K-LOSS ROBUST CODIAGNOSABILITY OF DISCRETE-EVENT SYSTEMS

Vinicius de Souza Lima Oliveira

Dissertação apresentada ao Corpo Docente do Departamento de Engenharia Elétrica da Escola Politécnica da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre.

Orientador: Marcos Vicente de Brito Moreira

Rio de Janeiro
Setembro de 2022

K-LOSS ROBUST CODIAGNOSABILITY OF DISCRETE-EVENT SYSTEMS

Vinicius de Souza Lima Oliveira

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

Examinada por:

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

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

Prof. Marcelo Teixeira, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2022

de Souza Lima Oliveira, Vinicius

K-Loss Robust Codiagnosability of Discrete-Event Systems / Vinicius de Souza Lima Oliveira. – Rio de Janeiro: UFRJ/Escola Politécnica, 2022.

IX, 46 p.: il.; 29, 7cm.

Orientador: Marcos Vicente de Brito Moreira

Dissertação – UFRJ/Escola Politécnica/ Departamento de Engenharia Elétrica, 2022.

Referências Bibliográficas: p. 43 – 46.

1. Discrete Event Systems. 2. Fault Diagnosis. 3. Communication Networks. I. Vicente de Brito Moreira, Marcos. II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Departamento de Engenharia Elétrica. III. *K*-Loss Robust Codiagnosability of Discrete-Event Systems.

"Rather than love, than money,
than fame, give me truth"
Henry David Thoreau

Agradecimentos

Gostaria de agradecer primeiramente à Deus por me dar saúde e a oportunidade de concluir mais esta etapa em minha vida.
Agradeço à minha mãe Marta Maria por todo sacríficio, esforço e amor dedicado a mim durante sua vida e a todo incentivo a minha educação.
Agradeço ao meu orientador Marcos Moreira por todos os ensinamentos passados, pelas horas investidas e por acreditar em mim ao longo dos anos de graduação e pós graduação.
Agradeço ao Felipe Cabral por toda contribuição em meus trabalhos, pelos conselhos e suporte.
Agradeço à Maynara Aredes pela parceria e apoio ao longo do período letivo da pós graduação.
Agradeço também à COPPE-UFRJ, seu corpo docente e administração, e a todos aqueles que contribuíram para que eu finalizasse esta importante etapa.

Resumo da Dissertação apresentada à Escola Politécnica/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre

K-LOSS ROBUST CODIAGNOSABILITY OF DISCRETE-EVENT SYSTEMS

Vinicius de Souza Lima Oliveira

Setembro/2022

Orientador: Marcos Vicente de Brito Moreira

Departamento: Engenharia Elétrica

O problema de diagnóstico robusto contra perda de observações tem sido frequentemente estudado e diversas abordagens tem sido propostas na literatura, considerando arquiteturas centralizadas e descentralizadas. Nos métodos de diagnóstico robustos propostos atualmente, assume-se que alguns canais de comunicação entre sensores e diagnosticadores são confiáveis e as leituras dos sensores são sempre comunicadas aos diagnosticadores, enquanto os demais sensores, ou canais de comunicação entre sensores e diagnosticadores, estão sujeitos a falhas. Nesses trabalhos, são consideradas falhas permanentes ou intermitentes, e são obtidos os respectivos modelos das plantsa sujeitas à essas falhas. Uma característica do método de diagnóstico robusto considerando falhas intermitentes é que o sensor ou canal de comunicação defeituoso pode ou não se recuperar da falha, e falhas permanentes também são representadas no modelo da planta sujeita a falhas intermitentes. No entanto, em alguns casos, a falha de comunicação é temporária, *i.e.*, o canal de comunicação sempre se recupera da falha após um número limitado de perdas consecutivas de observação, como falhas devido a congestionamento de tráfego de dados ou perda temporária de conexão. Neste artigo, formulamos um problema de diagnóstico robusto onde assumimos que após um determinado número máximo de perdas consecutivas de observação de eventos em um canal de comunicação, ele deve se recuperar da falha e comunicar a observação dos eventos. A nova formulação leva a uma noção diferente de codiagnosabilidade robusta, chamada de codiagnosabilidade robusta à K-perdas. Apresentamos também um método para a verificação desta propriedade.

Abstract of Dissertation presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Master

GRADUATION PROJECT TITLE

Vinicius de Souza Lima Oliveira

September/2022

Advisor: Marcos Vicente de Brito Moreira

Department: Electrical Engineering

Recently, the problem of robust diagnosis against loss of event observations has been proposed in the literature, considering centralized and decentralized architectures. In the robust diagnosis methods proposed in the literature it is assumed that some communication channels between sensors and diagnosers are reliable and the sensor readings are always communicated to the diagnosers, while the other sensors, or communication channels between sensors and diagnosers, are subject to failures. In these works, permanent or intermittent failures are considered, and models of the plant subject to these failures are obtained. One characteristic of the robust diagnosis method considering intermittent failures is that the faulty sensor or communication channel may or may not recover from the failure, and permanent failures are also represented in the model of the plant subject to intermittent failures. However, in some cases, the communication failure is temporary, *i.e.*, the communication channel always recovers from the failure after a bounded number of consecutive observation losses, such as failures due to data traffic congestion or temporary connection loss. In this paper, we formulate a different problem of robust diagnosis where we assume that after a given maximum number of consecutive event observation losses in a communication channel, it must recover from the failure and communicate the observation of an event. The new formulation leads to a different notion of robust codiagnosability, called K-loss robust codiagnosability. We also present a method for the verification of this property.

Sumário

Lista de Figuras

Capítulo 1

Introduction

Automation systems are subject to several types of faults that can affect their expected behavior and reduce their reliability and performance. Therefore, the implementation of a fault diagnosis system of DESs is fundamental to identify the occurrence of a fault event in those types of system. In DES, events are defined as instantaneous occurrences, that can change the system state. In order to model DES, the most common formalisms are Automata and Petri nets [1–4]. Automata are directed graphs, in which states and events are represented by vertices and arcs. Petri nets are bipartite graphs, or bigraphs, in the sense that it has two types of nodes, defined as places and transitions, where nodes of the same type cannot be connected. In this work, automata are used to model DESs. In order to identify that a fault event has occurred, it is necessary to build a DES model of the faulty-free and post-fault behaviors of the system. Then, the fault occurrence can be diagnosed by tracking the observed traces generated by the system. Several works in the literature address the problem of fault diagnosis of DES modeled by automata [4–16].

Although fault diagnosis of DESs can be carried out using different types of architectures [4, 7, 17, 18], in the present work we consider only two of them: (i) the centralized architecture, where the occurrence of all observable events are communicated to a monolithic diagnoser and (ii) the decentralized architecture composed of several local diagnosers, where it is considered that the observation of the system events is distributed, and each local diagnoser observes part of the system events. The diagnosis methods presented in SAMPATH et al. [4], CARVALHO et al.[10, 11], CABRAL et al. [13], SANTORO et al. [14], consider that all system information regarding fault diagnosis, is available in a centralized architecture. However, there is a large number of systems where the diagnosis information is only available locally [7], which makes the decentralized [7–9, 19] more suitable for such systems. In DEBOUK et al. [7], different protocols for decentralized diagnosis are presented. The notion of diagnosability introduced in SAMPATH et al. [4] is extended to decentralized architectures, consisting of local diagnosers that communicate with a coordinator, in order to detect fault event occurrences. Several protocols for decentralized diagnosis, that determine the diagnostic information generated at each local site, the communication rules used by the local sites, and the decision rule for fault diagnosis applied by the coordinator are presented in DEBOUK et al. [7].

In the aforementioned works, the observations of the system events are communicated to diagnosers that detect and isolate the faults that have occurred in the system, within a bounded number of event observations. This communication can be carried out by using wired or wireless networks, which are susceptible to external interferences or malfunctioning. In addition, sensors may fail and the occurrence of events may not be transmitted to the diagnoser. When the fault to be detected is not the failure in the communication of the occurrence of system events, and information regarding event observations is lost, the diagnoser constructed based on the plant model can get stuck or provide an incorrect diagnosis decision [20]. Thus, for systems subject to loss of event observations, it is important to modify the diagnoser in order to be capable of diagnosing the system faults.

The problem of robust diagnosis, where the objective is to detect the occurrence of unobservable fault events using a set of sensors that themselves are subject to failures, such as, intermittent or permanent communication malfunction, has been addressed in [10, 11, 20–26]. The sensor communication failures can be divided into three types [27, 28]: (i) permanent failures; (ii) intermittent failures; and (iii) transient failures. Permanent failures are continuous and stable in time, and are, in general, related to hardware faults [27]. Intermittent failures are in general related to hardware faults or software malfunctioning [28]. This type of failure may eventually become permanent over time if the cause of the problem is not fixed by the maintenance staff. Different from the permanent and intermittent fault, the transient failures are not caused by an internal problem in the sensor or network. Transient failures are caused by external interferences, such as electromagnetic radiation, heat, weakness of wireless connections, data traffic congestion, and other environmental interferences [28–32]. They are often due to transient adverse conditions (e.g., a tunnel for GPS) but usually disappear quickly and are not considered a threat for the system's security. The main characteristic of this type of failure is that it is temporary and disappears after a short period of time. Thus, transient failures are part of the normal operation of sensors [30], and, when they occur, the diagnoser constructed based on the plant model can get stuck or provide an incorrect diagnosis decision. This shows that if it is not possible to guarantee that an external interference will not occur in the communication network, then the diagnoser constructed based on the plant model without taking into account transient failures cannot be used for diagnosis.

Several works address the problem of robust diagnosis of DESs against permanent or intermittent loss of observations [11, 20, 22, 33–36]. In [20], the problem of robust diagnosis of DES against permanent loss of event observations, considering a centralized architecture, is introduced, where it is supposed that any permanent sensor failure takes place prior to the first occurrence of the event recorded by this sensor. In [34], the same assumption is considered, and the robust diagnosis against permanent loss of observations is extended to the decentralized case. With a view to relaxing the assumption considered in [20, 34], allowing the sensor failure occurrence at any time, in [33], a new model of the plant subject to permanent sensor failures is proposed, leading to a different notion of diagnosis of DES subject to permanent sensor failures. The case of decentralized diagnosis of DES subject to permanent sensor failures, which also allows sensor failures at any time, is presented in [36].

In [11], the problem of robust diagnosis against intermittent loss of observations (RDILO) is formulated first for the centralized case, and then, extended to the decentralized case considering protocol 3 of [7], *i.e.*, the fault is diagnosed by at least one of the local diagnosers that infer the occurrence of the fault event based on their own observations. As in [20, 33, 34, 36], in [11], it is assumed that some of the system sensors are reliable and are always capable of communicating their readings to the diagnosers, while the other sensors, or communication channels between sensors and diagnosers, are subject to failures. In addition, it is assumed that if the sensor or communication channel fails, then it can or cannot recover from the failure, *i.e.*, the model proposed in [11] represents both the intermittent and permanent loss of event observations. The notions of robust diagnosability and robust codiagnosability against intermittent loss of observations are presented in [11].

Since the model proposed in [11] also represents the permanent loss of event observations, then, if we assume that the communication channel always recovers from failures after a bounded number of consecutive observation losses, *i.e.*, if we consider only temporary communication failures such as failures due to data traffic congestion, interference, or temporary connection loss, the notion of robust diagnosability proposed in [11] becomes very conservative. In addition, the case of unreliable communication of all observable events cannot be addressed using the method proposed in [11].

In this work, we formulate the problem of robust diagnosis against transient sensor communication failures. Since transient failures last for a short time, we assume that after a given maximum number of consecutive event observation losses in a communication channel, it must recover from the failure and communicate the observation of an event. The new formulation leads to a different notion of robust codiagnosability, called K-loss robust codiagnosability. A model of the plant subject to loss of observations is presented, and, based on this model, a method for the verification of K-loss robust codiagnosability is proposed. It is important to remark that it is the first work that addresses the effects of transient sensor communication failures in the fault diagnosis of DES. It is also important to remark that two papers have been published with the results of this work. In [37] we present the K-Loss robust diagnosability method and in [38] we extend it to the decentrilized case.

The present work is organized as follows: In Chapter 2 we present the notation and some background on fault diagnosis of DES. In Chapter 3, we present the notion of K-loss robust codiagnosability of DES, and propose a model of the plant that represents only temporary loss of observation of the system events. Finally, the conclusions are drawn in Chapter 4.

Capítulo 2

Fundamental Concepts of Discrete-Event Systems

This chapter presents theoretical foundations of discret event systems (DES) necessary for the understanding and elaboration of this work. The chapter is structured with the objective of dealing with the modeling and mathematical formalisms used to describe discrete-event systems.

In general, a system is defined as a set of elements combined by nature, or by man, in order to form a complex whole, performing a function that could not be performed by any of the components individually. The type of systems considered in this work are discrete event systems whose state space is a discrete set and whose state transitions are governed by the occurrence of events [1]. Events can be a specific action (such as someone pressing on a software button), a spontaneous occurrence (such as a system shutting down for unknown reason), or the result of a condition that is satisfied (such as the level of a temperature in a room exceeding a certain value).

Thus, DES is a dynamic system that evolves according to the occurrence of events and, in this way, a mathematical formalism capable of describing this type of system is necessary. This formalism must be able to determine the current state of the system and must have an evolution rule based on the occurrence of an event, or, more generally, a sequence of events.

Analogously, the set of events of a DES can be considered as the alphabet of the system. Thus, sequences of events form words and a set of words forms a language: in this sense, the set formed of all possible sequences generated by a system is called the generated language of the system. Languages determine the evolution of states in a DES from the occurrence of events and, therefore, have a function similar to that of differential equations to describe dynamic continous -time systems.

2.1 Languages

Before we introduce the concept of languages, we first present some notations. The set of events of a DES is represented by symbol Σ. The concatenation of events forms a trace, and the language of a system consists of the set of bounded length traces that can be executed by the system. A trace that does not contain any event is called the empty trace and is denoted by ε. The length of a trace s is represented by $\|s\|$ and, the length of the empty trace is equal to zero. In the sequel, we present the formal definition of a language [1].

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

Example 2.1 *Consider a system with event set $\Sigma = \{a, b\}$. The language $L = \{\varepsilon, a, ab, aab, abb\}$ is composed of five traces, and the length of the traces of L are $\|\varepsilon\| = 0, \|a\| = 1, \|ab\| = 2, \|aab\| = 3$ and $\|abb\| = 3$.*

Since languages are sets, the usual operations of sets such as union, intersection, difference, and complement, can be applied to languages. Moreover, there are other important operations that can be applied to languages and are presented in the sequel.

2.1.1 Languages Operations

The Kleene-closure operation over the event set Σ is represented as $\Sigma^\star$, and consists of all finite length traces that are constructed with elements of Σ, including the empty trace ε. Therefore, any language L defined over Σ is a subset of $\Sigma^\star$. This operation can also be applied to languages and is defined as follows.

