by FIFO queues. In Step 4.5.1(a), we form Y_j, which is the set of all indices corresponding to events in q transmitted by the same channel ch_{ij}. If Y_j is not empty, in Step 4.5.1(b), we define transitions from state (x, q) labeled by events in $\Sigma^s_{o_i}$, *i.e.*, events corresponding to the success of the observation transmission to the diagnoser, as follows. First, we compute $\upsilon = min(Y_j)$. Second, the successful observation event of q_υ, σ^s_υ, is computed according to Equation (4.3). It is important to notice that since υ is the smaller index of Y_j, then there is no change in the order of event observations that are transmitted through the same channel, which guarantees that each channel ch_{ij} is modeled by a first-in first out (FIFO) queue. After computing σ^s_υ, we set state $\hat{x_i}$ which is reached from (x, q) through σ^s_υ by applying *rem* operation. If new state $\hat{x_i}$ does not belong to the set of events X_i and queue F, we add $\hat{x_i}$ to the end of queue F. Finally, in Step 5, we compute the set of active events, for all states of G_i.

Remark 4.2 *The observable event set of G_i is $\Sigma_{i_o} = \Sigma^s_{o_i}$ and not Σ_{o_i}, and the 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.*

Example 4.3 *Let us consider the networked discrete event system with timing structure NDESWTS $= (G, t_{min}, T)$ shown in Figures 4.7(a) and 4.7(b), where $\Sigma = \{\sigma_f, a, b, c\}$, $\Sigma_{o_1} = \{a, b, c\}$. Applying Algorithm 4.1, by using as input G, $\Sigma_{o_{11}} = \{a\}$, $\Sigma_{o_{12}} = \{b, c\}$, $T = [T_{11}\ T_{12}] = [2\ 0.1]$ and t_{min}, defined according to Figure 4.2, we obtain automaton G_i, $i = 1$, depicted in Figure 4.8, whose construction can be explained as follows. In Step 1, we define the initial state of automaton $(x_0, 0)$,*

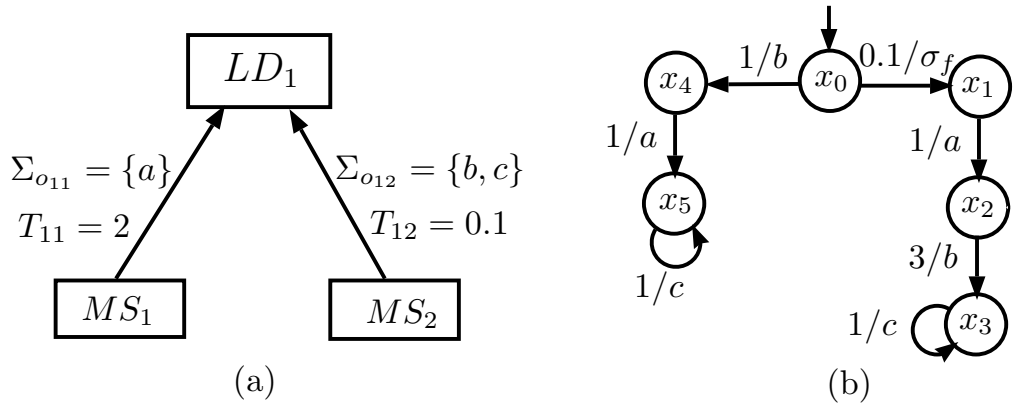


Figure 4.7: $NDESWTS = (G, t_{min}, T)$ for Examples 4.1 and 4.2, where communication delay structure (a) and automaton G with minimal time function t_{min} (b), which is considered again in Example 4.3.

where value 0 *second component indicates that no observation is being sent to the diagnoser. In Step 2, we form the sets* $\Sigma_{o_1} = \{a, b, c\}$, $\Sigma^s_{o_{11}} = \{a_s\}$, $\Sigma^s_{o_{12}} = \{b_s, c_s\}$, $\Sigma^s_{o_1} = \{a_s, b_s, c_s\}$, *and* $\Sigma_1 = \{\sigma_f, a, b, c, a_s, b_s, c_s\}$. *In Step 3, we create a queue of states* $F = [x_0, 0]$, *and thus, we repeat Step 4 until* F *becomes empty. In Steps 4.1 and 4.2, we set* $(x, q) = (x_0, 0)$, $F = [\,]$ *and* $X_1 = \{(x_0, 0)\}$. *In Step 4.3, we create the set of indices* I_{o_1} *that contains the indices of the events which belongs to* Σ_{o_1} *inside* q. *Thus,* $I_{o_1} = \emptyset$ *since* $q = 0$. *In Step 4.4 (resp. Step 4.5), we define the transitions from state* (x, q), *labeled by events in* Σ *(resp.* $\Sigma^s_{o_1}$*) associated with the occurrences of events in the plant (resp. successful observations). In Step 4.4.1, we set* $FLAG = TRUE$. *The next step, Step 4.4.2, should be skipped since* $I_{o_1} = \emptyset$. *Since* $\Gamma(x_0) = \{\sigma_f, b\}$ *and* $FLAG = TRUE$, *we define, in Step* 4.4, *two transitions from state* $(x_0, 0)$ *labeled by events* b *and* σ_f, *which define new states* (x_4, b) *and* $(x_1, 0)$, *respectively, and, thus* $F = [(x_4, b), (x_1, 0)]$. *Notice that, event* b *(resp.* σ_f*) is added (resp. not added) to the second component of the reached state since it is an observable (resp. unobservable) event. Finally, since* $I_{o_1} = \emptyset$, *then sets* Y_j, *to be formed in Step 4.5 for each communication channel* ch_{ij}, *for* $j = 1, 2$ *are also empty. Therefore, no transition will be defined in Step 4.5.*

Assume that after some iterations of Step 4, state (x_2, a) *is the first state of queue* F. *Then, in Step 4.3, we obtain* $I_{o_1} = \{1\}$ *and* $I_{t,r} = \emptyset$. *In Steps 4.4.2(a) and 4.4.2(b), we compute* $minet(a) = 0$ *and* $\rho = ms(a) = 1$, *respectively. Notice that*

even $b \in \Gamma(x_2)$*, transition labeled by* b *is not defined in state* (x_2, a)*. This so because, in Step 4.4.2(c),* $minet(a) + t_{min}(x_2, b) = 3 > T_{11} = 2$*. In Step 4.5, since* $I_{o_1} = \{1\}$ *and* $v = 1$ *(corresponding to the position of event* a *in* q*), we define the state to be reached by* (x_2, a) *as* $f_1((x_2, a), \psi_1(a)) = f_1((x_2, a), a_s) = (x_2, rem(a, 1)) = (x_2, 0)$*, and thus, transition labeled by* a_s *is defined in state* (x_2, a)*. This means that event* b *cannot occur before the observation of* a*. Let us now assume that* $(x, q) = (x_5, a)$ *at the beginning of Step 4. Then, in Step 4.3, we obtain* $I_{o_1} = \{1\}$ *and* $I_{t,r} = \emptyset$*. In Steps 4.4.2(a) and 4.4.2(b), we compute* $minet(a) = 0$ *and* $\rho = ms(a) = 1$*, respectively. Notice that transition labeled by* c *is defined in state* (x_5, a) *in Figure 4.8. This so because,* $c \in \Gamma(x_5)$ *and* $(minet(a) + t_{min}(x_5, c)) = 1 < T_{11} = 2$*. We define the state to be reached by* (x_5, a) *due to event* c *as* $(f(x_5, c), add(a, x_5, c)) = (x_5, a1c)$*. In Step 4.5, since* $I_{o_1} = \{1\}$ *and* $v = 1$*, we define the state to be reached by* (x_5, a) *as* $f_1((x_5, a), \psi_1(a)) = f_1((x_5, a), a_s) = (x_5, rem(a, 1)) = (x_5, 0)$*, and thus, transition labeled by* a_s *is defined in state* (x_5, a)*.*

To conclude the example, let us consider state $(x_5, a1c)$*. Then, in Step 4.3, we obtain* $I_{o_1} = \{1, 3\}$ *and* $I_{t,r} = \{2\}$*. In Steps 4.4.2(a) and 4.4.2(b), we compute* $minet(a) = 1$*,* $minet(c) = 0$*,* $\rho_a = 1$ *and* $\rho_c = 2$*, respectively. Notice that even* $c \in \Gamma(x_5)$*, transition labeled by* c *is not defined in state* $(x_5, a1c)$*. This so because, in Step 4.4.2(c),* $(minet(a) + t_{min}(x_5, c)) = 2 = T_{11} = 2$ *and* $(minet(c) + t_{min}(x_5, c)) = 1 > T_{12} = 0.1$*. In Step 4.5, two states can be reached by state* $(x_5, a1c)$*: (i)* $f_1((x_5, a1c), \psi_1(a)) = f_1((x_5, a1c), a_s) = (x_5, rem(a1c, 1)) = (x_5, c)$*, and (ii)* $f_1((x_5, a1c), \psi_1(c)) = f_1((x_5, a1c), c_s) = (x_5, rem(a1c, 3)) = (x_5, a1)$*.*

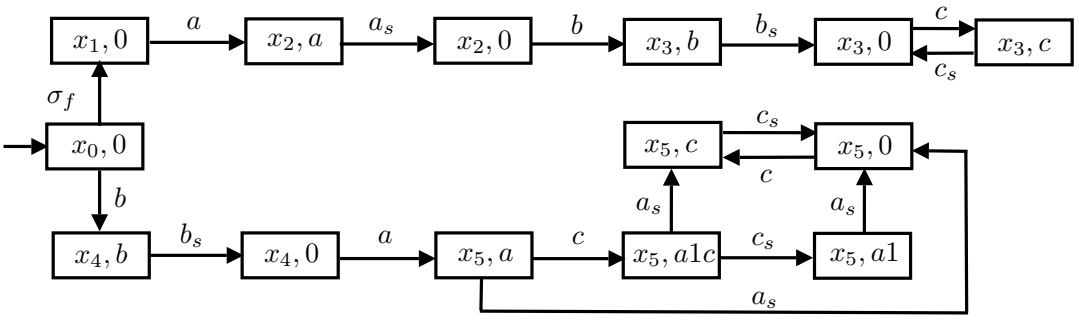


Figure 4.8: Example of automaton G_1 of Example 4.3.

Remark 4.3 *There are some simple observations we can immediately make regarding the previous example. First,* $L(G_1) = \{\sigma_f aa_s bb_s(cc_s)^p,\ bb_s aa_s(cc_s)^q,\ bb_s acc_s a_s(cc_s)^{q-1}, bb_s aca_s c_s(cc_s)^{q-1}\}$, $p, q \in \mathbb{N}$. $L(G_1)$ *is equal to* $\chi_1(s_1) \cup \chi_i(s_2)$ *by checking the conditions of Definition 4.2 for Example 4.2, where* $s_1 = \sigma_f abc^p$ *and* $s_2 = bac^p$. *Since* $L(G) = \{s_1, s_2\}$, $L(G_1) = \chi_1(L(G))$. *In this section, we will get into formal details in order to prove that the equality* $L(G_i) = \chi_i(L(G))$ *always holds.*

Second, notice that from failure trace $s_1 = \sigma_f abc^p$, *we can obtain trace* $P_{1s}(\chi_1(s_1)) = s_{1s} = a_s b_s c_s^p$ *which is actually observed by local diagnoser* LD_1. *From normal trace* $s_2 = bac^q$ *we can obtain traces* $P_{1s}(\chi_1(s_2)) = \{s_{2s}, s'_{2s}\}$, *such that* $s_{2s} = b_s a_s c_s^q$ *and* $s'_{2s} = b_s c_s a_s c_s^{q-1}$. *Thus, for inspection, we can point out that language* G_1 *is diagnosable by* LD_1, *since local diagnoser* LD_1 *will be sure of the failure occurrence* σ_f *after receiving the information of first event* a_s *(traces* s_{2s} *and* s'_{2s} *starts with event* b_s*). Formally, we will define codiagnosability of networked discrete event systems with timing structure (NDESWTS) in Section 4.3.*

The following results concern automaton G_i obtained by Algorithm 4.1.

Lemma 4.1 *Let* $w \in L(G_i)$ *and define* $(x, q) = f_i(x_{0_i}, w)$. *Then:*

(a) $x = f(x_0, P_i(w))$;

(b) $q = 0$ *if, and only if, every event* σ *in* w *that belongs to* Σ_{o_i} *has its occurrence either successfully transmitted* σ_{s_i} *in* w. *Otherwise,* $q = q_1 q_2 \cdots q_k \in \mathcal{Q}_i$, *where* $q_1 \in \Sigma_{o_i}$ *and every* $q_k \in \Sigma_{o_i}$, $k \in \{1, 2, \ldots, k\}$, *corresponds to one occurrence of event* q_k *in* w *which is still being transmitted, with* $minet(q_k)$ *(stated in Step 4.4.2(c) of Algorithm 4.1) equal to the minimal time interval elapsed since the occurrence of* q_k *in the plant.*

Proof. The proof is done by induction in the length of the traces $w \in L(G_i)$.

- *Basis step*: According to Step 1 of Algorithm 4.1, the initial state of G_i is equal to $x_{0_i} = (x_0, 0)$. Thus, for $w = \varepsilon$, $f_i(x_{0_i}, w) = (x_0, 0)$, which agrees with the facts that: $(a) f(x_0, P_i(\varepsilon)) = x_0$, and (b) there is no event in w whose occurrence has not been transmitted.

- *Induction hypothesis*: $\forall w \in L(G_i)$, such that $|w| \leq p$, $f_i(x_{0_i}, w) = (f(x_0, P_i(w)), q)$, where $q = 0$ if, and only if, every event σ in w that belongs to Σ_{o_i} has its occurrence successfully transmitted σ_{s_i} in w. Otherwise, $q = q_1 q_2 \ldots q_l \in \mathcal{Q}_i$ where $q_1 \in \Sigma_{o_i}$ and every $q_k \in \Sigma_{o_i}$, $k \in \{1, 2, \ldots, l\}$, corresponds to one occurrence of event q_k in w which is still being transmitted, with $minet(q_k)$ equal to the minimal time interval elapsed since the occurrence of q_k in the plant.

- *Inductive step*: Consider a trace $w\sigma \in L(G_i)$ such that $|w| = p$ and $\sigma \in \Sigma_i$. We will prove initially item (a) and, after that, item (b).

(a) Notice that, according to the induction hypothesis, the first component of state $f_i(x_{0_i}, w)$ is equal to $f(x_0, P_i(w))$. Let us first consider the case when $\sigma \in \Sigma$. Then, according to Step 4.4 of Algorithm 4.1, $\sigma \in \Gamma(f(x_0, P_i(w)))$ and the first component of the reached state is equal to $f(f(x_0, P_i(w)), \sigma) = f(x_0, P_i(w)\sigma) = f(x_0, P_i(w\sigma))$. Let us now consider the case when $\sigma \in \Sigma^s_{o_i}$. Since, according to Steps 4.5 of Algorithm 4.1, the transitions of G_i labeled by events in $\Sigma^s_{o_i}$ do not modify the first component of the state, the first component of $f_i(x_{0_i}, w\sigma)$ is equal to $f(x_0, P_i(w))$, which is equal to $f(x_0, P_i(w\sigma))$ since $P_i(w) = P_i(w\sigma)$.

(b) Let q denote the second component of state $f_i(x_{0_i}, w)$. Then, according to the induction hypothesis, q satisfies part (b) of the lemma statement with respect to trace w. According to Algorithm 4.1 the second component of the state reached from state $f_i(x_{0_i}, w)$ by a transition labeled by an event $\sigma_i \in \Sigma_i$ is determined as follows:

$(b1)$ If $\sigma_i = \sigma \in \Sigma$, then, according to Step 4.4, the second component of the reached state is $add(q, x, \sigma)$, where, according to Definition 4.3(a), function *add* links, to the right of q, either trace $t_{min}(x, \sigma)\sigma$, if $\sigma \in \Sigma_{o_i}$, or $t_{min}(x, \sigma)$, if $\sigma \in \Sigma_{uo_i}$, and also removes the largest prefix formed only with non-negative real numbers. In this case, the second component of the state reached from $f_i(x_{0_i}, w)$ by the transition labeled by σ will be as follows: $t_{min}(x, \sigma)$ must be added to the right of q after the occurrence of σ in the plant to enforcing that $minet(q_k)$, defined according Step

4.4.2(c) of Algorithm 4.1, be equal to the minimal time interval elapsed since the occurrence of the k-th event of the second component of the reached state; in addition, σ has to be added to q when $\sigma \in \Sigma_{o_i}$, since its occurrence has not been observed in $w\sigma$, whereas, in the case when $\sigma \in \Sigma_{uo_i}$, nothing else has to be added to q.

