VERIFICATION OF GENERALIZED ROBUST DIAGNOSABILITY OF DISCRETE EVENT SYSTEMS

Lahis El Ajouze Azeredo Coutinho

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

Orientador: Lilian Kawakami Carvalho

Rio de Janeiro
Junho de 2017

VERIFICATION OF GENERALIZED ROBUST DIAGNOSABILITY OF DISCRETE EVENT SYSTEMS

Lahis El Ajouze Azeredo Coutinho

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

Examinada por:

Prof. Lilian Kawakami Carvalho, D.Sc.

Prof. Patrícia Nascimento Pena, D.Sc.

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

RIO DE JANEIRO, RJ – BRASIL
JUNHO DE 2017

Coutinho, Lahis El Ajouze Azeredo

Verification of Generalized Robust Diagnosability of Discrete Event Systems/Lahis El Ajouze Azeredo Coutinho. – Rio de Janeiro: UFRJ/COPPE, 2017.

IX, 73 p.: il.; 29, 7cm.

Orientador: Lilian Kawakami Carvalho

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2017.

Referências Bibliográficas: p. 71 – 73.

1. Discrete event systems. 2. Fault diagnosis. 3. Sensor failures. 4. Robust diagnosability. I. Carvalho, Lilian Kawakami. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

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

VERIFICAÇÃO DA DIAGNOSTICABILIDADE ROBUSTA GENERALIZADA DE SISTEMAS A EVENTOS DISCRETOS

Lahis El Ajouze Azeredo Coutinho

Junho/2017

Orientador: Lilian Kawakami Carvalho

Programa: Engenharia Elétrica

Este trabalho aborda o problema de diagnosticabilidade robusta generalizada (DRG) de sistemas a eventos discretos (SEDs) descritos por uma classe de autômatos em que cada elemento da classe gera uma linguagem distinta. A definição de DRG e o algoritmo para sua verificação propostos anteriormente na literatura foram atualizados, resultando em um novo algoritmo com menor complexidade computacional que o anterior. Baseado nesse algoritmo, uma nova condição necessária e suficiente para diagnosticabilidade robusta generalizada foi apresentada. Quatro abordagens diferentes sobre diagnosticabilidade de SEDs foram analisadas: o problema de diagnosticabilidade de sistemas a eventos discretos sujeitos a falhas permanentes de sensores *(i)*; o problema de diagnosticabilidade robusta de sistemas a eventos discretos sujeitos a perdas permanentes *(ii)* e intermitentes *(iii)* de observação; e o problema de verificação da diagnosticabilidade robusta de sistemas a eventos discretos parcialmente observados *(iv)*. Mecanismos de transformação foram propostos para cada problema analisado com o objetivo de demonstrar que as abordagens de *(i)* a *(iv)* são casos particulares da diagnosticabilidade robusta generalizada proposta nesse trabalho.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

VERIFICATION OF GENERALIZED ROBUST DIAGNOSABILITY OF DISCRETE EVENT SYSTEMS

Lahis El Ajouze Azeredo Coutinho

June/2017

Advisor: Lilian Kawakami Carvalho

Department: Electrical Engineering

This work addresses the problem of generalized robust diagnosability (GRD) of discrete event systems (DESs) described by a class of automata, where each automaton in the class generates a distinct language. The definition of GRD and the algorithm for its verification previously proposed in literature were updated, resulting in an algorithm with smaller computational complexity than the previous one. Based on this algorithm, a new necessary and sufficient condition for generalized robust diagnosability was presented. Four different approaches on diagnosability of DESs were analyzed: the problem of diagnosability of discrete event systems subject to permanent sensor failures *(i)*; the problem of robust diagnosis of discrete event systems against permanent *(ii)* and intermittent *(iii)* loss of observations; and the problem of verification of robust diagnosability for partially observed discrete event systems *(iv)*. Transformation mechanisms for each analyzed problem were proposed with the purpose of demonstrating that all approaches *(i)* - *(iv)* are particular cases of the generalized robust diagnosability definition proposed in this work.

Contents

List of Figures

Chapter 1

Introduction

Ever since industry emerged it has been seeking for constant improvement of operations and optimization of processes. The concept of *automation* came to light over the last years as an alternative capable of reducing labor, energy, material expenses and waste whilst enhancing quality, precision and accuracy. After General Motors has established its automation department in 1947 [1], the new theory started to be spread out and widely applied to manufacturing, traffic, database, computer and several other types of systems. These systems may be classified in many different aspects, but this work will focus on discrete-state, event-driven systems commonly called as discrete event systems (DESs) and hereby modeled by *language* and *automata*, mathematical formalisms able to represent all the aspects of DESs [2].

As technology evolved, these systems became larger and more complex requiring stringent standards of performance and reliability as well as being intolerant to occurrences of failures. Sophisticated and systematic methods for the timely and accurate diagnosis and isolation of a system failure were therefore developed generating a event-based concept of diagnosability of discrete event systems in the context of failure diagnosis problem, as introduced by LIN [3] and SAMPATH *et al.* [4]. In [4], deterministic automata called *diagnosers* were proposed with the purposes of performing online detection and isolation of system failures and offline verification of the diagnosability properties of a system. This second task may be also performed using *verifiers*, which are deterministic automata that demands less computational complexity to perform the same task of a diagnoser as proposed in [5–7], and the references therein.

The idea of failure diagnosis of discrete event systems has received expressive attention from the scientific community as in [3, 4, 8–15], which has formulated and addressed a variety of problems derived from the failure diagnosis framework. Recently, there have been some works on sensor failures in supervisory control of DESs [16, 17]; on various notions of *robust* diagnosis of discrete event systems in the presence of potentially faulty sensors, such as in [18–23], where the robustness may

be due to the maintenance of the diagnosability property after the failure of a sensor, a communication failure between sensor and supervisor, an electrical bad linkage, etc. Besides, in [24, 25] fault diagnosis under unreliable conditions are presented.

KANAGAWA and TAKAI [26] have considered a failure diagnosis problem for discrete event systems subject to permanent sensor failures. A notion of diagnosability subject to permanent sensor failures has been introduced with respect to a certain nondeterministic observation mask, along with an aggregated Mealy automaton with a deterministic and state-dependent observation mask that is defined to perform verification of the proposed diagnosability definition. Then, it has been shown that the diagnosability of the original system subject to permanent sensor failures and the diagnosability of the aggregated Mealy automaton were equivalent and, finally, the delay bound within which the occurrence of any failure string can be detected subject to permanent sensor failures has been computed.

CARVALHO *et al.* [20] have considered the problem of diagnosing the occurrence of a certain unobservable fault event in the operation of partially-observed discrete event systems subject to permanent loss of observations modeled by finite state automaton. The permanent loss of observations has been assumed to be due to the failure of certain sensors monitoring originally observable events at the outset of the system, that is, before the occurrence of the event it is monitoring. Since the diagnostic engine is not necessarily aware of such failure, a previous definition of robust diagnosability of a given fault event despite the possibility of permanent loss of observations has been explored, and a polynomial time verification algorithm to verify robust diagnosability has been proposed. Besides, a methodology to perform online diagnosis in this scenario, using a set of partial diagnosers, has been also presented.

CARVALHO *et al.* [21] has considered that bad sensor operation or communication failure between sensors and the diagnoser to be regarded as loss of observations of events initially assumed as observable, which would lead to the diagnoser standing still or even reporting some wrong information regarding the fault occurrence. It has been assumed therefore that intermittent loss of observations may occur - a problem here referred to as robust diagnosis of discrete event systems against intermittent loss of observations - and an automaton model based on a new language operation called dilation has been proposed. A necessary and sufficient condition for robust diagnosability in terms of the language generated by the original automaton has been presented along with two tests for robust language diagnosability: one based on diagnosers and the other one using verifiers. The work has been finally extended to robust codiagnosability against intermittent loss of observations.

TAKAI [23] has studied robust failure diagnosis of discrete event systems and has considered that, given a set of possible models, each of which has its own nonfailure

specification, there exists a single diagnoser that detects any occurrence of a failure within a uniformly bounded number of steps for all possible models - which has been called as a robust diagnoser. A notion of robust diagnosability has been introduced along with a proof that it served as a necessary and sufficient condition for the existence of a robust diagnoser. An algorithm for verifying the robust diagnosability condition has been finally presented.

CARVALHO *et al.* [27] dealt with the problem of robust diagnosability of discrete event systems described by a class of automata, where each automaton generates a distinct language. A new generalized robust diagnosability (GRD) definition was introduced to generalize all previous definitions of robust diagnosability. Robustness here stands for the ability of the language diagnosability property to remain unchanged even considering uncertainties in the model of the system [22] and in the set of observable events [28]. A necessary and sufficient condition for the new definition is also presented, as well as a polynomial time verification algorithm. This definition is the central line of this work.

The objective here is to improve the verification algorithm proposed in [27] and to show how this definition is capable of encompassing many sophisticated problems addressed so far, namely *(i)* the problem of diagnosability of discrete event systems subject to permanent sensor failures [26]; the problem of robust diagnosis of discrete event systems against *(ii)* permanent [20] and *(iii)* intermittent [21] loss of observations; and *(iv)* the problem of verification of robust diagnosability for partially observed discrete event systems [23]. The results of this work have shown that, by relaxing some assumptions and updating the generalized robust diagnosability definition, as well as improving the computational complexity of the verification algorithm proposed by CARVALHO *et al.* [27], all the abovementioned problems may be addressed and solved using the generalized robust diagnosability definition. Transformation mechanisms were developed for each one of the referred problems so that it could be easily demonstrated how they can be addressed by the GRD definition. More importantly, even with the transformation mechanisms it is more advantageous in terms of computational complexity to solve these problems using the GRD definition.

This work is organized as follows. In Chapter 2 we present the discrete event systems theory based on [2]. In Chapter 3 we present the different approaches on fault diagnosis of discrete event systems proposed by TAKAI [23], CARVALHO *et al.* [20], CARVALHO *et al.* [21], TAKAI [23] and CARVALHO *et al.* [27]. In Chapter 4, we update the generalized robust diagnosability definition; propose a new verification algorithm for the GRD definition; and develop transformation mechanisms for each problem presented in Chapter 3 - based in the inputs from each paper - that returns a language based framework applicable to the generalized robust diagnosability def-

inition. Then, we evaluate whether or not the GRD definition is able to encompass all the presented problems. Finally, in Chapter 5 we draw some conclusions about this work and propose a future line of work to be pursued.

Chapter 2

Discrete event systems theory

Discrete event systems (DESs) are dynamic discrete-state, event-driven systems in which the state evolution depends exclusively on the occurrence of asynchronous discrete events over time, as defined by CASSANDRAS and LAFORTUNE [2]. The *states* are part of a discrete event set and describe the behavior of the system at an instant time t, whereas the *event* represents an instantaneous occurrence, mostly related to an action taken such as the press of a button, and is responsible for the transitions from one state to another - therefore the term *event-driven*.

The main particularity of DESs is that, unlike continuous-state systems, they are not necessarily associated with time. The system is observed in intervals of time and a change in observation is only perceived after the occurrence of an event. Hence, these systems are not properly represented by differential equations, which caused several formalisms to be created so that the system could be modeled.

In order to illustrate a DES, consider a very simple system consisting of a parking-garage gate as shown in Figure 2.1. This gate may be opened or closed. Consider the gate to be opened, which we may call O. Consider also that there exists a remote control with two buttons responsible for managing the gate: one button opens and the other one closes the gate. Only by the press of the open button the gate status may change from O to closed, or C, and *vice-versa* if the close button is pressed. Note that the gate may be in two different *states*, O and C and, unless a button is pressed, *i.e.*, an *event* O_{pen} or C_{lose} occurs, the gate stays exactly how it is, regardless of the time period that has passed. Thus, we are presenting a discrete-state, event-driven dynamic system in which the evolution depends solely on the occurrence of events.

This work will focus on a language-based approach based on set theory to mathematically formalize discrete event systems. This approach is graphically represented by state transition structures detailing the states, the transitions and the events responsible for the transitions to occur. Even though it is easy to encounter many other formalisms in literature, the one to be here presented is *automata* along with

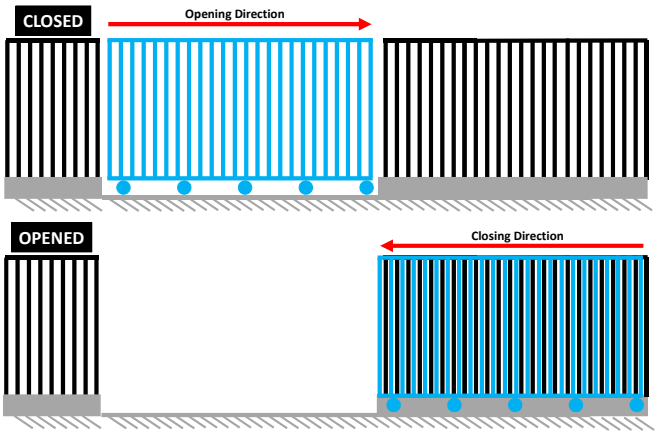


Figure 2.1: Illustrative parking-garage gate discrete event system.

the concept of *languages* that can both model a discrete event system.

This chapter is organized as follows. In Section 2.1 we present the concept of language along with its notations, definitions and operations and in Section 2.2 we present the concept of automata. In Sections 2.3 and 2.4 we respectively introduce the related concepts of diagnosability of discrete event systems and verification of decentralized diagnosability of DES. Finally, we conclude with some final comments in Section 2.5.

2.1 Languages

The concept of language is based on set theory and is relevant to discrete event system modeling since the sets of states and events are discrete and finite to the extent of this work. Specifically focusing on the event set Σ, it is possible to think of a sequence of events. If events are labeled by letters, and Σ is taken as an *alphabet*, the strings that represent sequences of events can be taken as *words*. Given a system, all the sequences of events that are possible to occur can then be faced as the *language* spoken by this system. That line of thinking starts explaining language modeling.

2.1.1 Notation and Definitions

As it was already introduced, the event set Σ of a DES is here treated as an alphabet assumed to be finite. Events are generically represented by the letter e, and the

empty string (or word) is represented by the symbol ε. Note that an empty alphabet has no elements, that is, not even ε is included. A sequence, string or word is denoted by s, and its length is given by $|s|$. By convention, the length of ε is zero.

Definition 2.1 *A* language *defined over an event set Σ is a set of finite-length strings formed from events in Σ.* □

A language defined over a set Σ is a subset of Σ^*, an operation called *Kleene-closure* which denotes the set of all possible finite-length sequences of events in Σ, including ε. Particularly, $\emptyset$, Σ and Σ^* itself are languages.

Let $s = abc$ denote a sequence where a, b and $c \in \Sigma^*$. Then, referred to s, a is the prefix, b is a substring and c is the suffix. Besides, s/a denotes the suffix of s after the prefix a.

2.1.2 Operations on Languages

Once languages are described as sets, all the usual set operations are also applicable to them. Besides, new operations are defined as follows.

- Concatenation: Let L_a, $L_b \subseteq \Sigma^*$, then $L_aL_b := \{s \in \Sigma^* : (s = s_as_b) \wedge (s_a \in L_a) \wedge (s_b \in L_b)\}$.
- Prefix-closure: Let $L \subseteq \Sigma^*$, then $\bar{L} := \{s \in \Sigma^* : (\exists t \in \Sigma^*)[st \in L]\}$.
- Kleene-closure: Let $L \subseteq \Sigma^*$, then $L^* := \{\varepsilon\} \cup L \cup LL \cup LLL....$
- Projections: $P : \Sigma_l^* \to \Sigma_s^*$, $\Sigma_s \subset \Sigma_l$, where $P(\varepsilon) := \varepsilon$,

$$P(e) := \begin{cases} e, & \text{if } e \in \Sigma_s \\ \varepsilon, & \text{if } e \in \Sigma_l \setminus \Sigma_s, \end{cases}$$

  and $P(se) := P(s)P(e)$ for $s \in \Sigma_l^*, e \in \Sigma_l$.

From the definition it is easy to see that the projection, often called as *natural projection*, takes a string formed from the larger event set - here defined as Σ_l - and *erases* events from it that do not belong to the smaller event set Σ_s.

Likewise, it is possible to define the inverse operation, with respect to the corresponding inverse map [1]: $P^{-1} : \Sigma_s^* \to 2^{\Sigma_l^*}$, where $P^{-1}(t) := \{s \in \Sigma_l^* : P(s) = t\}$. In other words, given a string of events in the smaller event set Σ_s, the inverse projection returns the set of all strings from the larger event set Σ_l that project, with P, to the given string.

[1] The notation 2^A means the power set of A, *i.e.*, the set of all subsets of a finite set A.

Both projection P and inverse projection P^{-1} may be extended to languages by simply applying them to all the strings in the language. For $L \subseteq \Sigma_l^*$, $P(L) := \{t \in \Sigma_s^* : (\exists s \in L)[P(s) = t]\}$, and for $L_s \subseteq \Sigma_s^*$, $P^{-1} := \{s \in \Sigma_l^* : (\exists t \in L_s)[P(s) = t]\}$.

2.2 Automata

The modeling formalism here presented is centered on the automaton, which is a device defined as a six-tuple capable of representing and manipulating languages.

Definition 2.2 *A deterministic automaton, denoted by G, is a six-tuple:*

$$G = (X, \Sigma, f, \Gamma, x_0, X_m) \tag{2.1}$$

where X is the set of states, Σ is the set of events, $f : X \times \Sigma \to X$ is the transition function, that is usually partial on its domain, $\Gamma : X \to 2^{\Sigma}$ is the active event (or feasible event) function, x_0 is the initial state, and $X_m \subseteq X$ is the set of marked states. □

The *state transition diagram* is the oriented graph representation of an automaton and helps shaping up all the math involved with DESs. In the state transition diagrams, the vertices of the graph are circles and represent the different states of the system. The transitions connecting them are represented as arcs and labeled by symbols, that are the events of the system. Note that it is possible to have selfloops, which in this case are transitions that start and end at the same state. The initial state has an arrow pointing to it, and the marked states are represented by double circles. In case the marked states are not defined, assume, without loss of generality, that $X_m = \emptyset$.

An example of an automaton and its state transition diagram is provided.

Example 1 *Let G be a deterministic automaton which state transition diagram can be seen in Figure 2.2. The sets of states and events are respectively given by $X = \{x_0, x_1, x_2, x_3, x_4\}$ and $\Sigma = \{a, b, c\}$. The transition function is defined as: $f(x_0, a) = x_1$; $f(x_0, b) = x_2$; $f(x_1, c) = x_3$; $f(x_3, b) = x_3$; $f(x_2, a) = x_4$; $f(x_4, a) = x_0$, causing the active event functions to be: $\Gamma(x_0) = \{a, b\}$; $\Gamma(x_1) = \{c\}$; $\Gamma(x_2) = \{a\}$; $\Gamma(x_3) = \{b\}$; $\Gamma(x_4) = \{a\}$. Finally, the initial state is x_0.*

The generated and marked languages of an automaton are described according to Definition 2.3.

Definition 2.3 *A language generated by an automaton G is given by:*

$$L(G) := \{s \in \Sigma^* : f(x_0, s) \text{ is defined}\}, \tag{2.2}$$

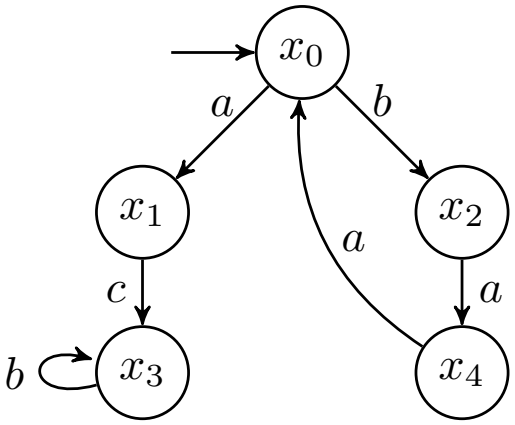


Figure 2.2: State transition diagram from Example 1.

While the language marked by G is given by:

$$L_m(G) := \{s \in L(G) : f(x_0, s) \in X_m\}. \tag{2.3}$$

□

Starting at the initial state, and following along the state transition diagram, all the directed possible paths are represented by the language generated by G, $L(G)$. The transitions composing each path are labelled by events. The concatenation of these events is the string corresponding to this path, in a way that $s \in L(G)$ if and only if it corresponds to an admissible path in the state transition diagram. It is important remarking that $L(G)$ is prefix-closed by definition, and that it is possible having events defined in Σ that do not appear in the state transition diagram. That indicates that these events will not figure in $L(G)$.

The language marked by G, $L_m(G)$, is a subset of $L(G)$ corresponding to all the sequences s in the state transition diagram labelling paths that ends in a marked state, that is, for which $f(x_0, s) \in X_m$. Note that $L_m(G)$ is not necessarily prefix-closed.

Another relevant definition to be presented involves the concept of *blocking*. While inspecting the state transition diagram, whenever a path takes to a state where no further events can be executed, it is called a *deadlock*. Analogously, a path that takes to a set of strongly connected[2] unmarked states is actually taking to a *livelock*.

Definition 2.4 *An automaton G is said to be blocking if*

$$\overline{L_m(G)} \subset L(G) \tag{2.4}$$

[2]In graph theory, a pair of vertices in a directed graph is said to be strongly connected if there is a path in each direction between them [29].

and nonblocking when

$$\overline{L_m(G)} = L(G) \tag{2.5}$$

□

In case a state transition diagram representing a language has no deadlock states in it, this associate language is said to be *live*.

It is possible that different automata generate and mark the same language. These are *language-equivalent* automata.

Definition 2.5 *Automata G_1 and G_2 are said to be language-equivalent if*

$$L(G_1) = L(G_2) \text{ and } L_m(G_1) = L_m(G_2) \tag{2.6}$$

□

Automaton theory is not restricted to deterministic automata. It is possible to have an automaton with more than one initial state, as well as having transitions labeled by the event ε. Moreover, a single event may label multiple transitions from a state x. Whenever an automaton meets at least one of these three requirements it is said to be nondeterministic.