Definition 2.2 *(Kleene-closure) Let $L \subseteq \Sigma^\star$, the Kleene-closure operation $L^\star$ is given by:*

$$L^\star = \{\varepsilon\} \cup L \cup LL \cup LLL \cup \dots$$

An important operation applied to traces and, consequently, to languages is the concatenation. A trace $s = abba$, for example, can be constructed by the concatenation of two traces ab and ba. Moreover, the empty trace ε is considered the identity element of the concatenation operation and, therefore, the trace ab is the concatenation of ε and ab, i.e., $\varepsilon ab = ab\varepsilon = ab$. This operation can also be formally defined for languages.

Definition 2.3 *(Concatenation) Let $L_a, L_b \subseteq \Sigma^\star$. The concatenation operation L_aL_b is defined as:*

$$L_aL_b = \{s = s_as_b : (s_a \in L_a) \text{ and } (s_b \in L_b)\}$$

The concatenation operation, when applied to languages L_a and L_b, generates a set containing the concatenation of each trace of set L_a with each trace of set L_b.

Consider a trace $s = abc$, where $a, b, c \in \Sigma^\star, a$ is a prefix of s, b is a subtrace of s and c if a suffix of s. Notice that, since $a, b, c \in \Sigma^\star$, then ε is always a prefix, a subtrace and a suffix of s. Now, the definition of prefix-closure of a language L can be stated.

Definition 2.4 *(Prefix-closure) Let $L \subseteq \Sigma^\star$, the prefix-closure operation $\bar{L}$ is given by:*

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

The prefix-closure of a language L is the set composed of all prefixes of all traces of L, thus $L \subseteq \bar{L}$. If $L = \bar{L}, i.e.$, if all prefixes of all traces of language L are also elements of L, this language is said to be prefix-closed.

Other important operations applied to traces and languages are the natural projection and the inverse projection, presented in the sequel.

Definition 2.5 *(Projection) Consider Σ_s and Σ_l, such that $\Sigma_s \subset \Sigma_l$. The natural projection $P_s^l : \Sigma_l^\star \to \Sigma_s^\star$ is defined recursively as follows:*

$$P_s^l(\varepsilon) = \varepsilon,$$
$$P_s^l(\sigma) = \begin{cases} \sigma, & \text{if } \sigma \in \Sigma_s \\ \varepsilon, & \text{if } \sigma \in \Sigma_l \backslash \Sigma_s \end{cases}$$
$$P_s^l(s\sigma) = P_s^l(s)P_s^l(\sigma), \text{ for all } s \in \Sigma_l^\star, \sigma \in \Sigma_l$$

Where the operator \represents set difference. The projection operation $P_s^l(s)$ erases all events $\sigma \in \Sigma_l \backslash \Sigma_s$ from the traces $s \in \Sigma_l^\star$. This operation can be extended to languages by applying the operation to all traces of the language.

The inverse projection operation is defined as follows.

Definition 2.6 *(Inverse projection) The inverse projection $P_s^{l^{-1}} : \Sigma_s^\star \to 2^{\Sigma_l^\star}$ is defined as:*

$$P_s^{l^{-1}}(t) = \left\{s \in \Sigma_l^\star : P_s^l(s) = t\right\}.$$

For a given trace $t \in \Sigma_s^\star$, the inverse projection operation $P_s^{l^{-1}}(t)$ generates a set formed of all traces s that can be constructed with the events of Σ_l whose projection P_s^l results in the trace t. This operation can also be extended to languages by applying the operation to all traces that belong to the language.

The language of a DES represents all traces that the system is capable of executing, i.e., it can be used to represent the system behavior. However, mainly in large and complex systems, the representation of the behavior of systems using only their languages is not easy and viable to work with. Therefore, it is necessary to use another formalism to describe DESs to facilitate the manipulation and analysis of systems with complex behavior. In this work we use automata to represent DESs, which are detailed in the next section.

2.2 Automata

An automaton is a device that is capable of representing a language according to well-defined rules [1, 2], and is formally defined as follows.

Definition 2.7 *(Automaton) A deterministic automaton, denoted by G, is a five-tuple:*

$$G = (Q, \Sigma, f, q_0, Q_m)$$

where Q is the set of states, Σ is the finite set of events, $f : Q \times \Sigma \rightarrow Q$ is the transition function, q_0 is the initial state, and Q_m is the set of marked states.

For the sake of simplicity, when the set of marked states Q_m is the empty set, i.e., $Q_m = \emptyset$, it will be omitted in the representation of the automaton.

We can also define $\Gamma_G : Q \rightarrow 2^{\Sigma}$ as the function of active events of a state of G, i.e., $\Gamma_G(q)$ is the set of all events $\sigma \in \Sigma$ for which the transition function $f(q, \sigma)$ is defined.

Automata can be represented by state transition diagrams, which are oriented graphs capable of reproducing all characteristics defined in G. The state transition diagram is formed of vertices, represented by circles, and edges, represented by arcs. The vertices represent the states of the system, and the edges represent the transitions between these states, which are labeled with events of Σ in order to represent which event correspond to each state transition. The initial state of the automaton is represented by an arc without an origin state. Example 2.2 shows an automaton and its state transition diagram.

Example 2.2 *Consider automaton G with state set $Q = \{0, 1, 2\}$ and event set $\Sigma = \{a, g\}$. The transition function of G is defined as: $f(0, a) = 1, f(0, g) = 0, f(1, g) = 2, f(2, a) = 1$ and, therefore, the active event function is given by: $\Gamma_G(0) = \{a, g\}, \Gamma_G(1) = \{g\}, \Gamma_G(2) = \{a\}$. The initial state q_0 is 0 and the set of marked states is $Q_m = \{1\}$. The state transition diagram of automaton G is shown in Figure 2.1.*

We can also define a path in an automaton G as a sequence $(q_1, \sigma_1, q_2, \ldots, q_{n-1}, \sigma_{n-1}, q_n)$, where $\sigma_i \in \Sigma, q_{i+1} = f(q_i, \sigma_i), i = 1, 2, \ldots, n-1$.

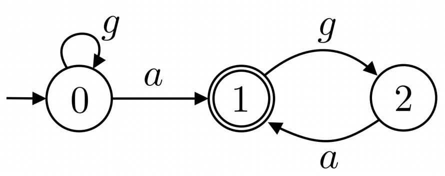

Figura 2.1: State transition diagram of automaton G of Example 2.2.

A path $(q_1, \sigma_1, q_2, \ldots, q_{n-1}, \sigma_{n-1}, q_n)$ is said to be cyclic, if $q_1 = q_n$. The set of states of a cyclic path forms a cycle.

Another important definition is the generated and marked languages of an automaton, presented as follows.

Definition 2.8 *(Generated and marked languages) The generated language of an automaton $G = (Q, \Sigma, f, q_0, Q_m)$ is defined as*

$$\mathcal{L}(G) = \{s \in \Sigma^\star : f(q_0, s) \ \textit{ is defined } \}$$

The marked language of G is defined as

$$\mathcal{L}_m(G) = \{s \in \mathcal{L}(G) : f(q_0, s) \in Q_m\}.$$

Notice that, in Definition 2.8, the domain of the transition function is considered to be extended, i.e., $f : Q \times \Sigma^\star \to Q$. In addition, notice that for any G such that $Q \neq \emptyset, \varepsilon \in \mathcal{L}(G)$.

In general, the language generated by $G, \mathcal{L}(G)$, is composed of all traces that, starting from the initial state, can be concatenated by following the transitions of the state transition diagram. Therefore, since a trace in $\mathcal{L}(G)$ is only feasible if all its prefixes are also feasible, the generated language $\mathcal{L}(G)$ is prefix-closed by definition. Moreover, if f is a total function over its domain, then $\mathcal{L}(G) = \Sigma^\star$. In this work, the language generated by $G, \mathcal{L}(G)$, is also referred to as L.

The marked language of $G, \mathcal{L}_m(G)$, is a subset of L, which contains all traces s that reach a marked state, i.e., all traces s such that $f(q_0, s) \in Q_m$. In this case, $\mathcal{L}_m(G)$ is not necessarily prefix-closed, since Q_m is not necessarily equal to Q.

The generated language of an automaton $G = (Q, \Sigma, f, q_0)$ is said to be live if $\Gamma_G(q) \neq \emptyset$ for all $q \in Q$.

In the following, we introduce some operations that can be applied to automata.

2.2.1 Operations on automata

There are several operations that can be used to modify the state transition diagram of a single automaton, or compose two or more automata [1]. These operations are separated into two groups: unary and composition operations [1].

Unary Operations

Unary operations are applied to a single automaton, in order to alter appropriately its state transition diagram, without change the automaton event set. In the sequel we present the definition of two unary operations.

Definition 2.9 *(Accessible part) Consider automaton* $G = (Q, \Sigma, f, q_0, Q_m)$.

The accessible part of $G, Ac(G)$*, is defined as:*

$$Ac(G) = (Q_{ac}, \Sigma, f_{ac}, q_0, Q_{ac,m}),$$

where $Q_{ac} = \{q \in Q : (\exists s \in \Sigma^\star) [f(q_0, s) = q]\}, Q_{ac,m} = Q_m \cap Q_{ac}$*, and* $f_{ac} : Q_{ac} \times \Sigma \to Q_{ac}$*. The transition function* f_{ac} *corresponds to* f *restricted to the smaller domain of the accessible states* Q_{ac}*.*

The operation of taking the accessible part of an automaton G erases the states that are not reachable from the initial state q_0 and its related transitions.

It is important to remark that the generated language of an automaton G is not modified with this operation. The formal definition of the coaccessible part of an automaton G is presented as follows [1].

Definition 2.10 *(Coaccessible part) Consider automaton* $G = (Q, \Sigma, f, q_0, Q_m)$. *The coaccessible part of* $G, \mathrm{CoAc}(G)$*, is defined as:*

$$\mathrm{CoAc}(G) = (Q_{coac}, \Sigma, f_{coac}, q_{0,\ coac}, Q_m),$$

where $Q_{coac} = \{q \in Q : (\exists s \in \Sigma^\star) [f(q, s) \in Q_m]\}, q_{0,\ coac} = q_0$ *if* $q_0 \in Q_{coac}$ *and* $q_{0,\ coac}$ *is not defined if* $q_0 \notin Q_{coac}$*, and* $f_{coac} : Q_{coac} \times \Sigma \to Q_{coac}$*. The operation of taking the coaccessible part of automaton G deletes all states q such that a path from q to a marked state does not exist.*

It is important to notice that the generated language of G can be reduced by applying the coaccessible part, i.e., $\mathcal{L}(\mathrm{CoAc}(G)) \subseteq \mathcal{L}(G)$, while the marked language is not modified.

Composition Operations

Composition operations applied to DESs modeled by automata allow us to combine two or more automata, resulting in a single automaton. Moreover, using composition operations it is possible to construct the model of a global system from the models of its individual components. In the following, we present two important composition operations [1].

Definition 2.11 *(Product composition) Let* $G_1 = (Q_1, \Sigma_1, f_1, q_{0,1}, Q_{m_1})$ *and* $G_2 = (Q_2, \Sigma_2, f_2, q_{0,2}, Q_{m_2})$ *be two automata. The product of* G_1 *and* G_2 *results in the automaton*

$$G_1 \times G_2 = Ac\left(Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, f_{1\times 2}, \left(q_{0,1}, q_{0,2}\right), Q_{m_1} \times Q_{m_2}\right)$$

where

$$f_{1\times 2}\left(\left(q_1, q_2\right), \sigma\right) = \begin{cases} \left(f_1\left(q_1, \sigma\right), f_2\left(q_2, \sigma\right)\right) & \text{if } \sigma \in \Gamma_{G_1}\left(q_1\right) \cap \Gamma_{G_2}\left(q_2\right) \\ \text{undefined,} & \text{otherwise.} \end{cases}$$

In the product composition an event can only occurs in the resulting automaton $G_1 \times G_2$ if it occurs simultaneously in G_1 and G_2. For this reason, the product operation is also known as a complete synchronous composition operation.

Due to the complete synchronization of the product operation, the generated language of $G_1 \times G_2$ is the intersection of the generated languages of the automata used in the composition, i.e., $\mathcal{L}\left(G_1 \times G_2\right) = \mathcal{L}\left(G_1\right) \cap \mathcal{L}\left(G_2\right)$. If $\Sigma_1 \cap \Sigma_2 = \emptyset$, then $\mathcal{L}\left(G_1 \times G_2\right) = \{\varepsilon\}$.

In general, systems are formed of several components that work together and whose event sets have private events, representing the internal behavior of each component, and common events, that represent the coupling behavior between components. The common way to obtain the global model of a system from the models of its components is applying the parallel composition, in which it is possible to maintain the private behavior of each component and capture the synchronism between the components. The formal definition of parallel composition operation is presented in the sequel.

Definition 2.12 *(Parallel composition) Let* $G_1 = (Q_1, \Sigma_1, f_1, q_{0,1}, Q_{m_1})$ *and* $G_2 = (Q_2, \Sigma_2, f_2, q_{0,2}, Q_{m_2})$ *be two automata. The parallel composition of* G_1 *and* G_2 *results in automaton*

$$G_1 \| G_2 = Ac\left(Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, f_{1\|2}, \left(q_{0,1}, q_{0,2}\right), Q_{m_1} \times Q_{m_2}\right)$$

where

$$f_{1\times 2}\left(\left(q_1,q_2\right),\sigma\right)=\begin{cases}\left(f_1\left(q_1,\sigma\right),f_2\left(q_2,\sigma\right)\right) & \textit{if } \sigma\in\Gamma_{G_1}\left(q_1\right)\cap\Gamma_{G_2}\left(q_2\right);\\ \left(f_1\left(q_1,\sigma\right),q_2\right) & \textit{if } \sigma\in\Gamma_{G_1}\left(q_1\right)\backslash\Sigma_2;\\ \left(q_1,f_2\left(q_2,\sigma\right)\right) & \textit{if } \sigma\in\Gamma_{G_2}\left(q_2\right)\backslash\Sigma_1;\\ \textit{undefined,} & \textit{otherwise.}\end{cases}$$

The parallel composition synchronizes the common events of components, i.e., an event $\sigma \in \Sigma_1 \cap \Sigma_2$ can only occur in the resulting automaton $G_1 \| G_2$ if it occurs in G_1 and G_2 simultaneously. On the other hand, private events of each automaton, i.e., the events in $(\Sigma_1 \backslash \Sigma_2) \cup (\Sigma_2 \backslash \Sigma_1)$, can be executed whenever possible in G_1 and G_2.

It is important to notice that if $\Sigma_1 = \Sigma_2$, then $G_1 \| G_2 = G_1 \times G_2$, since all transitions can only occur synchronously. In order to correctly define the language generated by $G_1 \| G_2$, it is necessary to consider the natural projections $P_i = (\Sigma_1 \cup \Sigma_2)^\star \rightarrow \Sigma_i^\star$, for $i = 1, 2$. Based on these projections, the generated language of $G_1 \| G_2$ is equal to $\mathcal{L}\left(G_1 \| G_2\right) = P_1^{-1}\left(\mathcal{L}\left(G_1\right)\right) \cap P_2^{-1}\left(\mathcal{L}\left(G_2\right)\right)$

An example of the product and parallel composition operations is presented in the sequel.