($b2$) If $\sigma_i = \sigma_\upsilon^s \in \Sigma_{o_i}^s$, then, according to Step 4.5 the second component of the reached state is $rem(q, \upsilon)$, where υ is the index of the first occurrence of event $\psi_i^{-1}(\sigma_\upsilon^s)$ in q and, according to Definition 4.3(d), function rem removes q_υ from q, and also removes the largest prefix formed only with non-negative real numbers. Thus, the occurrence of an event $\sigma_\upsilon^s \in \Sigma_{o_i}^s$ represents the successful of the observation of an event in w, namely, it models the successful of the observation of event q_υ stored in q. Thus, we can conclude that we must remove q_υ from q to obtain the second component of the reached state, as done by using function rem in Algorithm 4.1. In addition, notice that, functions add and rem are defined by using function cut, which removes, from $q \in \mathcal{Q}_i$, the largest prefix formed only with non-negative real numbers.

Finally, notice that in ($b1$) and ($b2$), functions add and rem guarantee that the first element of the second component of the reached state belongs to Σ_{o_i}, if the second component of the reached state has at least one element belonging to Σ_{o_i}, or that the second component of the reached state is equal to 0, otherwise. In both cases, the lemma statement holds true, and this completes the proof. ■

Based on Algorithm 4.1, we can state the following theorem related to the observation of the language generated by G_i.

Theorem 4.1 $L(G_i) = \chi_i(L(G))$.

Proof. The proof is done by induction in the length of the traces $w \in \Sigma_i^*$.

- *Basis step*: Let $w = \varepsilon$. Then, $w \in \chi_i(L(G))$ since $P_i(\varepsilon) = \varepsilon \in L(G)$ and ε satisfies Conditions 2 and 3 of Definition 4.2. In addition, we can also conclude that $w \in L(G_i)$ since the initial state of G_i is defined $((x_0, 0))$.

- *Induction hypothesis*: For all traces $w \in \Sigma_i^*$ such that $|w| \leq p$, $w \in L(G_i) \Leftrightarrow w \in \chi_i(L(G))$.

- *Inductive step*: Let $w\sigma_i \in \Sigma_i^*$ be such that $|w| = p$ and $\sigma_i \in \Sigma_i$. From Definition 4.2, if $w \notin \chi_i(L(G))$ then $w\sigma_i \notin \chi_i(L(G))$. In addition, since $L(G_i)$ is, by definition, prefix-closed, if $w \notin L(G_i)$ then $w\sigma_i \notin L(G_i)$. From induction hypothesis, we can only consider the case when $w \in L(G_i)$ and $w \in \chi_i(L(G))$. This implies, according to Definition 4.2, that there exists $s \in L(G)$ such that $s = P_i(w)$ and that w satisfies Conditions 2 and 3 of Definition 4.2. We will first consider the case when $\sigma_i = \sigma \in \Sigma$, and, in the sequel, the case when $\sigma_i = \sigma_v^s \in \Sigma_{o_i}^s$.

(*i*) $\sigma_i = \sigma \in \Sigma$: in this case, Condition 3 of Definition 4.2 is satisfied for $w\sigma$ since it is also satisfied for w. Regarding Condition 1 of Definition 4.2, notice that transitions from state $f_i(x_{0_i}, w)$ labeled by events in Σ are defined, in Step 4.4 of Algorithm 4.1, for those events that belong to $\Gamma(f(x_0, s))$ since, according to statement (a) of Lemma 4.1, the first component of state $f_i(x_{0_i}, w)$ is equal to $f(x_0, s)$. It makes sense, since $P_i(w\sigma) = s\sigma$, $w\sigma$ satisfies Condition 1 of Definition 4.2 only for trace $s\sigma$, which implies that $s\sigma$ must be in $L(G)$ for $w\sigma$ to be in $\chi_i(L(G))$, or equivalently, σ must be in $\Gamma(f(x_0, s))$. To check if $w\sigma$ satisfies Condition 2, let q denote the second component of state $f_i(x_{0_i}, w)$, and consider the problem of verifying the possibility of occurrence of event $\sigma \in \Sigma$ before the observation of one of the events belonging to Σ_{o_i} that form q. According to Step 4.4, this verification is carried out by checking if $q = 0$ or, when $q \neq 0$, by comparing, for every $q_k \in \Sigma_{o_i}$ that forms q, the minimal time elapsed since the occurrence of q_k with the delay bound of the channel that send the occurrences of q_k to the diagnoser. Thus, the transition labeled with an event $\sigma \in \Gamma(f(x_0, s))$ from state $f_i(x_{0_i}, w)$ is defined if, and only if, either $q = 0$ or, for every $q_k \in \Sigma_{o_i}$ that forms q, the delay bound $T_{i\rho}, \rho = ms(q_k)$ is higher than $minet(q_k) + t_{min}(f(x_0, P_i(w)), \sigma)$. Notice that, in accordance with statement (*b*) of Lemma 4.1, checking this condition is equivalent to verify if every event in Σ_{o_i}, that has occurred in $w\sigma$ and whose observation has not occurred, satisfies Equation 4.11. In addition, since w satisfies Condition 2, every event in w whose occurrence has been observed in w also satisfies Equation 4.11. Therefore, we can conclude that, when $\sigma \in \Sigma$, $w\sigma \in L(G_i)$, $w\sigma \in \chi_i(L(G))$.

(*ii*) $\sigma_i = \sigma_v^s \in \Sigma_{o_i}^s$: in this case, $P_i(w\sigma_v^s) = P_i(w)$, which implies that $w\sigma_v^s$ satisfies Condition 1 of Definition 4.2 for $s \in L(G)$. In addition, $w\sigma_v^s$ also satisfies Condition 2 of Definition 4.2 since it is satisfied for w. Thus, it remains to check if Condition 3 holds true for trace $w\sigma_v^s$. To do so, let us consider a transition from state $f_i(x_{0_i}, w)$, labeled by event $\sigma_v^s \in \psi_i(\Sigma_{o_i})$, carried out in Step 4.5, which is repeated for each communication channel ch_{ij}, $j = 1, 2, \ldots, m$. In Step 4.5, the set of indices Y_j is computed with respect to the second component of state $f_i(x_{0_i}, w)$, denoted by q, and set Σ_{o_i}. Notice that, in accordance with Lemma 4.1, $w\sigma_v^s$ satisfies Equation (4.10) if, and only if, $\exists \psi_i^{-1}(\sigma_v^s) \in q$. Thus, when Y_j is nonempty, index $v = min(Y_j)$, computed in Step 4.5, defines event q_v that corresponds to the first event in q whose occurrence is sent through channel ch_{ij}. As a consequence, $\psi_i(q_v)$ is the only one event in $\psi_i(\Sigma_{o_i})$ such that $w\psi_i(q_v)$ satisfies Equations (4.10) and (4.11), and, according to Step 4.5, it is also the unique event in $\psi_i(\Sigma_{o_i})$ that is used to create a new transition from state $f_i(x_{0_i}, w)$. Therefore, it can be concluded that, when $\sigma_i = \sigma_v^s \in \Sigma_{o_i}^s$, $w\sigma_i \in L(G_i) \Leftrightarrow w\sigma_i \in \chi_i(L(G))$, which completes the proof. ■

Remark 4.4 *(Complexity to compute automaton G_i) Given a networked discrete event system with timing structure $NDESWTS = (G, t_{min}, T)$ let us define the following variables: $T' = max\{T_{ij} : T_{ij} \in \mathbb{R}_+^*\}$ and $t = min(t_{min}(x, \sigma))$, for all $x \in X$, $\sigma \in \Sigma$, and $\mathcal{T} = max\{z \in \mathbb{Z} : z < T'/t\}$, where $\mathbb{Z}$ is the set of integer numbers. In order to compute the maximal number of states of G_i, let us consider that $link(q, p)$ only concatenates q and p. Namely, it does not add the last element of q with the first element of p when they are real numbers, and thus, every state of G_i has the following form:*

(*i*) *either the form (x, q_0) where $x \in X$ and $q_0 \in \Sigma_{o_i} \cup \{0\}$;*

(*ii*) *or the form $(x, q_0 e_1 e_2 \ldots e_k)$ where $x \in X$, $q_0 \in \Sigma_{o_i}$ and, for $r = 1, 2, \ldots, k$, either e_r is a real number (the minimal activation time that corresponds to either an event in Σ_{uo_i} or an event in Σ_{o_i} whose transmission has been finished), or e_r is a real number concatenated with an event belonging to Σ_{o_i} (which corresponds to a minimal activation time and its associated event in Σ_{o_i} whose occurrence is still being sent).*

In the worst case, G_i will have $|X|(|\Sigma_{o_i}|+1)$ states for (i) and $|X|^2|\Sigma_{o_i}|(|\Sigma_{o_i}|+|\Sigma|)^k$ states with the form (ii). In addition, by assuming that all minimal activation times are equal to t and all maximal observation delays are equal to $\mathcal{T}$, it can be seen that $k \leq \mathcal{T}$ since, if $k > \mathcal{T}$, then the maximal observation delay of event q_0 is violated. Thus, we can conclude that, in the worst case:

$$|X|(|\Sigma_{o_i}|+1) + |X|^2|\Sigma_{o_i}| \sum_{k=1}^{\mathcal{T}} (|\Sigma_{o_i}|+|\Sigma|)^k.$$

Therefore, X_i is:

$$O\left(|X|^2|\Sigma_{o_i}|(|\Sigma_{o_i}|+|\Sigma|)^{\mathcal{T}+1}\right).$$

Finally, it is worth noting that the number of states in the worst case may only decrease if we compute $link(q,p)$ as determined by Definition 4.3, i.e., by adding the last element of q with the first element of p when they are real numbers.

4.2.2 An Equivalent Untimed Model of NDESWTS Subject to Communication Delays and Intermittent Loss of Observations

We will now extend the model developed in Algorithm 4.1 to also take into account the intermittent loss of observation of events in the communication channels. In order to do so, we will use the dilation function introduced in [54].

Let us partition the set of observable events associated with diagnoser LD_i as $\Sigma_{i_o} = \Sigma_{i,ilo} \dot{\cup} \Sigma_{i,nilo}$, 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 $\Sigma^s_{i,ilo} = \psi^{-1}(\Sigma_{i,ilo})$ and $\Sigma^s_{i,nilo} = \psi^{-1}(\Sigma_{i,nilo})$, where ψ^{-1} denotes the inverse function of ψ. 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:

$$\Sigma^s_{o_i} = \Sigma^s_{i,ilo} \dot{\cup} \Sigma^s_{i,nilo} \tag{4.18}$$

where the events of $\Sigma^s_{i,ilo}$ and $\Sigma^s_{i,nilo}$ denote the successful transmission to diagnoser LD_i of the events of $\Sigma_{i,ilo}$ and $\Sigma_{i,nilo}$, respectively. Let us now define the following sets: *(i)* the set of unobservable events that models the intermittent loss of observation of events $\sigma_s \in \Sigma^s_{i,ilo}$ as $\Sigma^{s'}_{i,ilo} = \{\sigma'_s : \sigma_s \in \Sigma^s_{i,ilo}\}$ and; *(ii)* set

$\Sigma'_i = \Sigma_i \cup \Sigma^{s'}_{i,ilo}$. The dilation function $D_{s_i} : \Sigma^*_i \to 2^{(\Sigma'_i)^*}$ is defined in a recursive way as: $D_{s_i}(\varepsilon) = \{\varepsilon\}$; $D_{s_i}(\sigma) = \{\sigma\}$, if $\sigma \in \Sigma_i \setminus \Sigma^s_{i,ilo}$, $D_{s_i}(\sigma) = \{\sigma, \sigma'\}$, if $\sigma \in \Sigma^s_{i,ilo}$; $D_{s_i}(s_i\sigma) = D_{s_i}(s_i)D_{s_i}(\sigma), \forall s_i \in \Sigma^*_i, \forall \sigma \in \Sigma_i$. 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 which models both, all possible ordering of observation of events $\sigma \in \Sigma_{o_i}$ due to communication delays and the intermittent loss of observation of events $\sigma \in \Sigma_{i,ilo}$. This automaton will be defined as follows:

$$G'_i = (X_i, \Sigma'_i, f'_i, \Gamma'_i, x_{0_i}, \emptyset), \tag{4.19}$$

where $\Gamma'_i(x_i) = D_{s_i}[\Gamma_i(x_i)]$, $\forall x_i \in X_i$, and $f'_i(x_i, \sigma') = f_i(x_i, \sigma)$, if $\sigma' \in \Sigma^{s'}_{i,ilo}$, and $f'_i(x_i, \sigma) = f_i(x_i, \sigma)$, if $\sigma \in \Sigma_i \setminus \Sigma^{s'}_{i,ilo}$. Notice that, if $\Sigma_{i,ilo} = \emptyset$, $G'_i = G_i$, which implies that $D_{s_i}[L(G_i)] = L(G_i)$. It is worth remarking that the set of observable events of G'_i is $\Sigma'_{i_o} = \Sigma^s_{o_i}$ and the set of unobservable events of G'_i is $\Sigma'_{i_{uo}} = \Sigma \cup \Sigma^{s'}_{i,ilo}$. Regarding the events of G'_i, the following projections can be defined.

$$P'_{is} := \Sigma'^*_i \to \Sigma^{s*}_{o_i}. \tag{4.20}$$

$$P'_i := \Sigma'^*_i \to \Sigma^*. \tag{4.21}$$

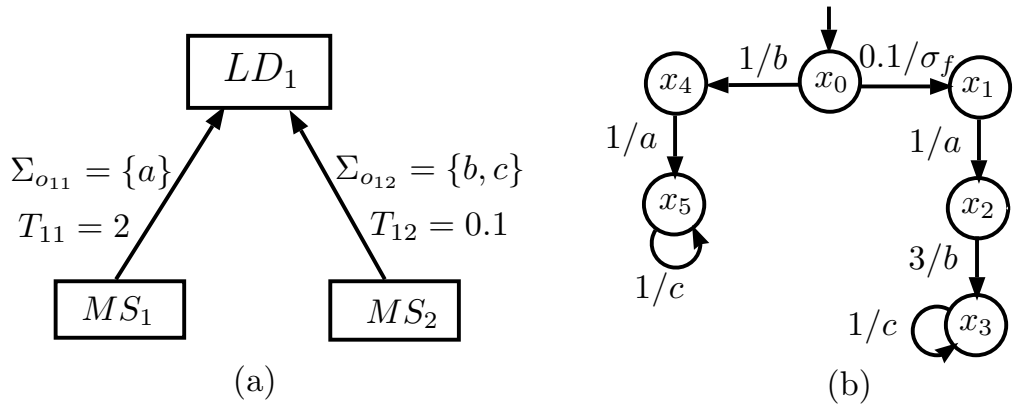


Figure 4.9: $NDESWTS = (G, t_{min}, T)$ for Examples 4.1–4.3, where communication delay structure (a) and automaton G with minimal time function t_{min} (b), which is considered again in Example 4.4.

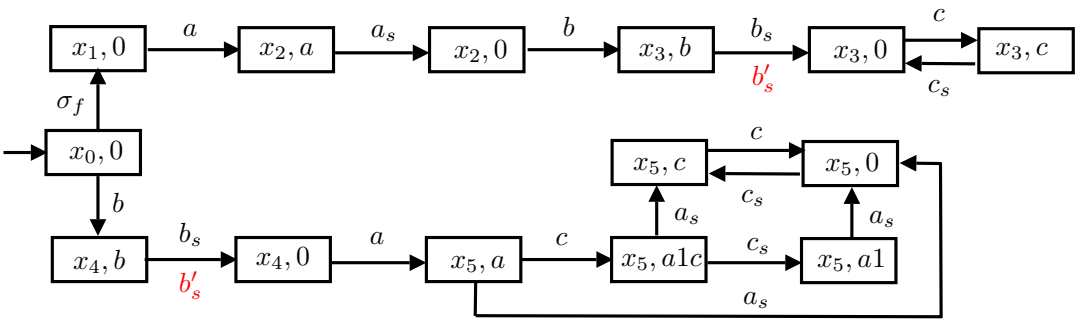


Figure 4.10: Example of automaton G'_1 of Example 4.4.