Definition 2.6 *A nondeterministic automaton, denoted by G_{nd}, is a six-tuple:*

$$G_{nd} = (X, \Sigma \cup \{\varepsilon\}, f_{nd}, \Gamma, X_0, X_m) \tag{2.7}$$

where:

- $f_{nd} : X \times \{\Sigma \cup \varepsilon\} \to 2^X$, *that is,* $f_{nd}(x, e) \subseteq X$ *whenever it is defined.*

- *The initial state x_0 may be a set of states, i.e., $X_0 \subseteq X$.* □

Defining the generated and marked languages by a nondeterministic automaton demands a few previous steps. For starts, the transition function needs to be extended to the domain $X \times \Sigma^*$, denoted by $f_{nd}^{ext}(x, \varepsilon)$. Further on, the $\varepsilon - reach$ of a state x is defined to be the set of all states that can be reached from x by following transitions labelled by ε in the state transition diagram. The proper notation is $\varepsilon R(x)$. By convention, $x \in \varepsilon R(x)$. The definition is extended to a subset of states $B \subseteq X$,

$$\varepsilon R(B) = \bigcup_{x \in B} \varepsilon R(x). \tag{2.8}$$

Recursively, $f_{nd}^{ext}(x, \varepsilon)$ is constructed as follows

$$f_{nd}^{ext}(x, \varepsilon) := \varepsilon R(x) \tag{2.9}$$

For $u \in \Sigma^*$ and $e \in \Sigma$,

$$f_{nd}^{ext}(x, ue) := \varepsilon R[z : z \in f_{nd}(y, e) \text{ for some state } y \in f_{nd}^{ext}(x, u)] \tag{2.10}$$

Now that all the requirements have been met, the generated and marked languages of a nondeterministic automaton are defined as follows.

Definition 2.7 *The language generated by a nondeterministic automaton G_{nd} is*

$$L(G_{nd}) := \{s \in \Sigma^* : (\exists x \in x_0)[f_{nd}^{ext}(x, s) \textit{ is defined}\} \tag{2.11}$$

While the language marked by G_{nd} is given by:

$$L_m(G_{nd}) := \{s \in L(G_{nd}) : (\exists x \in x_0)[f_{nd}^{ext}(x, s) \cap X_m \neq \emptyset\}. \tag{2.12}$$

□

It is possible for a deterministic automaton to generate and mark the same languages of a nondeterministic automaton. The automata are then said to be *language equivalent*, and the deterministic automaton is called the *observer*. The procedure for building an observer automaton is proposed.

The observer of a G_{nd} is defined as $Obs(G_{nd}) = (X_{obs}, \Sigma, f_{obs}, x_{0,obs}, X_{m,obs})$ and is built in Algorithm 2.1 [2].

Algorithm 2.1 Observer automaton

Input: $G_{nd} = (X, \Sigma \cup \{\varepsilon\}, f_{nd}, \Gamma, x_0, X_m)$
Output: $Obs(G_{nd}) = (X_{obs}, \Sigma, f_{obs}, x_{0,obs}, X_{m,obs})$

1: Define $x_{0,obs} := \varepsilon R(x_0)$ and set $X_{obs} = x_0$
2: For each B $\in X_{obs}$ and $e \in \Sigma$, define $f_{obs}(B, e) := \varepsilon R(\{x \in X : (\exists x_e \in B)[x \in f_{nd}(x_e, e)]\})$ whenever $f_{nd}(x_e, e)$ is defined for some $x_e \in B$. In this case, add the state $f_{obs}(B, e)$ to X_{obs}. If $f_{nd}(x_e, e)$ is not defined for any $x_e \in B$, then $f_{obs}(B, e)$ is not defined
3: Repeat Step 2 until the entire accessible part of G_{obs} has been constructed
4: $X_{m,obs} := \{B \in X_{obs} : B \cap X_m \neq \emptyset\}$

The observer automaton is an important tool in the study of partially-observed systems, that are the ones where some events are defined as unobservable and behave exactly like an ε-transition of a nondeterministic automaton.

The model for a partially-observed DES is a deterministic automaton whose set of events is $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$, a partition of Σ, where Σ_o and Σ_{uo} are, respectively, the set of observable and unobservable events. This event set is used to build an observer automaton just like before. The notion of ε-reach is now generalized by the notion of *unobservable reach*, that is

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

Extended to the set of states $B \subseteq X$,

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

Let $G = (X, \Sigma, f, \Gamma, x_0, X_m)$ denote a partially observed automaton with the partitioned event set. $Obs(G_{nd}) = (X_{obs}, \Sigma, f_{obs}, \Gamma_{obs}, x_{0,obs}, X_{m,obs})$ is built in Algorithm 2.2 [2].

Algorithm 2.2 Observer automaton that models a partially-observed DES

Input: $G = (X, \Sigma, f, \Gamma, x_0, X_m)$
Output: $Obs(G) = (X_{obs}, \Sigma, f_{obs}, x_{0,obs}, X_{m,obs})$

1: Define $x_{0,obs} := UR(x_0)$ and set $X_{obs} = x_{0,obs}$
2: For each B $\in X_{obs}$ and $e \in \Sigma_o$, define $f_{obs}(B,e) := UR(\{x \in X : (\exists x_e \in B)[x \in f(x_e, e)]\})$ whenever $f(x_e, e)$ is defined for some $x_e \in B$. In this case, add the state $f_{obs}(B,e)$ to X_{obs}. If $f(x_e, e)$ is not defined for any $x_e \in B$, then $f_{obs}(B,e)$ is not defined
3: Repeat Step 2 until the entire accessible part of G_{obs} has been constructed
4: $X_{m,obs} := \{B \in X_{obs} : B \cap X_m \neq \emptyset\}$

2.2.1 Operations on Automata

In order to properly analyze a DES modeled by an automaton it is necessary the definition of a set of operations useful for modifying the state transition diagram. Besides, operations capable of combining two or more automata are primordial once working with components of a system. The following operations are said to be unary and do not alter the elements in Σ.

Accessible Part

The accessible part is an operation that deletes all the unreachable states of an automaton G, starting from the initial state x_0. It takes along all the related transitions, and is formally defined as:

$$
\begin{aligned}
Ac(G) &:= (X_{ac}, \Sigma, f_{ac}, \Gamma_{ac}, x_0, X_{ac,m}) \text{ where} \\
X_{ac} &= \{x \in X : (\exists s \in \Sigma^*)[f(x_0, s) = x]\} \\
X_{ac,m} &= X_m \cap X_{ac} \\
f_{ac} &: X_{ac} \times \Sigma^* \to X_{ac}
\end{aligned}
$$

Note that the operation restrains the domain of the transition function to X_{ac} and does not alter $L(G)$ e $L_m(G)$.

Coaccessible Part

A state $x \in X$ is said to be coaccessible whenever there is a path starting from state x and ending in a marked stated belonging to X_m. The operation erases all the states in G that are not coaccessible, along with all its related transitions. Formally:

$$
\begin{aligned}
CoAc(G) &:= (X_{coac}, \Sigma, f_{coac}, \Gamma_{coac}, x_{0,coac}, X_m) \text{ where} \\
X_{coac} &= \{x \in X : (\exists s \in \Sigma^*)[f(x, s) \in X_m]\} \\
x_{0,coac} &:= \begin{cases} x_0 & \text{if } x_0 \in X_{coac} \\ \text{undefined} & \text{otherwise} \end{cases} \\
f_{coac} &: X_{coac} \times \Sigma^* \to X_{coac}
\end{aligned}
$$

Note that $L(CoAc(G)) \subseteq L(G)$. However, $L_m(CoAc(G)) = L_m(G)$.

Trim Operation

An automaton that is at the same time accessible and coaccessible is said to be trim, formally defined as:

$$
Trim(G) := CoAc[Ac(G)] = Ac[CoAc(G)]. \tag{2.13}
$$

Complement

Consider a trim deterministic automaton G that marks $L \subseteq \Sigma^*$ and, thus, generates $\overline{L}$. The automaton marking the language $\Sigma^* \setminus L$ is here denoted by G^{comp} and is built in two steps. The first one actually defines a complete automaton making f to be a total function f_{tot}. In order to do so, a new state x_d is added to X, known as the *dump state*. All undefined $f(x, e)$ in G is then assigned to x_d.

$$f_{tot}(x,e) = \begin{cases} f(x,e), & \text{if } e \in \Gamma_1(x) \\ x_d, & \text{otherwise} \end{cases}$$

Setting $f_{tot}(x_d, e)\ \forall e \in \Sigma$, the complete automaton is built, such that $L(G_{tot)} = \Sigma^*$ and $L_m(G_{tot}) = L$

$$G_{tot} = (X \cup x_d, \Sigma, f_{tot}, \Gamma_{tot}, x_0, X_m)$$

The next step towards building the complement is to change the marking status of all states in G_{tot} by marking all unmarked states, including x_d, and unmarking all marked states. The step yields the definition

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

From this point, the operations are named composition operations.

Product Operation

The product operation, also called the completely synchronous composition, is denoted by $\times$. The product between two automata G_1 e G_2 results in:

$$\begin{aligned} G_1 \times G_2 &:= Ac(X_1 \times X_2, \Sigma_1 \cup \Sigma_2, f, \Gamma_{1\times 2}, (x_{01}, x_{02}), X_{m,1} \times X_{m,2}) \text{ where} \\ f((x_1,x_2),e) &:= \begin{cases} (f_1(x_1,e), f_2(x_2,e)), & \text{if } e \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ \text{undefined}, & \text{otherwise} \end{cases} \\ \Gamma_{1\times 2}(x_1,x_2) &= \Gamma_1(x_1) \cap \Gamma_2(x_2). \end{aligned}$$

According to the definition of product composition, the transitions of both automata need to be synchronized with a common event, that is, an event $e \in \Sigma_1 \cap \Sigma_2$. An event will then only occur in $G_1 \times G_2$ if, and only if, it occurs simultaneously in G_1 e G_2.

The states of $G_1 \times G_2$ are denoted in pairs, in which the first component is the current state of G_1 and the second component is the current state of G_2. Besides, the generated and marked languages of $G_1 \times G_2$ are:

$$L(G_1 \times G_2) = L(G_1) \cap L(G_2), \tag{2.14}$$

$$L_m(G_1 \times G_2) = L_m(G_1) \cap L_m(G_2). \tag{2.15}$$

Parallel Composition

The parallel composition is also called synchronous composition and is represented by $||$. The parallel composition allows private transitions to occur, synchronizing only transitions labeled by common events.

The parallel composition between the two automata G_1 and G_2 results in:

$$G_1||G_2 \;:=\; Ac(X_1 \times X_2, \Sigma_1 \cup \Sigma_2, f, \Gamma_{1||2}, (x_{01}, x_{02}), X_{m,1} \times X_{m,2})$$

where

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

$$\Gamma_{1||2}(x_1,x_2) \;=\; [\Gamma_1(x_1) \cap \Gamma_2(x_2)] \cup [\Gamma_1(x_1) \setminus \Sigma_2] \cup [\Gamma_2(x_2) \setminus \Sigma_1].$$

In parallel composition, a common event, that is, belonging to $\Sigma_1 \cap \Sigma_2$, will occur only if both automata execute them at the same time. On the other hand, the private events, that is, the ones in $(\Sigma_1 \setminus \Sigma_2) \cup (\Sigma_2 \setminus \Sigma_1)$, may occur whenever it is possible.

If $\Sigma_1 = \Sigma_2$, then the parallel composition plays the same roll as the product operation.

In order to properly characterize both generated and marked languages of the automaton resulted from the parallel composition, a few definitions are needed:

$$P_i : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_i^* \text{ for } i = 1, 2. \tag{2.16}$$

Based on these projections, the resulting languages are:

$$L(G_1||G_2) \;=\; P_1^{-1}[L(G_1)] \cap P_2^{-1}[L(G_2)], \tag{2.17}$$

$$L_m(G_1||G_2) \;=\; P_1^{-1}[L_m(G_1)] \cap P_2^{-1}[L_m(G_2)]. \tag{2.18}$$

2.3 Diagnosability of discrete event systems

Consider the system shown in Figure 2.1. Suppose that one is entering the illustrated parking-garage and when it presses the open button it realizes that the parking-garage was stuck closed, or, reversely, by the press of the close button it is noticed that the gate is stuck opened. Using the language and automata formalisms presented here, we propose in Figure 2.3 a model for the parking-garage gate system considering these possible failures.

The system has a discrete-state set $X = \{C, O, S_c, S_o\}$ where S_c and S_o stands for *stuck closed* and *stuck opened* respectively. The event set of the system is given by $\Sigma = \{C_{lose}, O_{pen}, \sigma_f\}$, in which $\sigma_f \in \Sigma_f$, $\Sigma_f \subset \Sigma$ is modeling the possible failures

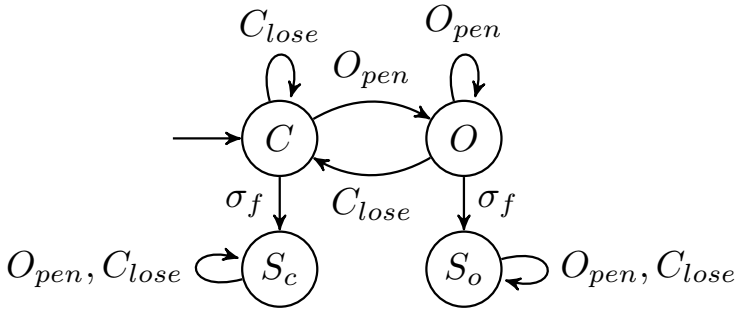


Figure 2.3: Automaton modeling the parking-garage system.

in the system. In case a person presses the close button and the gate does not closes, this person is able to drop the car and manually have the gate closed.

Assume now that two sensors are added to the system. The first one, positioned at the point where the gate is closed, capable of informing whether or not the gate is completely closed and, analogously, the second one monitoring the opening of the gate is positioned at the point where the gate is completely opened. Besides, consider that the open and close button will only work after the right information sent by each sensor, respectively the signal showing that the gate is completely closed and completely opened. Let us consider that, due to a failure in the motor responsible for opening the gate, the gate gets stuck closed. The sensor responsible for informing that the gate is completely closed does not inform this to the diagnoser and hence, even upon the press of the open button, the gate will not open.

Although this is a simple and visually detectable failure, many much more complex other failures has led the scientific community to deal with the problem of diagnosis of discrete event systems, which consists basically in detecting a failure in a system in order to improve performance and reliability. In this work we will deal with the event-based property of diagnosability of discrete event systems concerning the detection of the occurrence of a failure event.

Let $G = (X, \Sigma, f, \Gamma, x_0, X_m)$ model a system. The event set Σ is partitioned as $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$, where Σ_o and Σ_{uo} denote the set of observable and unobservable events, respectively, and let $\Sigma_f \subseteq \Sigma_{uo}$ be the fault events set that corresponds to all failures that may occur in the referred system. In addition, assume that the set of failure events can be partitioned as $\Sigma_f = \bigcup_{i=1}^{l} \Sigma_{f_i}$ where Σ_{f_i} represents the set of failure events of the same type. Assume, without loss of generality that $l = 1$, *i.e.*, there is only one failure type Σ_f. Let the language generated by G be denoted as $L(G) = L$ and assume that L is live. Let G_N be the subautomaton of G that represents the nonfailure behavior of the system and let $L(G_N) = K$, which is a prefix-closed language formed with all traces of L that do not contain any failure event from the set Σ_f. The language-based diagnosability definition may now be

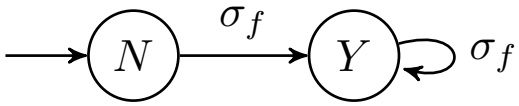


Figure 2.4: Faulty label automaton A_l.

presented.

Definition 2.8 *(Diagnosability of discrete event systems) Let L be the prefix closed language generated by G and $K \subset L$ denote the prefix-closed language generated by G_N. Define the projection $P_o : \Sigma^* \to \Sigma_o^*$ and let Σ_f be the set of failure events. Then, L is diagnosable with respect to P_o and Σ_f if, and only if,*

$$(\exists n \in \mathbb{N})(\forall s \in L \setminus K)(\forall st \in L \setminus K, |t| \geq n)$$
$$\Rightarrow (\forall w \in P_o^{-1}(P_o(st)) \cap L, w \in L \setminus K).$$

□

According to Definition 2.8, L is diagnosable with respect to P_o and Σ_f if, and only if, for all traces st of arbitrarily long length after the occurrence of a failure event, there does not exist a trace $s \in K$ such that $P_o(s) = P_o(st)$.

The property of a language of being or not diagnosable is tested through the use of two test automata: *diagnosers*, capable of performing online detection and isolation of failures and offline verification of the diagnosability properties of the system, and *verifiers*, that perform the best offline verification and are the main focus of this work.

Diagnosers are deterministic automata based on a given system model G whose event set is formed with the observable events of G. The states of a deterministic automaton G_d have labels Y and N attached to the states of G to indicate whether or not the fault event has occurred. The diagnoser is formally given by

$$G_d = (X_d, \Sigma_o, f_d, \Gamma_d, x_{0,d}) = Obs(G||A_l, \Sigma_o) \tag{2.19}$$

where A_l is the two state label automaton shown in Figure 2.4. Note that their computational complexity is exponential in the number of states.

When a diagnoser is in a certain (resp. normal) state, it is certain that a fault has (resp. has not) occurred. However, if the diagnoser is in an uncertain state, it is not sure whether or not the fault event has occurred. If there exists a cycle formed with uncertain states where the diagnoser can forever remain, then it will never be able to diagnose the fault occurrence; on the other hand if somehow it always leaves this cycle of uncertain states, then this cycle is not indeterminate. The indeterminate cycle definition proposed by SAMPATH *et al.* [4] becomes useful.

Definition 2.9 *(Indeterminate cycles of G_d) A set of uncertain states $\{x_{d_1}, x_{d_2}, \ldots, x_{d_p}\} \subset X_d$ forms an indeterminate cycle if the following conditions hold true:*

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

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

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

Let us refer the indeterminate cycles just defined as *indeterminate observed cycles*. It is worth remarking that the concept will be useful for the comprehension of the work to be presented.

2.4 Verification of decentralized diagnosability of discrete event systems

Since the main topic of this work is a verification algorithm, we may present the polynomial time verification algorithm for decentralized diagnosis of discrete event systems proposed by MOREIRA *et al.* [6].

Verifiers are automata constructed according to a given algorithm that are capable of performing offline language diagnosis of discrete event systems. Specifically, we will focus on the type of verifier that searches for cycles in the original system with a given property. They are an advantageous alternative to diagnosers when performing offline diagnosis since they can be built out of a polynomial time algorithm compared to the exponential complexity from the diagnosers.

The verifier to be here presented is capable of performing the verification of both codiagnosability and centralized diagnosability of discrete event systems with a polynomial time computational complexity and the feature of only searching for traces that leads to violation of the diagnosability definition. The verifier is constructed according to Algorithm 2.3. The following theorem states the correctness of Algorithm 2.3 [6].

Theorem 2.1 *Let L and $K, (K \subset L)$, denote the prefix-closed languages generated, respectively, by G ad G_N. Assume there are m local sites with projections $P_{o_i} : \Sigma^* \to \Sigma_{o_i}^*$, $i \in I$, and let Σ_f be the set of failure events. Then, L is not*

Algorithm 2.3 Verification of decentralized diagnosability of DES

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

Output: $G_V = (X_V, \Sigma_{R_1} \cup \Sigma_{R_2} \cup \ldots \Sigma_{R_m} \cup \Sigma, f_V, \Gamma_V, x_{0,V})$

Let G be a deterministic automaton and Σ_f the set of failure events. Assume that, for each local diagnoser, Σ is partitioned as $\Sigma = \Sigma_{o_i} \dot{\cup} \Sigma_{uo_i}$, $i \in I$, $I = \{1, 2, \ldots, m\}$, where Σ_{o_i} and Σ_{uo_i} are the observable and unobservable event sets for each local diagnoser, respectively.

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

1.1: Define $\Sigma_N = \Sigma \setminus \Sigma_f$

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

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

1.4: Redefine the event set of G_N as Σ_N, i.e., $G_N = (X_N, \Sigma_N, f_N, \Gamma_N, x_{0,N})$.

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

2.1: Define $A_l = (X_l, \Sigma_f, f_l, \Gamma_l, x_{0,l})$, where $X_l = \{N, Y\}, x_{0,l} = N, f_l(N, \sigma_f) = Y$ and $f_l(Y, \sigma_f) = Y$, for all $\sigma_f \in \Sigma_f$.

2.2: Compute $G_l = G||A_l$ and mark the states of G_l whose second coordinate is equal to Y.

2.3: Compute the failure automaton[3] $G_F = CoAc(G_l)$.

3: Define function $R_i : \Sigma_N \to \Sigma_{R_i}$ as [4]:

$$R_i(\sigma) = \begin{cases} \sigma, \text{ if } \sigma \in \Sigma_{o_i} \cup \Sigma_f \\ \sigma_{R_i}, \text{ if } \sigma \in \Sigma_{uo_i} \setminus \Sigma_f \end{cases} . \tag{2.20}$$

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

4: Compute the verifier automaton $G_V = G_{N,1}||G_{N,2}|| \ldots ||G_{N,m}||G_F = (X_V, \Sigma_{R_1} \cup \Sigma_{R_2} \cup \Sigma, f_V, \Gamma_V, x_{0,V})$. Notice that a state of G_V is given by $x_V = (x_{N,1}, x_{N,2}, x_F)$, where $x_{N,1}, x_{N_2}$ and x_F are states of $G_{N,1}, G_{N,2}$ and G_F, respectively and $x_F = (x, x_l)$, where x and x_l are states of G and A_l, respectively.

5: Verify the existence of a cycle[5] $cl = (x_V^k, \sigma_k, x_V^{k+1}, \ldots, x_V^l, \sigma_l, x_V^k)$, where $l \geq k \geq 0$, in G_V satisfying the following conditions:

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

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

codiagnosable with respect to P_{o_i} *and* Σ_f *if and only if there exists a cycle in* G_V, $cl = (x_V^k, \sigma_k, x_V^{k+1}, \ldots, x_V^l, \sigma_l, x_V^k)$, *where* $l \geq k \geq 0$, *satisfying the following conditions:*

$$\exists j \in \{k, k+1, \ldots, l\} \text{ s.t. for some } x_V^j, (x_l^j = Y) \wedge (\sigma_j \in \Sigma). \tag{2.21}$$

The computational complexity of Algorithm 2.3 is $O(m \times |X|^{m+1} \times (|\Sigma| - |\Sigma_f|))$, which shows that it requires polynomial time in the number of states and events of G but like all other methods in literature, it has exponential complexity in the number of local diagnosers.