Example 2.3 *Consider automata G_1 and G_2 presented in Figure 2.2 and 2.3, respectively. The event set of G_1 and G_2 are, respectively, $\Sigma_1 = \{a, b\}$ and $\Sigma_2 = \{a, c\}$. Computing the product and parallel compositions of automata G_1 and G_2, we obtain automata $G_{prod} = G_1 \times G_2$ and $G_{par} = G_1 \| G_2$, respectively, presented in Figure 2.4 and Figure 2.5. Notice that since the only common event of G_1 and G_2 is event a, i.e., $\Sigma_1 \cap \Sigma_2 = \{a\}$, automaton G_{prod} has only transitions labeled with event a, while in automaton G_{par} it is possible to observe the concurrent behavior of G_1 and G_2, represented by transitions labeled with events b and c.*

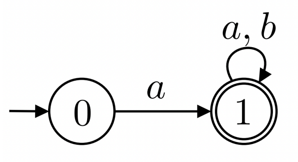

Figura 2.2: automaton G_1 of Example 2.3.

In the following, we present an important characteristic that must be taken into account when we use automata for modeling real systems.

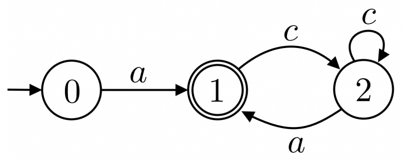

Figura 2.3: automaton G_2 of Example 2.3.

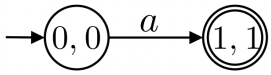

Figura 2.4: automaton G_{prod} of Example 2.3.

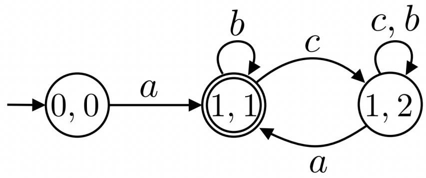

Figura 2.5: automaton G_{par} of Example 2.3.

2.2.2 Automata with partially observable events

In real systems it is not always possible to detect the occurrence of all events, due to limitations of the sensors used in the system. Events that do not have an associated sensor, such as fault events that do not cause immediate change in sensors readings, are called unobservable events. With the view to representing this, the event set Σ can be partitioned as $\Sigma = \Sigma_o \cup \Sigma_{uo}$, where Σ_o is the set of observable events and Σ_{uo} is the set of unobservable events. The observed language of an automaton G can be defined as $P_o(\mathcal{L}(G))$, where $P_o : \Sigma^\star \to \Sigma_o^\star$ is the natural projection.

In order to analyze a system with unobservable events, it is important to define the concept of unobservable reach of a state q, denoted as $UR(q)$. The unobservable reach of a given state $q \in Q$ represents the set of states that can be reached from q after the occurrence of a trace formed only of unobservable events, and it is formally defined as follows.

Definition 2.13 *(Unobservable reach) The unobservable reach of a state $q \in Q$, represented by $UR(q)$, is defined as:*

$$UR(q) = \{y \in Q : (\exists t \in \Sigma_{uo}^\star)\,[f(q,t) = y]\}\,.$$

The unobservable reach can also be defined for a set of states $B \in 2^Q$ as:

$$UR(B) = \bigcup_{q \in B} UR(q).$$

From the definitions of observed language and unobservable reach, it is possible to compute a deterministic automaton that generates the observed language of G with respect to $\Sigma_o, P_o(\mathcal{L}(G))$. This automaton is called observer of G and is denoted by $\mathrm{Obs}\,(G, \Sigma_o)$.

Definition 2.14 *(Observer automaton) The observer of automaton G with respect to the set of observable events $\Sigma_o, \mathrm{Obs}\,(G, \Sigma_o)$, is given by:*

$$\mathrm{Obs}\,(G, \Sigma_o) = (Q_{obs}, \Sigma_o, f_{obs}, q_{0,obs}, Q_{m,obs})$$

where $q_{obs} \subseteq 2^Q$. fobs, $q_{0,obs}$ and $Q_{m,obs}$ are obtained by following the steps of Algorithm 1 [6, 39]

Algorithm 1: Construction of Observer automaton.

Input : $G = (Q, \Sigma, f, q_0, Q_m)$
Output: $\mathrm{Obs}\,(G, \Sigma_o) = (Q_{obs}, \Sigma_o, f_{obs}, q_{0,obs}, Q_{m,obs})$

1 Define $q_{0,obs} \leftarrow UR\,(q_0)\,.Q_{obs} \leftarrow \{q_{0,obs}\}$ and $\widetilde{Q}_{\mathrm{obs}} \leftarrow Q_{\mathrm{obs}}\,.$;
2 $\bar{Q}_{obs} \leftarrow \widetilde{Q}_{obs}$ and $\widetilde{Q}_{obs} \leftarrow \emptyset$;
3 **for** $B \in \bar{Q}_{obs}$ **do**
4     $\Gamma_{obs}(B) \leftarrow \left(\bigcup_{q \in B} \Gamma_G(q)\right) \cap \Sigma_o$;
5   **for** $\sigma \in \Gamma_{obs}(B)$ **do**
6         $f_{\mathrm{obs}}\,(B, \sigma) \leftarrow UR(\{q \in Q : (\exists y \in B)[q = f(y, \sigma)]\})$;
7         $\widetilde{Q}_{obs} \leftarrow \widetilde{Q}_{obs} \cup f_{obs}(B, \sigma)$;
8   **end**
9 **end**
10 $Q_{obs} \leftarrow Q_{obs} \cup \widetilde{Q}_{obs}$;
11 Repeat steps 2 to 4 until all acessible part of $\mathrm{Obs}\,(G, \Sigma_o)$ is constructed.

We present now an example with the observer $Obs\,(G, \Sigma_o)$ of a system modeled by automaton G.

Example 2.4 *Consider automaton G presented in Figure 2.6 . The set of events is given by $\Sigma = \{a, b, \sigma_{uo}\}$, where $\Sigma_o = \{a, b\}$ and $\Sigma_{uo} = \{\sigma_{uo}\}$, and the set of states of G is $Q = \{0, 1, 2, 3\}$. The observer of G, $Obs\,(G, \Sigma_o)$, is shown in Figure 2.7. Let us assume that the system has executed trace $s = a\sigma_{uo}b$, then the observed trace is $P_o(s) = ab$, where $P_o : \Sigma^\star \rightarrow \Sigma_o^\star$. Notice that the state reached in $Obs(G^*, \Sigma_o)$ after the observation of trace ab is $\{2, 3\}$, which is the state estimate of G after* *observation of trace s. As it can be seen in Figure 2.7, each state of the observer $Obs(G, \Sigma_o)$ is the state estimate of G after the observation of a trace.*

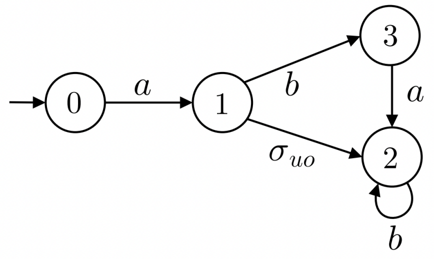

Figura 2.6: Automaton G of Example 2.4

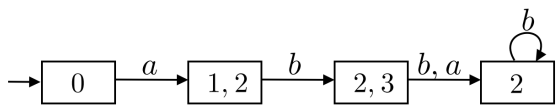

Figura 2.7: observer automaton of G, Obs (G, Σ_o) of Example 2.4

2.3 Fault Diagnosis of DES

Systems are subject to faults that can alter their expected behavior. Thus, it is necessary to define mechanisms that are capable of diagnosing the occurrence of fault events. In this section we present some preliminary concepts regarding diagnosis for DESs that will be important for the develpment of the following this work.

Let $\Sigma_f \subseteq \Sigma_{uo}$ be a set of events associated with faults of the system. In general, the set $\Sigma_f = \bigcup \Sigma_{f_i}, i = 1, 2, \ldots, n$, where i means the types of fault that can occur in the plant and each set Σ_{f_i} is formed of events that model faults that are somehow correlated.

For the sake of simplicity, in this work we assumed that there is only one fault event, i.e., $\Sigma_f = \sigma_f$. There is no loss of generality in the results presented in this work by making this assumption since, for systems with more than one fault type, each fault type can be considered separately.

In the sequel, we present the definition of fault-free and fault traces of a system.

Definition 2.15 *(Fault and Faulty-free traces) A fault trace is a trace of events s such that σ_f is one of the events that form s. A faulty-free trace, on the other hand, does not contain the event σ_f.*

We present below the formal definition of language diagnosability L [4].

Definition 2.16 *(Language Diagnosability)Let L be a language generated by an automaton G and suppose that L is live. Thus, L is diagnosable with respect to projection $P_o : \Sigma^\star \rightarrow \Sigma_o^\star$ and $\Sigma_f = \{\sigma_f\}$ if the following condition holds true [4]:*

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

where the diagnosis D is

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

According to definition 2.16, the language generated by an automaton G will be diagnosable with respect to the set of observable events Σ_o, projection P_o and the set of fault events $\Sigma_f = \{\sigma_f\}$, if the occurrence of event σ_f can be detected after a finite number of transitions after the occurrence of σ_f using traces of observable events only.

Depending on how the information about the dynamic evolution of the system is available, the diagnosis system can be divided in different classes [40]: Centralized, Decentralized, Distributed, Modular and Synchronous. In the present work, we will consider two of those possible classes: Centralized and Decentralized diagonis. We present the definition of this classes of automata in the sequel [7].

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

2.3.1 Centralized Diagnosis

The centralized diagnoser of a plant G is an automaton that can be used to verify the diagnosability of L and also for fault diagnosis, and it is presented in [4] and [1]. This diagnoser is constructed based on automaton G_l computed from the plant model G, where G_l is obtained by labeling the states of G according to the traces generated by the system, such that if a state of G is reached by a trace that contains the fault event σ_f, then it is labeled with F, otherwise it is labeled with N. After G_l has been obtained, the diagnoser automaton G_d is computed by making the observer of G_l with respect to its observable events, $G_d = Obs\,(G_l, \Sigma_o)$. The diagnoser automaton G_d is formally defined as follows.

Definition 2.17 *(Diagnoser automaton) The diagnoser automaton G_d obtained for system G with respect to the fault set Σ_f and observable events set Σ_o is given by:*

$$G_d = (Q_d, \Sigma_o, f_d, q_{0,d})$$

where $Q_d \subseteq 2^{Q \times \{N,F\}}$.

The transition function f_d, and the initial state $q_{0,d}$ are defined according to Algorithm 2, presented below.

Algorithm 2: Construction of Diagnoser Automaton.

Input : $G = (Q, \Sigma, f, q_0)$
Output: Diagnoser Automaton G_d
1 Define automaton $A_l = (Q_l, \Sigma_l, f_l, q_{0,l})$ where $Q_l = \{N, F\}$, $f_l(N, \sigma_f) = F$, $f_l(F, \sigma_f) = F$ and $q_{0,l} = N$;
2 Compute Automaton $G_l = G \parallel A_l$;
3 Compute the diagnoser automaton $G_d = Obs(G_l, \Sigma_o)$

It is important to notice that automaton G_l generates the same language as automaton G. Moreover, the states of G_l are of the form $q_l = (q, N)$, such that $q \in Q$, if q is reached by a faulty-free trace, and $q_l = (q, F)$ if q is reached by a fault trace. The generated language of G_d is the natural projection of the generated language of G, L, i.e., $\mathcal{L}(G_d) = P_o(L)$.

Since G_d is constructed from the observer automaton of G_l, the states of G_d are state estimates of G_l after the observation of a trace. If G_d reaches a state labeled only with the label F, the fault event has certainly occurred and it is diagnosed. A state of G_d labeled only with N indicates that the fault has not been executed by the system. States of G_d that have the labels N and F are called uncertain states, indicating that after the observation of a trace, a fault trace or a faulty-free trace with the same projection has been executed by the system.

In order to use G_d to verify the diagnosability of L, it is necessary to search for indeterminate cycles in G_d. An indeterminate cycle is an uncertain cycle, i.e., a cycle formed by uncertain states, that is associated with at least two cycles in G_l, one that has only states labeled with N, and one that has only states labeled with F. If there is an indeterminate cycle in G_d, then the language generated by G, L, is not diagnosable, otherwise, L is diagnosable.

Example 2.5 *Consider a plant represented by automaton G, where $\Sigma_o = \{a, b, c\}$ and $\Sigma_{uo} = \{\sigma_f\}$, shown in Figure 2.8. In order to obtain its diagnoser, the parallel composition $G \| A_l$ is first computed, which is shown in Figure 2.9. After that, we obtain the observer automaton of $G \| A_l$. The diagnoser of plant G, G_d, is shown* *in Figure 2.10. Notice that the initial state of* $G_d, \{1N, 3F\}$*, has both labels* Y *and* N*. This happens because event* σ_f *is unobservable and, if it occur, the diagnoser will not realize its occurrence; then the system can either be in states* $1N$ *or* $3F$*. As a consequence, the diagnoser cannot state, for sure, whether the fault event has occurred, i.e., it is uncertain with respect to the occurrence of the fault event. On the other hand, another event that can occur when the system is in the initial state is the observable event* b*; thus, if event* b *occurs, the diagnoser must indicate that the system is in state* $\{2N\}$ *and, thus, it will be certain that the fault event has not occurred. Supposing that from the initial state event* σ_f *has occurred in the plant, the diagnoser will remain in state* $\{1N, 3F\}$*, but, when event* a *occurs, then the diagnoser changes the current state,* $\{1N, 3F\}$*, to state* $\{4F\}$*, meaning that the diagnoser is now certain that event* σ_f *occurred.*

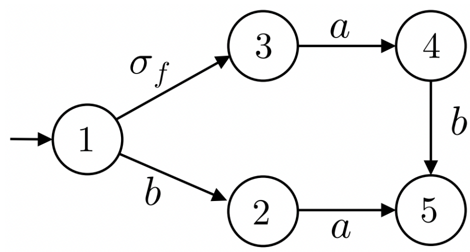

Figura 2.8: Automaton G of Example 2.5

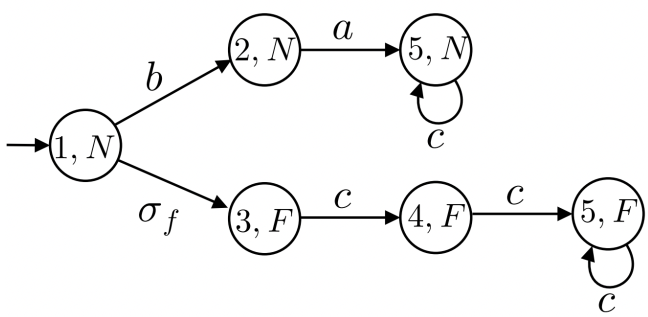

Figura 2.9: Parallel composition between G and A_l of Example 2.5

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

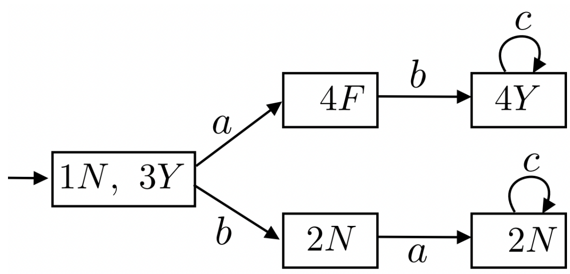

Figura 2.10: Automaton $G_d = Obs(G \parallel A_l)$ of Example 2.5

2.3.2 Decentralized Diagnosis

In order to solve the problems associated with the distributed nature of some systems, in [7] the decentralized structure preseted in Figure 2.11 has been proposed. In this structure, the reading of sensors is decentralized. Each diagnoser module observes part of the system events based on the information from the sensors connected to it, i.e., based on the observable events of each diagnoser Σ_{o_i} where $i = \{1, 2, \ldots, n\}$, where n is the number of diagnosers. Each diagnoser processes the received information and communicates the result to the coordinator. The coordinator receives the information from the diagnosers and processes it according to well defined rules and makes a decision with respect to the fault occurrence.