After giving some intuition diagnosability of NDESWTS to the reader, we can now formally state the concept in next section.

Example 4.4 *Let us consider again the networked discrete event system with timing structure NDESWTS* $= (G, t_{min}, T)$ *shown in Figures 4.9(a) and 4.9(b), where* $\Sigma = \{\sigma_f, a, b, c\}$, $\Sigma_{o_1} = \{a, b, c\}$ *and* G_1 *obtained in Example 4.3, depicted in Figure 4.8. Assume that* $\Sigma^s_{1,ilo} = \{b_s\}$ *is the set of events subject to intermittent loss of observation, and* $\Sigma^s_{1,nilo} = \{a_s, c_s\}$ *is the set of events that are not subject to intermittent loss of observation. Thus, we can form the set of unobservable events that model the intermittent loss of observation as* $\Sigma^{s'}_{1,ilo} = \{b'_s\}$. *We can then obtain automaton* G'_1, *depicted in Figure 4.10, by applying the dilation function over the language of automaton* G_1, *depicted in Figure 4.8. Notice that automaton* G'_1 *is formed by adding to* G_1 *transitions labeled by* b'_s *in parallel with the transitions labeled by* b_s.

Regarding traces observed by local diagnoser LD_1, *we can make the following observation. If we apply dilation function to failure trace* $s_1 = \sigma_f abc \in L(G)$, *we obtain* $D_{s_1}(\chi_1(s_1)) = \{\sigma_f aa_s bb_s(cc_s)^p, \sigma_f aa_s bb'_s(cc_s)^p, \}$, $p \in \mathbb{N}$, *such that trace* $sd_1 = \sigma_f aa_s bb'_s(cc_s)^p$ *represents the trace with lost of observation of event* b. *This trace has the following projection over the set* $\Sigma^s_{o_1}$:

$$P'_{1s}(sd_1) = a_s c_s^p.$$

As consequence, only events a_s *and* c_s *can be observed by local diagnoser* LD_1 *for this trace. On the other hand, if we apply dilation function to normal trace* $s_2 = \sigma_f abc \in L(G)$, *we obtain* $D_{s_1}(\chi_1(s_2)) = \{bb_s aa_s(cc_s)^p$, $bb'_s aa_s(cc_s)^p, bb_s acc_s a_s(cc_s)^{p-1}, bb'_s acc_s a_s(cc_s)^{p-1}, bb_s aca_s c_s(cc_s)^{p-1}, bb'_s aca_s c_s(cc_s)^{p-1}\}$.

Notice that there exists a trace $sd_2 = bb'_s aa_s(cc_s)^p \in D_{s_1}(\chi_1(s_2))$ *that has the following projection over the set* $\Sigma^s_{o_1}$*:*

$$P'_{1s}(sd_2) = a_s c_s^p = P'_{1s}(sd_1).$$

Thus, since the information of the occurrence of event b *may be lost (modeled by event* b'_s*), diagnoser* LD_1 *does not make sure whether failure* σ_f *occurs or not. Therefore,* $L(G'_1)$ *is not diagnosable, for inspection.*

4.3 NDESWTS Codiagnosability

As seen in previous section, change of order of observation of a trace $s \in L(G)$ due to communication delays to change the diagnosability decision. This is formalized in the following definition.

Definition 4.4 *A prefix-closed language* L*, generated by* G*, is said to be NDESWTS codiagnosable with respect to* χ_i*,* D_{s_i}*, and* P'_{is}*, for* $i = 1, \ldots, N_s$*, and* Σ_f *if:*

$$(\exists z \in \mathbb{N})(\forall s \in \Psi(\Sigma_f))(\forall t \in L/s, |t| \geq z) \Rightarrow$$
$$(\exists i \in \{1, \ldots, N_s\})(\forall \omega \in P'^{-1}_{is}(P'_{is}[D_{s_i}(\chi_i(st))]) \cap L)(\Sigma_f \in \omega_i).$$

□

According to Definition 4.4, language L is not NDESWTS codiagnosable if there exists a failure trace s and an arbitrarily long length trace t, such that there exist traces $s_i t_i \in D_{s_i}(\chi_i(st))$, $i = 1, 2, \ldots, N_s$, where $s_i t_i$ is not necessarily different from $s_j t_j$ for $i, j \in \{1, 2, \ldots, N_s\}$ and $s_{i_N} \in D_{s_i}(\chi_i(\omega_i))$, with $\Sigma_f \notin \omega_i$, satisfying $P'_{is}(s_i t_i) = P'_{is}(s_{i_N})$, for all $i \in \{1, \ldots, N_s\}$. This means that a language L is not NDESWTS codiagnosable if there exist a failure trace st, with arbitrarily long length after the occurrence of the failure event, and normal traces ω_i, for $i = 1, \ldots, N_s$, such that, the change in the order of observation and the loss of observation of events create ambiguous observations in all local diagnosers.

We now present necessary and sufficient conditions for network codiagnosability and propose two tests to verify this property: the first one based on diagnosers, and a second one, based on verifiers.

4.3.1 NDESWTS Diagnoser

Centralized case

To check the diagnosability of NDESWTS discrete event systems with timing structure, we use diagnoser developed in Section 3.1 by applying Algorithm 3.1 with the input G_i (without loss of observation) or G'_i (with events subject to loss of observation). Since $G'_i = G_i$ if $\Sigma_{i,ilo} = \emptyset$, *i.e.*, G_i is particular case of G'_i, from now on, we will always assume, without loss of generality, G'_i as input of algorithms in order to deal with (co)diagnosability of networked discrete event systems with timing structure. Thus, let us instead rewrite Equation (3.1) as follows.

$$G'_{scc_i} = G'_{d_i} || G'_{\ell_i}, \tag{4.22}$$

where $G'_{d_i} = obs(G'_{\ell_i}, \Sigma^s_{o_i})$ and $G'_{\ell_i} = G'_i || A_\ell$.

Therefore, we can rewrite Theorem 3.1 as follows.

Theorem 4.2 *The language L generated by automaton G is NDESWTS diagnosable with respect to χ_i, P'_{is}, D_{s_i} and $\Sigma_f = \{\sigma_f\}$ if, and only if, G'_{scc_i}, $i \in \{1, \ldots, N_s\}$ does not have strongly connected components formed with states (x'_{d_i}, x'_{ℓ_i}), such that x'_{d_i} is uncertain and x'_{ℓ_i} is an Y-labeled state, i.e., $x'_{\ell_i} = (x'_i, Y)$, where $x'_i \in X'_i$.*

Proof: The proof is straightforward from Theorem 3.1 by considering G'_i as input of Algorithm 3.1, such that $\Sigma^s_{o_i}$ is the set of observable events $\Sigma \dot{\cup} \Sigma^{s'}_{i,ilo}$ is the set of unobservable events. □

Remark 4.5 *It is worth remarking that Facts 3.1 and 3.2 are still valid, i.e., $L(G'_{scc_i}) = L(G'_{\ell_i}) = L(G'_i)$ and for every state (x'_{d_i}, x'_{ℓ_i}) of G'_{scc_i}, $x'_{\ell_i} \subseteq x'_{d_i}$.*

The following example illustrates the application of Algorithm 3.1 to check the diagnosability of networked discrete event systems with timing structure.

Example 4.5 *Let us consider again the networked discrete event system with timing structure $NDESWTS = (G, t_{min}, T)$ shown in Figures 4.2(a) and 4.2(b), where $\Sigma = \{\sigma_f, a, b, c\}$, $\Sigma_{o_1} = \{a, b, c\}$, such that $\Sigma^s_{o_1} = \{a_s, b_s, c_s\}$. First, we consider the case without observation losses. As a consequence, $\Sigma_{1,ilo} = \emptyset$ and $G'_1 = G_1$. From G_1, depicted in Figure 4.8, we can sequentially obtain: (i) automaton $G_{\ell_1} = G_1 || A_\ell$,*

depicted in Figure 4.11; (ii) diagnoser automaton $G_{d_1} = obs(G_{\ell_1}, \Sigma^s_{o_1})$*, depicted in Figure 4.11; (iii) and finally, automaton* $G_{scc_1} = G_{d_1}||G_{\ell_1}$*, depicted in Figure 4.13. According to Theorem 4.2, the language L generated by automaton G is NDESWTS diagnosable with respect to* χ_1*,* P_{1s}*, and* $\Sigma_f = \{\sigma_f\}$ *since* G_{scc_1} *does not have strongly connected components formed with states* (x_{d_1}, x_{ℓ_i})*, such that* x_{d_1} *is uncertain and* x_ℓ *is an Y-labeled state. Notice that we also obtained the same conclusion, for inspection, based on the language of* G_1 *in Example 4.3. Let us consider now, as Example 4.4,* $\Sigma_{1,ilo} = \{b\}$*, i.e., the observation of event b is subject to loss. Thus, from* G'_1*, depicted in Figure 4.10, we can obtain automaton* $G'_{\ell_1} = G'_1||A_\ell$*, depicted in Figure 4.14 and diagnoser automaton* $G'_{d_1} = obs(G'_{\ell_1}, \Sigma^s_{o_1})$*, depicted in Figure 4.14. Automaton* G'_{scc_1} *can be obtained by performing the parallel composition between* G_{d_1} *and* G_{ℓ_1}*. Since 25 states of* G'_{scc_1} *are labeled by a lot of states of* G'_1*, we only shows, in Figure 4.16, the path of automaton* G'_{scc_1} *that contains a strongly connected component formed by states* $(\{(x_5,0)N, (x_5,c)N, (x_3,0)Y, (x_3,c)Y\}, (x_3,0)Y)$ *and* $(\{(x_5,0)N, (x_5,c)N, (x_3,0)Y, (x_3,c)Y\}, (x_3,c)Y)$*, where* x_d *is uncertain and* x_ℓ *is an Y-labeled state. Therefore, L is not NDESWTS diagnosable with respect to* χ_1*,* P'_{1s}*,* D_{s_1} *and* $\Sigma_f = \{\sigma_f\}$*.*

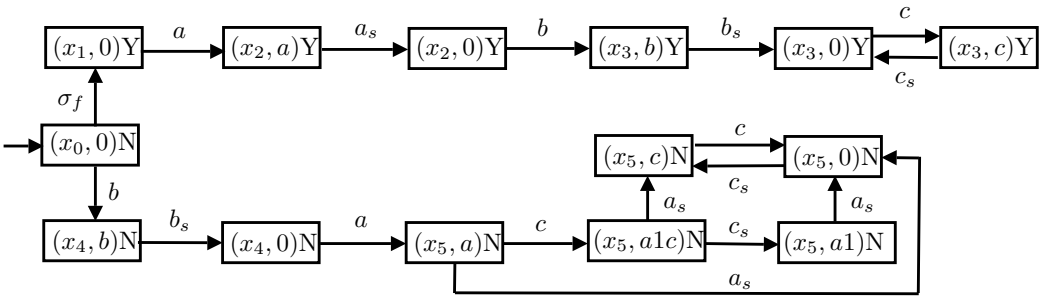


Figure 4.11: Automaton G_{ℓ_1} of Example 4.5.

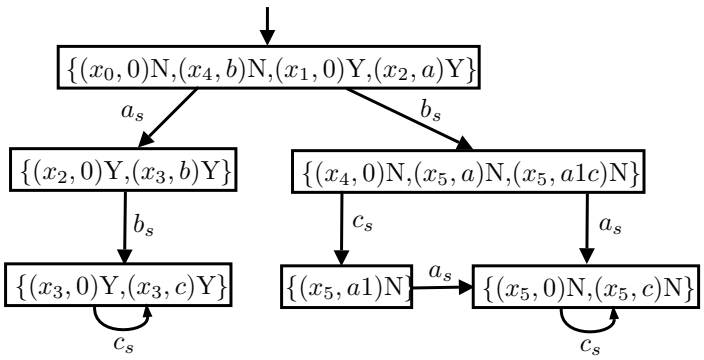


Figure 4.12: Automaton G_{d_1} of Example 4.5.

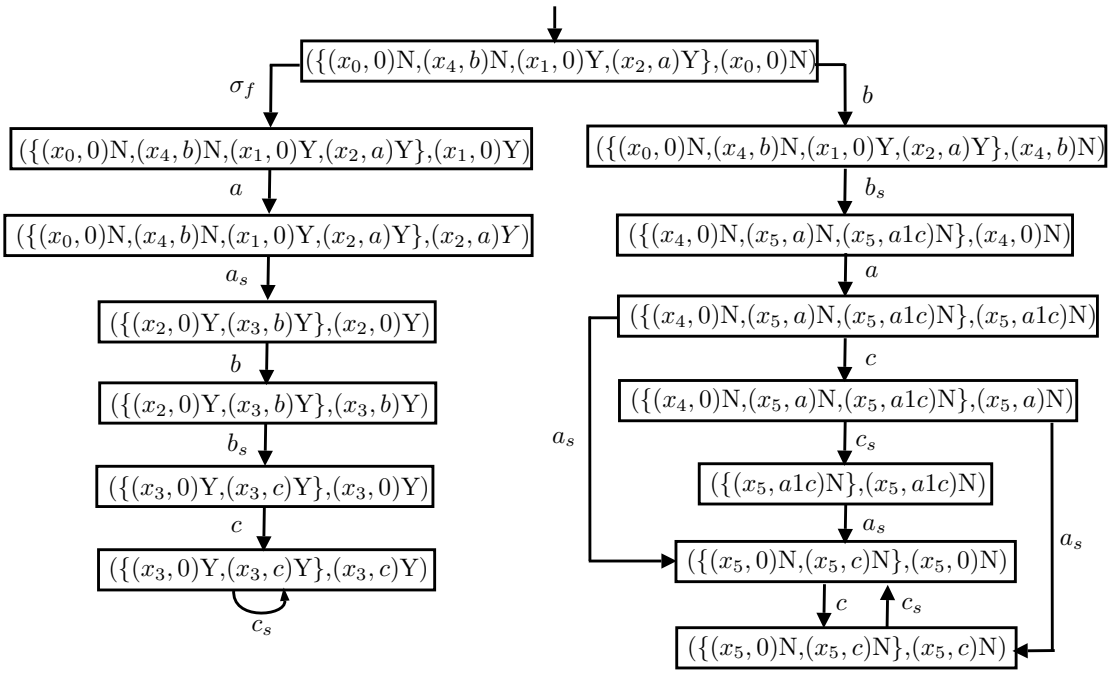


Figure 4.13: Automaton G_{scc_1} of Example 4.5.

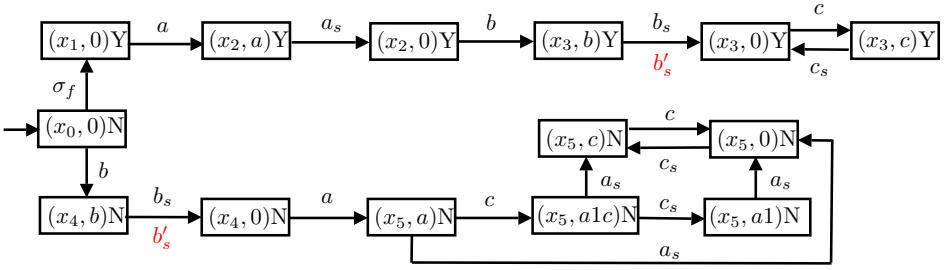


Figure 4.14: Automaton G'_{ℓ_1} of Example 4.5.

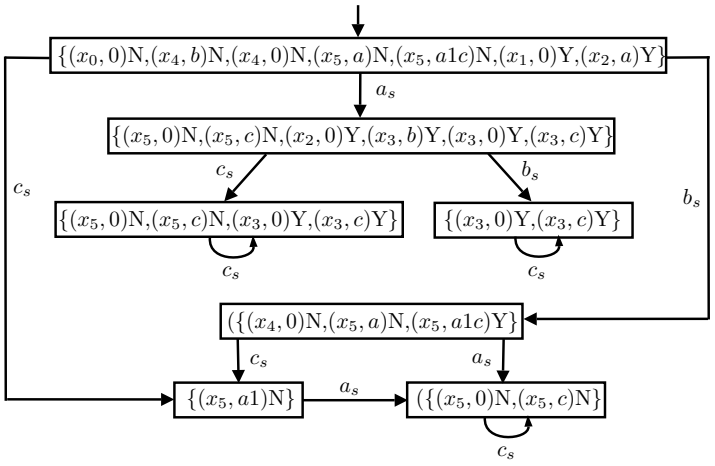


Figure 4.15: Automaton G'_{d_1} of Example 4.5.