Remark 1 *Compared to diagnosers, verifiers are much better options for offline verification of the language diagnosability property of discrete event systems since many proposed algorithms proposed in literature require polynomial-time computational complexity whereas diagnosers require exponential complexity in the number of states of a system model.*

2.5 Final comments

This chapter presented a fragment of the theory of discrete event systems that is relevant to the comprehension of this work. It has covered simple introductory concepts such as systems, events and states. Moreover, it has presented the concepts of languages and automata in a way to mathematically formalize discrete event systems [2]. Finally, it has introduced the concepts of failure diagnosis of discrete event systems and the verification of the language diagnosability definition using diagnosers and verifiers.

Chapter 3

Different approaches on fault diagnosis of discrete event systems

In this chapter we will present different approaches on fault diagnosis of discrete event systems. Each subsection will explain a portion of a paper available in literature regarding different addressed problems. In Subsection 3.1 we learn about the problem of diagnosability of DESs subject to permanent sensor failures [26]. In Subsection 3.2 we will refer to robust diagnosability of DESs subject to permanent sensor failures[20]. In Subsection 3.3 the work considers the problem of robustly diagnose a system subject to intermittent sensor failures [21]. In Subsection 3.4 we deal with the problem of verifying the robust diagnosability of partially observed DESs [23] and in Subsection 3.5 we finally present the definition of generalized robust diagnosability [27].

3.1 Diagnosability of discrete event systems subject to permanent sensor failures

KANAGAWA and TAKAI [26] considered the problem of failure diagnosis of discrete event systems that are subject to permanent sensor failures by proposing a notion of diagnosability with respect to a nondeterministic observation mask. It also proposed a verification mechanism based on an aggregated Mealy automaton with a deterministic and state-dependent observation mask and showed how the diagnosability of the aggregated Mealy automaton is equivalent to the diagnosability of the original system. Finally, the paper computed the delay bound for detecting the occurrence of a failure string subject to permanent sensor failures.

The definition of diagnosability subject to permanent sensor failures proposed by KANAGAWA and TAKAI [26] demands that a few concepts are introduced so that it can be entirely comprehended. Let us consider the observation mask

$M : \Sigma \to \Delta \cup \{\varepsilon\}$ where Δ is the set of observable symbols (not to be confused with the set of observable events). The observation mask M can be extended to $M : \Sigma^* \to \Delta^*$ in the usual manner. The occurrence of an event σ is observed as $M(\sigma)$. If $M(\sigma) = \varepsilon$, an event is said to be unobservable; otherwise, σ is an observable event. Comparing the observation mask to the projection operation presented in Section 2.1.2, we conclude that the projection operation may be a particular case of the observation mask where the symbols in the set of observable symbols Δ are the events in the smaller event set Σ_s^*. Hence, instead of using the projection operation, KANAGAWA and TAKAI [26] use the observation mask in their diagnosability definitions.

In order to illustrate how the observation mask works, let us consider the motivating example [26] proposed by CASSANDRAS and LAFORTUNE [2] consisting on a pump, a valve, a controller, a pump pressure sensor and a valve flow sensor. The purpose of the diagnosis is to detect a failure such that the valve gets stuck closed.

Consider that the system is modeled by automaton G represented by Figure 3.1(a) where $\Sigma = \{c_1, c_2, c_3, o_1, o_2, v\}$ where $c_i, i = 1, 2, 3$, are commands issued by the controller; o_1 is a change in pressure sensor reading; o_2 is a change in flow sensor reading; and v models the failure when the valve gets stuck closed. The nonfailure behavior of G is modeled by automaton R in Figure 3.1(b), that generates the language K. After the occurrence of event v representing the failure of the valve, this will not open even with the first command of the controller (c_1). Thus, the sensor flow reading will remain the same, which is the same as saying that event o_2 will not occur after the occurrence of v.

Assume that the observation mask M is given by

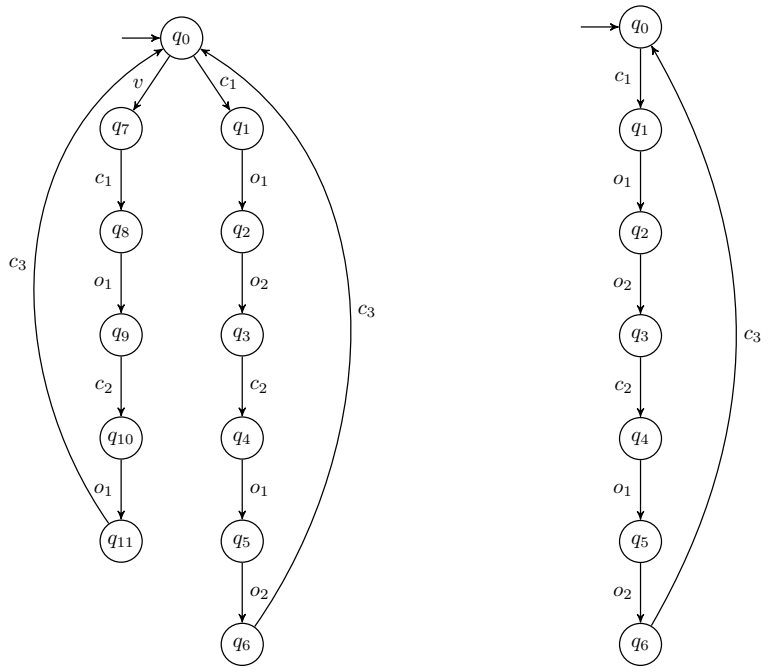
$$M(\sigma) = \begin{cases} \sigma, \text{ if } \sigma \in \Sigma - \{v\} \\ \varepsilon, \text{ if } \sigma = v \end{cases} \tag{3.1}$$

where $\Delta = \Sigma - \{v\}$. The mask demonstrates that all events are observable except for the failure event v.

It is assumed that a cyclical trace s^k occurs and that, after the occurrence of the sequence $s^k v c_1$, the pressure sensor fails permanently and the event o_1 becomes unobservable. The incident is modeled by the observation mask M_1 as follows.

$$M_1(\sigma) = \begin{cases} \sigma, \text{ if } \sigma \in \Sigma - \{o_1, v\} \\ \varepsilon, \text{ if } \sigma \in \{o_1, v\} \end{cases} \tag{3.2}$$

Let us consider any string $s^k v \in L(G) \setminus K$ that ends with a failure event v, where $s = c_1 o_1 o_2 c_2 o_1 o_2 c_3$ and $k \geq 0$. For its extension $s^k v c_1 o_1 c_2 \in L(G) \setminus K$, there is no nonfailure string $u \in K$ such that $M(u) = M(s^k v c_1 o_1 c_2) = s^k c_1 o_1 c_2$.



(a) G (b) R

Figure 3.1: Automata G and R from the motivating example.

This means that G is diagnosable with respect to K and M. However, after the permanent failure of the pressure sensor the string $s^k v c_1 o_1 c_2 \in L(G) - K$ is observed as $M(s^k v c_1) M_1(o_1 c_2) = s^k c_1 c_2$. For any nonfailure string $u \in K$ there are no strings $u_1, u_2 \in \Sigma^*$ such that $u = u_1 u_2$ and $M(u_1) M'(u_2) = s^k c_1 c_2$. That is, even if the event o_1 becomes unobservable due to the failure of the pressure sensor, the failure string $s^k v c_1 o_1 c_2 \in L(G) - K$ can be distinguished from any nonfailure string. This motivates the consideration of diagnosability subject to permanent sensor failures.

In [26] it is considered that the sensing capability of a diagnoser may degrade to permanent sensor failures and it is assumed that a permanent sensor failure occurs at most once, at any time during system operation, and that it is not detected by a diagnoser. This lost of sensor reading is modeled by a transition from the nominal observation mask M to a coarser observation mask M_i, where i stands for the number of sensor failures in a system. It is assumed that there are n different types of transitions from the nominal observation mask and therefore the resulting observation masks are denoted by $M_i : \Sigma \to \Delta \cup \{\varepsilon\}$ $(i \in I)$, where $I := \{1, 2, \ldots, n\}$ is the index set.

This transition between masks is here modeled as presented by ROHLOFF [16]. Consider a system modeled by G with observable event set $\Sigma_o = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ and nominal observation mask M

$$M(\sigma) = \begin{cases} \sigma, \text{ if } \sigma \in \Sigma_o \\ \varepsilon, \text{ otherwise} \end{cases} \tag{3.3}$$

where $\Delta = \Sigma_o$. In [16] it is considered that at most one observable event $\sigma_i \in \Sigma$ becomes unobservable due to the failure of the corresponding event sensor, *i.e.*, the failure of a sensor affects only one event observation, which can be described as the transition from the nominal mask M and M_i,

$$M_i(\sigma) = \begin{cases} \sigma, \text{ if } \sigma \in \Sigma_o - \{\sigma_i\} \\ \varepsilon, \text{ otherwise} \end{cases} \tag{3.4}$$

The observation mask $M^f : \Sigma^* \to 2^{\Delta^*}$ subject to permanent sensor failure is defined as

$$M^f(s) = \{M(s_1)M_i(s_2) \in \Delta^* \mid s_1 s_2 = s \ \wedge \ i \in I\} \tag{3.5}$$

where $M(s_1)M_i(s_2)$ denotes the observed string of $s = s_1 s_2$ when the transition from M to M_i occurs after the occurrence of s_1 [16]. Note that M^f is a nondeterministic observation mask since it assigns the set of possible observed strings to each $s \in \Sigma^*$. The set of all observable strings is given by

$$M^f(L(G)) = \bigcup_{s \in L(G)} M^f(s) \tag{3.6}$$

Let us consider the diagnosing problem of detecting the occurrence of any failure string in $L(G) - K$ within an uniformly bounded number of steps being G subject to permanent sensor failures. A diagnoser for G is formally defined as a function $D : M^f(L(G)) \to \{0, 1\}$. Whenever D is certain that a failure has occurred, it issues the answer 1; otherwise, the decision is 0. In order to correctly detect the occurrence of any failure string within an uniformly bounded number of steps this diagnoser D is required to satisfy conditions C_1 and C_2 as follows

$$\begin{aligned} &\textbf{C1)}\ \exists m \in \mathbb{N}, \forall s \in L(G) \setminus K, \forall t \in L(G)/s : [|t| \geq m \ \vee \ st \in L_d(G)] \Rightarrow \\ &\qquad [\forall \tau \in M^f(st) : D(\tau) = 1], \\ &\textbf{C2)}\ \forall s \in K, \forall \tau \in M^f(s) : D(\tau) = 0. \end{aligned} \tag{3.7}$$

where **C1** means that there exists $m \in \mathbb{N}$ such that the occurrence of any failure string is detected within m steps under any observed string and **C2** guarantees the decision "1" is issued after the occurrence of a failure string.

Taking this problem in consideration a diagnosability property subject to permanent sensor failures was defined as follows.

Definition 3.1 *The system G is said to be diagnosable with respect to a nonempty closed sublanguage $K \subseteq L(G)$ and the observation mask $M^f : \Sigma^* \to 2^{\Delta^*}$ if*

$$\exists m \in \mathbb{N}, \forall s \in L(G) \setminus K, \forall t \in L(G)/s : [|t| \geq m \vee st \in L_d(G)] \Rightarrow$$
$$[\forall u \in L(G) : M^f(st) \cap M^f(u) \neq \emptyset \Rightarrow u \notin K].$$

A theorem showing that diagnosability defined in Definition 3.1 is a necessary and sufficient condition for the existence of a diagnoser that satisfies conditions **C1** and **C2** is proposed.

Theorem 3.1 *Given a nonempty closed sublanguage $K \subseteq L(G)$ and the observation mask $M^f : \Sigma^* \to 2^{\Delta^*}$, there exists a diagnoser $D : M^f(L(G)) \to \{0, 1\}$ that satisfies **C1** and **C2** if, and only if, the system G is diagnosable with respect to K and M^f.*

If G be diagnosable with respect to K and M^f, then a diagnoser $D : M^f(L(G)) \to \{0, 1\}$ given as

$$D(\tau) = \begin{cases} 1, \text{ if } \forall u \in L(G) : \tau \in M^f(u) \Rightarrow u \notin K \\ 0, \text{ otherwise} \end{cases} \tag{3.8}$$

for each $\tau \in M^f(L(G))$ satisfies **C1** and **C2**.

The work presented by KANAGAWA and TAKAI [26] also covers the verification of the proposed definition through an aggregated Mealy automaton and the delay bound within which the occurrence of any failure string can be detected subject to permanent sensor failures. Since the remaining topics are not central to the extent of this work, we will only underline that the computational complexity of the presented state-dependent verification method is $O(n^2 \times |X| \times |X_N|^2 \times (n + |\Sigma|^2))$, where X_N stands for the number of states of the nonfailure specification automaton G_N that generates the sublanguage K.

3.2 Robust diagnosis of discrete event systems against permanent loss of observation

CARVALHO *et al.* [20] consider the problem of diagnosing the occurrence of a fault event in the operation of a partially-observed discrete event system subject to permanent loss of observation modeled by a finite state automaton. It assumes that certain sensors may fail at the outset and that the diagnoser is not aware of such failure - therefore generating the loss of observation. A previous definition of robust diagnosability is considered and a polynomial time verification algorithm sufficient for verifying the robust diagnosability of the system is presented. Furthermore, a methodology to perform online diagnosis using a set of partial diagnosers for a system subject to permanent sensor failure is provided.

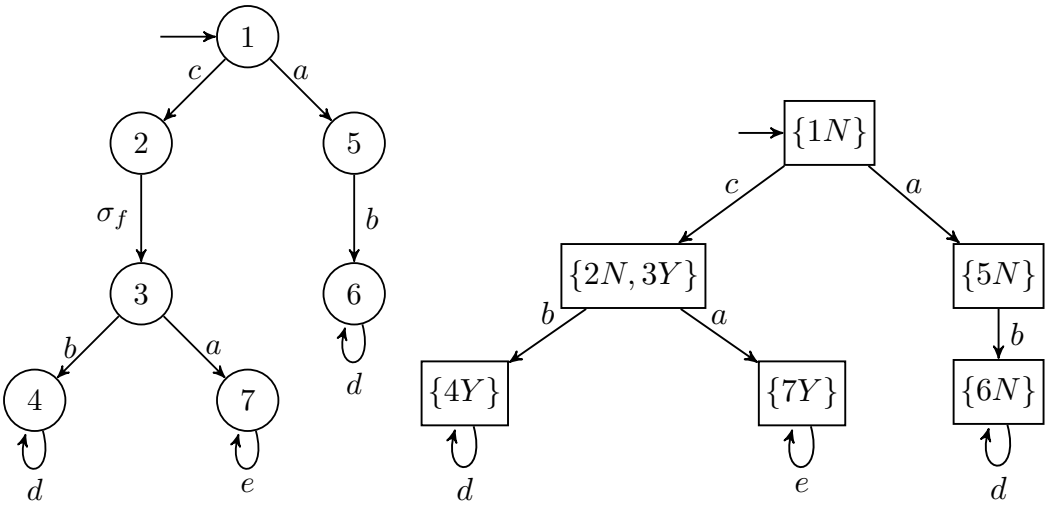


Figure 3.2: Automaton G and its diagnoser G_d.

The problem addressed in [20] is related to the one presented in Section 3.1 in a sense that it also considers that a sensor may permanently fail and compromise the fault diagnosis of a discrete event system. While KANAGAWA and TAKAI [26] considers that a sensor may fail at anytime during the system operation but only one sensor may fail at a time, CARVALHO *et al.* [20] consider that a sensor may only fail prior to the first occurrence of the event it is monitoring. However, in [20] the robustness of the definition allows more than one sensor to fail at the same time.

In order to clarify the concept, let us consider the system G depicted in Figure 3.2 along with its diagnoser automaton G_d.

Since there are only cycles in certain states of G_d, the language L generated by G is said to be diagnosable. Suppose that the trace $s_Y = c\sigma_f a e^n$ has occurred and that the sensor responsible for readings of the event c fails. When the trace s_Y occurs, the diagnoser will not be able to observe events c or σ_f before the occurrence of a. Hence, it will update to the state $5N$ where it will remain upon occurrence of e^n. Even though the language was originally diagnosable with respect to P_o and Σ_f, after the failure of the sensor the diagnoser becomes incapable of correctly diagnosing the system, demanding not only a modification on the diagnoser but also in the diagnosability definition. Just like the sensor responsible for monitoring event c has failed, the other sensors could have failed as well, demanding not only a new definition but one robust enough to encompass all the failure occurrences.

Hence, let us take the following assumptions into account.

- **A3.** L is diagnosable with respect to $P_o : \Sigma^* \to \Sigma_o^*$ and $\Sigma_f = \{\sigma_f\}$.
- **A4**[1]**.** Any loss of observations when it occurs, takes place before the first occurrence of the (initially observable) event associated with the sensor that

[1]Not restrictive in the case of a cyclical system that observably returns to its initial state.

has failed, and it is permanent, i.e., the event remains unobservable.

Consider also the definition proposed by BASILIO *et al.* [28].

Definition 3.2 *(Diagnosis basis) A set $\Sigma'_o \subseteq \Sigma_o$ is a diagnosis basis if L is also diagnosable with respect to projection $P'_o : \Sigma^* \to \Sigma'^*_o$ and $\Sigma_f = \{\sigma_f\}$. If for any nonempty subset $\Sigma''_o \subseteq \Sigma'_o$, L is not diagnosable with respect to projection $P''_o : \Sigma^* \to \Sigma''^*_o$ and $\Sigma_f = \{\sigma_f\}$ then Σ'_o is a minimal diagnosis basis.*

According to Definition 3.2, the sets that are diagnosis bases ensure diagnosability of L and thus use the redundancy of the events in $\Sigma_o \setminus \Sigma'_o$ to provide some robustness to the diagnosing system. This leads to the definition of robust diagnosability against permanent loss of observations.

Definition 3.3 *(Robust diagnosability against permanent loss of observations) Let $\Sigma_{db} = \{\Sigma_{o_1}, \Sigma_{o_2}, \ldots, \Sigma_{o_m}\}$, where Σ_{o_i}, $i \in I_m$ are either minimal or nonminimal diagnosis bases for L. Define the set*

$$\Sigma_{rob} = \{\Sigma_{uo_1}, \Sigma_{uo_2}, \ldots, \Sigma_{uo_m}\}, \tag{3.9}$$

where

$$\Sigma_{uo_i} = \Sigma_o \setminus \Sigma_{o_i},\ i \in I_m. \tag{3.10}$$

Then L is robustly diagnosable with respect to projections $P_{o_1}, P_{o_2}, \ldots, P_{o_m}$, where $P_{o_i} : \Sigma^* \to \Sigma^*_{o_i}$, and $\Sigma_f = \{\sigma_f\}$, or equivalently, with respect to permanent loss of observation of the events of all sets Σ_{uo_i}, $i \in I_m$, and $\Sigma_f = \{\sigma_f\}$, if the following condition holds true:

$$\begin{gathered}(\exists n \in \mathbb{N})(\forall s \in L \setminus K)(\forall t \in L/s) \\ (\|t\| \geq n \Rightarrow R_D),\end{gathered} \tag{3.11}$$

where the robust diagnosability condition R_D is given as

$$\begin{gathered}(\forall i, j \in \{1, 2, \ldots, m\}, i \neq j) \\ (\nexists\, \omega_j \in L)[\Sigma_f \notin \omega_j \wedge P_{o_i}(st) = P_{o_j}(w_j)].\end{gathered}$$

The idea behind Definition 3.3 is that since L is diagnosable with respect to $P_{o_i} : \Sigma^* \to \Sigma^*_{o_i}$, and $\Sigma_f = \{\sigma_f\}$, and assuming that all partial diagnosers for Σ_{o_k}, $k \in I_m$ are running simultaneously and have access to all available sensors, any partial diagnoser, say Σ_{o_i}, only performs properly if all events in Σ_{uo_i} become unobservable, *i.e.*, observation of all events in Σ_{uo_i} is lost. In this case, while some partial diagnosers may get stuck, others may continue running, since it is possible that the intersections of the languages generated by two different partial diagnosers

be nonempty. This implies that it is possible that an arbitrarily long trace s_Y that contains the fault event has the same projection over, say $\Sigma_{o_i}^*$ and $\Sigma_{o_j}^*$, where the former takes G_{d_i} to a certain state whereas the latter takes G_{d_j} to a normal state. In this case, according to Definition 3.3, L is not robustly diagnosable against permanent loss of observation of the events in Σ_{uo_i} and Σ_{uo_j}.

Even though it is not part of the central objective of this work, the paper presented by CARVALHO *et al.* [20] is followed by the proposition of verification of robust diagnosability against permanent loss of observations using verifiers, resulting in a computational complexity of $O(m^2|X|^2|\Sigma|)$, and the online implementation of robust diagnosers concerning the robust language diagnosability definition presented.

3.3 Robust diagnosis of discrete event systems against intermittent loss of observations

CARVALHO *et al.* [21] assume that intermittent loss of observations may occur during the fault diagnosis of a discrete event system. Bad sensor operation can make sensors fail to report event occurrences. Besides, bad electrical linkage and possible atmospheric interference in the communication channels of a system may lead to loss of communication between sensors and the system's diagnoser. These occurrences would be responsible for intermittent loss of observations during fault diagnosis of DESs. Hence, in [21] a robust diagnosability against intermittent loss of observations definition is presented, along with two tests for robust diagnosability - one using diagnosers and the other one using verifiers - and also a modeling mechanism for systems with intermittent behavior. Finally, the results are extended to robust codiagnosability against intermittent loss of observations.

The problem here addressed is better comprehended through the observation of a motivating example.