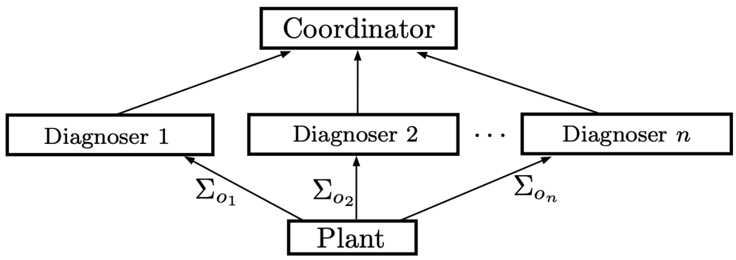

Figura 2.11: Decentralized arcquitecture

The diagnosis of a language L depends on a set of elements called protocol, which is composed of rules used to generate local diagnosis, communication rules between the module and the coordinator, decision rules used by the coordinator to diagnose the fault, and the projections $P_{o_i} : \Sigma^\star \to \Sigma_{o_i}^\star, i = 1, 2, \ldots, n$, associated with each diagnoser. In protocol 3 of [7], a fault is diagnosed when, at least, one local diagnoser identifies its occurrence. In the sequel, we present the definition of codiagnosability of a language L, but, before it, we will review the notion of normal and fault traces.

According to Definition 2.15 a fault trace is a trace of events s such that σ_f is one of its events and a normal trace, on the other hand, does not contain event σ_f.

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

Definition 2.18 *(Language codiagnosability) Let L and $L_N \subset L$ be prefix closed languages generated by G and G_N, respectively, and $P_{o_i} : \Sigma^\star \to \Sigma_{o_i}^\star, i = 1, \ldots, n$ projection operations. Then, L is codiagnosable with respect to the projections P_{o_i}e Σ_f if*

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

According to Definition 2.18 L is codiagnosable with respect to P_{o_i} and Σ_f if, and only if, for all fault traces $s_F = st$ of arbitrarily long length after the occurrence of the fault event, there do not exist traces $s_{N_i} \in L_N$, where s_{N_j} is not necessarily different from s_{N_k} for $j \neq k$, such that $P_{o_i}(s_{N_i}) = P_{o_i}(s_F)$, for all $i \in \{1, 2, \ldots, n\}$.

2.3.3 Codiagnosability Verification

Codiagnosability verification of the language of a discrete event system is the first step to develop a fault diagnosis system for a DES. Some works in literature have addressed the problem of codiagnosability verification of a DES [8][39]. In the present work, we will adopt the algorithm presented in [39] as the basis for the problem of robust diagnosability against intermittent loss of observation to be considered later in this work. We use algorithm below, originally presented in [39]

We present the necessary and sufficient condition for codiagnosability of DES proposed in [39] as follows.

Theorem 2.1 *Let L and L_N be $(L_N \subset L)$ the prefix closed languages generated by G and G_N, respectively, and let Σ_f be the set of fault events. Then, L is not diagnosable with respect to $P_{o_i} : \Sigma^\star \to \Sigma_{o_i}^\star, i = 1, \ldots, n$, and Σ_f if, and only if, there exists a cyclic path $cl := \left(y_V^k, \sigma_k, y_V^{k+1}, \ldots, y_V^l, \sigma_l, y_V^k\right)$, where $l \geq k > 0$, in V, that satisfies the following condition:*

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

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

Example 2.6 *Consider the system G depicted in Figure 2.12 and suppose we want to verify the codiagnosability of L with respect to $P_{o_i} : \Sigma^\star \to \Sigma_{o_i}^\star, i = 1, 2$ and*

Σ_f, *where* $\Sigma = \{a, b, c, \sigma_f\}$, $\Sigma_{o_1} = \{a, c\}, \Sigma_{o_2} = \{b, c\}$, *and* $\Sigma_f = \{\sigma_f\}$. *In steps 1 and 2, automata* G_N *and* G_F *presented in Figure 2.13 and 2.14, respectively, are computed. In the sequel, automata* $G_{N,1}$ *and* $G_{N,2}$ *are built in Step 3. In this example, automata* $G_{N,1}$ *and* $G_{N,2}$ *are equal to automaton* G_N *and, thus, are omitted. Finally, the verifier automaton* G_V *is shown in Figure 2.15. Notice that there are no cycles in* G_V *satisfying conditions (2.7). Therefore, the language generated by* G *is codiagnosable with respect to* P_{o_i} *and* Σ_f.

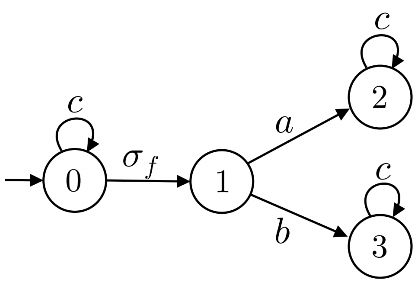

Figura 2.12: Automaton G of Example 2.6

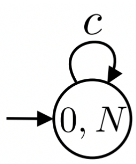

Figura 2.13: Automaton G_N of Example 2.6

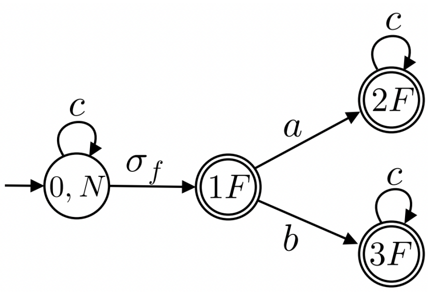

Figura 2.14: Automaton G_F of Example 2.6

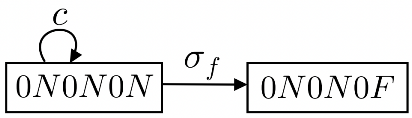

Figura 2.15: Automaton G_V of Example 2.6

2.4 Robust diagnosis against intermittent loss of observations

The problem of robust diagnosis against intermittent loss of observations, considering the decentralized diagnosis architecture proposed in Protocol 3 of [7], is introduced in [11], where it is considered that sensors, or the communication between sensors and local diagnosers, may fail intermittently. In this case, the set of observable events is partitioned as $\Sigma_o = \Sigma_{ilo} \dot{\cup} \Sigma_{nilo}$, where Σ_{ilo} is the set of events subject to intermittent loss of observations, and Σ_{nilo} is the set of observable events that are not subject to loss of observations. It is important to remark that, in [11], if the communication of an event $\sigma \in \Sigma_{ilo}$ to a local diagnoser fails, then the communication of σ to the other local diagnosers that also observe σ will also fail. In order to characterize the intermittent loss of observations, set $\Sigma'_{ilo} = \{\sigma' : \sigma \in \Sigma_{ilo}\}$ is created, where σ' is an unobservable event that models the loss of observation of event σ due to sensor malfunction or communication fault.

The following definition is presented in [11] to obtain the language observed by the local diagnosers, in the decentralized architecture, due to the intermittent loss of observations of the events in Σ_{ilo}.

Definition 2.19 *(Dilation) Let $\Sigma = \Sigma_{ilo} \dot{\cup} \Sigma_{nilo} \dot{\cup} \Sigma_{uo}$, $\Sigma'_{ilo} = \{\sigma' : \sigma \in \Sigma_{ilo}\}$, and $\Sigma_{dil} = \Sigma \cup \Sigma'_{ilo}$. Then, the dilation function is the mapping $D : \Sigma^\star \to 2^{\Sigma^\star_{dil}}$ where*

$$
\begin{aligned}
&D(\varepsilon) = \varepsilon,\\
&D(\sigma) = \begin{cases} \sigma, & \text{if } \sigma \in \Sigma \backslash \Sigma_{ilo},\\ \{\sigma, \sigma'\}, & \text{if } \sigma \in \Sigma_{ilo}, \end{cases}\\
&D(s\sigma) = D(s)D(\sigma), s \in \Sigma^\star, \sigma \in \Sigma.
\end{aligned}
$$

□

The dilation operation D can be extended from traces to languages by applying it to all traces in the language, that is, $D(\mathcal{L}(G)) = \bigcup_{s \in \mathcal{L}(G)} D(s)$. Language $D(\mathcal{L}(G))$ describes the behavior of plant G, subject to intermittent loss of observations of the events in Σ_{ilo}, under the assumptions considered in [11].

According to Definition 18, the language observed by the local diagnosers, when the communication of the sensors associated with Σ_{ilo} are subject to intermittent faults, is given by $P_{dil,o_i}(D(\mathcal{L}(G)))$, for $i = 1, 2, \ldots, n$, where $P_{dil,o_i} : \Sigma_{dil}^{\star} \to \Sigma_{o_i}^{\star}$ is a projection. In the sequel, we present the definition of robust codiagnosability against intermittent loss of observations presented in [11].

Definition 2.20 *(Robust codiagnosability of DES against intermittent loss of observations) The live language $\mathcal{L}(G)$ is robustly codiagnosable with respect to dilation D, projections $P_{dil,o_i} : \Sigma_{dil}^{\star} \to \Sigma_{o_i}^{\star}$, $i = 1, 2, \ldots, n$, and Σ_f, if the following holds true:*

$$(\exists z \in \mathbb{N})(\forall s \in \mathcal{L}(G) \backslash L_N)(\forall st \in \mathcal{L}(G) \backslash L_N, \|t\| \geq z) \Rightarrow R_I,$$

where the robust codiagnosability condition R_I is

$$\begin{aligned}&(\exists i \in \{1, 2, \ldots, n\})\\ &[P_{dil,o_i}(D(st)) \cap P_{dil,o_i}(D(\omega)) = \emptyset, \forall \omega \in L_N].\end{aligned}$$

□

According to Definition 2.20, a system is said to be robustly codiagnosable with respect to D, P_{dil,o_i}, $i = 1, 2, \ldots, n$, and Σ_f if, and only if, there do not exist an arbitrarily long length fault trace st, and fault-free traces $\omega_i \in L_N$, $i = 1, 2, \ldots, n$, such that the dilation of st, represented by $D(st)$, generates a trace with the same local observation as a trace in the dilation of ω_i, $D(\omega_i)$, for all $i \in \{1, 2, \ldots, n\}$.

It is important to remark that, according to Definition 18, $D(\sigma) = \{\sigma, \sigma'\}$ for all events $\sigma \in \Sigma_{ilo}$. Thus, after any occurrence of an event $\sigma \in \Sigma_{ilo}$, it is possible to observe this event, or to not observe it, which is represented by event $\sigma' \in \Sigma'_{ilo}$, *i.e.*, the observation of event σ can be, at any time, permanently lost or eventually recovered after losing it. This shows that the dilation function proposed in [11] models intermittent and permanent loss of observations. However, in some cases, only temporary losses may occur in the system. In these cases, the method proposed in [11] leads to a conservative result, and cannot be used. We propose in the next section a new definition of robust diagnosis that does not encompass the case of permanent fault of communication. The following example illustrates the concepts of robust diagonosability against intermittent loss of observations, presented in definition 2.20.

Example 2.7 *Consider automata G_1 and G_2 whose state transition diagrams are depicted in Figure 2.16 and 2.17, respectively, and assume, for both automata, that $\Sigma_0 = \{a, b, c\}, \Sigma_{ilo} = \{a\}$ and $\Sigma_f = \{\sigma_f\}$. The objective here is to verify if the*

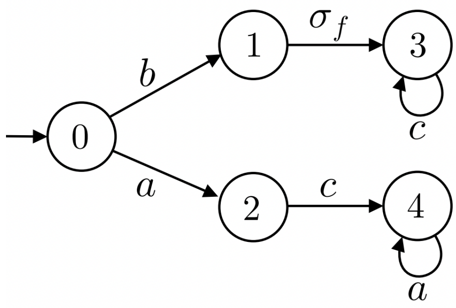

Figura 2.16: Automaton G_1 of Example 2.7

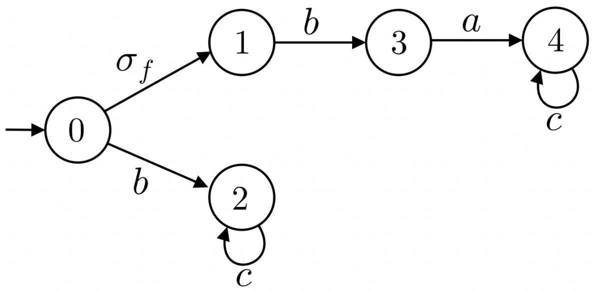

Figura 2.17: Automaton G_2 of Example 2.7

languages generated by G_1 and G_2 (L_1 and L_2, respectively) are robustly diagnosable with respect to D, P_o and $\Sigma_f = \{\sigma_f\}$.

Consider, initially, automaton G_1. From Figure 2.16, we see that the faulty traces of L_1 are $s'_Y = b\sigma_f c^n, n \in \mathbb{N}$. Following the steps in the robust diagnosability condition R_D given in Definition 21, we may conclude that $D(s'_Y) = \{b\sigma_f c^n\} \Rightarrow P_{dil,o}\,[D(s'_Y)] = \{bc^n\}$.

Let $L_{1,\ dil}$ denote the language generated by automaton $G_{1,\ dil}$, shown in Figure 2.18. It is not difficult to see that, since $L_{1,\ dil} = \overline{\{b\sigma_f\}\{c\}^* \cup \{ac\}\{b\}^* \cup \{a'c\}\{b\}^*}$, then $P^{-1}_{dil,o}\,\{P_{dil,o}[D(s'_Y)]\} \cap L_{1,\ dil} = \{b\sigma_f c^n\}$. Since $P^{-1}_{dil,o}\,\{P_{dil,o}\,[D(s_Y)]\} \cap L_{1,\ dil}$ has only the fault traces s'_Y, we may conclude that L_1 is robustly diagnosable with respect to D, P_o and $\Sigma_f = \{\sigma_f\}$.

Consider now the automaton G_2 depicted in Figure 2.17. In this case, the unique faulty traces of L_2 are $s''_Y = \sigma_f bac^n, n \in \mathbb{N}$. Following the robust diagnosability condition R_D, we have $D(s''_Y) = \{\sigma_f bac^n, \sigma_f ba'c^n\} \Rightarrow P_{dil,o}[D(s''_Y)] = \{bac^n, bc^n\}$.