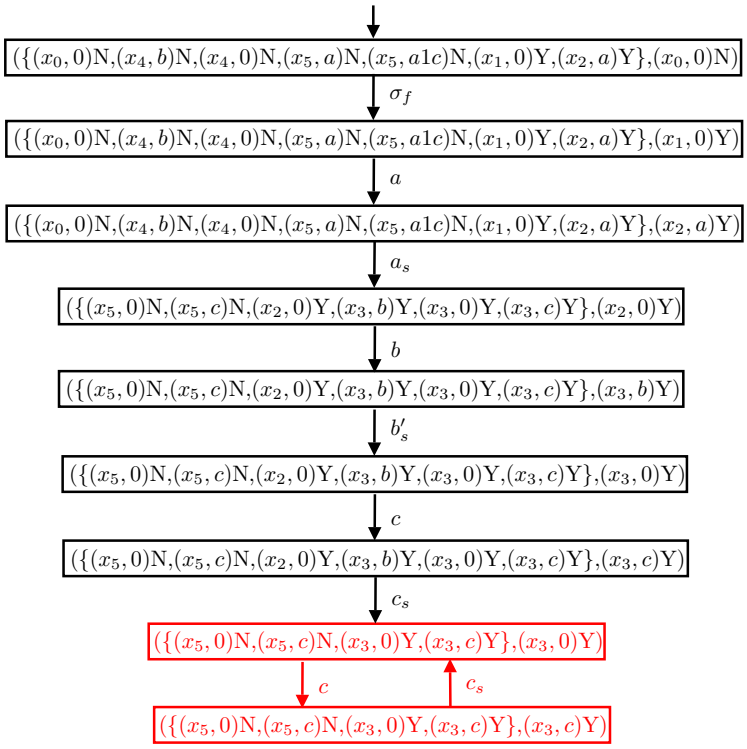


Figure 4.16: A path of automaton G'_{scc_1} of Example 4.5 that contains a strongly connected component where x_d is uncertain and x_ℓ is an Y-labeled state.

Decentralized case

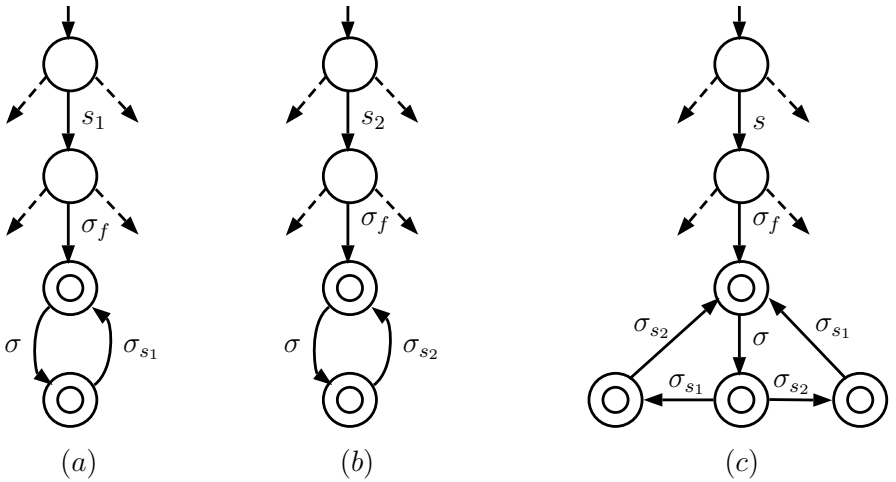
We will now address the codiagnosability of networked discrete event systems with timing structure. Let us give some intuition before formally stating the standard algorithm based on diagnoser developed in Section 3.1. To this end, consider networked discrete event system with timing structure $NDESWTS = (G, t_{min}, T)$ such that its NDESWTS architecture has, without loss of generality, two local diagnosers LD_1 and LD_2. In order to simplify the analysis, let us consider the system without losses of observation and assume now that we compute G_1 and G_2 according to Algorithm 4.1. We then apply Equation (4.22) to obtain G_{scc_1} and G_{scc_2}. Suppose that from Theorem 4.2, we verify that L is not NDESWTS diagnosable for both LD_1 and LD_2. Therefore, we can conclude that:

(i) there exists a trace $st \in L(G)$, where s is a failure trace and t is an arbitrarily long length trace, such that there exist traces $s_1\sigma_f t_1 \in D_{s_1}(\chi_1(st))$, and $s_{1_N} \in D_{s_1}(\chi_1(\omega_1))$, with $\Sigma_f \notin \omega_1$, satisfying $P'_{1s}(s_1\sigma_f t_1) = P'_{1s}(s_{1_N})$. According to Theorem 4.2, there exists a strongly connected component in automaton

G_{scc_1} formed with states (x_{d_1}, x_{ℓ_1}), where x_{d_1} is uncertain and x_{ℓ_1} is an Y-labeled state. In addition, assume that we mark the states that form this strongly connected component. An example of trace $s_1\sigma_f t_1 \in L(G_{scc_1})$, where $t_1 = (\sigma\sigma_{s_1})^p$ and $p \in \mathbb{N}$ is depicted in Figure 4.17(a). Dashed line arrows represent other possible traces of $L(G_{scc_1})$.

(*ii*) there exists a trace $s't' \in L(G)$, where s' is a failure trace and t' is an arbitrarily long length trace, such that there exist traces $s_2\sigma_f t_2 \in D_{s_2}(\chi_2(s't'))$, and $s_{2_N} \in D_{s_2}(\chi_2(\omega_2))$, with $\Sigma_f \notin \omega_2$, satisfying $P'_{2s}(s_2\sigma_f t_2) = P'_{2s}(s_{2_N})$, where $P_1 : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_1^*$ and $P_2 : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_2^*$. According to Theorem 4.2, there exists a strongly connected component in automaton G_{scc_2} formed with states (x_{d_2}, x_{ℓ_2}), where x_{d_2} is uncertain and x_{ℓ_2} is an Y-labeled state. In addition, assume that we mark the states that form this strongly connected component. An example of trace $s_2\sigma_f t_2 \in L(G_{scc_2})$, where $t_2 = (\sigma\sigma_{s_2})^p$ and $p \in \mathbb{N}$ is depicted in Figure 4.17(b). Dashed line arrows represent other possible traces of $L(G_{scc_2})$.

Now, suppose that $st = s't'$, *i.e.*, there exists a failure trace generated by plant G such that neither local diagnoser LD_1 nor LD_2 can diagnose the failure occurrence in trace st. This implies that L is not NDESWTS codiagnosable. As a consequence, there exists a strongly connected component in automaton $G_{scc_1}||G_{scc_2}$ formed with marked states $(x_{d_1}, x_{\ell_1}, x_{d_2}, x_{\ell_2})$, where x_{d_1} is uncertain, x_{ℓ_1} is an Y-labeled state,x_{d_2} is uncertain and x_{ℓ_2} is an Y-labeled state. The reason is explained as follows. Since $L(G_{scc_i}) = L(G_i)$, common events between $\Sigma_1 = \Sigma \cup \Sigma^s_{o_1}$ and $\Sigma_2 = \Sigma \cup \Sigma^s_{o_2}$ are the set of plant events Σ. As we mentioned Section 2.1.4, the parallel composition between automata synchronize the common events and allows the execution of private events, and thus, the traces of G_{scc_1} and G_{scc_2} that reach a strongly connected components formed by marked states also will be synchronized. Taking a look at Figure 4.17(c), we can see traces $s\sigma_f(\sigma\sigma_{s_1}\sigma_{s_2})^p$ and $s\sigma_f(\sigma\sigma_{s_2}\sigma_{s_1})^p$ of $G_{scc_1}||G_{scc_2}$, where $s \in P_1^{-1}[L(G_{scc_1})] \cap P_2^{-1}[L(G_{scc_2})]$. Notice that, since states of a parallel composition $G_{scc_1}||G_{scc_2}$ are marked only if are composed from marked states in both automata G_{scc_1} and G_{scc_2}, strongly connected components in automaton $G_{scc_1}||G_{scc_2}$ with marked states are the form $(x_{d_1}, x_{\ell_1}, x_{d_2}, x_{\ell_2})$, where x_{d_1} is uncertain, x_{ℓ_1} is an Y-labeled state, x_{d_2} is uncertain and x_{ℓ_2} is an Y-labeled state.



(a) (b) (c)

Figure 4.17: Hypothetical traces of G_{scc_1} (a); G_{scc_2} (b) and $G_{scc_1}||G_{scc_2}$ (c).

Another interesting aspect to consider is that the inverse direction is intuitive, *i.e.*, if $G_{scc_1}||G_{scc_2}$ has strongly connected components with marked states, x_{d_1}, x_{ℓ_1} and x_{d_2}, x_{ℓ_2}) are marked states of G_{scc_1} and G_{scc_2}, respectively, then L is not diagnosable with respect to local diagnosers LD_1 and LD_2 by the same failure trace. Therefore, L is not NDESWTS codiagnosable.

NDESWTS codiagnosability verification by using diagnoser can be formalized by the following algorithm.

Algorithm 4.2 *NDESWTS codiagnosability verification using diagnoser*

Input *Automaton* $G'_i = (X_i, \Sigma'_i, f'_i, \Gamma'_i, x_{i_0})$, *for* $i = 1, \ldots, N_s$.

Output *NDESWTS codiagnosability decision: Yes or No.*

STEP 1. *Compute automata* $G'_{scc_i} = G'_{d_i}||G'_{\ell_i}$, *for* $i \in \{1, \ldots, N_s\}$ *according to Equation* (4.22).

STEP 2. *Mark all strongly connected components of* G'_{scc_i}, $i \in \{1, \ldots, N_s\}$, *formed with states* (x'_{d_i}, x'_{ℓ_i}), *such that* x'_{d_i} *is uncertain and* x'_{ℓ_i} *is an Y-labeled state, i.e.,* $x'_{\ell_i} = (x'_i, Y)$, *where* $x'_i \in X'_i$.

STEP 3. *Compute automaton* $G^{NET}_{scc} = ||_{i=1}^{N_s} G'_{scc_i}$.

STEP 4. *Verify if there exists at least one strongly connected component formed with marked states in* G^{NET}_{scc}.

STEP 5. *If the answer is yes, then language L generated by automaton G is not NDESWTS codiagnosable with respect to χ_i, P'_{is}, D_{s_i}, $i = 1, \ldots, N_s$ and $\Sigma_f = \{\sigma_f\}$. Otherwise, L is NDESWTS codiagnosable.*

From Algorithm 4.2, the following theorem may be stated.

Theorem 4.3 *The language L generated by automaton G is NDESWTS codiagnosable with respect to χ_i, P'_{is}, D_{s_i}, $i = 1, \ldots, N_s$, and $\Sigma_f = \{\sigma_f\}$ if, and only if, automaton $G^{NET}_{scc} = ||_{i=1}^{N_s} G'_{scc_i}$ does not have strongly connected components formed with states of type $(x'_{d_1}, x'_{\ell_1}), (x'_{d_2}, x'_{\ell_2}), \ldots, (x'_{d_{N_s}}, x'_{\ell_{N_s}})$, such that x'_{d_i} is uncertain and x'_{ℓ_i} is an Y-labeled state, i.e., $x'_{\ell_i} = (x'_i, Y)$, where $x'_i \in X'_i$, for all $i \in \{1, \ldots, N_s\}$.*

Proof. The proof is similar to proof of Theorem 3.2 and it is presented as follows. ($\Rightarrow$) Let us assume that there exists a strongly connected component formed with states $(x'_{d_1}, x'_{\ell_1}), (x'_{d_2}, x'_{\ell_2}), \ldots, (x'_{d_{N_s}}, x'_{\ell_{N_s}})$, such that x'_{d_i} is uncertain and x'_{ℓ_i} is an Y-labeled state, *i.e.*, $x'_{\ell_i} = (x'_i, Y)$, where $x'_i \in X'_i$, for all $i \in \{1, \ldots, N_s\}$.

Since $G^{NET}_{scc} = ||_{i=1}^{N_s} G'_{scc_i}$, $L(G^{NET}_{scc}) = \bigcap_{i=1}^{N_s} P_i'^{-1} L(G'_{scc_i})$, such that $P'_i : \bigcup_{i=1}^{N_s} \Sigma_i'^* \rightarrow \Sigma_i'^*$, for $i \in \{1, 2, \ldots, N_s\}$. Thus, by construction, there exists a trace $s'_i \in G'_{scc_i}$, $i = 1, 2, \ldots, N_s$ that reaches a strongly connected component formed with states of type (x'_{d_i}, x'_{ℓ_i}), such that x'_{d_i} is uncertain and x'_{ℓ_i} is an Y-labeled state, *i.e.*, $x'_{\ell_i} = (x'_i, Y)$, where $x'_i \in X'_i$. Therefore, according to Theorem 4.2, L is not NDESWTS diagnosable with respect to χ_i, P'_{is}, D_{s_i} and $\Sigma_f = \{\sigma_f\}$, $i = \{1, \ldots, N_s\}$.

($\Leftarrow$) Assume now that the language L generated by automaton G is not NDESWTS codiagnosable with respect to χ_i, P'_{is}, D_{s_i}, $i = 1, \ldots, N_s$, and $\Sigma_f = \{\sigma_f\}$. Thus, according to Definition 4.4, there exists a failure trace s and an arbitrarily long length trace t, such that there exist traces $s_i t_i \in D_{s_i}(\chi_i(st))$, $i = 1, 2, \ldots, N_s$, and $s_{i_N} \in D_{s_i}(\chi_i(\omega_i))$, with $\Sigma_f \notin \omega_i$, satisfying $P'_{is}(s_i t_i) = P'_{is}(s_{i_N})$, for all $i \in \{1, \ldots, N_s\}$. As a consequence, the language L generated by automaton G is NDESWTS diagnosable with respect to χ_i, P'_{is}, D_{s_i} and $\Sigma_f = \{\sigma_f\}$. In addition, since $L(G'_{scc_i}) = L(G'_i)$ (Remark 4.5), $s_i t_i \in L(G'_{scc_i})$. According Theorem 4.2, automata G'_{scc_i}, $i \in \{1, \ldots, N_s\}$ have strongly connected components formed with states (x'_{d_i}, x'_{ℓ_i}), such that x'_{d_i} is uncertain and x'_{ℓ_i} is an Y-labeled state, *i.e.*, $x'_{\ell_i} = (x'_i, Y)$, where $x'_i \in X'_i$. Since trace $s_i t_i \in L(G'_{scc_i})$ and $P'_i(s_i t_i) = st$, where

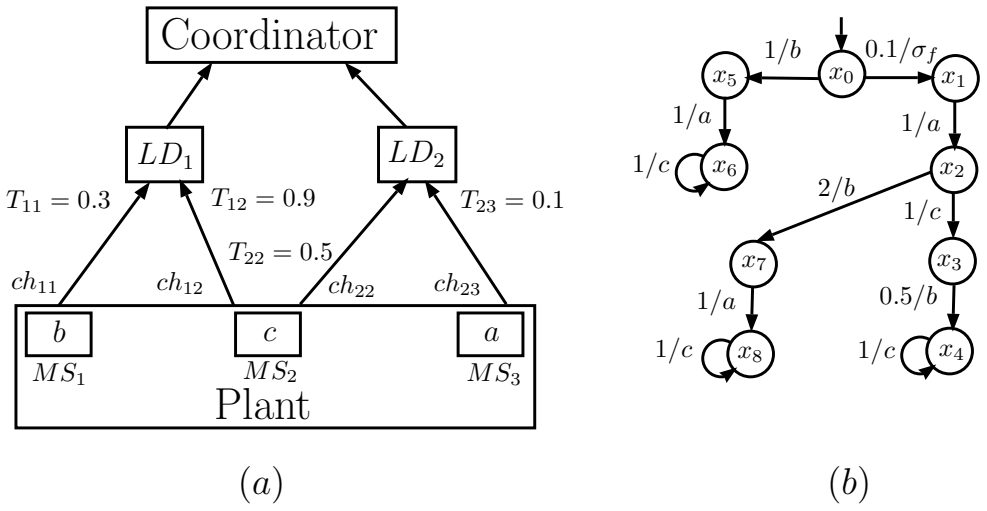


Figure 4.18: $NDESWTS = (G, t_{min}, T)$ for Example 4.6: communication delay structure (a) and automaton G with minimal time function t_{min} (b).