Example 2 *Figures 3.3(a) and 3.3(b) show the state transition diagrams of an automaton G, for which $\Sigma = \{a, b, c, d, e, \sigma_f\}$, $\Sigma_o = \{a, b, c, d, e\}$ and $\Sigma_f = \{\sigma_f\}$, and the corresponding diagnoser G_d, respectively. It is immediate to see that, since G_d has no indeterminate cycles, the language generated by G is diagnosable with respect to P_o and Σ_f.*

Assume, initially, that, for some $n \in \mathbb{N}$, the trace $s'_Y = c\sigma_f abd^n$ has been generated and suppose that the occurrence of event c has not been recorded somehow. Since event c has become unobservable, the first event occurrence to be recognized by G_d is a, which takes the diagnoser state to $\{5N\}$. When the next events of s'_Y

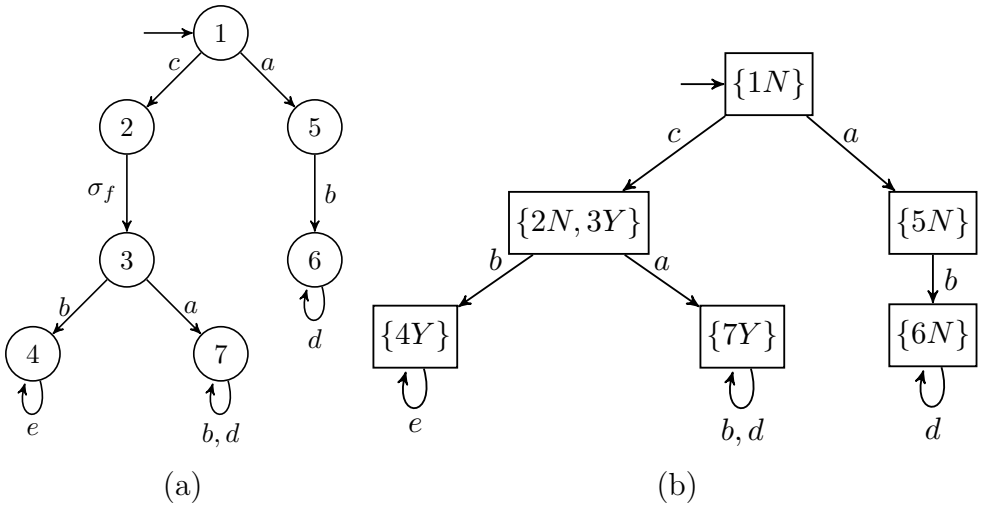


Figure 3.3: Automaton G (a) and its corresponding diagnoser G_d (b).

occur, the diagnoser moves to state $\{6N\}$, where it stays as long as event d continues to occur, therefore displaying wrong information regarding the occurrence of σ_f. Assume, now, that, for some $n \in \mathbb{N}$, trace $s''_Y = c\sigma_f c e^n$ has been generated and assume also that event c is subject to intermittent loss of observation. If the first occurrence of event c is not recognized by the diagnoser, then G_d remains in its initial state. If, in the sequel, the communication between the system and the diagnoser is somehow restored before the second occurrence of event c, then when c occurs for the second time, the diagnoser moves to state $\{2N, 3Y\}$. Notice that since the next event of s''_Y to occur is e, which is not in the active event set of $\{2N, 3Y\}$, the diagnoser stands still in an uncertain state, and, once again, provides wrong information regarding the fault occurrence. This anomalous behavior suggests that the system model should be modified to take into account intermittent loss of observations due to sensor malfunction or communication failure between sensors and diagnoser. This necessity for a new modeling led to the definition of the dilation operation.

Definition 3.4 *(Dilation) Let $\Sigma = \Sigma_{ilo} \dot{\cup} \Sigma_{nilo} \dot{\cup} \Sigma_{uo}$ be a partition of Σ, where Σ_{ilo} is the set of observable events associated with intermittent loss of observations and Σ_{nilo} denotes the set of observable events not subject to intermittent loss of observations and let $\Sigma'_{ilo} = \{\sigma' : \sigma \in \Sigma_{ilo}\}$ and $\Sigma_{dil} = \Sigma \cup \Sigma'_{ilo}$. The dilation D is the mapping*

$$\begin{array}{rcccl} D & : & \Sigma^* & \rightarrow & 2^{(\Sigma_{dil})^*} \\ & & s & \mapsto & D(s), \end{array} \tag{3.12}$$

where

$$\begin{aligned} D(\varepsilon) &= \{\varepsilon\}, \\ D(\sigma) &= \begin{cases} \{\sigma\}, \ \textit{if } \sigma \in \Sigma \backslash \Sigma_{ilo}, \\ \{\sigma, \sigma'\}, \ \textit{if } \sigma \in \Sigma_{ilo}, \end{cases} \\ D(s\sigma) &= D(s)D(\sigma), s \in \Sigma^*, \sigma \in \Sigma. \end{aligned} \tag{3.13}$$

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

$$D(L) = \bigcup_{s \in L} D(s). \tag{3.14}$$

Consider a system G with an associated event set Σ_{ilo}. For each event $\sigma_{ilo} \in \Sigma_{ilo}$ labeling a transition, the dilation operation will add a new transition to G parallel to the original transition labeled by σ'_{ilo} instead. The operation extended to every event which monitoring sensor is subject to intermittent loss of observation originates G_{dil}, a deterministic automaton that maintains the same observable event set Σ_o from G.

A very important result may then be presented as a theorem showing that G_{dil} models the behavior of a system G subject to intermittent loss of observations by establishing a relationship between the languages generated by G and G_{dil}, L and L_{dil}, respectively. The proof of Theorem 3.2 is in [21].

Theorem 3.2 *Let $G_{dil} = (X, \Sigma_{dil}, f_{dil}, \Gamma_{dil}, x_0)$ be a deterministic automaton obtained from G. Then, $L_{dil} = L(G_{dil}) = D(L)$.*

Consider automaton G from Example 2. Consider that $\Sigma = \{a, b, c, d, e, \sigma_f\}$ and $\Sigma_{ilo} = \{c\}$. If we apply the dilation operation to the referred example, we obtain the automaton G_{dil} described in Theorem 3.2, as shown in Figure 3.4.

Since the dilation operation has been fully presented, we may now introduce the definition of robust diagnosability of DES subject to intermittent loss of observations.

Definition 3.5 *(Robust diagnosability of DES subject to intermittent loss of observations) A prefix-closed and live language L, generated by an automaton G, is robustly diagnosable with respect to dilation D, projection $P_{dil,o} : \Sigma^*_{dil} \to \Sigma^*_o$ and $\Sigma_f = \{\sigma_f\}$ if the following holds true:*

$$(\exists n \in \mathbb{N})(\forall s \in L \setminus K)(\forall t \in L/s)(\|t\| \geq n \Rightarrow R_D),$$

where the robust diagnosability condition R_D is

$$(\nexists \omega \in L)[(P_{dil,o}(D(st)) = P_{dil,o}(D(\omega))) \wedge (\Sigma_f \notin \omega)]. \tag{3.15}$$

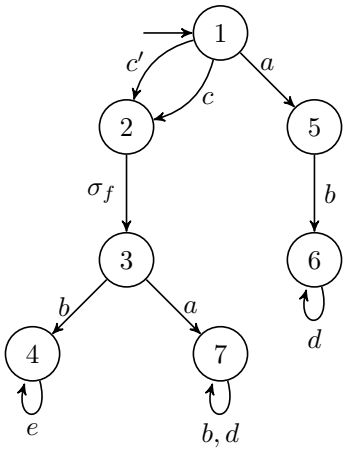


Figure 3.4: Automaton G_{dil} resulted from the application of the dilation operation to G where $\Sigma_{ilo} = \{c\}$.

Remark 2 *Note that if $\Sigma_{ilo} = \emptyset$ then $L_{dil} = L$, $D(st) = \{st\}$ and $P_{dil,o}$ reduces to P_o. In this case, Definition 3.5 reduces to the usual definition of language diagnosability introduced in [4].*

The work presented in [21] extends to robust diagnoser construction methodologies and application of verification algorithms to robustly diagnose a language subject to intermittent loss of observations. Even though the main concepts regarding this work were already presented, we may highlight that the computational complexity to build a robust diagnoser $G_{dil,d}$ is, in the worst case, $O(2^{|X|} \times |\Sigma|)$ where X and Σ denote the state and event sets of the original system G, respectively. Meanwhile, the worst case for the construction of $G_{dil,t}$ leads to a complexity of $O(2^{n|X|} \times |\Sigma|)$. Finally, the computation complexity using the verifier proposed by MOREIRA *et al.* [6] requires polynomial time in the number of states and events of G and is, in the worst case, $O(|X|^2 \times |\Sigma|)$ for the diagnosability test and $O(m \times |X|^{n+1} \times |\Sigma|)$ for the codiagnosability test.

3.4 Verification of robust diagnosability for partially observed discrete event systems

The paper presented by TAKAI [23] considers the robust failure diagnosis of discrete event systems. From a given set of possible models and its corresponding nonfailure specifications, the work suggests the existence of a single diagnoser, so called *robust diagnoser*, for the detection of the occurrence of a failure, in all possible models, within a uniformly bounded number of steps. A notion of robust diagnosability is introduced and proved that is serves as a necessary and sufficient condition for the existence of a robust diagnoser. Finally, the work presents an algorithm for verifying

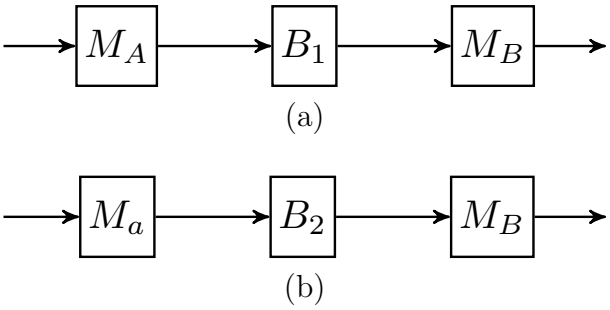


Figure 3.5: Two configurations $C1$ (a) and $C2$ (b) of the manufacturing line.

the proposed robust diagnosability condition.

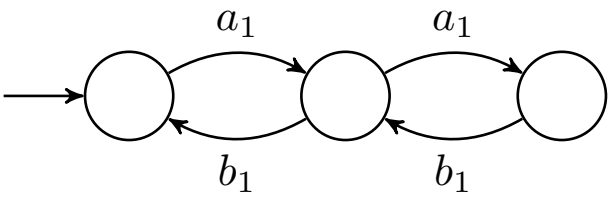
Motivated by the possibility of developing a robust tool capable of encompassing modeling uncertainties, multiple configuration systems such as manufacturing systems and at the same time reduce costs of implementation and maintenance, the work proposed was remarking since it considered a set of possible models, different of what was considered by the majority of the research community back then.

Assume that a system has an exact model that belongs to a set of n possible models $\{G_i | i \in I\}$, $I = \{1, 2, \ldots, n\}$. Since G_i are all possible models of a same system, the event set Σ is the a common finite set for all possibilities. Besides, each G_i has an associated G_{k_i} generating the nonempty closed sublanguage $K_i \subseteq \text{Ł}(G_i)$ that models the nonfailure behavior of the system.

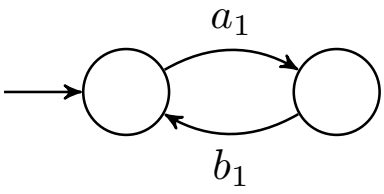
Consider the simple manufacturing line depicted in Figure 3.5. Configuration $C1$ (resp. $C2$) consists of two machines M_A and M_B (resp. M_a and M_B) and a two-slot buffer B_1 (resp. one-slot buffer B_2). The machine M_A can process two parts simultaneously, while M_a and M_B cannot process more than one part at a time. The machines M_A, M_B, and M_a are modeled by automata G_{M_A}, G_{M_B} and G_{M_a} as shown in Figure 3.6 where the event labels represent the following actions:

- a_1: M_A (resp. M_a) starts processing a part in C_1 (resp. C_2),
- b_1: M_A (resp. M_a) completes processing and passes a part to B_1 (resp. B_2) in C_1 (resp. C_2),
- a_2: M_B takes a part from B_1 (resp. B_2) and starts processing in C_1 (resp. C_2),
- b_2: M_B completes processing.

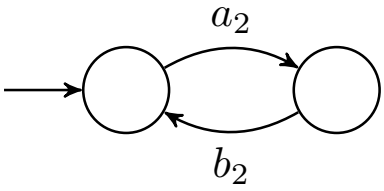
Automata models G_{B_1} and G_{B_2} of the buffers B_1 and B_2, respectively, are shown in Figure 3.7. For each configuration, consider a nonfailure specification such that the overflow and underflow of the buffer do not occur. The behaviors of C_1 and C_2 are described by the synchronous compositions $G_1 = G_{M_A} || G_{M_B}$ and $G_2 = G_{M_a} || G_{M_B}$, respectively. Also, the nonfailure specification for G_1 and G_2 are modeled



(a) G_{M_A}

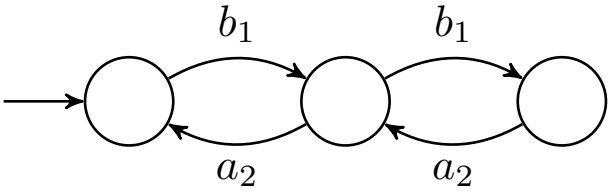


(b) G_{M_a}

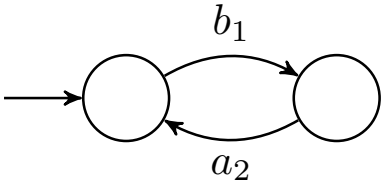


(c) G_{M_B}

Figure 3.6: Automata models of machines M_A, M_a and M_B.



(a) G_{B_1}



(b) G_{B_2}

Figure 3.7: Automata models of the buffers B_1 and B_2

by $G_{K_1} = G_{M_A}||G_{M_B}||G_{B_1}$ and $G_{K_2} = G_{M_a}||G_{M_B}||G_{B_2}$. Note that, for example, a string $a_1b_1a_1b_1$ is a nonfailure one of G_1 but it is a failure one of G_2. Thus, in this example, for each configuration, its own nonfailure specification language K_i, $(i \in \{1, 2\})$ has to be specified.

The problem of synthesizing a *robust* diagnoser is considered to satisfy the following two conditions:

C3. $(\forall i \in I)(\exists m_i \in \mathcal{N})(\forall s \in L(G_i) \setminus K_i)(\forall t \in L(G_i)/s)[|t| \geq m_i \vee st \in L_d(G_i)] \Rightarrow [\forall j \in I : M^{-1}M(st) \cap L(G_j) \subseteq L(G_j) \setminus K_j]$,

C4. $(\forall i \in I)(\forall s \in K_i)D(M(s)) \neq 1$.

That is, a robust diagnoser satisfying C3 and C4 correctly detects the occurrence of any failure in any possible model. In order to characterize the existence of such a robust diagnoser, the notion of *robust diagnosability* may be introduced.

Definition 3.6 *The set $\{G_i|i \in I\}$ of possible models is said to be robustly diagnosable with respect to a set of nonempty closed languages $\{K_i \subseteq L(G_i)|i \in I\}$ if*

$(\forall i \in I)(\forall m_i \in \mathcal{N})(\forall s \in L(G_i)KK_i)(\forall t \in L(G_i)/s)[|t| \geq m_i \vee st \in L_d(G_i)] \Rightarrow [\forall j \in I : M{-}1M(st) \cap L(G_j) \subseteq L(G_j) \setminus K_j]$.

Robust diagnosability of $\{G_i|i \in I\}$ with respect to $\{K_i \subseteq L(G_i)|i \in I\}$ requires that, for each possible model G_i there exists a nonnegative integer $m_i \in \mathcal{N}$ such that, for any failure string $s \in L(G_i) \setminus K_i$ and any extension $t \in L(G_i)/s$ with $|t| \geq m_i$ or $st \in L_d(G_i)$, any string $u \in M{-}1M(st) \cap L(G_j)$ indistinguishable from $stinG_j$ is also a failure string in $L(G_j) \setminus K_j$ for all possible models G_j. Note that, if $j \neq i$, it is possible that the set $M^{-1}M(st) \cap L(G_j)$ of indistinguishable strings is the empty set. On the other hand, diagnosability of G_i with respect to K_i only requires that strings indistinguishable from $stinG_i$ be failure ones in $L(G_i) \setminus K_i$. Thus, the following proposition holds.

Proposition 1 *If the set $\{G_i|i \in I\}$ of possible models is robustly diagnosable with respect to a set of nonempty closed languages $\{K_i \subseteq L(G_i)|i \in I\}$, then, for each $i \in I$, G_i is diagnosable with respect to K_i.*

The converse relation of Proposition 1 does not holds, i.e., G_i being diagnosable with respect to a given K_i does not guarantees $\{G_i|i \in I\}$ to be diagnosable with respect to $\{K_i \subseteq L(G_i)|i \in I\}$.

Another comparison to be made is regarding the robust diagnosability of the aggregated model G^* such that $L(G^*) = \bigcup_{i \in I} L(G_i)$.

Proposition 2 *Let G^* be an aggregated model such that $L(G^*) = \bigcup_{i\in I} L(G_i)$.If the set $\{G_i|i \in I\}$ of possible models is robustly diagnosable with respect to a set of nonempty closed languages $\{K_i \subseteq L(G_i)|i \in I\}$, then the aggregated model is diagnosable with respect to $\bigcup_{i\in I} K_i$.*

Proposition 2 shows how the diagnosability of the set $\{G_i|i \in I\}$ is stronger than of the aggregated model G^*. Hence, the converse relation of this proposition also does not hold.

Consider the following proposition showing that if nonfailure behavior and deadlocking behavior are *consistent* among all possible models, that is, if a string is a nonfailure (resp. deadlocking) string in some possible mode, then it is also a nonfailure (resp. deadlocking) string (if it is feasible) in other possible models, then robust diagnosability of G_i with respect to K_i is equivalent to diagnosability of an aggregated model G^* with respect to $\bigcup_{i\in I} K_i$.

Proposition 3 *Let G^* be an aggregated model such that $L(G^*) = \bigcup_{i\in I} L(G_i)$. Assume that, for each $i \in I$, a nonempty closed language $K \subseteq L(G)$ is specified as $K_i = k \cap L(G_i)$ by an nonempty closed language $K \subseteq \Sigma^*$, and $L_d(G_i) = L_d(G^*) \cap L(G_i)$. Then, the set $\{G_i|i \in I\}$ of possible models is robustly diagnosable with respect to a set of nonempty closed languages $\{K_i \subseteq L(G_i)|i \in I\}$ if and only if the aggregated model G^* is diagnosable with respect to $\bigcup_{i\in I} K_i$.*

Finally, consider the following theorem characterizing the existence of a robust diagnoser satisfying conditions C3 and C4.

Theorem 3.3 *There exists a robust diagnoser $D : \Delta^* \to \{0, 1\}$ satisfying conditions C3 and C4 for a set of nonempty closed languages $\{K_i \subseteq L(G_i)|i \in I\}$ if and only if the set $\{G_i|i \in I\}$ of possible models is robustly diagnosable with respect to $\{K_i \subseteq L(G_i)|i \in I\}$.*

The work proposed by TAKAI [23] is followed by the presentation of an algorithm for verifying robust diagnosability. Since it is not the main focus of this work, let us highlight that the computational complexity of the algorithm presented is equal to $O(|X_i| \times |X_{K_i}|^2 \times (\Pi_{j\neq i}|X_{K_j}|) \times |\Sigma|^{n+1})$.

3.5 Generalized robust diagnosability of discrete event systems

The paper presented by CARVALHO *et al.* [27] addressed the problem of diagnosability of discrete event systems considering not only one but two sources of

uncertainties to the system: the ones from the loss of observation of events and the ones from the modeling of the system. The work took into consideration two already addressed fault diagnosis problems presented by LIMA *et al.* [30] and TAKAI [22] and encompassed both by presenting a new definition of a generalized robust diagnosability. Moreover, they presented a necessary and sufficient condition for the generalized robust diagnosability and a polynomial time algorithm for the verification of the generalized robust diagnosability.

The work proposed by LIMA *et al.* [30] was continued until it reached the maturity presented in Section 3.2. The definition of robust diagnosability introduced in [30] was stated as follows.

Definition 3.7 *(Robust diagnosability against permanent sensor failures) Let L be the live prefix-closed language generated by automaton G and let $K \subset L$ denote the prefix-closed language generated by G_N (the subautomaton of G that models the non-failure behavior). In addition, assume that Σ_{o_i}, $i \in I_m$, $I_m = \{1, 2, \ldots, m\}$, is a diagnosis basis for L. Then, L is robustly diagnosable against permanent sensor failures with respect to projections $P_{o_i} : \Sigma^* \to \Sigma_{o_i}^*$, for $i \in I_m$, and $\Sigma_f = \{\sigma_f\}$, if and only if*

$$(\exists n \in \mathbb{N})(\forall s \in L \setminus K)(\forall st \in L \setminus K, |t| \geq n) \Rightarrow$$
$$(\forall i, j \in I_m, i \neq j)(\forall w \in K, P_{o_i}(st) \neq P_{o_j}(w)).$$

□

In accordance to Definition 3.7, a prefix-closed language L is not robustly diagnosable if there exist integers $i, j \in I_m$, $(i \neq j)$, and two traces $w, st \in L$, where w is a non-faulty trace and st can be made arbitrarily long after the fault event $\sigma_f \in \Sigma_f$, such that $P_{o_i}(st) = P_{o_j}(w)$.

Alternatively, TAKAI [22] brought light to the possibility of having a set of possible automata modeling a system and introduced another definition of robust diagnosability, similar to that presented in [30], assuming that: *(i)* the real system model belongs to a set of possible models $G_i = (X_i, \Sigma, f_i, \Gamma_i, x_{0_i}, X_{m_i})$, $i \in I_m$; *(ii)* each automaton model G_i generates a different language L_i and has a distinct non-faulty behavior, described by a nonempty closed sublanguage $K_i \subseteq L_i$; *(iii)* they all share the same observable event set Σ_o.

Taking both problems in consideration, CARVALHO *et al.* [27] have presented the so called generalized robust diagnosability definition for a class of different languages generated by a class of automata that models a specific system. The definition is as follows.