From automaton $G_{2,\ dil,\ ,}$ shown in Figure 2.19, we obtain $L_{2_{dil}} = \overline{\{a\}\{c\}^* \cup \{\sigma_f ba\}\{c\}^* \cup \{\sigma_f ba'\}\{c\}^*}$. Then, we have $P^{-1}_{dil\ ,o}\{P_{dil,o}[D(s''_Y)]\} \cap L_{2,\ dil} = \{\sigma_f bac^n, \sigma_f ba'c^n, bc^n\}$.

Since there is a normal trace in $P^{-1}_{dil,\ ,}\{P_{dil\ ,o}[D(s''_Y)]\} \cap L_{2,\ dil}$, we may conclude that L_2 is not robustly diagnosable with respect to D, P_0 and $\Sigma_f = \{\sigma_f\}$. The lack of robust diagnosability of L_2 with respect to D, P_o and $\Sigma_f = \{\sigma_f\}$ can be explained

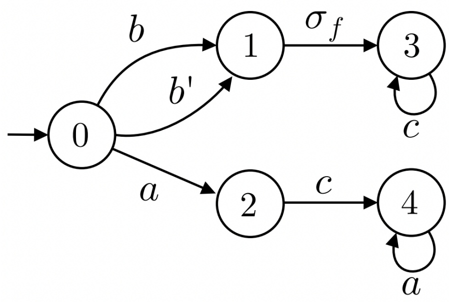

Figura 2.18: Automaton $G_{1_{dil}}$ of Example 2.7

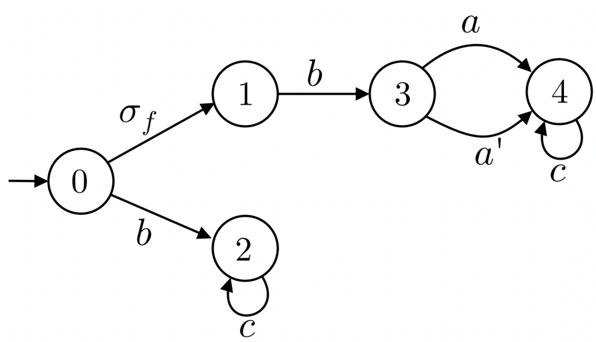

Figura 2.19: Automaton $G_{2_{dil}}$ of Example 2.7

as follows: it is not possible to assure if the normal traces bc^n occurred or the faulty traces $s''_Y = \sigma_f bac^n$ have occurred and, somehow, the observable event a has not been recorded by the diagnoser.

2.5 Final Remarks

In this chapter, the background of DESs, such as the definition of language, operations and the automaton formalism used to represent DESs were presented. Diagnoser Automata, which is used to verify the diagnosability of a language L was also presented, along with the concept of centralized and decentralized diagnosis and codiagnosability verification .

After presenting the background needed, we presented the concept of robust diagnosis against interminttent loss of observation [10], as a basis for the work presented in the following chapter, in wich we introduce the concept of K-loss robust codiagnosability that aims to address the problem decentralized diagnosis of DES against temporary faults in the communication.

Algorithm 3: Codiagnosability Verification

1 Input: $G = (X, \Sigma, \Gamma, f, x_0)$;

2 Output: $V = (X_V, \Sigma_V, \Gamma_V, f_V, x_{0_V})$;

3 Compute automaton G_N that models the normal behavior of G.

3.1 Define $\Sigma_N = \Sigma \backslash \Sigma_f$

3.2 Build the single state automaton $A_l^N = \big(\{N\}, \Sigma_N, f_l^N, x_{0,N}\big)$, where $f_l^N(N, \sigma) = N$, for all $\sigma \in \Sigma_N$, and $x_{0,N} = N$

- Build the automaton $G_N = G \times A_l^N$
- Redefine the set of events of G_N as Σ_N, i.e., $G_N = (X_N, \Sigma_N, f_N, (x_0, N))$;

4 Compute automaton G_F that models the fault behavior of automaton G.

- Build the label automaton $A_l = \big(\{N, F\}, \Sigma_f, f_l^{NF}, x_{0,NF}\big)$ where $x_{0,NF} = N$, $f_l^{NF}(N, \sigma_f) = F$, and $f_l^{NF}(F, \sigma_f) = F$, for all $\sigma_f \in \Sigma_f$
- Compute $G_l = G \| A_l$ and mark all states labeled with F.
- Compute $G_F = \mathrm{CoAc}\,(G_l)$. 3. Rename the unobservable events of G_{N_i}, as follows.

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

- Define the following set:

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

- Define $\Sigma_{R_i} = \Sigma_{o_i} \cup \Sigma_f \cup \Sigma'_{R_i}$.
- Define the following renaming function

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

where

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

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

7 Compute the verifier automaton $G_V = G_{R_1} \| G_{R_2} \| \ldots \| G_{R_n} \| G_F$

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

- $$\exists j \in \{k, k+1, \ldots, \ell\} \text{ such that, for some } y_V^j, \big(y_l^j = F\big) \wedge (\sigma_j \in \Sigma)$$
- If the answer is yes, them L is not codiagnosable with respect to P_{o_i} and Σ_f.

Capítulo 3

K-Loss Robust Codiagnosability

In the robust diagnosis methods proposed in the literature [10, 11, 16, 20, 23–25, 34] it is assumed that some communication channels between sensors and diagnosers are reliable and the sensor readings are always communicated to the diagnosers, while the other sensors, or communication channels between sensors and diagnosers, are subject to failures. In these works, permanent or intermittent failures are considered, and models of the plant subject to these failures are obtained. One characteristic of the robust diagnosis method considering intermittent failures proposed in [11, 20] is that the faulty sensor or communication channel may or may not recover from the failure, and permanent failures are also represented in the model of the plant subject to intermittent failures. However, in some cases, the communication failure is temporary, *i.e.*, the communication channel always recovers from the failure after a bounded number of consecutive observation losses, such as failures due to data traffic congestion or temporary connection loss. In this work, we formulate a different problem of robust diagnosis where we assume that after a given maximum number of consecutive event observation losses in a communication channel, it must recover from the failure and communicate the observation of an event. The new formulation leads to a different notion of robust codiagnosability, called K-loss robust codiagnosability. We also present a method for the verification of this property.

3.1 Problem formulation

In this work we consider the decentralized diagnosis scheme proposed in Protocol 3 of [7], and assume that each local diagnoser LD_i, $i = 1, \ldots, n$, receives information about the occurrence of observable events of the plant through different communication channels, as shown in Figure 3.1. Let Σ_{o_i} be the set of observable events of local diagnoser LD_i, and let $ch_{i,j}$, $j = 1, \ldots, \eta_i$, denote the communication channels that transmit the observation of the events belonging to Σ_{o_i} to diagnoser LD_i, where η_i is the number of channels that communicate the observation of events to LD_i.

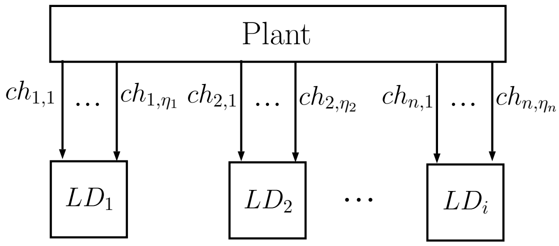

Figura 3.1: Decentralized diagnosis scheme.

Let $\Sigma_{o_{i,j}} \subset \Sigma_{o_i}$ denote the set of observable events that are communicated through channel $ch_{i,j}$. Assume that each event $\sigma_o \in \Sigma_{o_i}$ is transmitted through a unique communication channel to LD_i, *i.e.*, all sets $\Sigma_{o_{i,j}}$, for $j = 1, \ldots, \eta_i$, are disjoint. Thus, $\Sigma_{o_i} = \Sigma_{o_{i,1}} \dot\cup \Sigma_{o_{i,2}} \dot\cup \ldots \dot\cup \Sigma_{o_{i,\eta_i}}$.

We assume, in this work that all communication channels $ch_{i,j}$ may be subject to temporary loss of observations, *i.e.*, after a bounded number of consecutive event observation losses in channel $ch_{i,j}$, it must recover from the failure and communicate the observation of an event in $\Sigma_{o_{i,j}}$. Let $k_{i,j}$ denote the maximum number of consecutive losses of observation in channel $ch_{i,j}$, and form tuple $K_i = (k_{i,1}, k_{i,2}, \ldots, k_{i,\eta_i})$. Thus, the languages that are observed by local diagnosers LD_i, $i = 1, 2, \ldots, n$, depend on the maximum number of consecutive losses of observation described in $K = (K_1, K_2, \ldots, K_n)$. It is also important to remark that, differently from [11], an event observation can be successfully communicated to a local diagnoser, and not communicated to a different local diagnoser due to a temporary communication failure. The reason for this is that, in this work, we associate the temporary failures with the communication channels that are used to transmit the event observations to the local diagnosers, and not to faulty sensors.

The following example illustrates the observations of a trace of $\mathcal{L}(G)$ by two local diagnosers subject to temporary failures in the communication channels.

Example 3.1 *Consider the plant automaton depicted in Figure 3.2, where the set of events is given by $\Sigma = \{a, b, c, d, \sigma_f\}$ and the fault event set is $\Sigma_f = \{\sigma_f\}$. Consider that the decentralized diagnosis scheme is composed of two local diagnosers LD_i, $i = 1, 2$, that detect the occurrence of the fault event σ_f based on their own observations. Assume that two channels $ch_{1,1}$ and $ch_{1,2}$ are used to communicate the occurrence of the events in $\Sigma_{o_{1,1}} = \{a\}$ and $\Sigma_{o_{1,2}} = \{c\}$, respectively, to local diagnoser LD_1, and only one channel $ch_{2,1}$ communicates the occurrence of the events in $\Sigma_{2,1} = \{b, c, d\}$ to local diagnoser LD_2. Thus, $\Sigma_{o_1} = \{a, c\}$ and $\Sigma_{o_2} = \{b, c, d\}$. Let us also suppose that $K_1 = (k_{1,1}, k_{1,2}) = (1, 0)$ and $K_2 = (k_{2,1}) = (1)$, i.e., the communication of at most one event through channel $ch_{1,1}$ may be consecutively lost, no event is lost in channel $ch_{1,2}$, and the communication of at most one event through channel $ch_{2,1}$*

may be consecutively lost. It is important to remark that, in this example, if no loss of observation is considered in the communication channels, then the language of the system $\mathcal{L}(G)$ is codiagnosable with respect to P_{o_i}, $i = 1, 2$, and Σ_f.

Let us consider now that trace $s = abcad$ is executed by the system. Then, the following three traces may be observed by diagnoser LD_1: aca, ca, and ac. Trace aca is observed when there is no loss of observation in channel $ch_{1,1}$; trace ca is observed when the first occurrence of event a is not transmitted through channel $ch_{1,1}$; and trace ac is observed when the second occurrence of a is not transmitted through channel $ch_{1,1}$. Note that event c is always transmitted through channel $ch_{1,2}$, since $k_{1,2} = 0$. In addition, the following four traces may be observed by local diagnoser LD_2 due to communication failures in channel $ch_{2,1}$: bcd, cd, bd, and bc. □

Then, if the system executes trace $s = a\sigma_u bc$, the following six traces may be observed by the diagnoser: abc, ab, bc, ac, a, c. Sequence abc is observed when there is no loss of observation in both channels; trace ab (resp. bc) is observed when event c (resp. a) is not transmitted in channel ch_1; trace ac corresponds to the case that the observation of event b is lost in channel ch_2; trace a is observed when both b and c are lost; and trace c corresponds to the case that the observation of event b is lost in ch_2, event a is lost in ch_1, but since $k_1 = 1$, then event c must be transmitted through ch_1. □

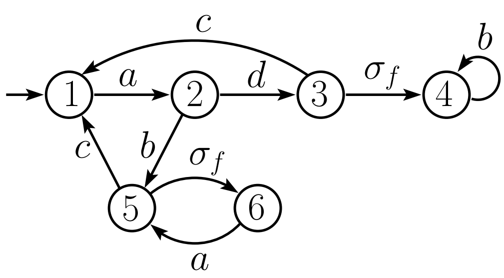

Figura 3.2: Automaton G of Example 3.1.

3.2 Definition of K-loss robust codiagnosability

In order to distinguish the occurrence of an event $\sigma \in \Sigma_{o_i}$ in the plant from its observation by local diagnoser LD_i, we create event σ_{s_i} that represents the successful observation of σ by LD_i. In this regard, let $\Sigma_{s_{i,j}} = \{\sigma_{s_i} : \sigma \in \Sigma_{o_{i,j}}\}$ be the set of events that are successfully communicated through channel $ch_{i,j}$, and let $\Sigma_{s_i} = \cup_{j=1}^{\eta_i} \Sigma_{s_{i,j}}$ denote the set of events successfully communicated to diagnoser LD_i. In addition, to represent the failure in the communication of an event $\sigma \in \Sigma_{o_i}$, let us create event σ_{l_i}. Thus, the set of events whose observation is lost in channel

$ch_{i,j}$ is defined as $\Sigma_{l_{i,j}} = \{\sigma_{l_i} : \sigma \in \Sigma_{o_{i,j}}\}$. Thus, the set of events that represents the loss of observation by diagnoser LD_i can be defined as $\Sigma_{l_i} = \cup_{j=1}^{\eta_i} \Sigma_{l_{i,j}}$. Then, the following set can be defined:

$$\Sigma_i = \Sigma \cup \Sigma_{l_i} \cup \Sigma_{s_i}, \tag{3.1}$$

where the events in $\Sigma \cup \Sigma_{l_i}$ are unobservable to local diagnoser LD_i, and the events in Σ_{s_i} are observable.

In order to obtain all possible observations of a trace $s \in \mathcal{L}(G)$ by diagnoser LD_i, it is necessary first to introduce the insertion function $I_i : \Sigma^\star \to 2^{\Sigma_i^\star}$, where $I_i(\varepsilon) = \varepsilon$, $I_i(\sigma) = \{\sigma\sigma_{l_i}, \sigma\sigma_{s_i}\}$, if $\sigma \in \Sigma_{o_i}$, $I_i(\sigma) = \{\sigma\}$, if $\sigma \in \Sigma_{uo_i}$, and $I_i(s\sigma) = I_i(s)I_i(\sigma)$, for all $s \in \Sigma^\star$ and $\sigma \in \Sigma$. Let us also consider the projection operations $P_i : \Sigma_i^\star \to \Sigma^\star$ and $P_{s,l}^{i,j} : \Sigma_i^\star \to (\Sigma_{l_{i,j}} \cup \Sigma_{s_{i,j}})^\star$, for $j = 1, \ldots, \eta_i$. Then, we can define the following function.