$P'_i := \Sigma'^{*}_i \rightarrow \Sigma^*$, for all $i \in \{1, \ldots, N_s\}$, when we compute $G^{NET}_{scc} = ||_{i=1}^{N_s} G'_{scc_i}$, trace st will synchronize the strongly connected components formed with states (x'_{d_i}, x'_{ℓ_i}), such that x'_{d_i} is uncertain and x'_{ℓ_i} is an Y-labeled state, *i.e.*, $x'_{\ell_i} = (x'_i, Y)$ for each G'_{scc_i}, forming a strongly connected component in G^{NET}_{scc} with states of type $(x'_{d_1}, x'_{\ell_1}), (x'_{d_2}, x'_{\ell_2}), \ldots, (x'_{d_{N_s}}, x'_{\ell_{N_s}})$, such that x'_{d_i} is uncertain and x'_{ℓ_i} is an Y-labeled state. ■

Example 4.6 *Consider $NDESWTS = (G, t_{min}, T)$ depicted in Figure 4.18(a) and 4.18(b). From Figure 4.18(a), it can be seen that the communication delay structure has two local diagnosers, LD_1 and LD_2, and three measurement sites, MS_1, MS_2 and MS_3. Let $\Sigma_{MS_1} = \{b\}$, $\Sigma_{MS_2} = \{c\}$ and $\Sigma_{MS_3} = \{a\}$, be the sets of events that the measurement sites MS_1, MS_2 and MS_3, respectively, detects. Assume that the set of observable events of local diagnoser LD_1 is $\Sigma_{o_1} = \{b, c\}$. Thus, the occurrence of the events in Σ_{o_1} are transmitted through communication channels ch_{11} and ch_{12}, i.e., $\Sigma_{o_{11}} = \{b\}$ and $\Sigma_{o_{12}} = \{c\}$. Assume now that the set of observable events of LD_2 is, $\Sigma_{o_2} = \{a, c\}$. Thus, the occurrence of the events in Σ_{o_2} are communicated through channels ch_{22} and ch_{23}, i.e., $\Sigma_{o_{22}} = \{c\}$ and $\Sigma_{o_{23}} = \{a\}$. Figure 4.18(b) depicts automaton G, where $\Sigma = \{a, b, c, \sigma_f\}$, and the minimal time function is given as: $t_{min}(x_0, \sigma_f) = 0.1$, $t_{min}(x_1, a) = t_{min}(x_2, c) = t_{min}(x_4, c) = t_{min}(x_0, b) = t_{min}(x_5, a) = t_{min}(x_6, c) = t_{min}(x_7, a) = t_{min}(x_8, c) = 1$, $t_{min}(x_2, b) = 2$ and $t_{min}(x_3, b) = 0.5$. Finally, the maximal delay matrix is*

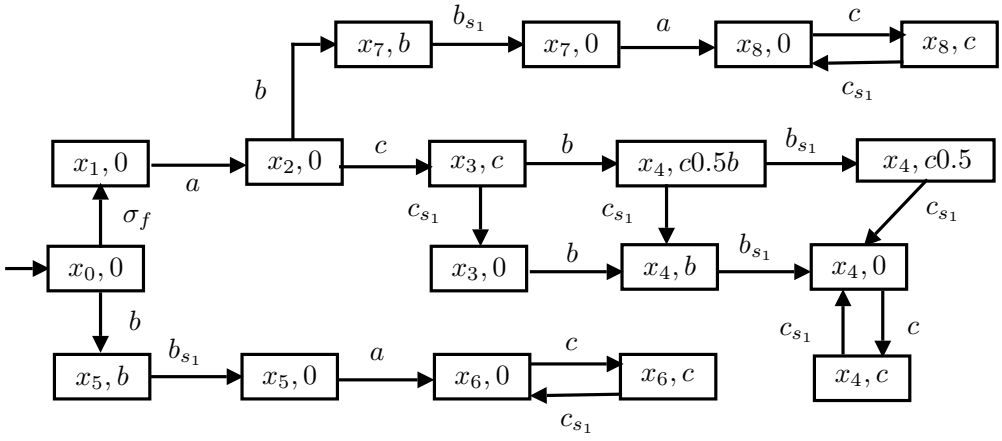


Figure 4.19: Automaton G_1 of Example 4.6.

$T = [T_{11}\ T_{12}\ T_{13};\ T_{21}\ T_{22}\ T_{23}]$, *where* $T_{11} = 0.3$, $T_{12} = 0.9$, $T_{13} = \infty$, $T_{21} = \infty$, $T_{22} = 0.5$ *and* $T_{23} = 0.1$. *It is not difficult to check that* $L(G)$ *is codiagnosable with respect to* P_{o_i} *and* Σ_f, $i = 1, 2$. *We will now verify if NDESWTS remains codiagnosable in the presence of communication delays and intermittent loss of observation.*

Automaton G_1, *shown in Figure 4.19, is computed according to Algorithm 4.1 as follows. In Step* 1, *we define the initial state of* G_1 *as the pair* $(x_0, 0)$, *where* x_0 *is the initial state of* G. *In Step 2, the set of events* $\Sigma^s_{o11} = \{b_{s_1}\}$, $\Sigma^s_{o12} = \{c_{s_1}\}$, $\Sigma^s_{o_1} = \{b_{s_1}, c_{s_1}\}$ *and* $\Sigma_1 = \{\sigma_f, a, b, c, b_{s_1}, c_{s_1}\}$ *are created. Step* 3 *forms queue* $F = [(x_0, 0)]$ *and, in Step* 4, *we establish new transitions and states. In Step 4, we obtain* $(x, q) = (x_0, 0)$, *set* $F = [\ ]$ *and* $X_1 = (x_0, 0)$ *in Steps* 4.1 *and* 4.2. *In Step* 4.3, *we create the set of indices* I_{o_1} *that records the indices of the observable events inside* q. *Thus,* $I_{o_1} = \emptyset$, *since* $q = 0$. *In Step* 4.4, *we compute, for all* $k \in I_{o_1}$, *the minimal elapsed time after the occurrence of event* q_k *in the plant, which is denoted as* $minet(q_k)$. *We define two transitions from state* $(x_0, 0)$ *labeled by events* σ_f *and* b, *since* $\Gamma(x_0) = \{\sigma_f, b\}$ *and* $q = 0$, *and the states reached by these transitions are, respectively,* $(x_1, 0)$ *and* (x_5, b). *We then add these states to queue* F, *which becomes* $F = [(x_1, 0), (x_5, b)]$. *Notice that, event* b (*resp.* σ_f) *is added* (*resp. not added*) *to the second component of the reached state* (x_5, b) $(resp.(x_1, 0))$ *because it is an observable event (resp. unobservable event). Since* $q = 0$, *no transition is defined in Step* 4.5. *The iterations are performed until* F *becomes empty. Notice that in the trace* $s = \sigma_f acbb_{s_1}c_{s_1} \in L(G_1)$ *occurs a change of order in the observation of local diagnoser* LD_1. *The plant generates events* c *and* b, *in this order, however, the local*

diagnoser LD_1 observes event b_{s_1} before event c_{s_1}.

The next step in the verification of the NDESWTS is the computation, according to Algorithm 4.1, of automaton G_2, depicted in Figure 4.20. Notice that, the set of observable and unobservable events of local diagnoser LD_1 are $\Sigma_{1_o} = \{b_{s_1}, c_{s_1}\}$ and $\Sigma_{1_{uo}} = \{a, b, c, \sigma_f\}$, respectively, and the set of observable and unobservable events of local diagnoser LD_2 are $\Sigma_{2_o} = \{a_{s_2}, c_{s_2}\}$ and $\Sigma_{2_{uo}} = \Sigma_{1_{uo}} = \{a, b, c, \sigma_f\}$, respectively.

Let us now assume that event b is subject to intermittent loss of observation only by local diagnoser LD_1, and event a is subject to intermittent loss of observation only by local diagnoser LD_2. Thus, for: (i) local diagnoser LD_1: $\Sigma_{1,ilo} = \{b\}$, $\Sigma_{1,nilo} = \{c\}$ and $\Sigma^{s'}_{1,ilo} = \{b'_s\}$; (ii) local diagnoser LD_2, $\Sigma_{2,ilo} = \{a\}$, $\Sigma_{2,nilo} = \{c\}$ and $\Sigma^{s'}_{2,ilo} = \{a'_s\}$. After forming those sets, we can compute automata G'_1 and G'_2, according to Equation (4.19), that model the communication delay and intermittent loss of observations of the events in $\Sigma_{1,ilo}$ and $\Sigma_{2,ilo}$, respectively. State transition diagrams corresponding to automata G'_1 and G'_2 are depicted in Figures 4.21 and 4.22, respectively. In order to check NDESWTS codiagnosability, according to Algorithm 4.2, we first to compute automata G'_{scc_1} and G'_{scc_2}. Since states of diagnosers have a long label, we rename states from G'_1 and G'_2 according to Table 4.6.

In Step 1 of Algorithm 4.2, we can compute automata $G'_{scc_1} = G'_{d_1}||G'_{\ell_1}$ and $G'_{scc_2} = G'_{d_2}||G'_{\ell_2}$ according to Equation (4.22), where G'_{ℓ_1}, G'_{ℓ_2} G'_{d_1}, G'_{d_2} are depicted in Figures 4.23, 4.24, 4.25 and 4.26 respectively. In Step 2, we mark all strongly connected components of G'_{scc_i}, $i \in \{1, 2\}$, formed with states (x'_{d_i}, x'_{ℓ_i}), such that x'_{d_i} is uncertain and x'_{ℓ_i} is an Y-labeled state, i.e., $x'_{\ell_i} = (x'_i, Y)$, where $x'_i \in X'_i$. Due to the size of automata G'_{scc_1} and G'_{scc_2}, we show in Figures 4.27 and 4.27 only the paths that reach strongly connected components with marked states in G'_{scc_1} and G'_{scc_2}, respectively. Since there exist those strongly connected components, according to Theorem 4.2, the language L is not NDESWTS diagnosable. We now check if L is NDESWTS codiagnosable or not. To this end, we need to check if some marked strongly connected component of G'_{scc_i}, $i \in \{1, 2\}$, "survives" to the parallel composition. In Step 3, we compute automaton $G^{NET}_{scc} = G'_{scc_1}||G'_{scc_2}$. Since this automaton has strongly connected components with marked states, as shown in Figure 4.29, L is not NDESWTS codiagnosable with respect to χ_i, P'_{is}, D_{s_i}, $i = 1, 2$ and $\Sigma_f = \{\sigma_f\}$.

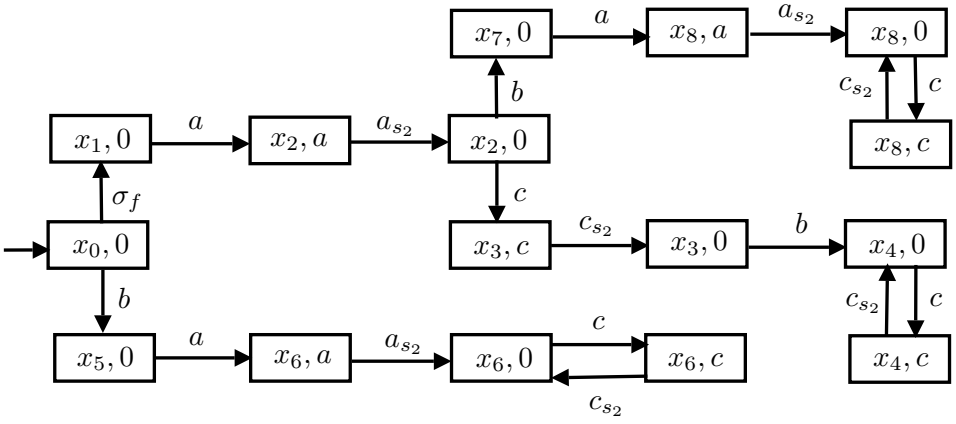


Figure 4.20: Automaton G_2 of Example 4.6.

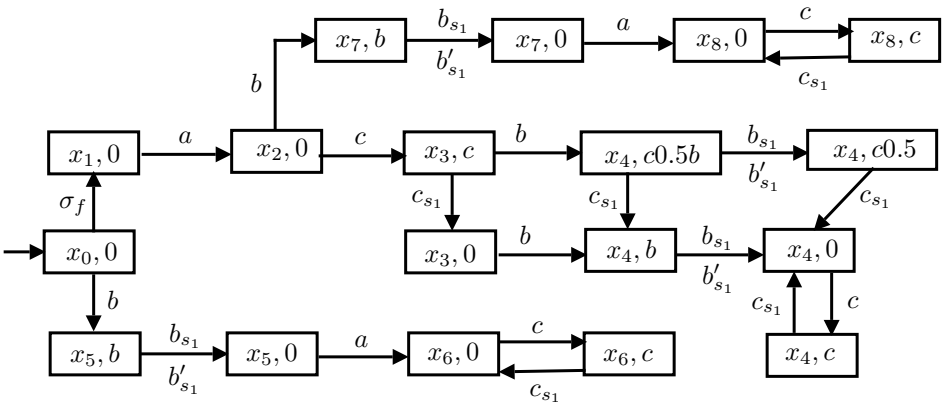


Figure 4.21: Automaton G'_1 of Example 4.6.

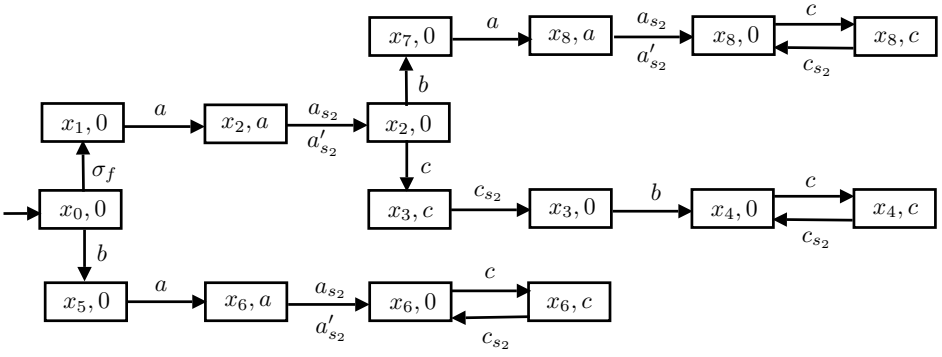


Figure 4.22: Automaton G'_2 of Example 4.6.

Table 4.1: Renaming states of G'_1 and G'_2 of Example 4.6.

State of G'_1	State of G'_2	No.	State of G'_1	State of G'_2	No.
$(x_0, 0)$	$(x_0, 0)$	0	$(x_8, 0)$	(x_8, a)	9
(x_5, b)	$(x_5, 0)$	1	(x_8, c)	$(x_8, 0)$	10
$(x_5, 0)$	(x_6, a)	2	(x_3, c)	(x_8, c)	11
$(x_6, 0)$	$(x_6, 0)$	3	$(x_4, c0.5b)$	(x_3, c)	12
(x_6, c)	(x_6, c)	4	$(x_4, c0.5)$	$(x_3, 0)$	13
$(x_1, 0)$	$(x_1, 0)$	5	$(x_4, 0)$	$(x_4, 0)$	14
$(x_2, 0)$	(x_2, a)	6	(x_4, c)	(x_4, c)	15
(x_7, b)	$(x_2, 0)$	7	(x_4, b)	—	16
$(x_7, 0)$	$(x_7, 0)$	8	$(x_3, 0)$	—	17

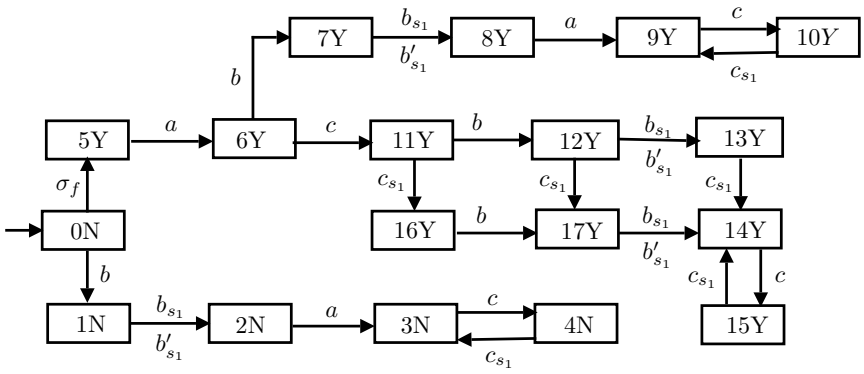


Figure 4.23: Automaton G'_{ℓ_1} of Example 4.6.