Definition 3.8 *(Generalized robust diagnosability) Let $L_i \subseteq \Sigma^\star$ be the language generated by G_i, $i \in I_m$, $I_m = \{1, 2, \ldots, m\}$, and assume that L_i is live. In addition,*

assume that each model $i \in I_m$ has a projection $P_{o_i} : \Sigma^\star \to \Sigma_{o_i}^\star$, and that L_i is diagnosable with respect to P_{o_i} and Σ_f. Let us denote G_{N_i} as the subautomaton of G_i that models the non-faulty behavior of the corresponding model, and $K_i \subset L_i$ the language generated by G_{N_i}. Then,

$$\mathbb{L} = \{L_i : i \in I_m\},$$

the class of all possible languages generated by the class of automata

$$\mathbb{G} = \{G_i : i \in I_m\},$$

is robustly diagnosable with respect to projection P_{o_i}, $i \in I_m$, and $\Sigma_f = \{\sigma_f\}$, if and only if

$$\begin{aligned}&(\forall i \in I_m)(\exists n_i \in \mathbb{N})(\forall s_i \in L_i \setminus K_i)(\forall s_i t_i \in L_i \setminus K_i, |t_i| \geq n_i)\\ &\Rightarrow (\forall j \in I_m)(\forall w_j \in K_j, P_{o_i}(s_i t_i) \neq P_{o_j}(w_j)).\end{aligned}$$

□

Before the verification algorithm was deployed, the renaming function $R_i : \Sigma \to \Sigma_{R_i}$, for $i \in I_m$, was presented. The function was proposed by MOREIRA *et al.* [6] and is used for the development of the verification algorithm.

$$R_i(\sigma) = \begin{cases} \sigma, \text{ if } \sigma \in \Sigma_{o_i} \cup \Sigma_f \\ \sigma_{R_i}, \text{ if } \sigma \in \Sigma_{uo_i} \setminus \Sigma_f \end{cases}. \tag{3.16}$$

Notice that function R_i can be extended to domain $\Sigma^\star$ in the usual way, as follows: $R_i(\varepsilon) = \varepsilon$, and $R_i(s\sigma) = R_i(s)R_i(\sigma)$, $\forall s \in \Sigma^\star$ and $\forall \sigma \in \Sigma$. As a consequence, R_i can also be extended to a language $L \subseteq \Sigma^\star$ by simply applying it to all strings in L.

Based on R_i, we can also define the inverse renaming function, as follows:

$$\begin{aligned} R_i^{-1} : \ \Sigma_{R_i} &\to \Sigma \\ \sigma_{R_i} &\mapsto \sigma, \end{aligned}$$

where $\sigma_{R_i} = R_i(\sigma)$, with the following extension to domain $\Sigma_{R_i}^\star$: $R_i^{-1}(s_{R_i}\sigma_{R_i}) = R_i^{-1}(s_{R_i})R_i^{-1}(\sigma_{R_i})$ for all $s_{R_i} \in \Sigma_{R_i}^\star$ and $\sigma_{R_i} \in \Sigma_{R_i}$, and $R_i^{-1}(\varepsilon) = \varepsilon$.
A polynomial time algorithm for the verification of the robust diagnosability was finally presented with computational complexity equal to $O(m|X_i|\prod_{j=1,j\neq i}^{m} |X_j|(|\Sigma| - |\Sigma_f|))$ where m is the number of possible models $G_i \in \mathbb{G}$.

The following theorem proves the correctness of Algorithm 3.1.

Algorithm 3.1 Verification of generalized robust diagnosability of DES

Inputs: $G_i = (X_i, \Sigma, f_i, \Gamma_i, x_{i,0})$, $i \in I_m$

Output: V_i

1: For each model $G_i = (X_i, \Sigma, f_i, \Gamma_i, x_{i,0})$, $i \in I_m$, build automaton $G_{R_i} = (X_i, \Sigma_{R_i}, f_{R_i}, \Gamma_{R_i}, x_{i,0})$, where $\Sigma_{R_i} = R_i(\Sigma)$, $\Gamma_{R_i}(x) = R_i[\Gamma_i(x)]$, and $f_{R_i}(x, R_i(\sigma)) = f_i(x, \sigma)$ for all $x \in X_i$ and $\sigma \in \Gamma_i(x)$.

2: Compute the failure automaton $\mathcal{F}_i$ as follows:

2.1: Build the faulty label automaton $A_\ell = (X_{A_\ell}, \Sigma_f, f_{A_\ell}, \Gamma_{A_\ell}, x_{0,A_\ell})$, where $X_{A_\ell} = \{N, Y\}$, $x_{0,A_\ell} = N$, and $f_{A_\ell}(N, \sigma_f) = Y$ and $f_{A_\ell}(Y, \sigma_f) = Y$.

2.2: Compute $\tilde{G}_{R_i} = G_{R_i} \| A_\ell$ and mark all states of $\tilde{G}_{R_i}$ that have the second component equal to Y.

2.3: Compute the faulty automaton $\mathcal{F}_i = CoAc(\tilde{G}_{R_i}) = (X_{\mathcal{F}_i}, \Sigma_{R_i}, f_{\mathcal{F}_i}, \Gamma_{\mathcal{F}_i}, x_{0,\mathcal{F}_i})$.

2.4: Redefine the event set of $\mathcal{F}_i$ as $\Sigma_{\mathcal{F}_i} = \Sigma_{R_i} \cup \Sigma_o$.

3: Build the non-faulty automaton H_{R_i} as:

3.1: Define $\Sigma_{Z_i} = \Sigma_{R_i} \setminus \Sigma_f$, and build automaton $Z_i = (\{N\}, \Sigma_{Z_i}, f_{Z_i}, N)$, composed of a single state with a self-loop labeled with all events in Σ_{Z_i}.

3.2: Construct $H_{R_i} = G_{R_i} \times Z_i = (X_{H_{R_i}}, \Sigma_{R_i}, f_{H_{R_i}}, \Gamma_{H_{R_i}}, x_{0,H_{R_i}})$.

3.3: Redefine the event set of each H_{R_i} as $\Sigma_{H_{R_i}} = \Sigma_{R_i} \setminus \Sigma_f$.

4: Construct the augmented automaton $\mathcal{H}_i = (X_{\mathcal{H}_i}, \Sigma_{\mathcal{H}_i}, f_{\mathcal{H}_i}, \Gamma_{\mathcal{H}_i}, x_{0,\mathcal{H}_i})$ from automaton H_{R_i} as follows:

4.1: Define $\Sigma_{\mathcal{H}_i} = \Sigma_{H_{R_i}} \cup \Sigma_o$.

4.2: Define $x_{0,\mathcal{H}_i} = x_{0,H_{R_i}}$.

4.3: Add a new state D_i to the state space of H_{R_i}. Thus, $X_{\mathcal{H}_i} = X_{H_{R_i}} \cup \{D_i\}$.

4.4: For each $x_{\mathcal{H}_i} \in X_{H_{R_i}}$ define:

$$f_{\mathcal{H}_i}(x_{\mathcal{H}_i}, \sigma) = \begin{cases} f_{H_{R_i}}(x_{\mathcal{H}_i}, \sigma), \text{ if } \sigma \in \Gamma_{H_{R_i}}(x_{H_{R_i}}) \\ D_i, \text{ if } \sigma \in \Sigma_o \setminus \Gamma_{H_{R_i}}(x_{H_{R_i}}) \\ \text{undefined, otherwise} \end{cases}, \tag{3.17}$$

and for $x_{\mathcal{H}_i} = D_i$ define:

$$f_{\mathcal{H}_i}(x_{\mathcal{H}_i}, \sigma) = \begin{cases} D_i, \text{ for all } \sigma \in \Sigma_o \\ \text{undefined, otherwise} \end{cases}. \tag{3.18}$$

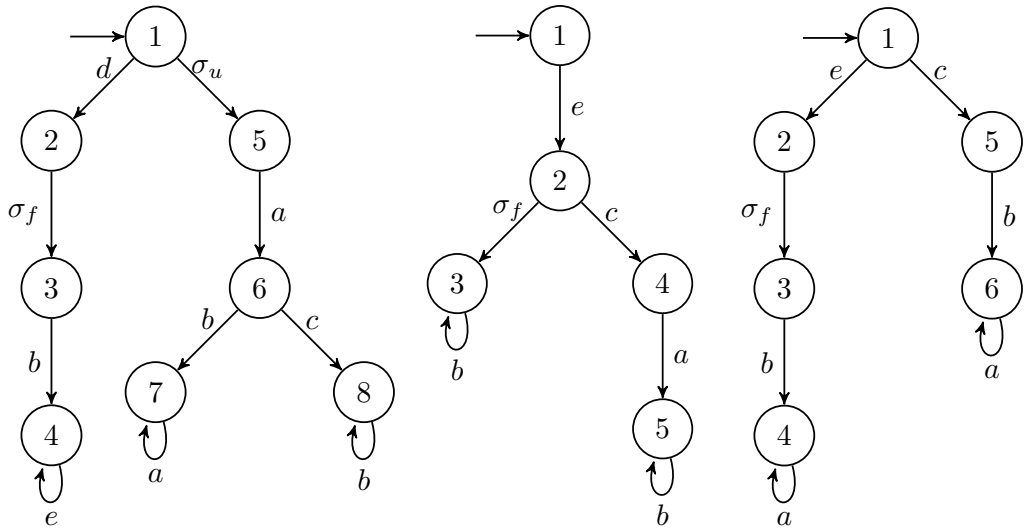
5: For $i \in I_m$, compute verifier V_i whose jth state $x_{V_{ij}} \in X_{\mathcal{F}_i} \times (\times_{q=1, q\neq i}^{m} X_{\mathcal{H}_q})$ and the jth state of $X_{\mathcal{F}_i}$ is $x_{\mathcal{F}_{ij}} \in X_i \times X_{A_\ell}$, by making a composition of $\mathcal{F}_i$, $\mathcal{H}_1$, ..., $\mathcal{H}_{i-1}$, $\mathcal{H}_{i+1}$, ..., $\mathcal{H}_m$, following the same procedure as for the parallel composition $\mathcal{F}_i \| (\|_{j=1, j\neq i}^{m} \mathcal{H}_j)$, except that if state $(x_{\mathcal{F}_i}, D_1, \ldots, D_{i-1}, D_{i+1}, \ldots, D_m)$ is reached, where $x_{\mathcal{F}_i} \in X_{\mathcal{F}_i}$, then its active event set is forced to be the empty set[2], *i.e*,

$$\Gamma_{V_i}(x_{\mathcal{F}_i}, D_1, \ldots, D_{i-1}, D_{i+1}, \ldots, D_m) = \emptyset.$$

6: Verify the existence of a cycle $cl_i = (x_{V_{ik}}, \sigma_k, x_{V_{ik+1}}, \sigma_{k+1}, \ldots, \sigma_l, x_{V_{ik}})$, where $l \geq k > 0$, in V_i satisfying the following condition:

$$\exists j \in \{k, k+1, \ldots, l\} \text{ s.t. } (\sigma_j \in \Sigma_{R_i}) \wedge (x_{\mathcal{F}_{ij}} = \{x_{i_j}, Y\}).$$

If the answer is yes, then the class $\mathbb{L}$ is not robustly diagnosable with respect to projections P_{o_i}, $i \in I_m$, and Σ_f. Otherwise, the class $\mathbb{L}$ is robustly diagnosable.



(a) G_1 (b) G_2 (c) G_3

Figure 3.8: Class of automata $\mathbb{G} = \{G_1, G_2, G_3\}$.

Theorem 3.4 *The class $\mathbb{L}$ is not robustly diagnosable with respect to P_{o_i}, $i \in I_m$, and Σ_f if and only if there exists a cycle $cl_i = (x_{V_{ik}}, \sigma_k, x_{V_{ik+1}}, \sigma_{k+1}, \ldots, \sigma_l, x_{V_{ik}})$, where $l \geq k > 0$, in at least one verifier V_i, $i \in I_m$, satisfying the following condition:*

$$\exists j \in \{k, k+1, \ldots, l\} \text{ s.t. } (\sigma_j \in \Sigma_{R_i}) \wedge (x_{\mathcal{F}_{ij}} = \{x_{i_j}, Y\}). \tag{3.19}$$

The generalized robust diagnosability definition and the verification algorithm proposed are better comprehended through the observance of an illustrating example.

Example 3 *Let $\mathbb{G} = \{G_1, G_2, G_3\}$ be the class of automata shown in Figure 3.8, and assume that $\Sigma = \{a, b, c, d, e, \sigma_u, \sigma_f\}$ is the set of all events used in the modeling of the system. In addition, let $\Sigma_{o_1} = \{a, b, c\}$, $\Sigma_{o_2} = \{a, b, c, e\}$ and $\Sigma_{o_3} = \{a, b, d, e\}$ be, respectively, the observable event sets of G_1, G_2 and G_3. The objective here is to verify if the class $\mathbb{L}$ of languages generated by the automata in $\mathbb{G}$ is robustly diagnosable with respect to P_{o_i}, $i \in I_m$, and $\Sigma_f = \{\sigma_f\}$.*

Initially, note that $\Sigma_o = \bigcup_{i=1}^{3} \Sigma_{o_i} = \{a, b, c, d, e\}$. Now, according to Algorithm 3.1, the first step is to obtain automaton G_{R_i}, $i = 1, 2, 3$, by renaming the events in $\Sigma_{uo_i} \setminus \Sigma_f$. Therefore events d, e and σ_u should be renamed, respectively, as d_{R_1}, e_{R_1} and $\sigma_{u_{R_1}}$, in G_1, and event c as c_{R_3} in G_3. Notice that no event needs to be renamed in G_2. The state transition diagram of automata G_{R_i}, $i = 1, 2, 3$, are not shown since they are identical to those of G_i, $i = 1, 2, 3$, except for the above renaming.

The next step of Algorithm 3.1 is to compute the faulty automata $\mathcal{F}_i$, $i = 1, 2, 3$. Following Steps 2.1–2.4, automata $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{F}_3$, shown in Figure 3.9, are obtained. Notice that, although only events b, d_{R_1}, e_{R_1} and σ_f appear in the state transition

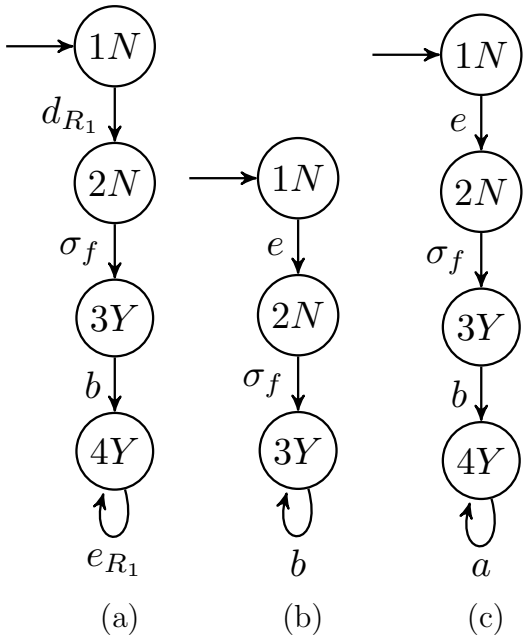


Figure 3.9: Faulty automata $\mathcal{F}_1$ (a), $\mathcal{F}_2$ (b), and $\mathcal{F}_3$ (c).

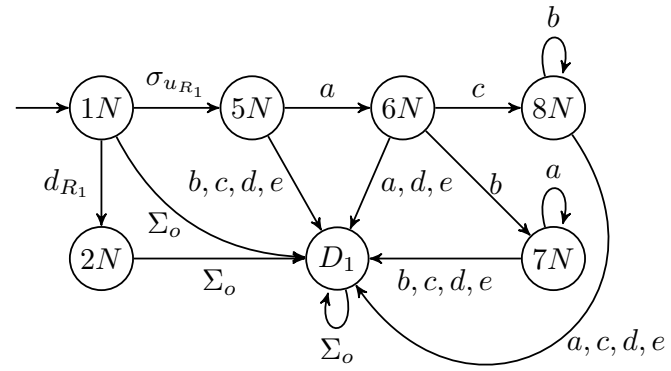
diagram of $\mathcal{F}_1$, its event set is $\Sigma_{\mathcal{F}_1} = \Sigma_{R_1} \cup \Sigma_o = \{a, b, c, d, e, d_{R_1}, e_{R_1}, \sigma_{u_{R_1}}, \sigma_f\}$. Same analysis can be carried out for $\mathcal{F}_2$ and $\mathcal{F}_3$ leading to $\Sigma_{\mathcal{F}_2} = \{a, b, c, d, e, d_{R_2}, \sigma_{u_{R_2}}, \sigma_f\}$ and $\Sigma_{\mathcal{F}_3} = \{a, b, c, d, e, c_{R_3}, \sigma_{u_{R_3}}, \sigma_f\}$.

Having computed the automata that model the faulty behavior of G_{R_1}, G_{R_2} and G_{R_3}, the next step is to obtain the non-faulty automata H_{R_1}, H_{R_2} and H_{R_3} that accounts for the non-faulty behavior of G_{R_1}, G_{R_2} and G_{R_3}, and, in the sequel to obtain the augmented automata $\mathcal{H}_1$, $\mathcal{H}_2$ and $\mathcal{H}_3$. Following Step 4 of Algorithm 3.1, the state transition diagrams depicted in Figure 3.10 are computed. Regarding the event sets of the augmented automata $\mathcal{H}_1$, $\mathcal{H}_2$ and $\mathcal{H}_3$, it is worth remarking that $\Sigma_{\mathcal{H}_1} = \Sigma_{\mathcal{F}_1} \setminus \{\sigma_f\}$, $\Sigma_{\mathcal{H}_2} = \Sigma_{\mathcal{F}_2} \setminus \{\sigma_f\}$ and $\Sigma_{\mathcal{H}_3} = \Sigma_{\mathcal{F}_3} \setminus \{\sigma_f\}$.

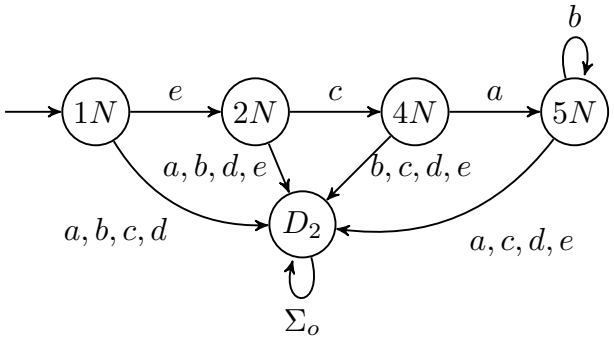
Proceeding in accordance with Step 5 of Algorithm 3.1, the verifier automata V_1, V_2 and V_3 must be computed. Figures 3.11(a) and 3.11(b) show the state transition diagram of verifiers V_1 and V_3, respectively; the state transition diagram of verifier V_2 has been omitted since it leads to a conclusion similar to that drawn from V_3.

Finally, to verify if $\mathbb{L}$ is robustly diagnosable with respect to P_{o_i}, $i \in I_m$, and $\Sigma_f = \{\sigma_f\}$, we follow Step 6 of Algorithm 3.1. In doing so, we first consider verifier V_1 shown in Figure 3.11(a). Notice that cycle $(4YD_23N, e_{R_1}, 4YD_23N)$ is formed with an event in Σ_{R_1}. Thus, although verifiers V_2 (not shown in the paper) and V_3 (depicted in Figure 3.11(b)) do not have any cycle with events in Σ_{R_2} and Σ_{R_3}, respectively, whose first components of their states have fault labels, we may conclude that $\mathbb{L}$ is not robustly diagnosable with respect to P_{o_i}, $i \in I_m$, and $\Sigma_f = \{\sigma_f\}$.

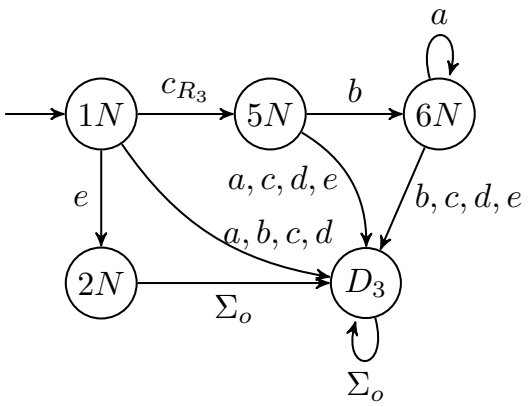
A close examination of verifier V_1 reveals the traces responsible for the non-robust



(a)



(b)



(c)

Figure 3.10: Augmented non-faulty automata $\mathcal{H}_1$ (a), $\mathcal{H}_2$ (b), and $\mathcal{H}_3$ (c).

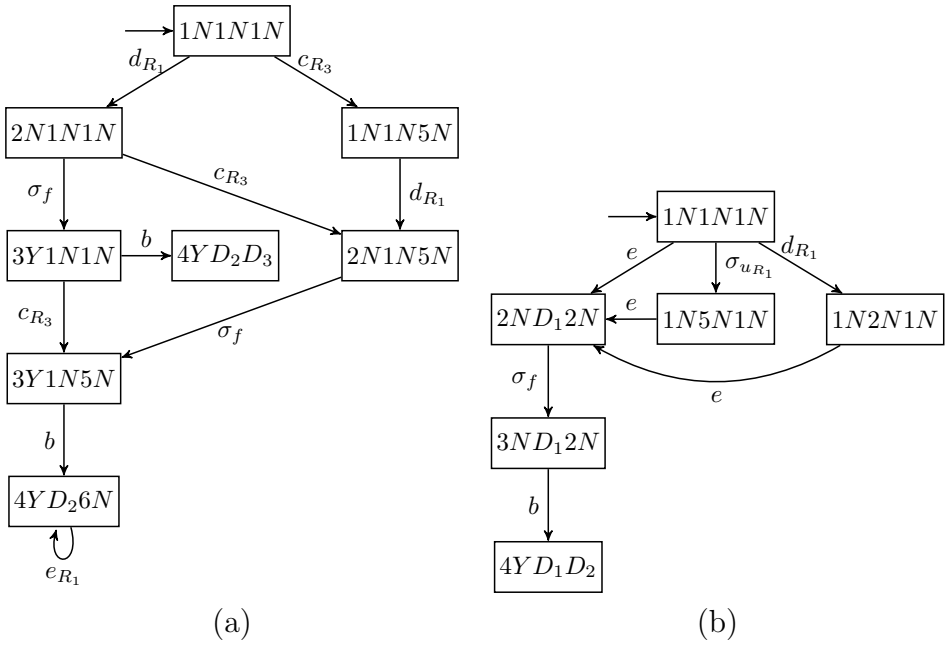


Figure 3.11: Verifier automata V_1 (a) and V_3 (b).

diagnosability. Notice that trace $s_{V_1} = d_{R_1}\sigma_f c_{R_3} b e^n_{R_1}$, where n can be arbitrarily large, takes V_1 from its initial state to state $4YD_23N$ and cycles over this state. Since $s_{\mathcal{F}_1} = P_{\mathcal{F}_1}(s_{V_1}) = d_{R_1}\sigma_f b e^n_{R_1}$ and $s_{\mathcal{H}_3} = P_{\mathcal{H}_3}(s_{V_1}) = c_{R_3}b$, then after inverse rename we obtain $s_1 = d\sigma_f b e^n \in L_1$ and $s_3 = cb \in L_3$. Furthermore, $P_{o_1}(s_1) = P_{o_3}(s_3) = b$, which implies that when trace s_1 occurs, it is not possible to conclude that the system is either in state 4 of G_1, which is after the occurrence of fault event σ_f, or in state 6 of G_3, which is in a normal path.