Definition 3.1 *A function that models the temporary failure in communication channels $ch_{i,j}$, $j = 1, \ldots, \eta_i$, from plant G to local diagnoser LD_i, such that the maximum number of consecutive losses of observation of each channel $ch_{i,j}$ is $k_{i,j}$, is a mapping*

$$\begin{aligned} \Psi_i : \Sigma^\star &\to 2^{\Sigma_i^\star} \\ s &\mapsto \Psi_i(s) \end{aligned}$$

where $w \in \Psi_i(s)$, if w satisfies the following conditions:

(i) $P_i(w) = s$;

(ii) $w \in I_i(s)$;

(iii) For all $j \in \{1, 2, \ldots, \eta_i\}$, we have that for all $\mu'_j, \mu'''_j \in (\Sigma_{l_{i,j}} \cup \Sigma_{s_{i,j}})^\star$ and $\mu''_j \in \Sigma_{l_{i,j}}^\star$, such that $P_{s,l}^{i,j}(w) = \mu'_j\mu''_j\mu'''_j$, then $\|\mu''_j\| \leq k_{i,j}$. □

The domain of function Ψ_i can be extended to consider languages as usual, *i.e.*, $\Psi_i(L) = \cup_{s \in L} \Psi_i(s)$, for all languages $L \subseteq \Sigma^\star$.

In Condition *(i)*, $P_i(w)$ removes from $w \in \Psi_i(s)$ all events inserted by function I_i to represent the successful or unsuccessful communication of an event to local diagnoser LD_i. If $P_i(w) = s$, then we know that w has been obtained from s. Condition *(ii)* ensures that w has been obtained using function I_i. Thus, only after the occurrence of an observable event $\sigma \in \Sigma_{o_i}$, event $\sigma_{l_i} \in \Sigma_{l_i}$ or $\sigma_{s_i} \in \Sigma_{s_i}$ can be generated.

Finally, Condition *(iii)* establishes that the maximum number of consecutive events of $\Sigma_{l_{i,j}}$ in w is $k_{i,j}$, for all $j \in \{1, 2, \ldots, \eta_i\}$, which shows that the maximum number of consecutive losses in each channel $ch_{i,j}$ represented in Ψ_i is $k_{i,j}$. Thus, the language observed by diagnoser LD_i is given by $P_{s_i}(\Psi_i(\L(G)))$, where $P_{s_i} : \Sigma_i^\star \rightarrow \Sigma_{s_i}^\star$ projects a trace from $\Psi_i(\L(G))$ to a trace in $\Sigma_{s_i}^\star$, observable by diagnoser LD_i.

Example 3.2 *Let us consider the same plant presented in Example 3.1, where* $\Sigma_{o_1} = \{a, c\}$ *and* $\Sigma_{o_2} = \{b, c, d\}$. *In addition, let us consider, as in Example 3.1, that* $K_1 = (1, 0)$ *and* $K_2 = (1)$. *Consider that the system executes trace* $s = abcad$. *Then, according to Definition 3.1, the following traces can be generated due to the temporary communication losses in channel* $ch_{1,1}$, $\Psi_1(s) = \{aa_{s_1}bcc_{s_1}aa_{s_1}d, aa_{l_1}bcc_{s_1}aa_{s_1}d, aa_{s_1}bcc_{s_1}aa_{l_1}d\}$, *whose projection to* $\Sigma_{s_1}^\star$ *is* $P_{s_1}(\Psi_1(s)) = \{a_{s_1}c_{s_1}a_{s_1}, c_{s_1}a_{s_1},\ a_{s_1}c_{s_1}\}$, *which corresponds to the possible observations of* LD_1. *In addition, due to temporary losses in channel* $ch_{2,1}$, *the following traces can be generated* $\Psi_2(s) = \{abb_{s_2}cc_{s_2}add_{s_2}, abb_{l_2}cc_{s_2}add_{s_2}, abb_{s_2}cc_{l_2}add_{s_2}, abb_{s_2}cc_{s_2}add_{l_2}\}$, *whose projection to* $\Sigma_{s_2}^\star$ *is* $P_{s_2}(\Psi_2(s)) = \{b_{s_2}c_{s_2}d_{s_2}, c_{s_2}d_{s_2}, b_{s_2}d_{s_2}, b_{s_2}c_{s_2}\}$, *which corresponds to the possible observations of* LD_2. □

In the sequel, we present the definition of K-loss robust codiagnosability.

Definition 3.2 *Let* $\mathcal{L}(G)$ *be the live language generated by* G, *and let* $ch_{i,j}$, *for* $i = 1, 2, \ldots, n$ *and* $j = 1, \ldots, \eta_i$, *be the communication channels between plant and local diagnosers* LD_i, *subject to the maximum consecutive losses of observation* $k_{i,j}$. *Then,* $\mathcal{L}(G)$ *is robustly codiagnosable with respect to* $K = (K_1, \ldots, K_n)$, Ψ_i, P_{s_i}, $i = 1, \ldots, n$, *and* Σ_f, *if*

$$(\exists z \in \mathbb{N})(\forall s \in \mathcal{L}(G) \backslash L_N)(\forall st \in \mathcal{L}(G) \backslash L_N, \|t\| \geq z) \Rightarrow R_K$$

where the K*-loss robust codiagnosability condition* R_K *is*

$$(\exists i \in \{1, 2, \ldots, n\})[P_{s_i}(\Psi_i(st)) \cap P_{s_i}(\Psi_i(\omega)) = \emptyset, \forall \omega \in L_N].$$

□

According to Definition 3.2, language $\mathcal{L}(G)$ is not robustly codiagnosable with respect to K, Ψ_i, P_{s_i}, $i = 1, \ldots, n$, and Σ_f, if there exist a fault trace st with arbitrarily long length after the occurrence of the fault event, and fault-free traces ω_i, $i = 1, \ldots, n$, such that the temporary loss of observations create ambiguous observations for all local diagnosers LD_i, $i = 1, \ldots, n$.

Remark 3.1 *It is important to remark that the concept of* K*-Loss robust diagnosability, introduced in [37] is a particular case of the* K*-Loss robust codiagnosability method presented in this chapter, when we consider the monolithic architecture diagnosis scheme or, equivalently, considering* $j = 1$ *in definition 3.1.*

3.2.1 Model of the plant subject to temporary event communication failures

We present in this section an automaton model of the plant subject to temporary loss of observations. In order to do so, we first model the behavior of each communication channel $ch_{i,j}$ considering the maximum number of consecutive losses $k_{i,j}$. The automaton model of the communication channel $ch_{i,j}$ is denoted as $\Delta_{i,j} = (Q_{ij}, \Sigma \cup \Sigma_{s_{i,j}} \cup \Sigma_{l_{i,j}}, f_{ij}, q_{ij,0}, Q_{ij,m})$, and is computed using Algorithm 4.

Algorithm 4: Construction of automaton $\Delta_{i,j}$.

Input : $k_{i,j}, \Sigma, \Sigma_{s_{i,j}}, \Sigma_{l_{i,j}}$
Output: $\Delta_{i,j} = (Q_{ij}, \Sigma \cup \Sigma_{s_{i,j}} \cup \Sigma_{l_{i,j}}, f_{ij}, q_{ij,0}, Q_{ij,m})$
1 $q_{ij,0} \leftarrow (\varepsilon, 0)$;
2 $c \leftarrow 0$, $Q_{ij} \leftarrow \emptyset$, $Q_{ij,m} \leftarrow \emptyset$;
3 **while** $c \leq k_{i,j}$ **do**
4     $Q_{ij} \leftarrow Q_{ij} \cup \{(\varepsilon, c)\}$, $Q_{ij,m} \leftarrow Q_{ij,m} \cup \{(\varepsilon, c)\}$;
5   **for** $\sigma \in \Sigma$ **do**
6   **if** $\sigma \in \Sigma_{o_{i,j}}$ **then**
7             $Q_{ij} \leftarrow Q_{ij} \cup \{(\sigma, c)\}$;
8             $f_{ij}((\varepsilon, c), \sigma) = (\sigma, c)$;
9             $f_{ij}((\sigma, c), \sigma_{s_i}) = (\varepsilon, 0)$;
10   **end**
11   **if** $\sigma \in \Sigma_{uo_i} \cup (\Sigma_{o_i} \setminus \Sigma_{o_{i,j}})$ **then**
12             $f_{ij}((\varepsilon, c), \sigma) = (\varepsilon, c)$;
13   **end**
14   **end**
15     $c \leftarrow c + 1$;
16 **end**
17 $c \leftarrow 0$;
18 **while** $c < k_{ij}$ **do**
19   **for** $\sigma \in \Sigma_{o_{i,j}}$ **do**
20         $f_{ij}((\sigma, c), \sigma_{l_i}) = (\varepsilon, c + 1)$;
21   **end**
22     $c \leftarrow c + 1$;
23 **end**

Each state of Q_{ij} is a tuple formed of the last event $\sigma \in \Sigma_{o_{i,j}}$ generated by the plant or ε, and a counter c, that indicates the number of consecutive losses of observation of σ. Thus, the initial state defined in line 1 of Algorithm 4 is equal to $(\varepsilon, 0)$ indicating that no event belonging to $\Sigma_{o_{i,j}}$ has been generated and the counter is set to 0. Then, in line 8, if an event $\sigma \in \Sigma_{o_{i,j}}$ is generated by the plant, a transition from state (ε, c) to state (σ, c) is created to indicate the occurrence of σ. To represent that σ is successfully transmitted to the diagnoser, in line 9 a new transition labeled with σ_{s_i} is created from (σ, c) to $(\varepsilon, 0)$, and the counter is reset.

In lines 11 to 13, self-loops are introduced in the states (ε, c), labeled with events in $\Sigma_{uo_i} \cup (\Sigma_{o_i} \setminus \Sigma_{o_{i,j}})$, to allow the occurrence of these events only in these states. By doing so, after the occurrence of an event $\sigma \in \Sigma_{o_{i,j}}$ in the plant, its observation represented by event σ_{s_i}, or the loss of observation of σ represented by σ_{l_i}, must be generated before another event occurrence in the plant. In lines 19 to 21, the loss of observation of event σ is represented by increasing counter c with the transition from state (σ, c) to $(\varepsilon, c+1)$, labeled with event σ_{l_i}. The maximum number of consecutive occurrences of $\sigma\sigma_{l_i}$, without the occurrence of trace $\sigma\sigma_{s_i}$ is, as it can be seen in lines 18 to 23, equal to $k_{i,j}$. It is important to remark that only the states of the form (ε, c) are marked in line 4.

Since, according to Algorithm 1, $(k_{i,j}+1)$ states of the form (ε, c), and $|\Sigma_{o_{i,j}}| \times (k_{i,j}+1)$ states of the form (σ, c), for $\sigma \in \Sigma_{o_{i,j}}$, are created in $\Delta_{i,j}$, then the number of states of $\Delta_{i,j}$ is $(|\Sigma_{o_{i,j}}|+1) \times (k_{i,j}+1)$. In addition, since there are transitions from states (ε, c) labeled with all events in Σ, and only two transitions at most leaving states (σ, c) labeled with σ_{l_i} and σ_{s_i}, then the maximum number of transitions of $\Delta_{i,j}$ is bounded by $(k_{i,j}+1) \times |\Sigma| + |\Sigma_{o_{i,j}}| \times 2$. Thus, the overall complexity of Algorithm 4 is $O(k_{i,j} \times |\Sigma|)$.

Example 3.3 *Consider the plant presented in Example 3.1, with* $\Sigma_{o_1} = \{a, c\}$, $\Sigma_{o_2} = \{b, c, d\}$, $K_1 = (1, 0)$ *and* $K_2 = (1)$. *Then,* $\Delta_{1,1}$, $\Delta_{1,2}$, *and* $\Delta_{2,1}$, *obtained using Algorithm 4, are depicted in Figures 3.3, 3.4, and 3.5, respectively.* □

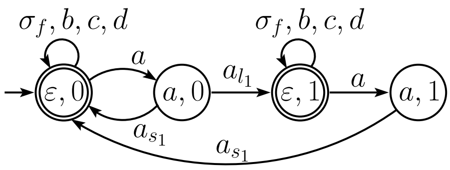

Figura 3.3: Automaton $\Delta_{1,1}$ of Example 3.3.

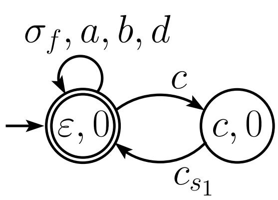

Figura 3.4: Automaton $\Delta_{1,2}$ of Example 3.3.

After computing the communication channel models $\Delta_{i,j}$, for $j = 1, 2, \ldots, \eta_i$, the model of the plant that generates the observation of the system events for local diagnoser LD_i, due to communication failures in the channels $ch_{i,j}$, can

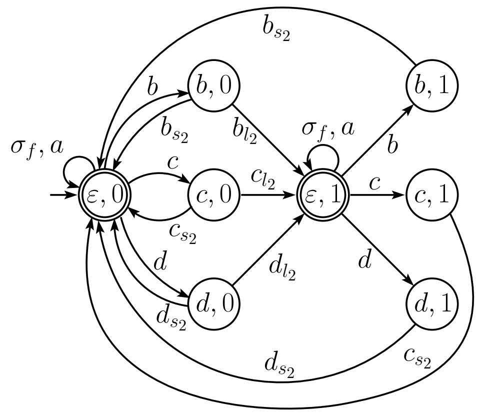

Figura 3.5: Automaton $\Delta_{2,1}$ of Example 3.3.

be obtained in two steps: (i) mark all states of the plant G; and (ii) compute $G_{t_i} = (Q_{t_i}, \Sigma_i, f_{t_i}, q_{t_i,0}, Q_{t_i,m}) = G \| \Delta_{i,1} \| \Delta_{i,2} \| \ldots \| \Delta_{i,\eta_i}$. The following theorem shows that the marked language of G_{t_i} is equal to $\Psi_i(\mathcal{L}(G))$.

Theorem 3.1 $\mathcal{L}_m(G_{t_i}) = \Psi_i(\mathcal{L}(G))$.

Proof. According to Algorithm 4, after the occurrence of an event $\sigma \in \Sigma_{o_{i,j}}$, communicated through channel $ch_{i,j}$, an unmarked state of G_{t_i} is reached, and then, event σ_{s_i} or σ_{l_i} must be generated, reaching a marked state of G_{t_i}. In addition, if an unobservable event $\sigma \in \Sigma_{uo_i}$ occurs, a marked state of G_{t_i} is also reached. Thus, all traces w reaching a marked state of G_{t_i} belongs to $I_i(s)$, where $s \in \Sigma^\star$ is the corresponding trace executed by the system. Moreover, for the same reason, we have that $P_i(w) = s$. Thus, Conditions (i) and (ii) of Definition 3.1 are satisfied. Finally, since, according to lines 18 to 23 of Algorithm 4, the maximum number of consecutive occurrences of subsequence $\sigma\sigma_{l_i}$, where $\sigma \in \Sigma_{o_{i,j}}$, counting only traces $\sigma\sigma_{l_i} \in (\Sigma_{o_{i,j}} \cup \Sigma_{l_{i,j}})^\star$, is equal to $k_{i,j}$ for all traces $w \in \mathcal{L}_m(G_{t_i})$, then Condition (iii) of Definition 3.1 is also satisfied. Thus, $\mathcal{L}_m(G_{t_i}) \subseteq \Psi_i(\mathcal{L}(G))$.