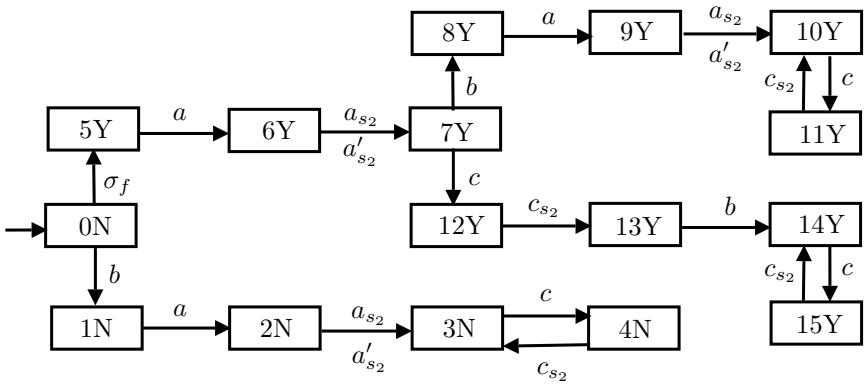


Figure 4.24: Automaton G'_{ℓ_2} of Example 4.6.

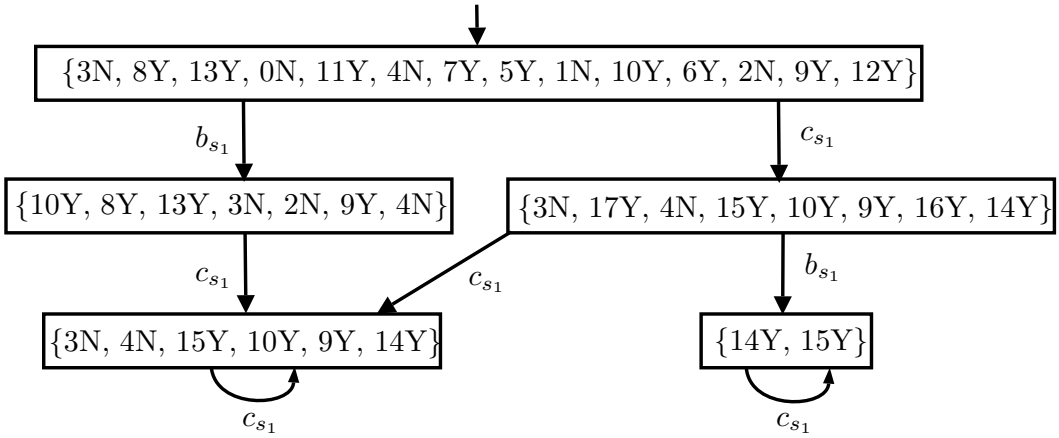


Figure 4.25: Automaton G'_{d_1} of Example 4.6.

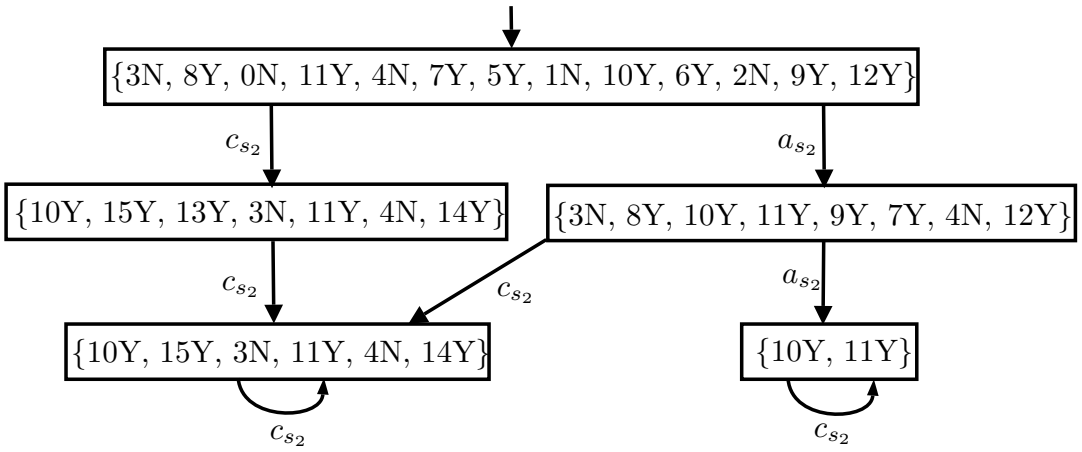


Figure 4.26: Automaton G'_{d_2} of Example 4.6.

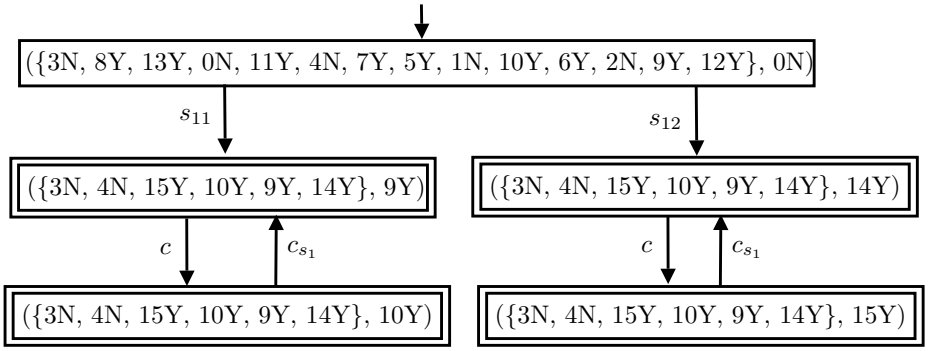


Figure 4.27: Paths that reach strongly connected components with marked states in G'_{scc_1} of Example 4.6, where $s_{11}, s_{12} \in L(G_{scc_1})$.

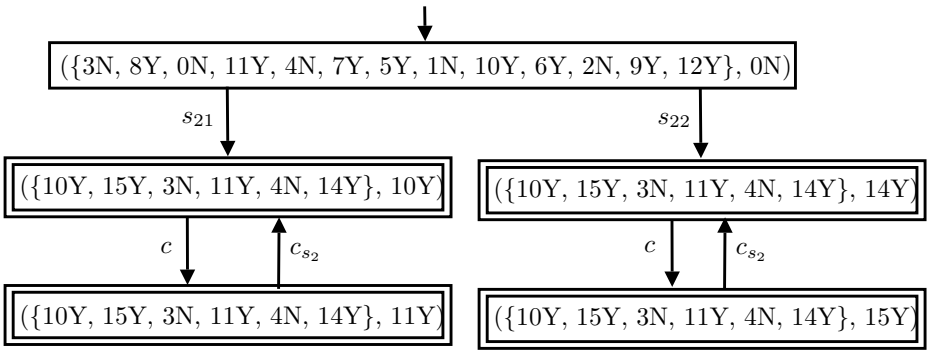


Figure 4.28: Paths that reach strongly connected components with marked states in G'_{scc_2} of Example 4.6, where $s_{21}, s_{22} \in L(G_{scc_2})$.

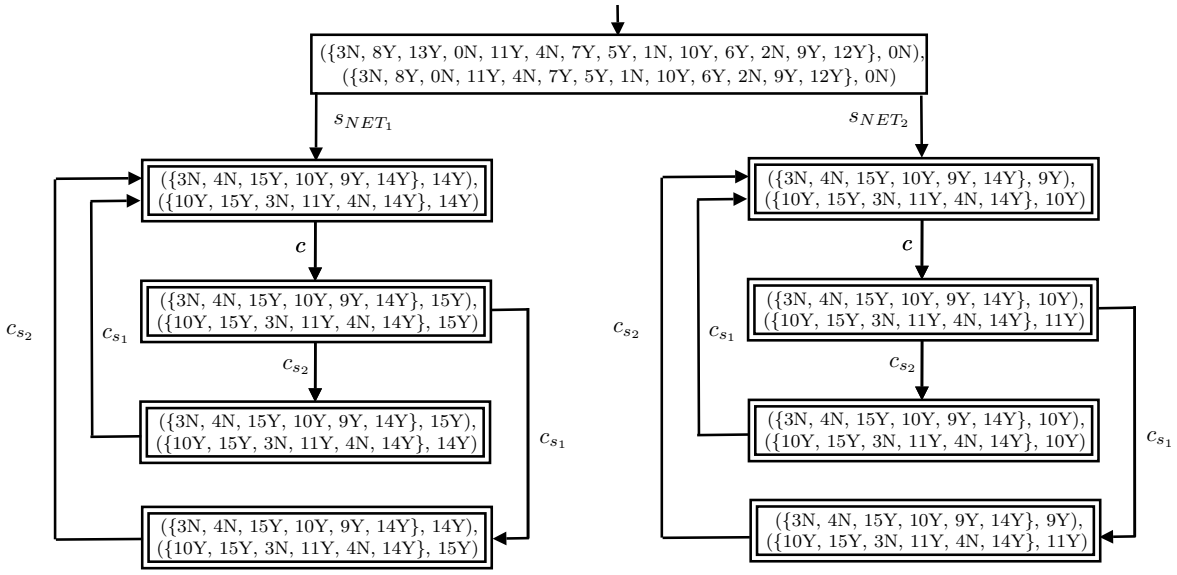


Figure 4.29: Paths that reach strongly connected components with marked states of G^{NET}_{scc} of Example 4.6, where $s_{NET_1}, s_{NET_2} \in L(G^{NET}_{scc})$.

Remark 4.6 *(Computational complexity of Algorithm 4.2) The computational complexity of Algorithm 4.2 is based on the computation of G^{NET}_{scc} that is performed by parallel composition between automata G'_{scc_i}, $i = 1, \ldots, N_s$. Since each G'_{scc_i} is the parallel composition between G'_{d_i} and G'_{ℓ_i} we can construct Table 4.6 to show the maximum number the states and transitions of all automata that must be computed to obtain G^{NET}_{scc} from G'_i. It is worth remarking that $|X_i|$ (the number of states of G'_i) as function of $|X|$ (number of states of the plant) was computed in Remark 4.4.*

Table 4.2: Computational complexity of Algorithm 4.2.

	No. of states	No. of transitions
G'_i	$\|X_i\|$	$\|X_i\|\|\Sigma'_i\|$
A_ℓ	2	2
G'_{ℓ_i}	$2\|X_i\|$	$2\|X_i\|\|\Sigma'_i\|$
G'_{d_i}	$\|X_{d_i}\| = 2^{2\|X_i\|}$	$2^{2\|X_i\|}\|\Sigma^s_{o_i}\|$
G'_{scc_i}	$2\|X_{d_i}\|\|X_i\|$	$2\|X_{d_i}\|\|X_i\|\|\Sigma'_i\|$
G^{NET}_{scc}	$\prod_{i=1}^{N_s} 2\|X_i\|\|X_{d_i}\|$	$(\prod_{i=1}^{N_s} 2\|X_i\|\|X_{d_i}\|\|\Sigma'_i\|$
Complexity		$O(\prod_{i=1}^{N_s} \|X_i\|\|X_{d_i}\|\|\Sigma'_i\|)$

4.3.2 NDESWTS Verifier

We will now present an algorithm for the verification of NDESWTS codiagnosability of DES based on the same idea as the verifier proposed in [47]. To this end, we first present the definition of the one-to-one event renaming function, as follows.

$$\rho_i : \Sigma'_{i_N} \to \Sigma'_{i_\rho} \tag{4.23}$$

$$\sigma \mapsto \rho_i(\sigma) = \begin{cases} \sigma_{\rho_i}, \text{ if } \sigma \in (\Sigma \cup \Sigma^{s'}_{i,ilo}) \setminus \Sigma_f \\ \sigma, \text{ if } \sigma \in \Sigma^s_{o_i}. \end{cases}$$

where $\Sigma'_{i_N} = \Sigma'_i \setminus \Sigma_f$, for $i = 1, \ldots, n$. The domain of function ρ_i can be extended to $\Sigma'^*_{i_N}$ as usual, *i.e.*, $\rho_i(s\sigma) = \rho_i(s)\rho_i(\sigma)$, for all $s \in \Sigma'^*_{i_N}$ 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 4.3 *NDESWTS codiagnosability verification using verifier*

Input *Automaton* $G'_i = (X_i, \Sigma'_i, f'_i, \Gamma_i, x_{i_0})$, *for* $i = 1, \ldots, N_s$.

Output *Automaton* $V = (X_V, \Sigma_V, f_V, \Gamma_V, x_{0,V}, X_{V_m})$

Step 1. *Compute automaton* $G'_{i,N} = (X'_{i_N}, \Sigma'_{i_N} f'_{i_N}, \Gamma'_{i_N}, (x_{i_0}, N), \emptyset)$, *where* $\Sigma'_{i_N} = \Sigma'_i \setminus \Sigma_f$, *for* $i = 1, \ldots, N_s$, *that models the normal behavior of automaton* G'_i *as presented in Algorithm 2.2;*

STEP 2. *Compute automaton* $G'_{i,F} = (X'_{i_F}, \Sigma'_i, f'_{i_F}, \Gamma'_{i_F}, (x_{i_0}, N), X_{i_{F_m}})$, *for* $i = 1, \ldots, N_s$, *that models the failure behavior of* G'_i *as presented in Algorithm 2.2;.*

STEP 3. *Construct automaton* $G'_{i,\rho} = (X'_{i_N}, \Sigma'_{i_\rho}, f'_{i_\rho}, \Gamma'_{i_\rho}, (x_{i_0}, N), \emptyset)$ *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* $x_{i_N} \in X'_{i_N}$.

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

STEP 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:*

$$\begin{aligned} &\exists j \in \{k, k+1, \ldots, \ell\} \text{ such that, for some} \\ &\quad y^j_{V_i} = (x^j_i, N, y^j_i, Y), \wedge(\sigma_j \in \Sigma'_i) \end{aligned} \tag{4.24}$$

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

STEP 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 strongly connected components that contain cyclic paths* cl_i *which violate condition (4.24).*

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

STEP 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* V, $\ell \geq k > 0$, *that satisfies the following condition:*

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

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

The idea behind Algorithm 4.3 is similar to Algorithm 2.2. In Steps 1 and 2, we compute automata $G'_{i,N}$ and $G'_{i,F}$ that that models the normal and failure behavior of G'_i. In Step 3, we compute automaton $G'_{i,\rho}$ from $G'_{i,N}$ by renaming its

unobservable events. In Step 4, we compute each verifier automaton $\bar{V}_i = G'_{i,\rho} \| G'_{i,F}$ for $i = \{1, \ldots, N_s\}$. This verifier has the same idea as the verifier presented in Algorithm 2.2: it is the parallel composition of the system with faults and the system without faults, with a synchronization on the observable events, being, in this case, $\Sigma^s_{o_i}$. In Step 5, we find the cyclic paths in $\bar{V}_i$ with uncertain states $(x_i^j N y_i^j Y)$ and events not renamed. In Step 6, we form V_i from $\bar{V}_i$ by marking those uncertain states that belongs to the strongly connected components that the contain cyclic paths which violate condition (4.24). Finally, in Steps 7 and 8, we compute the verifier automaton $V = V_1 \| \ldots \| V_{N_s}$ and we check if some marked cyclic path survives in automaton V. If the answer is yes, then L is not NDESWTS codiagnosable. Otherwise, L is NDESWTS codiagnosable.