3.6 Final comments

This chapter presented five different works exploring problems on diagnosability of discrete event systems. The comprehension of the papers here introduced is key for the understanding of the developed transformation formalisms to be presented in Chapter 4. We have started presenting the work proposed by KANAGAWA and TAKAI [26] concerning diagnosability of DESs subject to permanent sensor failures. Then, we presented the concepts of robust diagnosis of DESs against permanent and intermittent loss of observations proposed in [20] and [21], respectively. We have also explored the verification of robust diagnosability of partially observed DESs presented by TAKAI [23] and finally presented the generalized robust diagnosability paper proposed by CARVALHO *et al.* [27].

Chapter 4

Verification of generalized robust diagnosability on discrete event systems

In this chapter we will update the generalized robust diagnosability definition in Section 4.1 and present a new verification algorithm in Section 4.2 that has improved computational complexity compared to the one present in Algorithm 3.1, as shown in Section 4.3. In Section 4.4 we will access each problem presented in Chapter 3 and develop transformation mechanisms from the addressed problems of Chapter 3 to the generalized robust definition input format that allow us to demonstrate how the generalized robust diagnosability definition encompasses all addressed problems so far.

4.1 Updates on the generalized robust diagnosability definition

Considering that new problems on fault diagnosis of discrete event systems were addressed after the publication of the generalized robust diagnosability definition in 2011 such as the works proposed by KANAGAWA and TAKAI [26] and CARVALHO *et al.* [21] we had to update the already presented Definition 3.8 so that it could easily encompass all problems presented so far. The changes are related to the following assumptions.

A1. L_i is live

A2. L_i is diagnosable with respect to projection $P_{o_i} : \Sigma^\star \to \Sigma_{o_i}^\star$ and Σ_f, for $i \in I_m$, $I_m = \{1, 2, \ldots, m\}$.

The first change made was to remove assumption **A1**, which states that an input language L_i is live, whereas the second change was to include in the testing range of the definition the assumption **A2**, that considers L_i to be diagnosable with respect to projection $P_{o_i} : \Sigma^\star \to \Sigma^\star_{o_i}$ and Σ_f, for $i \in I_m$, $I_m = \{1, 2, \ldots, m\}$.

Removing assumption **A1** was already done in [13], [14] and [6]. Let us consider a deterministic automaton G that generates the language L that is not live and its projection $P_o : \Sigma^* \to \Sigma^*_o$. Since L is not live, then the system necessarily has deadlock states. In order to turn L into a live language we may add a new unobservable event σ_L to the event set Σ of automaton G and then add a self-loop labeled by σ_L to each one of the deadlock states of the system. Proceeding this way we have a new language L'_i that is live and that its traces have the same projection $P_o : \Sigma^* \to \Sigma^*_o$ as the ones from L_i, which preserves the partially observation of the system. Moreover, the new event prevents that the addition of self-loops labeled by previously defined events make the automaton non-deterministic.

Assumption **A2** considers that a given L_i is already diagnosable with respect to its own projection P_{o_i}. In practical standards that means saying that it was necessary that a test was performed to guarantee the diagnosability of L_i. Considering that we are dealing with a class of automata $\mathbb{G}$ that generates a class of languages $\mathbb{L}$, then we infer without loss of generality that the class must be tested for the codiagnosability of the decentralized diagnosers prior to the generalized robust diagnosability verification.

If we analyze Definition 3.8 we will see that, for the particular case in which $i = 1$, the definition is testing for the diagnosability of a single automaton just like proposed by SAMPATH *et al.* [4]. Furthermore, if we make $i = j$ we will be testing for the codiagnosability of local diagnosers as presented in [6, 13, 31]. In that sense, we have decided to include the tests for diagnosability and codiagnosability - particular cases of the generalized robust diagnosability definition - into the updated definition. The decision will allow us to easily demonstrate how the generalized robust diagnosability definition encompasses all addressed problems presented in Chapter 3.

Once we have explained the reason why we have made the changes on the previously presented definition, we may now present the updated definition of the generalized robust diagnosability.

Definition 4.1 *(Updated generalized robust diagnosability) Let $L_i \subseteq \Sigma^\star$ be the language generated by G_i, $i \in I_m$, $I_m = 1, 2, \ldots, m$. In addition, assume that each model $i \in I_m$ has a projection $P_{o_i} : \Sigma^\star \to \Sigma^\star_{o_i}$. Let us denote G_{N_i} as the subautomaton of G_i that models the non-faulty behavior of the corresponding model, and $K_i \subset L_i$ the language generated by G_{N_i}. Then,*

$$\mathbb{L} = \{L_i : i \in I_m\},$$

the class of all possible languages generated by the class of automata

$$\mathbb{G} = \{G_i : i \in I_m\},$$

is robustly diagnosable with respect to projection P_{o_i}, $i \in I_m$, *and* $\Sigma_f = \{\sigma_f\}$, *if and only if*

$$(\forall i \in I_m)(\exists n_i \in \mathbb{N})(\forall s_i \in L_i \setminus K_i)(\forall s_i t_i \in L_i \setminus K_i, |t_i| \geq n_i)$$
$$\Rightarrow (\forall j \in I_m)(\nexists w_j \in K_j)[\Sigma_f \notin w_j \wedge P_{o_i}(s_i t_i) = P_{o_j}(w_j)].$$

□

Once the definition has been updated we may now improve the computational complexity of the verifier algorithm as proposed in Section 3.2.

4.2 The updated verification algorithm for generalized robust diagnosability of discrete event systems

In this section we will present the new verification algorithm for the generalized robust diagnosability definition as presented in Definition 4.1. We will then provide a necessary and sufficient theorem that proves the correctness of the algorithm and conclude with an illustrating example.

Let us assume that the automata $G_i \in \mathbb{G}$ that are inputs for the algorithm have each one a corresponding set of observable events $\Sigma_{o_i} \in \Sigma_o$. Hence, $\Sigma_o = \{\Sigma_{o_1}, \Sigma_{o_2}, \ldots, \Sigma_{o_m}\}$. Additionally, let us assume, without loss of generality, that the failure events set $\Sigma_f = \{\sigma_f\}$ for all inputs to the algorithm are presented. We also refer to the renaming function presented in Equation 3.16.

Consider the algorithm proposed as follows where we intend to include the diagnosability, codiagnosability and robust diagnosability tests considering the given inputs and perform the offline verification of the language diagnosability property.

Before we present a theorem proving the correctness of Algorithm 4.1 we would like to explain the idea behind the steps and how it affects the computational complexity of the algorithm.

Generally, the idea behind this verification algorithm is to separate the normal and failure behavior models of a given system and, after a few adjustments, perform pairwise parallel compositions between them in order to identify faulty traces that may have the same projection as a normal trace, which would lead to the violation of the codiagnosability and (robust) diagnosability definition.

Algorithm 4.1 Updated verification of generalized robust diagnosability of DES

Inputs: A class of automata $\mathbb{G} = \{G_1, G_2, \ldots, G_m\}$, $i \in I_m$, where $G_i = (X_i, \Sigma_i, f_i, \Gamma_i, x_{0,i}, X_{m_i})$ and the event set Σ_i is partitioned as $\Sigma_i = \Sigma_{o_i} \dot{\cup} \Sigma_{uo_i}$.

Outputs: A class of automata $\mathbb{G}_{V_{ij}} = \{G_{V_{11}}, G_{V_{12}}, \ldots, G_{V_{mm}}\}$, $i, j \in I_m$

1: Build G_{F_i}, $i \in I_m$, where each G_{F_i} models the failure behavior of G_i as follows:

1.1: Construct the faulty label automaton $A_f = (X_f, \Sigma_f, f_f, \Gamma_f, x_{0,f}, X_{m,f})$ where $X_f = \{N, Y\}$, $x_{0,f} = N$, $X_{m,f} = \emptyset$, $f_f(N, \sigma_f) = Y$ and $f_f(Y, \sigma_f) = Y$ for all $\sigma_f \in \Sigma_f$.

1.2: Compute $\tilde{G}_{F_i} = G_i \| A_f$, $i \in I_m$, and mark all states of $\tilde{G}_{F_i}$ whose second component is Y.

1.3: Obtain the failure automaton $G_{F_i} = CoAc(\tilde{G}_{F_i}) = (X_{F_i}, \Sigma_i, f_{F_i}, \Gamma_{F_i}, x_{0,F_i}, X_{m,F_i})$, $i \in I_m$, where $CoAc(\tilde{G}_{F_i})$ denotes the coaccessible part of automaton $\tilde{G}_{F_i}$, i.e.,the automaton whose states are all coaccessible.

2: Build the set of normal behavior automata G_{N_i}, $i \in I_m$, as follows:

2.1: Define $\Sigma_{A_{N_i}} = \Sigma_i \setminus \Sigma_f$, and construct automaton $A_{N_i} = (\{N\}, \Sigma_{A_{N_i}}, f_{A_{N_i}}, N, \emptyset)$ composed of a single state with a self-loop labeled with all events in $\Sigma_{A_{N_i}}$.

2.2: Construct the nonfailure automaton $G_{N_i} = G_i \times A_{N_i} = (X_{N_i}, \Sigma_{A_{N_i}}, f_{N_i}, \Gamma_{N_i}, x_{0,N_i}, X_{m,N_i})$.

3: Build the renamed normal behavior automata $G^R_{N_i} = (X_{N_i}, \Sigma^R_{N_i}, f^R_{N_i}, \Gamma^R_{N_i}, x_{0,N_i}, X_{m,N_i})$, $i \in I_m$, where $\Sigma^R_{N_i} = R_i(\Sigma_i)$, $\Gamma^R_{N_i}(x) = R_i[\Gamma_{N_i}(x)]$, and $f^R_{N_i}(x, R_i(\sigma)) = f_{N_i}(x, \sigma)$ for all $x \in X_{N_i}$ and $\sigma \in \Gamma_{N_i}(x)$.

4: Construct the verifier automaton $G_{V_{ij}} = G_{F_i} || G^R_{N_i}$ for each $i \in I_m$, $i = j$, whose states are of the form $x_{V_{ii}} = (x_{F_i}, x_{N_i})$, where x_{F_i} and x_{N_i} are states of G_{F_i} and $G^R_{N_i}$ respectively and $x_{F_i} = (x, x_f)$ where $x \in X$ and $x_f \in \{N, Y\}$ and $x_{N_i} = (x, N)$.

5: **IF,** for any $G_{V_{ij}}$, exists a cycle $cl := (x^k_{V_{ii}}, \sigma_k, x^{k+1}_{V_{ii}}, \sigma_{k+1}, \ldots, \sigma_1, x^l_{V_{ii}})$, where $l \geq k > 0$ satisfying the following conditions:

$$\exists q \in \{k, k+1, \ldots, l\} \text{ s.t. } (x^q_l = Y) \wedge (\sigma_q \in \Sigma_i).$$

then the class of models $\mathbb{G}$ and the generated class of languages $\mathbb{L}$ are not robustly diagnosable with respect to P_{o_i} and Σ_f and the algorithm must stop; **ELSE,** continue for *Step* 6.

6: Build the renamed failure behavior automata $G^R_{F_i} = (X_{F_i}, \Sigma^R_{F_i}, f^R_{F_i}, \Gamma^R_{F_i}, x_{0,F_i})$, $i \in I_m$, where $\Sigma^R_{F_i} = R_i(\Sigma_i)$, $\Gamma^R_{F_i}(x) = R_i[\Gamma_{F_i}(x)]$, and $f^R_{F_i}(x, F_i(\sigma)) = f_{F_i}(x, \sigma)$ for all $x \in X_{F_i}$ and $\sigma \in \Gamma_{F_i}(x)$.

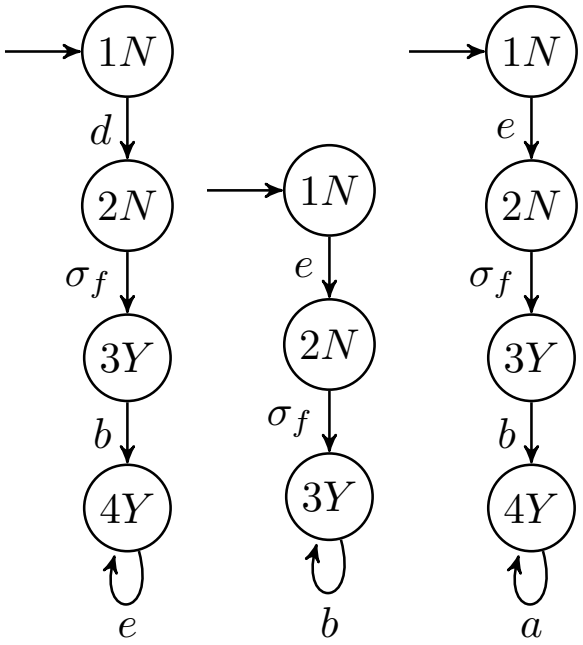
7: Define the event set $\Sigma_{R_i} = R_i(\Sigma_i), i \in I_m$ and redefine the event sets of the renamed normal and failure bahavior automata $G^R_{N_i}$ and $G^R_{F_i}$ as $\Sigma'_{R_i} = \Sigma_{R_i} \cup \Sigma_o$.

8: For each pair (i, j), $i, j \in I_m$, and $j \neq i$, construct the verifier automaton $G_{V_{ij}} = G^R_{F_i} || G^R_{N_j}$ whose states are of the form $x_{V_{ij}} = (x^R_{F_i}, x^R_{N_j})$, where x_{F_i} and x_{N_j} are states of $G^R_{F_i}$ and $G^R_{N_j}$ respectively and $x_{F_i} = (x, x_f)$ where $x \in X$ and $x_f \in \{N, Y\}$.

9: Test for the existence of a cyclic path $cl = (x^k_{V_{ij}}, \sigma_k, x^{k+1}_{V_{ij}}, \sigma_{k+1}, \ldots, \sigma_1, x^1_{V_{ij}})$, where $l \geq k > 0$ in at least one verifier $G_{V_{ij}}$, for $i \in I_m$, satisfying the following conditions:

$$\exists q \in \{k, k+1, \ldots, l\} \text{ s.t. } (x^q_f = Y) \wedge (\sigma_q \in \Sigma_{R_i}).$$

If such a cl exists, then $\mathbb{L}$ is not robustly diagnosable with respect to projections P_{o_i} and P_{o_j} or Σ_f.



(a) G_{F_1} (b) G_{F_2} (c) G_{F_3}

Figure 4.1: Automata from Step 1 of the updated verification algorithm.

Let us start from the inputs G_i in $\mathbb{G}$ and through Step 1 build the set of automata G_{F_i} that models the failure behavior of the corresponding system G_i, $i \in I_m$ for all scenarios in this algorithm. The construction of this failure behavior model takes three steps and depends on a faulty label automaton A_f composed by two labeling states and transitions labeled solely by the fault event σ_f. When A_f is parallel composed with each automaton G_i resulting on automata $\tilde{G}_{F_i}$ it marks the paths in G_i that belong to the failure behavior of the system. States are labeled with Y or N depending on their behavior status. G_{F_i} are finally obtained through the coaccessible part of $\tilde{G}_{F_i}$, i.e., the only paths that leads to marked states, the states labeled with Y after the parallel composition with A_f, are retained in these automata.

Let us consider the systems from Example 3.8 so that we can understand each step and compare it with the previously presented Algorithm 3.1. As stated in Section 3.5, a class of automata $\mathbb{G} = \{G_1, G_2, G_3\}$ is given in Figure 3.8 where $\Sigma = \{a, b, c, d, e, \sigma_u, \sigma_f\}$ is the set of all events used in the modeling of the system. In addition, $\Sigma_{o_1} = \{a, b, c\}$, $\Sigma_{o_2} = \{a, b, c, e\}$ and $\Sigma_{o_3} = \{a, b, d, e\}$. Plugging this example to the updated verification algorithm, *Step* 1 results in Figure 4.1. If we compare it to Figure 3.9, we clearly see that the difference consists on the renaming of the unobservable events of the faulty behavior automata G_{F_i} that occurs at the beginning of Algorithm 3.1 and here it will only happen in *Step* 6.

Next step of Algorithm 4.1 is directed to the construction of the normal behavior automata G_{N_i}, which is obtained through the product composition between each automaton G_i and a normal behavior labeling automaton A_{N_i} composed by only

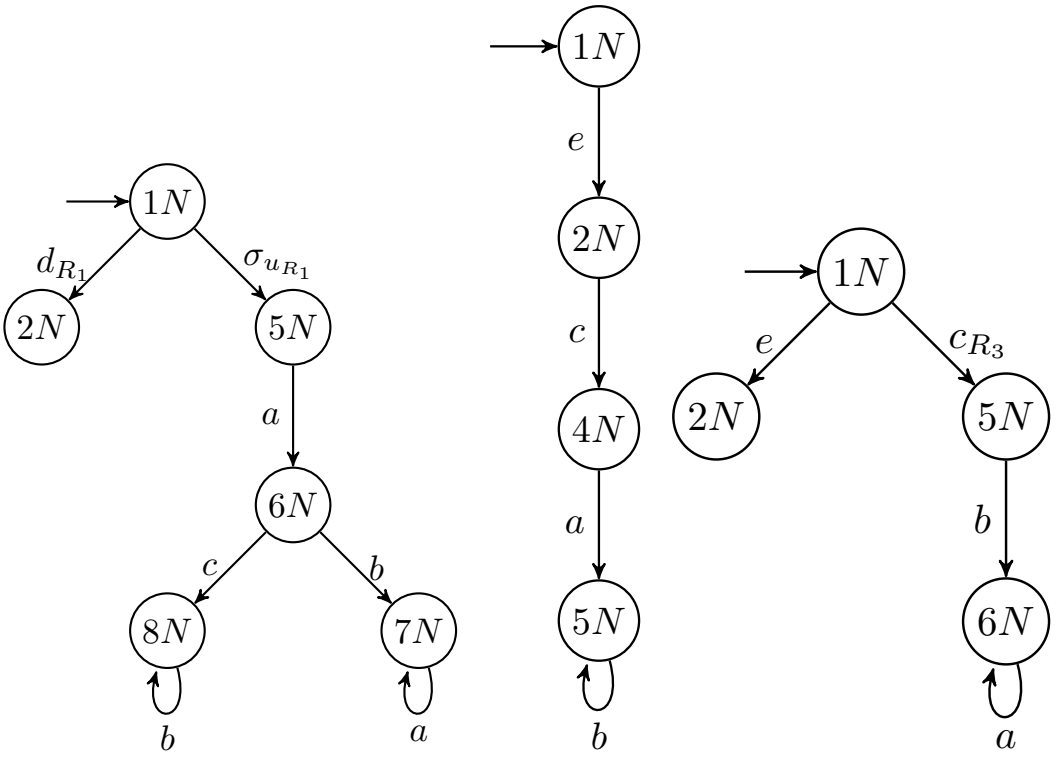


Figure 4.2: Automata from Step 2 and 3 of the updated verification algorithm.

one state with a selfloop labeled by all the events in Σ excluding the fault event. When a product composition is performed in this condition, there are no events in G_i that are private to it, except for the fault event σ_f and therefore the automaton G_{N_i} is guaranteed to model solely the normal behavior of the system.

Before the first verifiers are constructed, it becomes necessary that the events from the normal behavior automata G_{N_i} are renamed according to the function presented in Equation 3.16. By doing so, the unobservable events of G_{N_i} become private to the normal behavior model of the system. By the time the verifier is constructed, not only the parallel composition will result differently but most importantly the labeled events will denote whether or not such event occurred only in the normal behavior of the system or in both cases. The results from *Step* 2 and *Step* 3 are shown in Figure 4.2. Note how we do not need to construct the augmented automaton $\mathcal{H}_\rangle$ from Algorithm 3.1 in this updated version.

The first verifiers $G_{V_{ij}}$ for the cases where $i = j$ are constructed. Let us take a moment to understand what this means. By performing verifiers where $i = j$ the test performed is a parallel compositing between the failure and renamed normal behaviors of a same system G_i and the codiagnosability condition will be with respect to their natural projection P_{o_i} and the fault event set $\Sigma_f = \{\sigma_f\}$. This step was included in this new version so that the verifier would become more flexible to

encompass different approaches on fault diagnosis of discrete event systems. It is not a structural change since it was assumed that this test had already been performed in [27]. However, in case one of these initial tests show that a given language L_i is not diagnosable with respect to its own natural projection and the fault event set, then the entire work must be stopped since it is a necessary and sufficient condition for a robust diagnosability property that the diagnosability and codiagnosability properties are applicable to the languages tested. Verifiers V_{11}, V_{22} and V_{33} are shown in Figure 4.3. Please note how it was not a step taken in Algorithm 3.1 since it was assumed from the beginning that each automaton $G_i \in \mathbb{G}$ was already diagnosable with respect to P_{o_i} and Σ_f.