Let $w \in \Psi_i(\mathcal{L}(G))$. Then, according to Definition 3.1, w is formed of the concatenation of traces of the form: (i) σ, if $\sigma \in \Sigma_{uo_i}$; and (ii) $\sigma\sigma_{s_i}$ and $\sigma\sigma_{l_i}$, if $\sigma \in \Sigma_{o_i}$. In addition, there exists a trace $s \in \Sigma^\star$ such that $P_i(w) = s$, and the maximum number of consecutive occurrences of subsequence $\sigma\sigma_{l_i}$ in w, where $\sigma \in \Sigma_{o_{i,j}}$, counting only traces $\sigma\sigma_{l_i} \in (\Sigma_{o_{i,j}} \cup \Sigma_{l_{i,j}})^\star$, is $k_{i,j}$. Let us consider that trace s is executed by the system. According to Algorithm 4, after the occurrence of an event $\sigma \in \Sigma_{o_{i,j}}$, event σ_{s_i} or σ_{l_i} must be generated, and only after that, a marked state of G_{t_i} is reached. In addition, after the occurrence of an unobservable event $\sigma \in \Sigma_{uo_i}$, no event in $\Sigma_{s_i} \cup \Sigma_{l_i}$ is generated, and a marked state of G_{t_i} is reached. Moreover, according to lines 18 to 23 of Algorithm 4, the number of consecutive occurrences of $\sigma\sigma_{l_i}$, where

$\sigma \in \Sigma_{o_{i,j}}$, is limited to $k_{i,j}$. Thus, any trace $w \in \Psi_i(\mathcal{L}_m(G))$ belongs to the marked language of G_{t_i}, which implies that $\Psi_i(\mathcal{L}(G)) \subseteq \mathcal{L}_m(G_{t_i})$. ■

Theorem 3.1 shows that G_{t_i} can be used to obtain the language observed by diagnoser LD_i due to the loss of observations in the communication channels $ch_{i,j}$, for $j = 1, 2, \ldots, \eta_i$. Since $G_{t_i} = G\|\Delta_{i,1}\|\ldots\|\Delta_{i,\eta_i}$, the worst-case computational complexity of G_{t_i} is $O(|Q| \times 2^{\eta_i} \times |\Sigma|^{\eta_i+1} \times \prod_{j=1}^{\eta_i} k_{i,j})$.

Example 3.4 *The model of the plant G of Example 3.1 subject to temporary loss of observations for local diagnosers LD_1 and LD_2, $G_{t_1} = G\|\Delta_{1,1}\|\Delta_{1,2}$ and $G_{t_2} = G\|\Delta_{2,1}$, respectively, are presented in Figures 3.6 and 3.7.* □

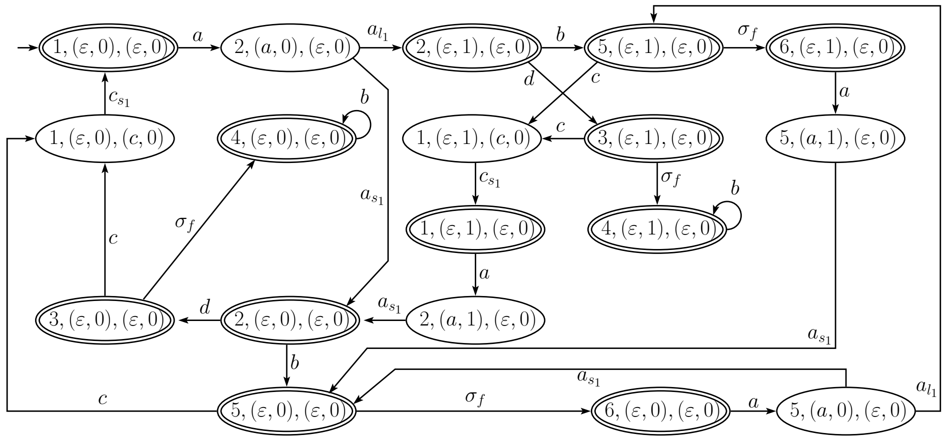

Figura 3.6: Automaton model $G_{t_1} = G\|\Delta_{1,1}\|\Delta_{1,2}$ of the system subject to loss of observations of Example 3.4.

3.3 K-loss robust codiagnosability verification

In order to present the K-loss robust codiagnosability verification method, it is first necessary to define the renaming function $\rho_i : \Sigma_i \setminus \Sigma_f \to \Sigma_{\rho_i}$, where

$$\rho_i(e) = \begin{cases} e^{\rho_i}, \text{ if } e \in (\Sigma \cup \Sigma_{l_i}) \setminus \Sigma_f \\ e, \text{ otherwise.} \end{cases} \tag{3.2}$$

The domain of ρ_i can be extended to $(\Sigma_i \setminus \Sigma_f)^\star$ as $\rho_i(se) = \rho_i(s)\rho_i(e)$, for all $s \in (\Sigma_i \setminus \Sigma_f)^\star$ and $e \in \Sigma_i \setminus \Sigma_f$, and $\rho_i(\varepsilon) = \varepsilon$. Function ρ_i can be applied to a language $M \subseteq (\Sigma_i \setminus \Sigma_f)^\star$ as $\rho_i(M) = \cup_{s \in M} \rho_i(s)$.

In Algorithm 5 we present the construction of the verifier automaton V for the verification of K-loss robust codiagnosability based on the verifier proposed in [39]. Note that, since, according to line 5 of Algorithm 5, $V = \|_{i=1}^n V_i$ and, acccording to line 4, $V_i = G_{\rho_i}\|G_{F_i}$, then each state of verifier V has the following

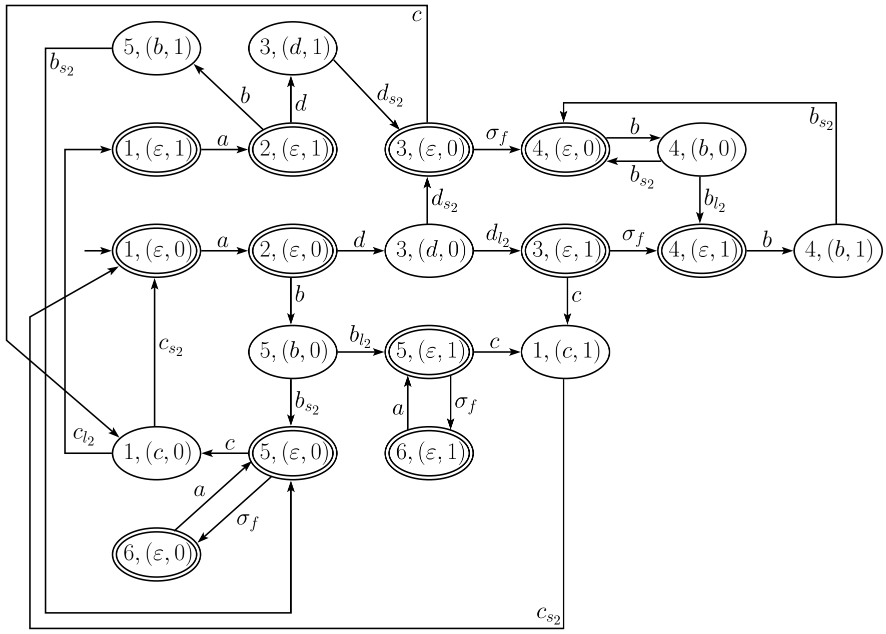

Figura 3.7: Automaton model $G_{t_2} = G \| \Delta_{2,1}$ of the system subject to loss of observations of Example 3.4.

Algorithm 5: Construction of Verifier V.

Input : $G_{t_i} = (Q_{t_i}, \Sigma_i, f_{t_i}, q_{t_i,0}, Q_{t_i,m})$, $i = 1, \ldots, n$

Output: $V = (Q_V, \Sigma_V, f_V, q_{V,0}, \emptyset)$

1 Compute the fault-free automata $G_{N_i} = (Q_{N_i}, \Sigma_i \setminus \Sigma_f, f_{N_i}, q_{N_i,0}, \emptyset)$, $i = 1, \ldots, n$, by eliminating all transitions of G_{t_i} labeled with σ_f, and taking its accessible part;

2 Compute automata $G_{F_i} = (Q_{F_i}, \Sigma_i, f_{F_i}, q_{N_i,0}, \emptyset)$ that model the fault behavior of automata G_{t_i}, for $i = 1, \ldots, n$, as presented in [39];

3 Compute the renamed fault-free automata $G_{\rho_i} = (Q_{N_i}, \Sigma_{\rho_i}, f_{\rho_i}, q_{N_i,0}, \emptyset)$, $i = 1, \ldots, n$, with $f_{\rho_i}(q_N, e^{\rho_i}) = f_{N_i}(q_N, e)$, where $e^{\rho_i} = \rho_i(e)$, for all $e \in \Sigma_i \setminus \Sigma_f$ and $q_N \in Q_{N_i}$;

4 Compute automata $V_i = G_{\rho_i} \| G_{F_i}$, $i = 1, \ldots, n$;

5 Compute verifier $V = \|_{i=1}^{n} V_i$.

---

form $q_V = ((q_{N_1}, q_{F_1}), (q_{N_2}, q_{F_2}), \ldots, (q_{N_n}, q_{F_n}))$, where $q_{N_i} \in Q_{N_i}$ and $q_{F_i} \in Q_{F_i}$, for $i = 1, \ldots, n$. In addition, according to [39], each state of G_{F_i}, computed in line 2 of Algorithm 5, has the form $q_{F_i} = (q_{t_i}, \ell)$, where q_{t_i} is a state of G_{t_i}, and $\ell \in \{N, F\}$ is a label that is assigned to state q_{F_i}. If q_{t_i} is reached after the occurrence of the fault event σ_f, then label F is assigned to q_{F_i}. Otherwise, label N is assigned to q_{F_i}. If state q_{F_i} of a state $q_{V_i} = (q_{N_i}, q_{F_i})$ has label F, then q_{V_i} is said to be a fault state of V_i, and if all states q_{F_i}, $i = 1, \ldots, n$, of a state q_V have label F, then q_V is said to be a fault state of V.

In the following lemma, we present a necessary and sufficient condition for the diagnosability of the language generated by the i-th local model, $\mathcal{L}(G_{t_i})$, with respect to P_{s_i} and Σ_f.

Lemma 3.1 *$\mathcal{L}(G_{t_i})$ is diagnosable with respect to P_{s_i} and Σ_f if, and only if, there does not exist a cyclic path $cl_i = (q^x_{V_i}, \sigma_x, q^{x+1}_{V_i}, \sigma_{x+1}, \ldots, q^y_{V_i}, \sigma_y, q^x_{V_i})$, $y \geq x > 0$, in verifier V_i, that satisfies the following condition:*

$$\begin{aligned} & q^p_{V_i} \text{ is a fault state } \forall p \in \{x, \ldots, y\}, \text{ and for some} \\ & p \in \{x, \ldots, y\}, \sigma_p \in \Sigma. \end{aligned} \tag{3.3}$$

Proof. The proof is presented in [39] for a generic plant G, projection P_o, and set of fault events Σ_f. ∎

Note that, since $V_i = G_{\rho_i} \| G_{F_i}$, and G_{ρ_i} is obtained from G_{N_i} by renaming the events in $\Sigma \cup \Sigma_{l_i}$, then each trace of V_i is associated with a fault-free trace $\omega_i \in \mathcal{L}(G_{N_i})$ and a fault trace $s_i \in \mathcal{L}(G_{F_i})$ such that $P_{s_i}(\omega_i) = P_{s_i}(s_i)$ [41]. Thus, the existence of a cyclic path cl_i in V_i satisfying condition (3.3) is a necessary and sufficient condition for the existence of an arbitrarily long length fault trace s_i that cannot be diagnosed by local diagnoser LD_i.

The necessary and sufficient condition for K-loss robust codiagnosability, based on verifier V computed in Algorithm 5, is presented as follows.

Theorem 3.2 *$\mathcal{L}(G)$ is robustly codiagnosable with respect to K, Ψ_i, P_{s_i}, $i = 1, \ldots, n$, and Σ_f if, and only if, there does not exist a cyclic path $cl = (q^x_V, \sigma_x, q^{x+1}_V, \sigma_{x+1}, \ldots, q^y_V, \sigma_y, q^x_V)$, $y \geq x > 0$, in verifier V, that satisfies the following condition:*

$$\begin{aligned} & q^p_V \text{ is a fault state, } \forall p \in \{x, \ldots, y\}, \text{ and for some} \\ & p \in \{x, \ldots, y\}, \sigma_p \in \Sigma. \end{aligned} \tag{3.4}$$

Proof. ($\Leftarrow$) Let us assume that $\mathcal{L}(G)$ is not robustly codiagnosable with respect to K, Ψ_i, P_{s_i}, $i = 1, \ldots, n$, and Σ_f. Then, according to Definition 3.2, there exist a fault trace s and a continuation t with arbitrarily long length, and fault-free traces ω_i, such that $P_{s_i}(\Psi_i(st)) \cap P_{s_i}(\Psi_i(\omega_i)) \neq \emptyset$, for all $i = 1, 2, \ldots, n$. This implies, according to Lemma 3.1, that all verifiers V_i have a cyclic path cl_i satisfying condition (3.3), associated with the same fault trace st. Since $V = \|_{i=1}^{n} V_i$, and the events that synchronize all verifiers V_i are only those belonging to Σ, then there is an arbitrarily long length trace in V, s_V, such that the projection $P_V(s_V) = st$, where $P_V : \Sigma_V^\star \to \Sigma^\star$. Since V is a finite automaton, and s is a fault trace, then there exists in V a cyclic path cl, associated with trace s_V, satisfying condition (3.4).