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

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

Theorem 4.4 *Language L is NDESWTS codiagnosable with respect to χ_i, D_{s_i}, P'_{is}, for $i = 1, \ldots, N_s$, 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 V satisfying the following condition:*

$$\begin{aligned} &x_V^q \in X_{V_m}, \forall q \in \{k, k+1, \ldots, \ell\}, \text{ and for some} \\ &q \in \{k, k+1, \ldots, \ell\}, \sigma_q \in \Sigma. \end{aligned} \tag{4.25}$$

Proof: ($\Rightarrow$) Suppose that language L is not NDESWTS codiagnosable with respect to χ_i, D_{s_i}, P'_{s_i}, for $i = 1, \ldots, N_s$, and Σ_f. Thus, according to Definition 4.4, there exists at least one arbitrarily long length trace st ($s \in \Psi(\Sigma_f)$, $t \in L/s$) and traces ω_i, $i = 1, \ldots, N_s$, where $\Sigma_f \notin \omega_i$ and ω_i is not necessarily distinct from ω_j, for $j = 1, \ldots, N_s$ 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 $i \in \{1, 2, \ldots, N_s\}$. Thus, according to Lemma 4.2 if L is not NDESWTS codiagnosable, there exist traces $s_i t_i \in L_m(G'_{i,F})$ 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 $i \in \{1, 2, \ldots, N_s\}$. As shown in [47], 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 (4.24), 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 $V = ||_{i=1}^n V_i$, then $L_m(V) = \bigcap_{i=1}^n P^{-1}_{VV_i}[L_m(V_i)]$, where $P_{VV_i} : \Sigma^*_V \to \Sigma^*_{V_i}$. Thus, $\bigcap_{i=1}^n P^{-1}_{VV_i}(v_i) \subseteq L_m(V)$. Let $v \in \bigcap_{i=1}^n P^{-1}_{VV_i}(v_i)$. 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 $i = 1, \ldots, N_s$, in $\bigcap_{i=1}^n P^{-1}_{VV_i}(v_i)$ are in Σ, then there will be a cyclic path in 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 V that ends with a cyclic path cl that satisfies condition (4.25), and let $v \in L_m(V)$ be the trace associated with p. Notice that, since $V = \|_{i=1}^n V_i$, then $L_m(V) = \bigcap_{i=1}^{N_s} {P_{VV_i}}^{-1}[L_m(V_i)]$, and $P_{VV_i}(v) = v_i \in L_m(V_i)$, for $i = 1, 2, \ldots, N_s$. Notice also that, the common events of traces $v_i \in L_m(V_i)$, for $i = 1, 2, \ldots, N_s$, are events $\sigma \in \Sigma$. Thus, since condition (4.25) 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 4.3, 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 V there exists one cyclic path cl_i in each verifier V_i, for $i = 1, \ldots, N_s$, that satisfies condition (4.24), *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 $P'_{is}(s_i t_i) = P'_{is}(s_{i_N})$, for all $i \in \{1, \ldots, N_s\}$. In order to show that L is not NDESWTS codiagnosable, notice that, since condition (4.25) 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 $V = \|_{i=1}^{N_s} V_i$, 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 $i \in \{1, \ldots, N_s\}$. Thus, according to Definition 4.4, L is not NDESWTS codiagnosable with respect to χ_i, D_{s_i}, P'_{is}, for $i = 1, \ldots, N_s$, and Σ_f. □

Example 4.7 *Consider again the system of Example 4.6 NDESWTS* $= (G, t_{min}, T)$

depicted in Figure 4.18(a) and 4.18(b). We now apply the Algorithm 4.3 to compute V_1 *and* V_2 *by using as input automata* G'_1 *and* G'_2 *depicted in Figures 4.21 and 4.22, respectively. Following Steps 1, 2, and 3 of Algorithm 4.3, automata* $G'_{1,\rho}$ *and* $G'_{1,F}$ *shown in Figures 4.30 and 4.31, respectively and automata* $G'_{2,\rho}$ *and* $G'_{2,F}$*, shown in Figures 4.32 and 4.33, respectively, are computed. Continuing Algorithm 4.3, verifiers* V_1 *and* V_2 *are computed. Due to the size of these automata, we show in Figures 4.34(a) and 4.34(b) only those paths that contain the cyclic paths* $cl_1 = (cc_{\rho_1}c_{s_1})^p$ *and* $cl_2 = (cc_{s_2}c_{\rho_2})^q$ *that satisfy condition* (8)*, where* $p, q \in \mathbb{N}$*,* $s_{V_1} \in L(V_1)$ *and* $s_{V_2} \in L(V_2)$*. After the computation of* V_1 *and* V_2*, we compute automaton* $V = V_1||V_2$*. We show in Figure 4.35 only the path of* V *that contains a cyclic path* $cl = (cc_{s_2}c_{\rho_2}c_{\rho_1}c_{s_1})^l$*, where* $l \in \mathbb{N}$*, associated with the cyclic paths* cl_1 *and* cl_2*. Therefore, language* L *is not NDESWTS codiagnosable with respect to* χ_i*,* D_{s_i}*,* P'_{s_i}*,* $i = 1, 2, \ldots, n$ *and* Σ_f*.*

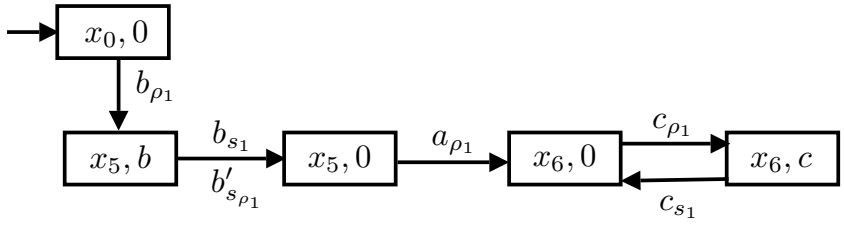


Figure 4.30: Automaton $G'_{1,\rho}$ of Example 4.7.

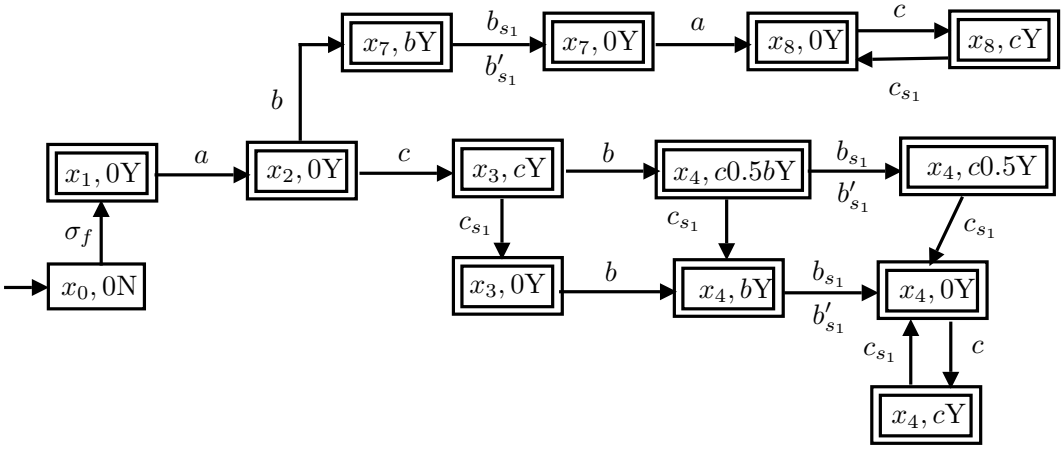


Figure 4.31: Automaton $G'_{1,F}$ of Example 4.7.

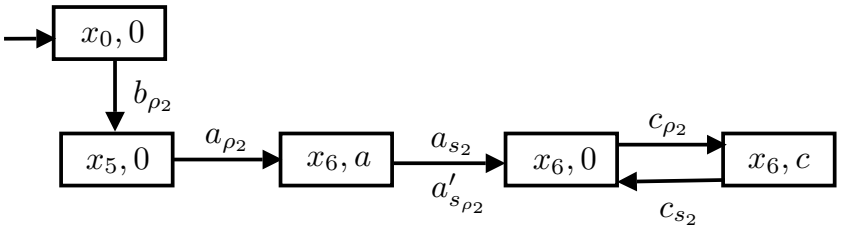


Figure 4.32: Automaton $G'_{2,\rho}$ of Example 4.7.

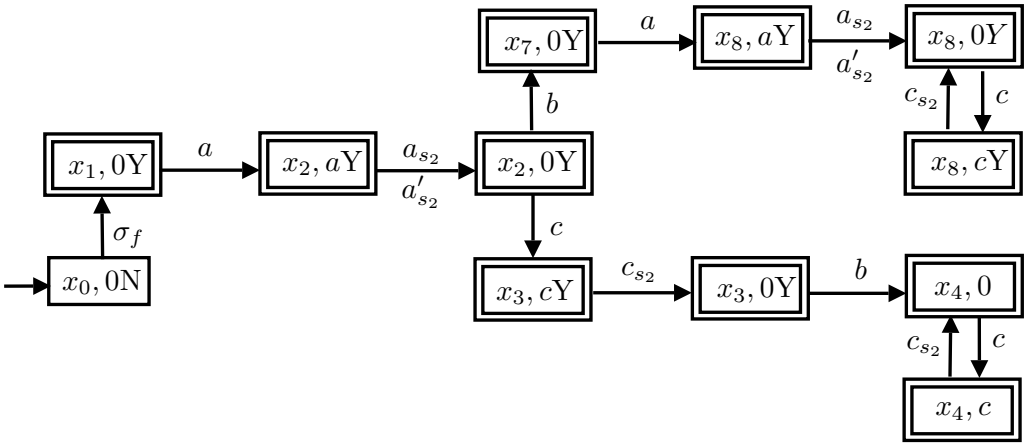
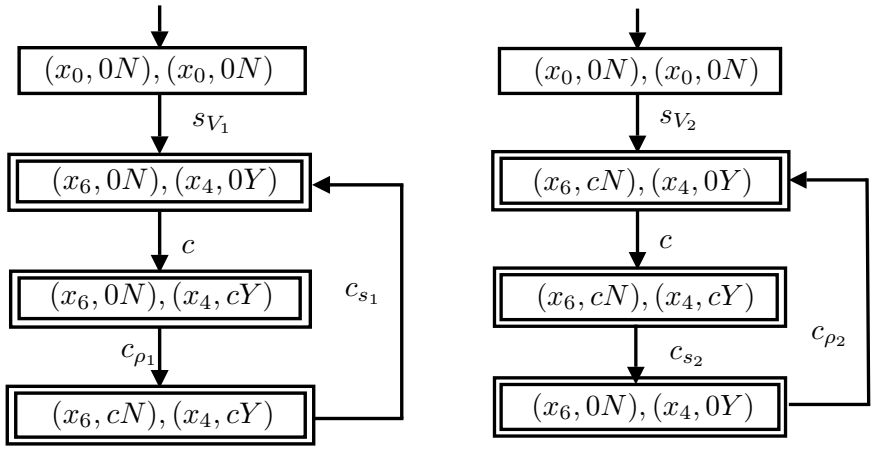


Figure 4.33: Automaton $G'_{2,F}$ of Example 4.7.



(a) (b)

Figure 4.34: Path of V_1 with cyclic path cl_1 (a) and path of V_2 with cyclic path cl_2 (b).

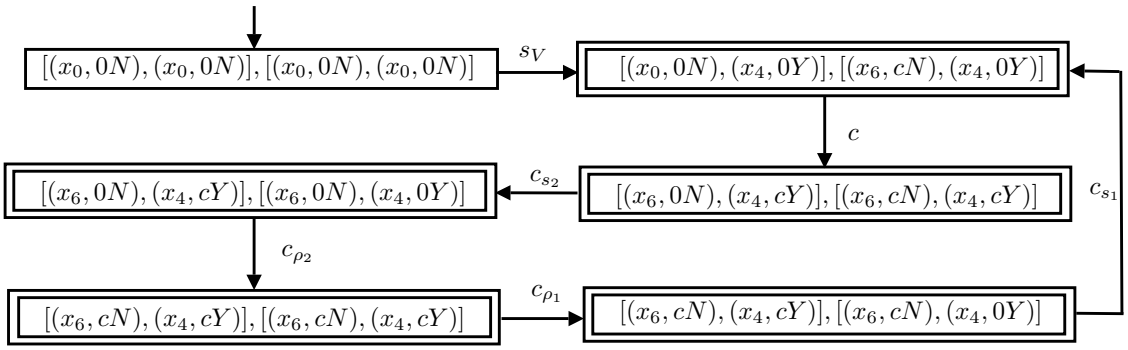


Figure 4.35: Path of V with cycle path cl Example 4.7.

Remark 4.7 *(Computational complexity of Algorithm 4.3) The computational complexity of Algorithm 4.3 is based on the computation of automaton V, which is the parallel composition between each V_1, ..., V_{N_s}. The computation of each V_i, $i = 1, 2, \ldots, N_s$, is similar to Algorithm 2.2 ([47]). Basically, we need to apply this algorithm by considering the input as G'_i. Table 4.7 shows the maximum number of states and transitions of all automata that must be computed in order to obtain the verifier automaton V for N_s local diagnosers according to Algorithm 4.3.*

Table 4.3: Computational complexity of Algorithm 4.3.

	No. of states	No. of transitions
G'_i	$\|X_i\|$	$\|X_i\|\|\Sigma'_i\|$
A_N	1	$\|\Sigma'_i\| - \|\Sigma_f\|$
G'_N	$\|X_i\|$	$\|X_i\|(\|\Sigma'_i\| - \|\Sigma_f\|)$
A_ℓ	2	$2\|\Sigma_f\|$
G'_{ℓ_i}	$2\|X_i\|$	$2\|X_i\|\|\Sigma'_i\|$
$G'_{i,F}$	$2\|X_i\|$	$2\|X_i\|\|\Sigma'_i\|$
$G'_{i,\rho}$	$\|X_i\|$	$\|X_i\|(\|\Sigma'_i\| - \|\Sigma_f\|)$
V_i	$2\|X_i\|^2$	$2\|X_i\|^2(2\|\Sigma'_i\| - \|\Sigma_f\|)$
V	$\prod_{i=1}^{N_s} 2\|X_i\|^2$	$\prod_{i=1}^{N_s} 2\|X_i\|^2(2\|\Sigma'_i\| - \|\Sigma_f\|)$
Complexity		$O(\prod_{i=1}^{N_s} \|X_i\|^2(\|\Sigma'_i\| - \|\Sigma_f\|))$

4.4 Concluding Remarks

In this chapter, we have introduced an architecture of networked discrete event systems with timing structure (NDESWTS) subject to delays and losses of observations of events between the measurement sites (MS) and local diagnosers (LD), and, for this purpose, we have introduced a new timed model that represents the dynamic behavior of the plant based on the, a priori, knowledge of the minimal firing time for each transition of the plant and on the maximal delays in the communication channels that connect the measurement sites (MS) and local diagnosers (LD). We converted this timed model in an untimed one, and, based on the untimed model, we presented necessary and sufficient conditions for NDESWTS codiagnosability and proposed two tests to its verification: the first one based on diagnosers, and a second one, based on verifiers.

Chapter 5

Conclusion

The main goal of this work was to propose the problem of codiagnosability of networked discrete event systems with timing structure (NDESWTS) in the presence of delays and loss of observations. In this regard, the main contributions of this work are described below:

- We introduce a new timed model that represents the dynamic behavior of the plant based on the, a priori, knowledge of the minimal firing time for each transition of the plant and on the maximal delays in the communication channels that connect measurement sites and local diagnosers.

- Methodology to construct an equivalent untimed model for NDESWTS. We provide algorithms for the computation of untimed deterministic finite-state automata that generate the extended languages associated with the plant and the diagnoser observation. It is worth noticing that the model for NDESWTS proposed here allows its application in other NDESWTS problems, for instance, it has also been used to study the supervisory control of NDESWTS subject to event communication delays and loss of observations [38, 39] or to deal with malicious attacks/intrusions to cyberphysical systems.

- We propose two tests to verify for codiagnosability of NDESWTS: the first one based on diagnosers, and and a second one, based on verifiers.

A preliminary version of the results obtained here was published in [82,83]. Other research topics addressed in this work were:

1. The introduction of a diagnoser-like G_{scc} to check the (co)diagnosability whose the advantages are presented as follows.

   - The test proposed here is that the search for indeterminate cycles is replaced with the search for strongly connected components.
   - The usual assumptions on language liveness and nonexistence of unobservable cycles of states connected with unobservable events only are no longer required.
   - this automaton test has in its event set both observable and unobservable events of the plant, hidden cycles are "exposed" in this approach.
   - this automaton provides all information to check (co)diagnosability, *i.e.*, diagnosers G_d and G_{test} do not carry enough information to determine if an observed cycle of uncertain states is an indeterminate cycle, since it is necessary to also perform a search for cycles in G.
   - Other advantage of the proposed test is diagnosability verification becomes a particular case of codiagnosability verification.
   - the test can be adapted to deal with codiagnosability of NDESWTS.

2. We proposed the extended verifier, developed to show not only the ambiguous paths but also those paths that lead to language diagnosis.

3. We applied diagnoser-like automaton G_{scc} and extend verifier G_{VT} to compute the maximum time a system takes to diagnose a failure occurrence, so called τ-codiagnosability, and the maximum number of event occurrences necessary to diagnose a failure occurrence, so-called K-codiagnosability.