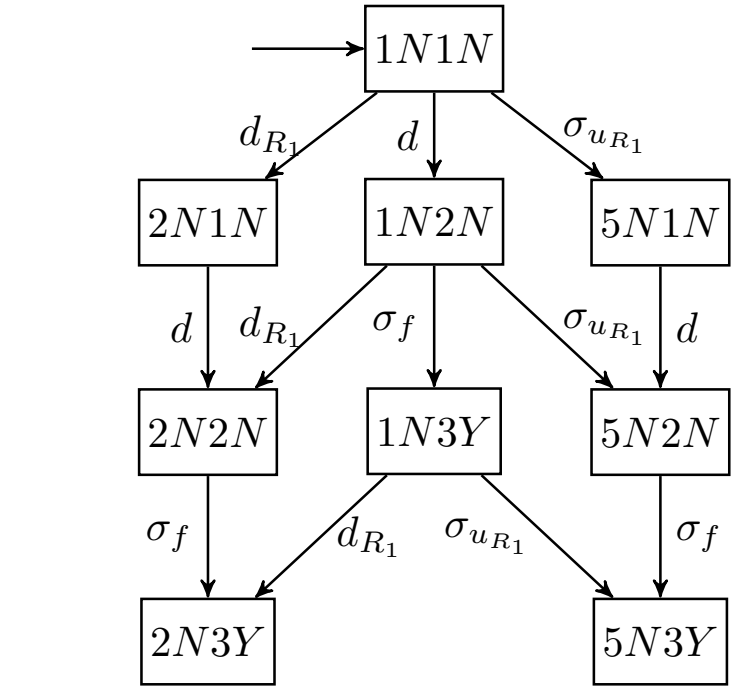
Remark 3 *The pairwise parallel composition for verifiers generation here presented in order to enhance the computational complexity of the verification algorithm for the generalized robust diagnosability definition was already proposed in [20].*

Remark 4 *Note that if we make $i = 1$ in Definition 3.8 we reduce the problem to the diagnosability of a discrete event system with respect to its projection P_o and Σ_f just like proposed by SAMPATH* et al. *[4]. Moreover, if we consider $i = j$ then we will be facing the codiagnosability verification problem addressed in [6, 13, 31]. Hence, we have included both testing possibilities in the new verification algorithm.*

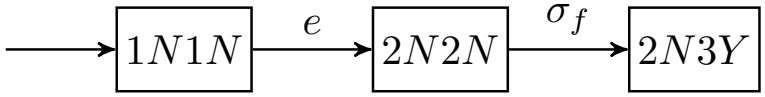
Assuming that all the languages were said to be diagnosable according to its step, we may follow to Step 6 that, just like was done in Step 3, renames the faulty behavior automata G_{F_i} through the renaming function given in Equation 3.16. The renaming relevance here is the same as we just now presented, with the slight difference that here the faulty behavior of a system will be parallel composed to the normal behavior of another system. Hence, labeling private events becomes key for a robust diagnosability definition otherwise it is impossible to detect the proper occurrence of events and the detection of violating cycles.

Next step, maybe the most important of the entire contribution here deployed, is to redefine the event sets of the renamed automata $G^R_{N_i}$ and $G^R_{F_i}$ to $\Sigma'_{R_i} = \Sigma_{R_i} \cup \Sigma_o$, where Σ_o, as stated in the beginning of this chapter, accounts for $\Sigma_o = \{\Sigma_{o_1}, \Sigma_{o_2}, \ldots, \Sigma_{o_m}\}$. It is important to rephrase that we are dealing here with a language-based diagnosability property. Hence, the entire diagnostic is based on the occurrence of traces, which are reduced to the events that conform the referred traces. Therefore, in case the event sets are not carefully adjusted, the parallel compositions that construct verifiers will be entirely inconclusive and there is no verification at all; that is why this is such an important step.

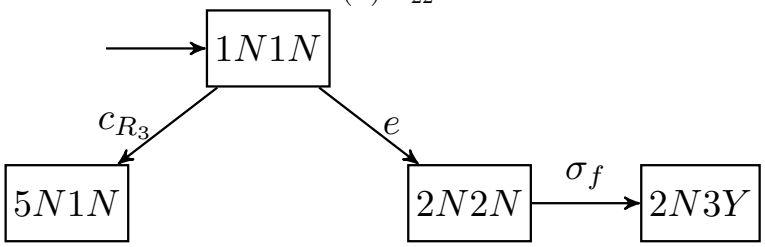
Finally, we perform the verifiers $G_{V_{ij}}$ for $i \neq j$ and test for the robust diagnosability property of the languages generated by G_i with respect to projections P_{o_i} and Σ_f. In order to illustrate *Step* 8, consider Figure 4.4.



(a) V_{11}

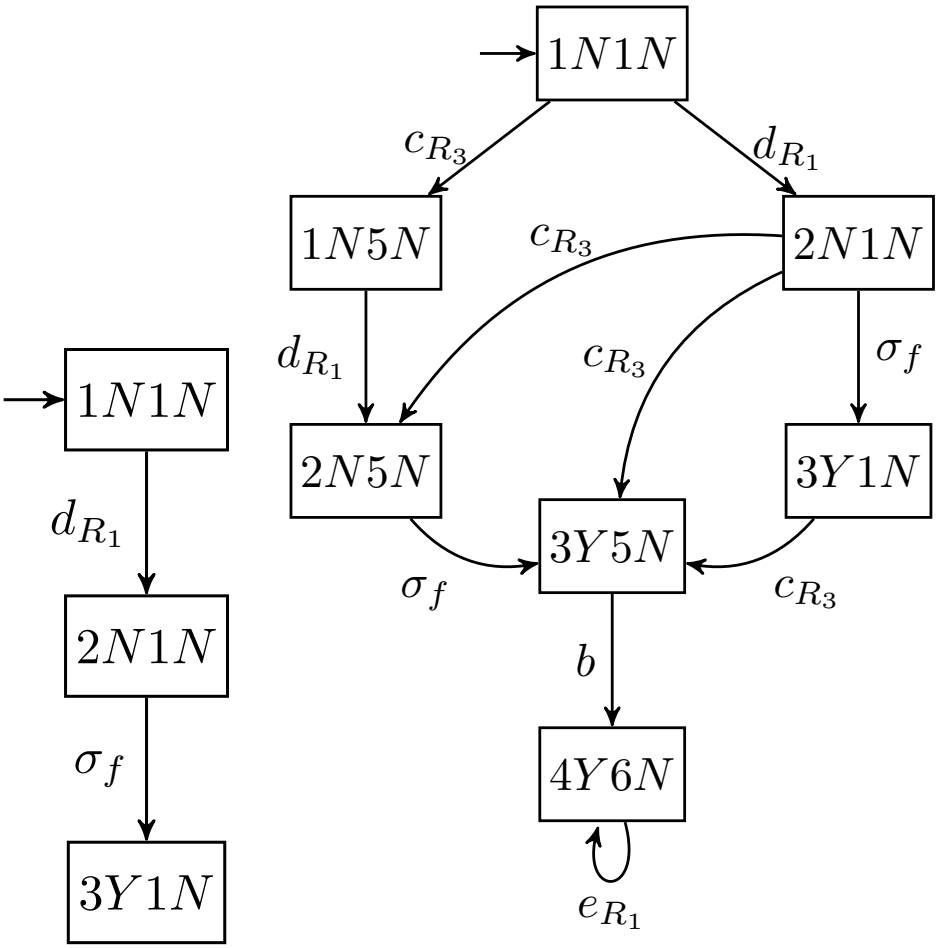


(b) V_{22}



(c) V_{33}

Figure 4.3: Verifiers from Step 4 of the updated verification algorithm.



(a) V_{12} (b) V_{13}

Figure 4.4: Verifiers from Step 8 of the updated verification algorithm.

Although it can not be seen through the figures presented, the most important change performed in this verifier comparing to the one presented in Algorithm 3.1 was to compute the verifiers in pairs instead of performing a parallel with all the possible models of the system. This change was key for the improvement of the computation complexity to be further detailed. As a consequence, the augmented automaton $\mathcal{H}$ is not necessary anymore here which simplifies a few steps and leaves the algorithm in a clearer and more objective form.

A theorem that proves the correctness of Algorithm 4.1 is presented as follows.

Theorem 4.1 *The class $\mathbb{L}$ is not robustly diagnosable with respect to P_{o_i}, $i \in I_m$, and Σ_f if and only if there exists a cycle $cl_i = (x_{V_{ik}}, \sigma_k, x_{G_{V_{ij}k+1}}, \sigma_{k+1}, \ldots, \sigma_l, x_{V_{ik}})$, where $l \geq k > 0$, in at least one verifier $G_{V_{ij}}$, $i, j \in I_m$ satisfying the following condition.*

$$\begin{aligned} &\exists p \in \{k, k+2, \ldots, l\} : (x_{V_{ij}} = ((x^p, Y), x_{N_j}) \ \wedge \\ &[(\sigma_p \in \Sigma_i), \ \textit{for} \ (i = j) \ \vee \ (\sigma_p \in \Sigma_{R_i}), \ \textit{for} \ (i \neq j)] \end{aligned} \tag{4.1}$$

where for $i = j, (x^p, Y)$ is a state from G_{F_i}, for $i \neq j, (x^p, Y)$ is a state from $G^R_{F_i}$ and x_{N_j} is a state from $G^R_{N_j}$.

Proof. For the case $i = j$, the proof is according to MOREIRA *et al.* [6]. The proof for the case $i \neq j$ is as follows.
($\Longleftarrow$) Let us assume that there exists a cyclic path $cl = (x^k_{V_{ij}}, \sigma_k, x^{k+1}_{V_{ij}}, \sigma_{k+1}, \ldots, \sigma_l, x^l_{V_{ij}})$, where $l \geq k > 0$, in verifier $G_{V_{ij}}$ satisfying condition (4.1). Since $x^p_{F_i} = (x^p, Y)$ for some $p \in \{k, k+1, \ldots, l\}$, then, from the construction of $G_{V_{ij}}$ it can be seen that $x^p_{F_i} = (x^p, Y)$ for all $p \in \{k, k+1, \ldots, l\}$. This implies that there exists a trace $s'_i t'_i \in L(G_{V_{ij}})$, such that s'_i contains the fault event, and $t'_i = (\sigma_k \sigma_{k+1} \ldots \sigma_l)^p$, $p \in \mathbb{N}$, where $|t'_i| > n$, $\forall n \in \mathbb{N}$.

Define now the following projection operations:

$$\begin{aligned} P_{F_i} &: \Sigma^R_{F_i} \cup \Sigma^R_{N_j} \to \Sigma^R_{F_i}, \\ P_{N_j} &: \Sigma^R_{F_i} \cup \Sigma^R_{N_j} \to \Sigma^R_{N_j}, \\ P &: \Sigma^R_{F_i} \cup \Sigma^R_{N_j} \to \Sigma_{o_i} \cap \Sigma_{o_j}, \\ P_i &: \Sigma_{R_i} \to \Sigma_{o_i}, \\ P_j &: \Sigma_{R_j} \to \Sigma_{o_j}. \end{aligned}$$

Notice that P_i and P_j become, respectively, equivalent to P_{o_i} and P_{o_j} if the renaming is removed.

Since $G_{V_{ij}} = G^R_{F_i} \| G^R_{N_j}$, then $L(G_{V_{ij}}) = P^{-1}_{F_i}[L(G^R_{F_i})] \cap P^{-1}_{N_j}[L(G^R_{N_j})]$, which implies that $s'_i t'_i \in P^{-1}_{F_i}[L(G^R_{F_i})]$. Let $\tilde{s}_i \tilde{t}_i = P_{F_i}(s'_i t'_i)$, where $\tilde{s}_i = P_{F_i}(s'_i)$ and $\tilde{t}_i = P_{F_i}(t')$.

Thus, since $P_{F_i}\left[P_{F_i}^{-1}(L(G_{F_i}^R))\right] = L(G_{F_i}^R)$, then $\tilde{s_i}\tilde{t_i} \in L(G_{F_i}^R)$. In addition, since $t_i' = (\sigma_k\sigma_{k+1}\ldots\sigma_l)^p$, $p \in \mathbb{N}$, where $|t_i'| > n$, $\forall n \in \mathbb{N}$, and, by assumption, there exists an event $\sigma_q \in \Sigma_{R_i}$ for $q \in \{k, k+1, \ldots, l\}$ and $\Sigma_{R_i} \subset \Sigma_{F_i}^R$, then the event sequence $\tilde{t_i} = P_{F_i}(t_i')$ also has arbitrarily long length, which implies that $\tilde{s_i}\tilde{t_i} \in L(G_{F_i}^R)$ also has arbitrarily long length after the occurrence of the fault event σ_f. Notice that $G_{F_i}^R$ is obtained from G_{F_i} after renaming the event set Σ as Σ_{R_i} and that G_{F_i} is the faulty behavior of G_i according to *Step* 1. Thus, there exists a fault trace $s_it_i \in L(G_i)$ with arbitrarily long length after a fault event $\sigma_f \in \Sigma_f$, such that $P_{o_i}(s_it_i) = P_i(\tilde{s_i}\tilde{t_i})$.

Let $\tilde{w_j} = P_{N_j}(s_i't_i')$. Since $s_i't_i' \in L(G_{V_{ij}})$, then $s_i't_i' \in P_{N_j}^{-1}[L(G_{N_j}^R)]$. In addition, $P_{N_j}[P_{N_j}^{-1}(L(G_{N_j}^R))] = L(G_{N_j}^R)$, which implies that $\tilde{w_j} \in \mathcal{L}(G_{N_j}^R)$. Notice that $G_{N_j}^R$ is obtained from G_{N_j} after renaming the events of Σ according to function R_j and that G_{N_j} models the normal behavior of G_j. Thus, there exists a trace $w_j \in L(G_j)$, where $\Sigma_f \notin w_j$, such that $P_{o_j}(w_j) = P_j(\tilde{w_j})$.

To conclude the proof, notice that

$$P(\tilde{s_i}\tilde{t_i}) = P[P_{F_i}(s_i't_i')] = P_{F_i}[P(s_i't_i')] = P(s_i't_i'),$$

and

$$P(\tilde{w_j}) = P[P_{N_j}(s_i't_i')] = P_{N_j}[P(s_i't_i')] = P(s_i't_i'),$$

and thus, $P(\tilde{s_i}\tilde{t_i}) = P(\tilde{w_j})$. Notice that according to Step 7 of Algorithm 4.1 the event sets of $G_{F_i}^R$ and $G_{N_j}^R$ are given as $\Sigma_{R_i} \cup \Sigma_o$ and $(\Sigma_{R_j} \setminus \Sigma_f) \cup \Sigma_o$, respectively. Thus, the renamed events of Σ_{R_i} and Σ_{R_j} are private events of $G_{F_i}^R$ and $G_{N_j}^R$, respectively, whereas the events in Σ_o belong to both automata. However, since the event traces of $G_{F_i}^R$ belong to $\Sigma_{R_i}^*$ and the traces of $G_{N_j}^R$ belong to $(\Sigma_{R_j} \setminus \Sigma_f)^*$ and $G_{V_{ij}} = G_{F_i}^R \| G_{N_j}^R$, then an event $\sigma \in \Sigma_o$ belongs to a trace $s \in L(G_{V_{ij}})$, if and only if $\sigma \in \Sigma_{o_i} \cap \Sigma_{o_j}$. Therefore, $P(\tilde{s_i}\tilde{t_i}) = P_i(\tilde{s_i}\tilde{t_i})$ and $P(\tilde{w_j}) = P_j(\tilde{w_j})$, which implies that there exists a trace $s_it_i \in L(G_i)$ of arbitrarily long length after the occurrence of the fault event and a nonfaulty trace $w_j \in L(G_j)$, such that $P_{o_i}(s_it_i) = P_{o_j}(w_j)$. Thus, robust diagnosability is violated.

($\Longrightarrow$) Suppose now that the class $\mathbb{L}$ is not robustly diagnosable with respect to P_{o_i}, $i \in I_m$, and Σ_f. Thus, there exists a trace $s_it_i \in L_i \setminus K_i$, where $\sigma_f \in s_i$ and $|t_i| > n_i$, $\forall n_i \in \mathbb{N}$, and $w_j \in K_j$, such that $P_i(s_it_i) = P_j(w_j)$, which implies that the observable events in s_it_i must all belong to $\Sigma_{o_i} \cap \Sigma_{o_j}$. According to Algorithm 4.1 it is easy to see that $L(G_{F_i}^R) = R_i(L(G_{F_i})) = R_i(L_i \setminus K_i)$ and that $L(G_{N_j}^R) = R_j(L(G_{N_j})) = R_j(K_j)$. Hence, we imply that $\tilde{s_i}\tilde{t_i} = R_i(s_it_i), \tilde{w_j} = R_j(w_j)$, $\tilde{s_i}\tilde{t_i} \in L(G_{F_i}^R)$, where $\sigma_f \in \tilde{s_i}$ and $|\tilde{t_i}| > n_i$, $\forall n_i \in \mathbb{N}$, and $\tilde{w_j} \in L(G_{N_j}^R)$, such that $P_i(\tilde{s_i}\tilde{t_i}) = P_j(\tilde{w_j})$.

We will show that $G_{V_{ij}}$ has a cyclic path that satisfies condition (4.1), and for this purpose, we split the proof in two parts, as follows:

Part I. We show that there exists an arbitrarily long length trace $s'_i t'_i \in L(G_{V_{ij}})$ such that $P_{F_i}(s'_i t'_i) = \tilde{s}_i \tilde{t}_i$ and $P_{N_j}(s'_i t'_i) = \tilde{w}_j$;
Part II. We prove that there exists a cyclic path cl, associated with trace $s'_i t'_i$, satisfying condition (4.1).

In order to prove part I, let us suppose that there exists a state in $G_{V_{ij}}$, $x_{V_{ij}} = (x_{F_i}, x_{N_j})$, reachable from the initial state $x_{0,V_{ij}}$ after the execution of a trace $u \in L(G_{V_{ij}})$, where $P_{F_i}(u)$ is in the prefix-closure of $\tilde{s}_i \tilde{t}_i$. Note that the state $x_{V_{ij}}$ always exists even if $u = \varepsilon$ where $x_{V_{ij}} = x_{0,V_{ij}}$. Now, let $\sigma_q \in \Sigma_{R_i}$ be a feasible event of x_{F_i}, such that $P_{F_i}(u)\sigma_q \in \overline{\{\tilde{s}_i \tilde{t}_i\}}$, and consider the problem of finding a state of $G_{V_{ij}}$, $\hat{x}_{V_{ij}}$, reachable from $x_{V_{ij}}$, that has σ_q as a feasible event. Two cases are possible:
(a) σ_q is an observable event of $\Sigma_{R_i} \cap \Sigma_{R_j}$;
(b) σ_q is an unobservable event of Σ_{R_i}; notice that in this case σ_q cannot be a renamed event of Σ_{R_j}.

Let us first consider case (a). In this case, σ_q will be a feasible event of $x_{V_{ij}}$ if and only if it is feasible for the corresponding state of $G^R_{N_j}$. Since $P_i(\tilde{s}_i \tilde{t}_i) = P_j(\tilde{w}_j)$, then σ_q will be feasible for some state of $G_{V_{ij}}$, $\hat{x}_{V_{ij}} = (x_{F_i}, \hat{x}_{N_j})$, after the occurrence of a finite trace from $(\Sigma_{R_j} \setminus \Sigma_{o_j})^*$. Consider now case (b), *i.e.*, $\sigma_q \in \Sigma_{R_i} \setminus \Sigma_{o_i}$. In this case, since self-loops labeled with all events in the set $\Sigma_{R_i} \setminus \Sigma_{o_i}$ are added to each state of $G^R_{N_j}$ in order to form the parallel composition $G_{V_{ij}} = G^R_{F_i} \| G^R_{N_j}$, we may conclude that σ_q is already feasible for $x_{V_{ij}} = (x_{F_i}, x_{N_j})$. Therefore, it can be seen that there exists an arbitrarily long trace $s'_i t'_i$ associated with $\tilde{s}_i \tilde{t}_i$ such that $s'_i t'_i \in P^{-1}_{F_i}(\tilde{s}_i \tilde{t}_i) \cap P^{-1}_{N_j}(\tilde{w}_j)$, which implies that $P_{F_i}(s'_i t'_i) = \tilde{s}_i \tilde{t}_i$ and $P_{N_j}(s'_i t'_i) = \tilde{w}_j$.

In order to prove part II, *i.e.*, that there exists a cyclic path cl in $G_{V_{ij}}$ whose first components of its states are faulty states and at least one of the events of the cyclic path belongs to Σ_{R_i}, let us assume, without loss of generality, that $\tilde{s}_i = P_{F_i}(s'_i)$ and $\tilde{t}_i = P_{F_i}(t'_i)$. Therefore, t'_i is also an arbitrarily long trace of $L(G_{V_{ij}})$. Notice that since $G_{V_{ij}}$ is a finite state automaton, t'_i must be associated with a cyclic path cl of $G_{V_{ij}}$ whose first components are faulty states. Any cyclic path cl in $G_{V_{ij}}$ must satisfy one of the following three cases:
(i) cl is associated with two cyclic paths, one in $G^R_{F_i}$ and another one in $G^R_{N_j}$;
(ii) cl is associated with a cyclic path in $G^R_{F_i}$ only, *i.e.*, with no cyclic path in $G^R_{N_j}$;
(iii) cl is associated with a cyclic path in $G^R_{N_j}$ only, *i.e.*, with no cyclic path in $G^R_{F_i}$.
If condition *(iii)* holds true, then all states of cl will have the same first component $x_{F_i} \in X_{F_i}$. Therefore $\nexists \sigma_q \in \Sigma_{R_i}$ such that σ_q is an event of the cyclic path cl, which contradicts part I of the proof. On the other hand, when either condition *(i)* or *(ii)* holds true, then, as shown in the above proof of part I, $\exists \sigma_q \in \Sigma_{R_i}$ in the cyclic path cl, which concludes the proof. □

4.3 Computational complexity of the new verifier automaton

The computational complexity of Algorithm 4.1 is obtained based on the steps necessary to generate each $G_{V_{ij}}$ from the new verifier presented. Table 4.1 shows the maximum number of states and transitions of all automata that must be computed in order to generate the verifiers considering as input a class of automata $\mathbb{G} = \{G_1, G_2, \ldots, G_m\},\ i \in I_m$. The *Step* 1 of the algorithm generates the failure behavior labeling automaton A_f, which has 2 states and 2 transitions; the parallel compositions $\tilde{G}_{F_i} = G_i||A_f$, for each $G_i \in \mathbb{G}$ which have maximum number of states equal to $2|X_i|$ and $2|X_i||\Sigma_i|$ number of transitions; and the failure automata G_{F_i} which is the coaccessible part of $\tilde{G}_{F_i}$ and therefore has in its worst case the same complexity as $\tilde{G}_{F_i}$. Before evolving to the next step, we have decided to include an automton G_j to the steps for two reasons. First, to clarify from which automaton each normal/failure behavior automaton is coming and also to make it more fare since it is part of the process. The complexity associated with G_j is $|X_j$ regarding the set of states and $|X_j||\Sigma_f|$ concerning the transitions. *Step* 2 generates the normal behavior labeling automata A_{N_j} which has only 1 state and a self-loop labeled by all the events σ such that $\sigma \in \Sigma_j \setminus \Sigma_f$; and the normal behavior automata $G_{N_j} = G_j \times A_{N_j}$ will have complexity of $|X_j|(|\Sigma_j| - |\Sigma_f|)$. In *Step* 3 the normal behavior automata are renamed according to the renaming function presented in Equation 3.16 and therefore has same computational complexity as the last step. *Step* 4 constructs the first verifiers $G_{V_{ij}} = G_{F_i}||G^R_{N_j}$ for $i = j$. The parallel composition has in its worst case maximum number of states equal to $2|X|^2$ since it is done in pairs and therefore maximum number of transitions equal to $2|X|^2[(|\Sigma| - |\Sigma_f|) + |\Sigma_f|]$. *Step* 6 only renames the faulty behavior automata preserving the complexity and *Step* 7 only redefines the event sets of $G^R_{N_i}$ and $G^R_{F_i}$ which does not impact significantly in terms of computation complexity. Finally in *Step* 8 the last $G_{V_{ij}}$, for $i \neq j$ are constructed. Since here $i \neq j$, the computation complexity is given by $2|X_i||X_j|(|\Sigma_i| + |\Sigma_j| - |\Sigma_f|)$.