($\Rightarrow$)Let us assume now that V has a cyclic path $cl = (q_V^x, \sigma_x, q_V^{x+1}, \sigma_{x+1}, \ldots, q_V^y, \sigma_y, q_V^x)$ satisfying condition (3.4). Let $t_V = (\sigma_x \sigma_{x+1} \ldots \sigma_y)^r$, where r is an arbitrarily large natural number, and $s_V \in \mathcal{L}(V)$ be the trace executed from the initial state of V until reaching state q_V^x. Since the states of cl are fault states, then σ_f belongs to s_V. Let $s = P_V(s_V)$ and $t = P_V(t_V)$. Since all events in Σ are common to all verifiers V_i and $V = \|_{i=1}^{n} V_i$, then there is in each verifier V_i, $i = 1, \ldots, n$, a trace $s_{V_i} t_{V_i}$ such that $P_V(s_{V_i} t_{V_i}) = st$. Since s is a fault trace, t has arbitrarily long length, and V_i are finite automata, then each V_i has a cyclic path cl_i satisfying condition (3.3), which shows that there exist fault-free traces $\omega_i \in \mathcal{L}(G_{N_i})$ and fault traces $s_i \in \mathcal{L}(G_{F_i})$, $i = 1, 2, \ldots, n$, associated with $s_{V_i} t_{V_i}$, such that $P_{s_i}(\omega_i) = P_{s_i}(s_i)$. Thus, the robust codiagnosability condition of Definition 3.2 is violated.$\square$

Note that, since $V^i = G_N^{\rho_i} \| G_F^i$, and $G_N^{\rho_i}$ is obtained from G_N^i by renaming the events in $\Sigma \cup \Sigma_l^i$, then each path of V^i is associated with a fault-free path $\omega^i \in \mathcal{L}(G_N^i)$ and a fault trace $s^i \in \mathcal{L}(G_F^i)$ such that $P_s^i(\omega^i) = P_s^i(s^i)$. Thus, as shown in [39], the existence of a cyclic path in V^i is a necessary and sufficient condition for the existence of an arbitrarily long length fault trace s^i that cannot be diagnosed by local diagnoser LD_i, which implies in the existence of a fault trace $s \in \mathcal{L}(G)$, where $s^i \in \Psi_i(s)$, that cannot be diagnosed by LD_i. In order to $\mathcal{L}(G)$ be robustly codiagnosable with respect to $K = (K_1, K_2, \ldots, K_n)$, Ψ_i, P_s^i, $i = 1, \ldots, n$, and Σ_f, all fault traces that cannot be diagnosed by a local diagnoser LD_i, must be diagnosed by at least another local diagnoser LD_j, where $j \neq i$ and $j \in \{1, 2, \ldots, n\}$. Let us firstly assume that $\mathcal{L}(G)$ is not robustly codiagnosable with respect to K, Ψ_i, P_s^i, $i = 1, \ldots, n$, and Σ_f. Then, according to Definition 3.2, there exists a fault trace $st \in \mathcal{L}(G)$, and fault-free traces $\omega_i \in \mathcal{L}(G)$, that generate in all verifiers V^i, $i = 1, \ldots, n$, cyclic paths cl^i satisfying. Since $V = \|_{i=1}^{n} V^i$, $P^i(s^i) = st$, and only the events in Σ are common to all V^i, then st generates a cyclic path cl in V satisfying the conditions presented in (3.4).

The verification of K-loss robust codiagnosability can be carried out constructing verifiers V_i for each local diagnoser LD_i, $i = 1, \ldots, n$, using the method proposed in [39], considering automaton G_{t_i} as the plant model, and Σ_s^i as the set of observable events. Then, a verifier $V = V_1 \| V_2 \| \dots \| V_n$ can be computed.

An important characteristic of verifier V is that the fault-free traces $s_{\rho_i} \in \mathcal{L}(G_{\rho_i})$ and the fault traces $s_{F_i} \in \mathcal{L}(G_{F_i})$, associated with a trace $s_V \in \mathcal{L}(V)$, can be easily obtained as $s_{\rho_i} = P_{V,s\rho_i}(s_V)$ and $s_{F_i} = P_{V,i}(s_V)$, where $P_{V,s\rho_i} : \Sigma_V^\star \to (\Sigma_{s_i} \cup \Sigma_{\rho_i})^\star$ and $P_{V,i} : \Sigma_V^\star \to \Sigma_i^\star$.

Example 3.5 *According to Algorithm 5, verifier V is computed using automata G_{t_1} and G_{t_2}. Firstly, automata G_{N_i} and G_{F_i} are obtained from automata G_{t_i}, for $i = 1, 2$. Then, verifiers $V_i = G_{\rho_i} \| G_{F_i}$ are computed. Finally, automaton $V = V_1 \| V_2$ is computed. In Figure 3.8, a fault path of verifier V is presented. Note that in Figure 3.8 there is a cyclic path cl that satisfies condition (3.4). Thus, $\mathcal{L}(G)$ is not robustly codiagnosable with respect to K, Ψ_i, P_{s_i}, $i = 1, 2$, and Σ_f. Note that the trace associated with the fault path shown in Figure 3.8 is $s_V = a^{\rho_1} a a_{s_1} d d_{l_2} \sigma_f b a^{\rho_2} b^{\rho_2} (b_{s_2} c^{\rho_2} c_{l_2}^{\rho_2} a^{\rho_2} b^{\rho_2} b)^m$, where $m \in \mathbb{N}$. Trace s_V is associated with the following traces: (i) $s_{\rho_1} = P_{V,s\rho_1}(s_V) = a^{\rho_1} a_{s_1} \in \mathcal{L}(G_{\rho_1})$; (ii) $s_{F_1} = P_{V,1}(s_V) = a a_{s_1} d \sigma_f b^{m+1} \in \mathcal{L}(G_{F_1})$; (iii) $s_{\rho_2} = P_{V,s\rho_2}(s_V) = a^{\rho_2} b^{\rho_2} (b_{s_2} c^{\rho_2} c_{l_2}^{\rho_2} a^{\rho_2} b^{\rho_2})^m \in \mathcal{L}(G_{\rho_2})$; and (iv) $s_{F_2} = P_{V,2}(s_V) = a d d_{l_2} \sigma_f b (b_{s_2} b)^m \in \mathcal{L}(G_{F_2})$. Traces s_{ρ_1} and s_{F_1} are observed as a_{s_1} by local diagnoser LD_1, and traces s_{ρ_2} and s_{F_2} are observed as $b_{s_2}^m$ by local diagnoser LD_2. Thus, since traces s_{F_1} and s_{F_2} are obtained from the same fault trace $s_F = a d \sigma_f b^{m+1}$ executed by the system, then $\mathcal{L}(G)$ is indeed not robustly codiagnosable according to Definition 3.2.* □

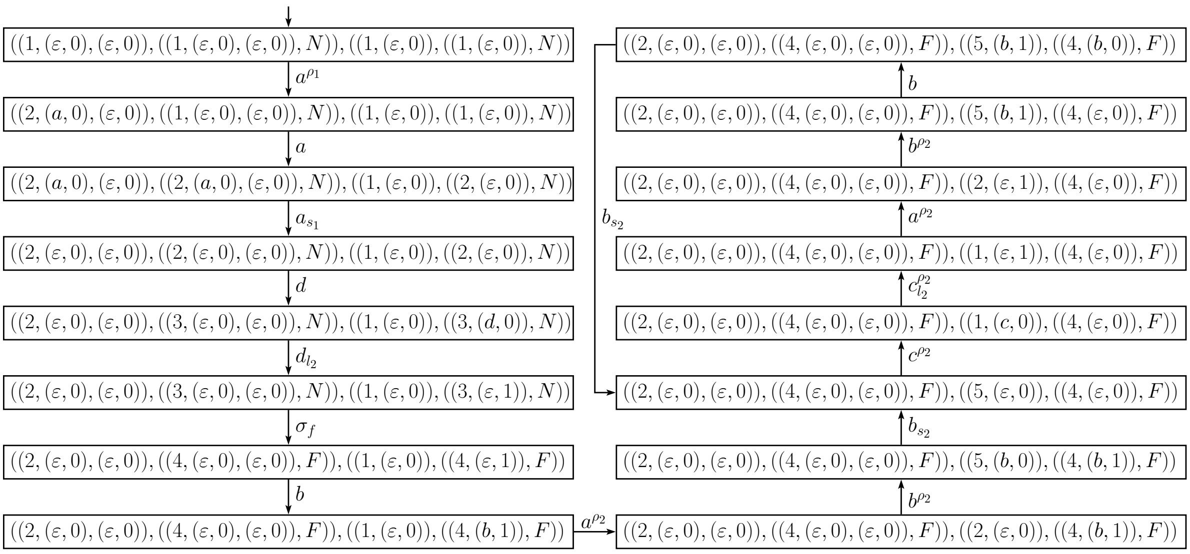

Figura 3.8: A fault path of verifier $V = V_1 \| V_2$ of the system subject to loss of observations of Example 3.5.

Since, as shown in [39], the number of states of V_i, $i = 1, 2, \dots, n$, is, in the worst case, $2 \times |Q_{t_i}|^2$, then the complexity of computing verifier $V = \|_{i=1}^n V_i$, is $O\left(2^n \times \left(\prod_{i=1}^n |Q_{t_i}|^2\right) \times \left(\sum_{i=1}^n |\Sigma_i|\right)\right)$.

3.4 Concluding Remarks

In this chapter, we start by formulating the problem of robust diagnosis against temporary loss of observation, in which we associate the temporary failures with the communication channels that are used to transmit the event obsvations to the local diagnosers. In the sequel, we proceed to present the definition of K-loss robust codiagonosability and an automaton model of the plant subject to temporary event communication failures, with the respective alogrithm used to model the behavior of each comunication channel $ch_{i,j}$. In the sequel, we presented the K-loss robust codiagnosability verification method and the algorithm used to construct the verifier V. We finish this chapter by presenting the complexity computing the verifier V. In all the steps aforementioned, examples were used to demonstrate the proposed methods and algorithms presented in this work.

Capítulo 4

Conclusion

In this work, we introduced the notion of K-loss robust codiagnosability to address the problem of robust decentralized diagnosis of Discrete-Event Systems against temporary failures in the communication of the observation of system events to the local diagnosers, where it is considered that the communication always recovers from the failure after a given maximum number of observation losses. Differently from [11], we assume in this work that an event observation can be successfully communicated to a local diagnoser, and not communicated to a different local diagnoser due to temporary communication failure, as we associate the temporary failures with the communication channels that transmit the events observations and not to faulty sensors.

The proposed method of K-loss robust codiagnosability differently from other methods proposed in the literature, allows considering the case that all communication channels between plant and local diagnosers are not reliable. In definition 3.1 we present a function that models the temporary failure in the communication channels and in the sequel we present the definition 3.2 of K-loss robust diagnosability.

Models of the plant for each local diagnoser, subject to temporary loss of observations, are also presented. Finally, by using these models, a K-loss robust codiagnosability verification method is proposed.

In summary, the main contributions of this work are as follows:

- The K-loss robust diagnosability method address a problem that is closer to reality of industrial plant operations when compared to the other methods proposed in the literature, as it models the temporary failure in the communication channels, considering that the communication always recovers after a maximum number of observation losses.

- Algorithms to compute the automaton that models the communication channel behavior.

- A method for verification of the K-loss robust diagnosability of DESs, the algorith that computethe verifier automaton for the verification of the proposed method and its computational complexity.

Referências Bibliográficas

[1] CASSANDRAS, C. G., LAFORTUNE, S. *Introduction to Discrete Event Systems*. Secaucus, NJ, Springer-Verlag New York, Inc., 2008.

[2] HOPCROFT, J. E., MOTWANI, R., ULLMAN, J. D. *Introduction to automata theory, languages, and computation*. Boston, Addison Wesley, 2006.

[3] LAWSON, M. V. *Finite automata*. CRC Press, 2003.

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

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

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

[7] 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.

[8] 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.

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

[10] CARVALHO, L. K., MOREIRA, M. V., BASILIO, J. C. "Generalized robust diagnosability of Discrete Event Systems", v. 44, n. 1, pp. 8737–8742, 2011.

[11] 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.

[12] BASILIO, J. C., LIMA, S. T. S., LAFORTUNE, S., et al. “Computation of minimal event bases that ensure diagnosability”, *Discrete Event Dynamic Systems: Theory And Applications*, v. 22, pp. 249–292, 2012.

[13] CABRAL, F. G., MOREIRA, M. V., DIENE, O., et al. “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.

[14] SANTORO, L. P. M., MOREIRA, M. V., BASILIO, J. C. “Computation of minimal diagnosis bases of Discrete-Event Systems using verifiers”, *Automatica*, v. 77, pp. 93–102, 2017.

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

[16] BASILE, F., CHIACCHIO, P., DE TOMMASI, G. “An efficient approach for online diagnosis of discrete event systems”, *IEEE Transactions on Automatic Control*, v. 54, n. 4, pp. 748–759, 2009.

[17] 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.

[18] CABRAL, F. G., M. M. V. “Synchronous Diagnosis of Discrete-Event Systems”, *Transactions on Automation Science and Engineering*, 2018.

[19] SCHMIDT, K. “Abstraction-based verification of codiagnosability for discrete event systems”, *Automatica*, v. 46, n. 9, pp. 1489–1494, 2010.

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

[21] ATHANASOPOULOU, E., L. L. H. C. “Probabilistic failure diagnosis in finite state machines under unreliable observations”, *8th In- ternational Workhop on Discrete Event Systems, Ann Arbor, Michigan, USA*, pp. 301–306, 2010.

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

[23] TAKAI, S. "Robust failure diagnosis of partially observed discrete event systems", *10th Word Congress International Federation of Automatic Control*, p. 1158–1165, 2010.

[24] LIMA, S. S., B. J. C. L. S. E. A. "Robust diagnosis of discrete-event systems subject to permanent sensor failures", *IFAC Proceedings Volumes*, v. 43, n. 12, pp. 90–97, 2010.

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

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

[27] ASIM, M., MOKHTAR, H., MERABTI, M. "A fault management architecture for wireless sensor network". In: *International Wireless Communications and Mobile Computing Conference*, pp. 779–785, 2008.

[28] MAHAPATRO, A., KHILAR, P. M. "Transient fault tolerant wireless sensor networks", *Procedia Technology*, v. 4, pp. 97–101, 2012.

[29] MUHAMMED, T., SHAIKH, R. A. "An analysis of fault detection strategies in wireless sensor networks", *Journal of Network and Computer Applications*, v. 78, pp. 267–287, 2017.

[30] PARK, J., IVANOV, R., WEIMER, J., et al. "Security of cyber-physical systems in the presence of transient sensor faults", *ACM Transactions on Cyber-Physical Systems*, v. 1, n. 3, pp. 1–23, 2017.

[31] TRAPP, M., SCHURMANN, B., TETTEROO, T. "Failure behavior analysis for reliable distributed embedded systems", *International Parallel and Distributed Processing Symposium*, 2002. doi: 10.1109/ipdps.2002.1016486.

[32] KAR, P., MISRA, S. "On the Effects of Transfaulty Sensor Nodes in Stationary Wireless Sensor Network Systems", *IEEE Sensors Journal*, v. 19, n. 13, pp. 5022–5029, 2019.

[33] KANAGAWA, N., TAKAI, S. "Diagnosability of discrete event systems subject to permanent sensor failures", *International Journal of Control*, v. 88, n. 12, pp. 2598–2610, 2015.

[34] TOMOLA, J. H. A., CABRAL, F. G., CARVALHO, L. K., et al. "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.

[35] NUNES, C. E. V., MOREIRA, M. V., ALVES, M. V. S., et al. "Codiagnosability of networked discrete event systems subject to communication delays and intermittent loss of observation", *Discrete Event Dynamic Systems*, v. 28, n. 2, pp. 215–246, 2018.

[36] WADA, A., TAKAI, S. "Verification of Codiagnosability for Decentralized Diagnosis of Discrete Event Systems Subject to Permanent Sensor Failures". In: *18th European Control Conference (ECC 2019)*, pp. 1726–1731, Napoli, Italy, 2019.

[37] OLIVEIRA, V. S. L., CABRAL, F. G., MOREIRA, M. V. "K-loss Robust Diagnosability of Discrete Event Systems", 2020.

[38] OLIVEIRA, V. S. L., CABRAL, F. G., MOREIRA, M. V. "K-loss Robust Codiagnosability of Discrete Event Systems". 2022.

[39] 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.

[40] CABRAL, F. G., MOREIRA, M. V. "Synchronous Diagnosis of Discrete-Event Systems", *IEEE Transactions on Automation Science and Engineering*, v. 17, n. 2, pp. 921–932, 2020.

[41] 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.