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

(i) Different timing structures of networked discrete event systems. In this topic, the idea is to consider a timing structure, where, instead of assuming minimal activation times of the plant transitions and maximal communication delays, we could assume a priori knowledge of the time intervals in which the plant transitions can occur and the lower and upper bounds on the communication delays, without relying on automata with guards.

(*ii*) Computation of τ- and K-codiagnosability of codiagnosable NDESWTS and a systematic way to compute the maximal delay in the communication channel necessary so as to the language becomes not NDESWTS codiagnosable.

(*iii*) The study of other NDESWTS problems by using the modeling proposed in this work, for example, opacity, prognosis, detectability, state estimation.

(*iv*) Application of NDESWTS codiagnosability to real manufacturing systems. Namely, in the context of manufacturing services [84], we can use the models proposed here to check the workflow, and thus, we can ensure the production quality by making the failure diagnosis to each part of manufacturing system automatically.

Bibliography

[1] GILCHRIST, A., *Industry 4.0: the industrial internet of things*. 1 ed. Berkeley, Apress, 2016.

[2] GONZALEZ, A. G., ALVES, M. V., VIANA, G. S., CARVALHO, L. K., BASILIO, J. C., "Supervisory Control-Based Navigation Architecture: A New Framework for Autonomous Robots in Industry 4.0 Environments", *IEEE Transactions on Industrial Informatics,* DOI: 10.1109/TII.2017.2788079, 2017.

[3] WANG, S., WAN, J., ZHANG, D., LI, D., ZHANG, C., "Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination", *Computer Networks*, v. 101, pp. 158–168, 2016.

[4] BAHETI, R., GILL, H., "Cyber-physical systems", *The impact of control technology*, v. 12, pp. 161–166, 2011.

[5] MARTIN, P., EGERSTEDT, M. B., "Hybrid systems tools for compiling controllers for cyber-physical systems", *Discrete Event Dynamic Systems*, v. 22, n. 1, pp. 101–119, 2012.

[6] LEE, J., BAGHERI, B., KAO, H.-A., "A cyber-physical systems architecture for industry 4.0-based manufacturing systems", *Manufacturing Letters*, v. 3, pp. 18–23, 2015.

[7] JIRKOVSKY, V., OBITKO, M., MARIK, V., "Understanding Data Heterogeneity in the Context of Cyber-Physical Systems Integration", *IEEE Transactions on Industrial Informatics*, v. 13, n. 2, pp. 660–667, 2017.

[8] RAMADGE, P. J., WONHAM, W. M., "The control of discrete-event systems", *Proceedings of the IEEE*, v. 77, n. 1, pp. 81–98, 1989.

[9] DAVID, R., ALLA, H., "Petri nets for modeling of dynamic systems: A survey", *Automatica*, v. 30, n. 2, pp. 175–202, 1994.

[10] CASSANDRAS, C. G., LAFORTUNE, S., "Discrete event systems- The state of the art and new directions", *Applied and computational control, signals, and circuits.*, v. 1, pp. 1–65, 1999.

[11] CASSANDRAS, C. G., LAFORTUNE, S., *Introduction to Discrete Events Systems*. 2 ed. New York, NY : USA, Springer, 2008.

[12] DAVID, R., ALLA, H., *Discrete, continuous, and hybrid Petri nets*. 2 ed. New York, NY : USA, Springer, 2010.

[13] WONHAM, W. M., "Supervisory control of discrete-event systems", *Encyclopedia of Systems and Control*, pp. 1396–1404, 2015.

[14] LIN, F., "Opacity of discrete event systems and its applications", *Automatica*, v. 47, n. 3, pp. 496–503, 2011.

[15] JACOB, R., LESAGE, J.-J., FAURE, J.-M., "Overview of discrete event systems opacity: Models, validation, and quantification", *Annual Reviews in Control*, v. 41, pp. 135–146, 2016.

[16] SHU, S., LIN, F., YING, H., "Detectability of discrete event systems", *IEEE Transactions on Automatic Control*, v. 52, n. 12, pp. 2356–2359, 2007.

[17] KUMAR, R., TAKAI, S., "Decentralized prognosis of failures in discrete event systems", *IEEE Transactions on Automatic Control*, v. 55, n. 1, pp. 48–59, 2010.

[18] LIN, F., "Diagnosability of discrete event systems and its applications", *Discrete Event Dynamic Systems: Theory and Applications*, v. 4, n. 1, pp. 197–212, 1994.

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

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

[21] 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, pp. 33–86, 2000.

[22] 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.

[23] GUNGOR, V. C., LAMBERT, F. C., "A survey on communication networks for electric system automation", *Computer Networks*, v. 50, n. 7, pp. 877–897, 2006.

[24] DIMITRIOS, H., WILLIAM, S., *Handbook of networked and embedded control systems*. Boston, USA, Birkhauser, 2008.

[25] HESPANHA, J. P., NAGHSHTABRIZI, P., XU, Y., "A survey of recent results in networked control systems", *Proceedings of the IEEE*, v. 95, n. 1, pp. 138–162, 2007.

[26] GUPTA, R. A., CHOW, M.-Y., "Networked control system: Overview and research trends", *IEEE transactions on industrial electronics*, v. 57, n. 7, pp. 2527–2535, 2010.

[27] BALEMI, S., "Input/output discrete event processes and communication delays", *Discrete Event Dynamic Systems*, v. 4, n. 1, pp. 41–85, 1994.

[28] 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.

[29] SADID, W. H., RICKER, L., HASHTRUDI-ZAD, S., "Robustness of synchronous communication protocols with delay for decentralized discrete-event control", *Discrete Event Dynamic Systems*, v. 25, n. 1-2, pp. 159–176, 2015.

[30] PARK, S.-J., CHO, K.-H., "Delay-robust supervisory control of discrete-event systems with bounded communication delays", *IEEE Transactions on Automatic Control*, v. 51, n. 5, pp. 911–915, 2006.

[31] 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.

[32] PARK, S.-J., CHO, K.-H., "Supervisory control of discrete event systems with communication delays and partial observations", *Systems & control letters*, v. 56, n. 2, pp. 106–112, 2007.

[33] LIN, F., "Control of networked discrete event systems: dealing with communication delays and losses", *SIAM Journal on Control and Optimization*, v. 52, n. 2, pp. 1276–1298, 2014.

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

[35] 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.

[36] SHU, S., LIN, F., "Deterministic Networked Control of Discrete Event Systems with Nondeterministic Communication Delays", *IEEE Transactions on Automatic Control*, v. 62, n. 1, pp. 190–205, 2017.

[37] SHU, S., LIN, F., "Predictive Networked Control of Discrete Event Systems", *IEEE Transactions on Automatic Control*, v. 62, n. 9, pp. 4698–4705, 2017.

[38] ALVES, M. V. S., CARVALHO, L. K., BASILIO, J. C., "Controle supervisório de sistemas a eventos discretos em rede temporizados". In: *XIII Simpósio Brasileiro de Automação Inteligente,* Porto Alegre, BRA, pp. 1382–1389, 2017.

[39] ALVES, M. V. S., CARVALHO, L. K., BASILIO, J. C., "Supervisory Control of Timed Networked Discrete Event Systems". In: *Proceedings of the 56th IEEE Conference on Decision and Control,* Melbourne, AUS, pp. 4859–4865, 2017.

[40] 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.

[41] QIU, W., KUMAR, R., "Distributed diagnosis under bounded-delay communication of immediately forwarded local observations", *IEEE Transactions on Systems, Man, and Cybernetics-Part A Systems and Humans*, v. 38, n. 3, pp. 628–643, 2008.

[42] NUNES, C. E. V., MOREIRA, M. V., ALVES, M. V. S., "Network codiagnosability of Discrete-Event Systems subject to event communication delays". In: *13th International Workshop on Discrete Event Systems (WODES),* Xi'an, China, pp. 217–223, 2016.

[43] NUNES, C. E. V., MOREIRA, M. V., ALVES, M. V. S., CARVALHO, L. K., BASILIO, J. C., "Codiagnosability of networked discrete event systems subject to communication delays and intermittent loss of observation", *Discrete Event Dynamic Systems,* DOI: 10.1007/s10626-017-0265-6, 2018.

[44] JIANG, S., HUANG, Z., CHANDRA, V., KUMAR, R., "A polynomial algorithm for testing diagnosability of discrete-event systems", *IEEE Transactions on Automatic Control*, v. 46, n. 8, pp. 1318–1321, 2001.

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

[46] YOO, T.-S., LAFORTUNE, S., "Polynomial-time verification of diagnosability of partially observed discrete-event systems", *IEEE Transactions on Automatic Control*, v. 47, n. 9, pp. 1491–1495, 2002.

[47] 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.

[48] MOHRI, M., "Weighted automata algorithms", *Handbook of weighted automata*, pp. 213–254, 2009.

[49] SU, R., VAN SCHUPPEN, J., ROODA., J. E., "The synthesis of time optimal supervisors by using heaps-of-pieces", *IEEE Transactions on Automatic Control*, v. 57, n. 1, pp. 105–118, 2012.

[50] 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.

[51] PENCOLÉ, Y., CORDIER, M.-O., "A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks", *Artificial Intelligence*, v. 164, n. 1, pp. 121–170, 2005.

[52] THORSLEY, D., TENEKETZIS, D., "Diagnosability of stochastic discrete-event systems", *IEEE Transactions on Automatic Control*, v. 50, n. 4, pp. 476–492, 2005.

[53] CABRAL, F. G., MOREIRA, M. V., DIENE, O., BASILIO, J. C., "A Petri Net Diagnoser for Discrete Event Systems Modeled by Finite State Automata", *IEEE Transactions on Automatic Control*, v. 60, n. 1, pp. 59–71, 2015.

[54] 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.

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

[56] CARVALHO, L. K., MOREIRA, M. V., BASILIO, J. C., "Diagnosability of intermittent sensor faults in discrete event systems", *Automatica*, v. 79, pp. 315–325, 2017.

[57] CLAVIJO, L. B., BASILIO, J. C., "Empirical studies in the size of diagnosers and verifiers for diagnosability analysis", *Discrete Event Dynamic Systems*, v. 27, n. 4, pp. 701–739, 2017.

[58] JOHNSON, D. B., "Finding All the Elementary Circuits of a Directed Graph", *SIAM Journal of Computing*, v. 4, n. 1, pp. 77–84, 1975.

[59] TARJAN, R., "Depth first search and linear graph algorithms", *SIAM Journal of Computer*, v. 1, n. 2, pp. 146–160, 1972.

[60] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., STEIN, C., *Introduction to Algorithms*. Cambridge, MA, MIT Press, 2007.

[61] TOMOLA, J. H., CABRAL, F. G., CARVALHO, L. K., MOREIRA, M. V., "Robust Disjunctive-Codiagnosability of Discrete-Event Systems Against Permanent Loss of Observations", *IEEE Transactions on Automatic Control*, v. 62, n. 11, pp. 5808–5815, 2017.

[62] YOO, T.-S., GARCIA, H., "Computation of fault detection delay in discrete-event systems". In: *Proceedings of the 14th International Workshop on Principles of Diagnosis,* Washington, D.C., USA, pp. 207—212, 2003.

[63] VIANA, G. S., BASILIO, J. C., MOREIRA, M. V., "Computation of the maximum time for failure diagnosis of discrete-event systems", *In Proc. of the American Control Conference,* Chicago, IL, pp. 396–401, 2015.

[64] BASILE, F., CABASINO, M. P., SEATZU, C., "Diagnosability Analysis of Labeled Time Petri Net Systems", *IEEE Transactions on Automatic Control*, v. 62, n. 3, pp. 1384–1396, 2017.

[65] CASSEZ, F., "The complexity of codiagnosability for discrete event and timed systems", *IEEE Transactions on Automatic Control*, v. 57, n. 7, pp. 1752–1764, 2012.

[66] ALUR, R., DILL, D. L., "A theory of timed automata", *Theoret. Comput. Sci.*, v. 126, n. 2, pp. 183–235, 1994.

[67] PAN, J., HASHTRUDI-ZAD, S., "Diagnosability test for timed discrete-event systems", *IEEE International Conference on Tools with Artificial Intelligence,* Arlington, VA, pp. 63–72, 2006.

[68] CASSEZ, F., "A note on fault diagnosis algorithms". In: *Proc. 48th IEEE Conf. Decision Control and 28th Chinese Control Conf.,* Shanghai, China, pp. 6941–6946, 2009.

[69] BRANDIN, B. A., WONHAM, W. M., "Supervisory control of timed discrete-event systems", *IEEE Transactions on Automatic Control*, v. 39, n. 2, pp. 329–342, 1994.

[70] TRIPAKIS, S., "Fault diagnosis for timed automata", in *Lecture Notes in Computer Sciences, In Formal Techniques in Real Time and Fault Tolerant Systems (FTRTFT),* New York: Springer-Verlag, v. 2469, 2002.

[71] MOREIRA, M. V., BASILIO, J. C., CABRAL, F. G., "'Polynomial time verification of decentralized diagnosability of discrete event systems' versus 'Decentralized failure diagnosis of discrete event systems: A critical appraisal"', *IEEE Transactions on Automatic Control*, v. 61, n. 1, pp. 178–181, 2016.

[72] PARK, S.-J., CHO, K.-H., "Nonblocking supervisory control of timed discrete event systems under communication delays: The existence conditions", *Automatica*, v. 44, n. 4, pp. 1011–1019, 2008.

[73] ZHANG, R., CAI, K., GAN, Y., WONHAM, W., "Delay-robustness in distributed control of timed discrete-event systems based on supervisor localisation", *International Journal of Control*, v. 89, n. 10, pp. 2055–2072, 2016.

[74] ZHAO, B., LIN, F., WANG, C., ZHANG, X., POLIS, M. P., WANG, L. Y., "Supervisory control of networked timed discrete event systems and its applications to power distribution networks", *IEEE Transactions on Control of Network Systems*, v. 4, n. 2, pp. 146–158, 2017.

[75] BASILIO, J. C., CARVALHO, L. K., MOREIRA, M. V., "Fault diagnosis of discrete event systems modeled as automata", *Sba: Controle & Automação Sociedade Brasileira de Automatica*, v. 21, n. 5, pp. 510–533, 2010.

[76] SANTORO, L. P., MOREIRA, M. V., BASILIO, J. C., "Computation of minimal diagnosis bases of Discrete-Event Systems using verifiers", *Automatica*, v. 77, pp. 93–102, 2017.

[77] BASILIO, J. C., LAFORTUNE, S., "Robust codiagnosability of discrete event systems", *In Proc. of the American Control Conference,* Saint Louis, MS, v. 10, pp. 2202–2209, 2009.

[78] BASILIO, J. C., SOUZA LIMA, S. T., LAFORTUNE, S., MOREIRA, M. V., "Computation of minimal event bases that ensure diagnosability", *Discrete Event Dynamic Systems: Theory and Applications*, v. 22, n. 3, 2012.

[79] VIANA, G. S., BASILIO, J. C., "Codiagnosability of discrete systems revisited: a new necessary and sufficient condition and its applications", *Automatica (submitted for publication)*, 2017.

[80] VIANA, G. S., MOREIRA, M. V., BASILIO, J. C., "Codiagnosability of cyber-physical systems modeled by weighted automata", *IEEE Transactions on Automatic Control (submitted for publication)*, 2018.

[81] CHO, H., MARCUS, S. I., "Supremal and maximal sublanguages arising in supervisor synthesis problems with partial observations", *Theory of Computing Systems*, v. 22, n. 1, pp. 177–211, 1989.

[82] VIANA, G. S., ALVES, M. V. S., BASILIO, J. C., "Codiagnosticabilidade de sistemas a eventos discretos em rede com informações de temporização entre ocorrência de eventos e atrasos nos canais de comunicação". In:

XIII Simpósio Brasileiro de Automacão Inteligente, Porto Alegre, BRA, pp. 1247–1254, 2017.

[83] VIANA, G. S., ALVES, M. V. S., BASILIO, J. C., "Codiagnosability of Timed Networked Discrete-Event Systems subject to event communication delays and intermittent loss of observation". In: *Proceedings of the 56th IEEE Conference on Decision and Control,* Melbourne, AUS, pp. 4211–4216, 2017.

[84] SILVA, J. R., NOF, S. Y., "Manufacturing service: from e-work and service-oriented approach towards a product-service architecture", *IFAC-PapersOnLine*, v. 48, n. 3, pp. 1628–1633, 2015.