Remark 5 *Note that the algorithms presented by MOREIRA* et al. *[6] and CARVALHO* et al. *[27] generate verifiers by performing a parallel composition of the failure behavior automaton with all the normal behavior automata G_{N_i} for each $G_i \in \mathbb{G}$. This mechanism demands more computational complexity than the pairwise approach proposed in CARVALHO* et al. *[20] and in Algorithm 4.1. Moreover, this new algorithm differs from the previous one presented by CARVALHO* et al. *[27] since it does not generate the automata $\mathcal{H}_i$ as proposed by MOREIRA* et al. *[6] and CARVALHO* et al. *[20]. The computational complexity of the polynomial time Algorithm*

Table 4.1: Computational Complexity of Algorithm 4.1

Automaton	States	Transitions
G_i	$\|X_i\|$	$\|X_i\|\|\Sigma_i\|$
A_f	2	2
$\tilde{G}_{F_i}$	$2\|X_i\|$	$2\|X_i\|\|\Sigma_i\|$
G_{F_i}	$2\|X_i\|$	$2\|X_i\|\|\Sigma_i\|$
G_j	$\|X_j\|$	$\|X_j\|\|\Sigma_j\|$
A_{N_j}	1	$\|\Sigma_j\| - \|\Sigma_f\|$
G_{N_j}	$\|X_j\|$	$\|X_j\|(\|\Sigma_j\| - \|\Sigma_f\|)$
$G^R_{N_j}$	$\|X_j\|$	$\|X_j\|(\|\Sigma_j\| - \|\Sigma_f\|)$
$G_{V_{ij}}, i = j$	$2\|X_i\|^2$	$2\|X_i\|^2[2\|\Sigma_i\| - \|\Sigma_f\|]$
$G^R_{F_i}$	$2\|X_i\|$	$2\|X_i\|\|\Sigma_i\|$
$G_{V_{ij}}, i \neq j$	$2\|X_i\|\|X_j\|$	$2\|X_i\|\|X_j\|(\|\Sigma_i\| + \|\Sigma_j\| - \|\Sigma_f\|)$

Complexity $O(m^2|X|^2|\Sigma|)$

4.1 is $O(m^2|X|^2|\Sigma|)$*, smaller than the previous generalized robust diagnosability verifier which has complexity equal to* $O\Big(m|X_i|\prod_{j\neq i}|X_j|(|\Sigma|-|\Sigma_f|)\Big)$ *and smaller than the verifier proposed by TAKAI [23] which has computational complexity equal to* $O(|Q_i| \times |Q_{K_i}|^2 \times (\Pi_{j\neq i}|Q_{k_j}|) \times |\Sigma|^{n+1})$.

Now that we updated the generalized robust diagnosability definition we may present transformation mechanisms capable of turning each already presented problem in Chapter 3 into a generalized robust diagnosability definition input. Moreover, we present a proof that the mechanism holds and that the new definition is necessary and sufficient to detect failures and then we present a test for each case.

4.4 Transformation mechanisms

According to what was presented in Section 3.5 along with the changes proposed in this work, the generalized robust diagnosability definition is capable of encompassing many fault diagnosis problems. Among these problems, we have selected four approaches that were presented in Chapter 3: diagnosis subject to permanent sensor failures [26], robust diagnosis subject to permanent [20] and intermittent [21] sensor failures, and verification of robust diagnosability of partially observed DESs [23]. Some adaptations were made to the original definition and verifier presented in Definition 3.8 and Algorithm 3.1 so that the GRD definition could be as robust and generalized as possible. Hence, let us now propose transformation mechanisms from problems and notations already presented so that it can be pluged into the

GRD definition and let us confirm the effectiveness of the GRD definition.

Following the same sequence of presentation in Chapter 3, let us start with the paper from KANAGAWA and TAKAI [26] regarding fault diagnosis of systems subject to permanent sensor failures.

4.4.1 Diagnosability of discrete event systems subject to permanent sensor failures

The paper presented in Section 3.1 has a notation a little bit different from the one used in this work; it works with the concept of masks instead of natural projections and observable symbols instead of events in the language diagnosability definition and conditions. However, the main idea behind the problem is clear and concerns the possibility of a sensor to fail during a system operation and intends to propose a language diagnosability definition capable of still being able to diagnose fault occurrences even upon a sensor failure. Note that this approach differs from the work proposed by CARVALHO *et al.* [20] since it does not necessarily assume that the sensor failure occurred prior to the first occurrence of the event that the given sensor is monitoring.

Algorithm 4.2 Transformation mechanism from the problem of diagnosability of DESs subject to permanent sensor failures to GRD

Inputs:

- $G = (X, \Sigma, f, \Gamma, x_0, X_m)$
- $M(\sigma)$, $M_i(\sigma)$, $i \in I_m$
- Σ_f

Output: $\mathbb{G} = \{G_1, G_2, \ldots, G_m\}$

1: Define $\Sigma_{sf_i} = \{\sigma \in \Sigma : M(\sigma) \neq \varepsilon \wedge M_i(\sigma) = \varepsilon\}$, $i \in I_m$

2: Define a new unobservable event σ_u and a new set of states $X' = \{x' : x \in X\}$

3: Compute $G_i = (X_i, \Sigma_i, f_i, \Gamma_i, x_0)$, $i \in I_m$, where:

- $X_i = X \cup X'$
- $\Sigma_i = \Sigma \cup \{\sigma_u\} \cup \lambda_i(\Sigma_{sf_i})$
- For $x \in X$,

$$f_i(x, \sigma) = \begin{cases} f(x, \sigma), & \text{if } \sigma \in \Sigma \\ x', & \text{if } \sigma = \sigma_u \end{cases}$$

  and

$$\Gamma_i(x) = \Gamma(x) \cup \{\sigma_u\}$$

- For $x' \in X'$, $\Gamma_i(x') = \Gamma(x)$ and $f_i(x', \sigma) = f_i(f(x, \sigma), \sigma_u)$, if $\sigma \in \Sigma$ and $f(x, \sigma)$ is defined

4: Redefine $f_i(x', \sigma) = f_i(x', \lambda_i(\sigma))$ and $\Gamma_i(x') = \lambda_i(\Gamma_i(x'))$ for every $\sigma \in \Sigma_{sf_i}$

5: Compute the class of automata $\mathbb{G} = \{G_1, G_2, \ldots, G_m\}$

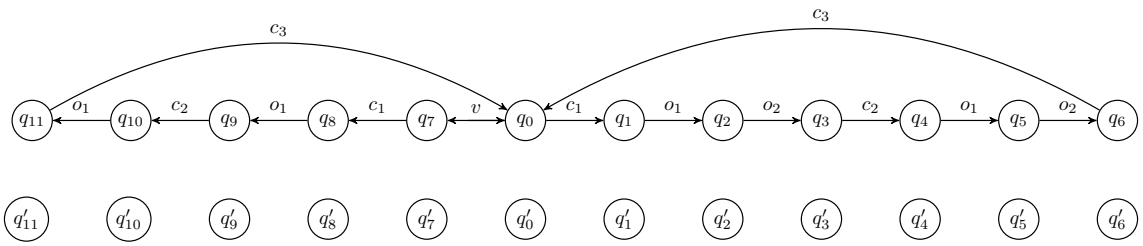


Figure 4.5: Step 2 of Algorithm 4.2.

The transformation mechanisms proposed is in Algorithm 4.2 and considers a system modeled by an automaton G with a defined fault events set Σ_f and associated observation masks $M(\sigma)$ and $M_i(\sigma)$, $i \in I_m$, $I_m = \{1, 2, \ldots, m\}$. Mask $M(\sigma)$ is inherent to the nature of the system whereas masks $M_i(\sigma)$ are related to observations after the permanent failure of a given sensor. The output, a class of automata $\mathbb{G}$ is dealt with the generalized robust diagnosability definition so that the language of the system may be tested.

Before analyzing the algorithm, consider the proposition of the renaming function λ_i:

$$\lambda_i(\sigma) = \begin{cases} \sigma, & \text{if } M_i(\sigma) \neq \varepsilon \vee \sigma \in \Sigma_f \\ \sigma', & \text{if } M_i(\sigma) = \varepsilon \wedge \sigma \notin \Sigma_f \end{cases}$$

Very similar to the renaming function R_i many times referred throughout this work, this function renames symbols that become unobservable in the observation masks M_i and that were originally observable, i.e., for a given event σ its associated symbol was observable according to $M(\sigma) \neq \varepsilon$ but $M_i(\sigma) = \varepsilon$.

In Step 1 we define the sensor subject to failure set Σ_{sf_i} for each mask $M_i(\sigma)$, $i \in I_m$. The idea is to detect which events originally observable in G becomes unobservable in the different masks $M_i(\sigma)$. In Step 2 we define a new unobservable event σ_u and also a new set of states X'. The new event will be responsible for labeling transitions between the original model G to models G_i generating languages with the same observability property as its associated masks M_i. Since we do not use Mealy automata, that was the alternative for mirroring the system model without interfering in the generated language. With the same purpose, the new states were created so that the model could be replicated. In Step 3 we finally create the models G_i which will generate the same language generated by the original system G but will also generate all the language generation possibilities concerning the associated mask M_i observability. Each G_i will then have the states of G duplicated; the same events of G plus the new σ_u event and also the events from its corresponding sensor subject to failure event set Σ_{sf_i} renamed through function λ_i. The transition functions and

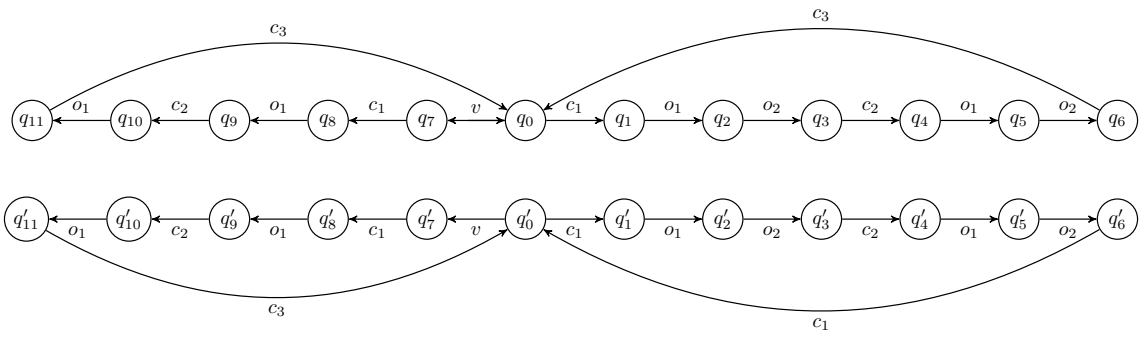


Figure 4.6: Replicating the original system transitions.

feasible event sets are then defined respecting replicating rules where transitions labeled by the new event σ_u go from the states $x_i \in X$ to their corresponding state $x'_i \in X'$; the transitions from the original system are replicated to the model G_i; and the events that were initially observable but become unobservable in M_i are renamed to label transitions in G_i.

The transformation is easier comprehended through an example. Let us consider the motivating example presented in Figure 3.1. In this case, the event set $\Sigma = \{c_1, c_2, c_3, o_1, o_2, v\}$ where $\Sigma_f = \{v\}$ and the mask is according to equation 3.1 where $\Delta = \Sigma - \{v\}$. Doing $i = 1$ let us now consider the mask (equation 3.1) from the motivating example where o_1 becomes unobservable. With all the inputs, let us generate the automaton G_1 through the steps of Algorithm 4.2.

We may first define $\Sigma_{sf_1} = \{o_1\}$ since event o_1 was originally observable and became unobservable in the corresponding observation mask M_1. Then, in Step 2 let us define the new event σ_u and the new set of states $X' = \{q'_0, q'_1, q'_2, q'_3, q'_4, q'_5, q'_6, q'_7, q'_8, q'_9, q'_{10}, q'_{11}\}$ according to Figure 4.5.

We may now step-by-step construct automaton G_1. According to the definition of the transition function of G_i in Algorithm 4.2 we may first replicate transitions from $x \in X$ to $x' \in X'$ labeled by the new unobservable event σ_u as shown in Figure 4.7 . Then, we shall replicate the transitions in G one time, as depicted in Figure 4.6. In Step 4 we must rename the events $\sigma_{sf_i} \in \Sigma_{sf_i}$ and the construction is done, as shown in Figure 4.8.

Suppose also that a mask M_2 is given where instead of o_1, the event o_2 becomes unobservable. The result of the algorithm is according to Figure 4.9 and the class of automata ready to be applied to the generalized robust definition is $\mathbb{G} = \{G_1, G_2\}$. Note that KANAGAWA and TAKAI [26] make no assumptions regarding the diagnosability of the original system with respect to K and M_f. Hence, verifiers for the cases where $i = j$ and $i \neq j$ will be generated assuring complete verification of the language diagnosability of the system.

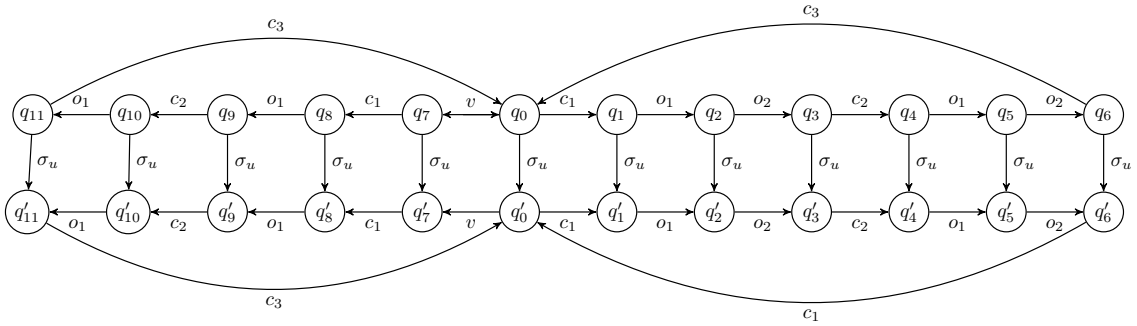


Figure 4.7: Including transitions labeled by σ_u from $x \in X$ to $x' \in X'$.

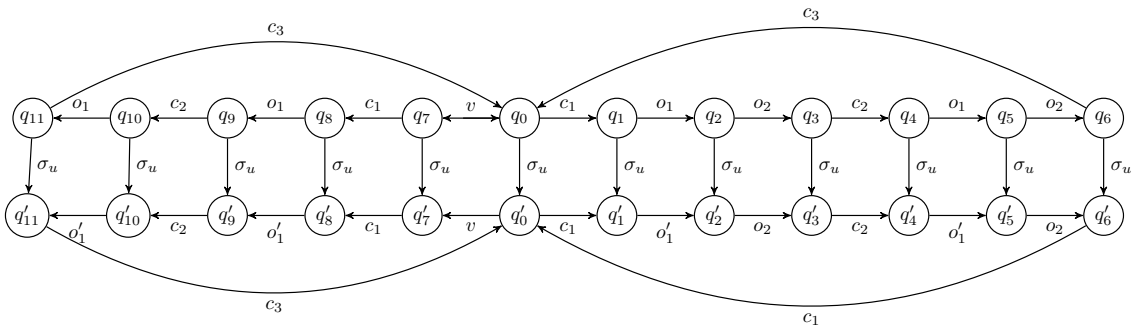


Figure 4.8: Automaton G_1 constructed according to Algorithm 4.2.

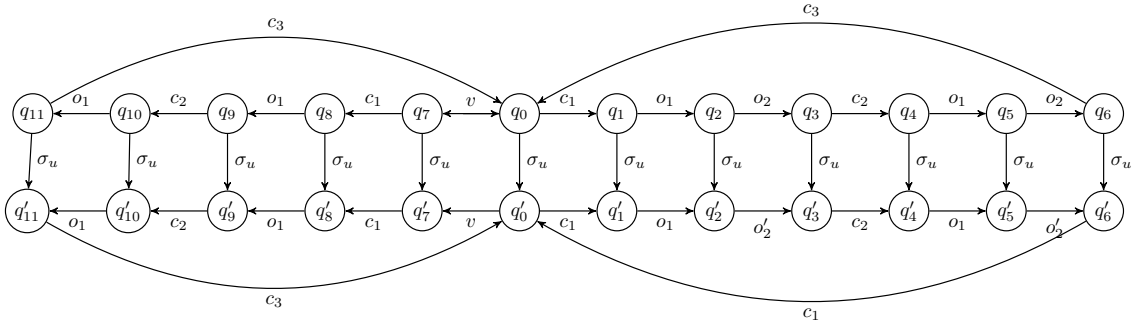


Figure 4.9: Automaton G_2 constructed according to Algorithm 4.2.

It is important to rephrase that the algorithm here presented preserves the modeling methodology proposed by KANAGAWA and TAKAI [26] and based in [16] performing only minor adaptations so that it can be applied as an input of the generalized robust diagnosability definition.

4.4.2 Robust diagnosis of discrete event systems against permanent loss of observations

The problem presented by CARVALHO *et al.* [20] is concerning a sensor failure occurring before the occurrence of the event monitored by the referred sensor that remains out of work throughout the operation of the system. It is very similar to the one presented in Section 4.4.1 and precisely a particular case of the problem described in Section 3.1.

Other then the sensor failure, a set of diagnosis basis Σ_{db} for a given system G is provided. The definition needs to be able to robustly diagnose a fault occurrence in this system considering all its diagnosis basis, i.e., providing the minimal conditions of operation that maintains the language of the system diagnosable.

Having the model G and the set of diagnosis basis Σ_{db} we may quickly solve this problem through the steps of Algorithm 4.3 and the application of the generalized robust diagnosability definition. Consider the algorithm given below.

Algorithm 4.3 Transformation mechanism from the problem of robust diagnosis of DESs against permanent loss of observations to GRD

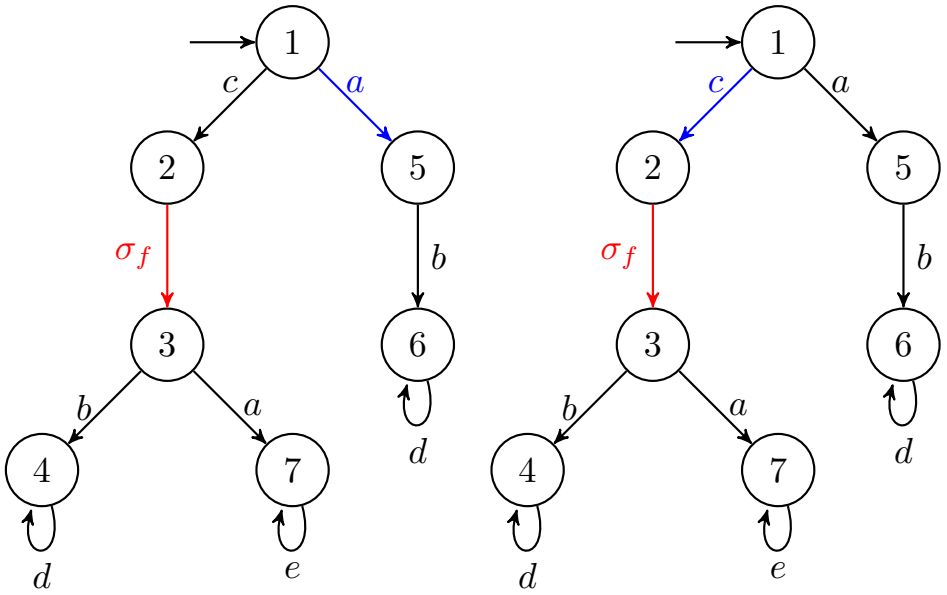
Inputs:

- $G = (X, \Sigma, f, \Gamma, x_0, X_m)$
- $\Sigma_{db} = \{\Sigma_{o_1}, \Sigma_{o_2}, \ldots, \Sigma_{o_m}\}$

Output: $\mathbb{G} = \{G_1, G_2, \ldots, G_m\}$

1: Define $G_i = (X, \Sigma, \Gamma, f, x_0)$, $i \in I_m$
2: For each G_i define the set of observable events as $\Sigma_{o_i} \in \Sigma_{db}$
3: Compute the class of automata $\mathbb{G} = \{G_1, G_2, \ldots, G_m\}$

The idea here is as simple as to build a particular model for each $\Sigma_{o_i} \in \Sigma_{db}$ in order to form a class of automata $\mathbb{G}$. By doing so we are reflecting the diagnosis basis in many models to be cross-checked. We may enhance understandability by considering the example shown in Figure 3.2. Consider automaton G to have $\Sigma = \{a, d, b, c, d, e, \sigma_f\}$, $\Sigma_o = \{a, b, c, d, e\}$, and $\Sigma_{db} = \{\{a, b, c\}, \{c, d, e\}, \{a, c, d\}, \{a, d, e\}, \{a, b, e\}, \{b, c, e\}, \{a, b, c, d\}, \{a, b, c, e\}, \{a, b, d, e\}, \{a, c, d, e\}, \{b, c, d, e\}, \Sigma_o\}$. The verification of the generalized robust diagnosability of the given system is done through the construction of one G_i to each $\Sigma_{o_i} \in \Sigma_{db}$, constructing a class of automata $\mathbb{G}$ and plugging to the GRD definition. As a matter of exemplification, let



(a) G_9 (b) G_{11}

Figure 4.10: Automata G_9 and G_{11} constructed according to Algorithm 4.3 where the blue transitions are labeled by unobservable events and the red ones are labeled by the fault event.

us depict in Figure 4.10 automata G_9 and G_{11} from the associated $\Sigma_9 = \{a, b, d, e\}$ and $\Sigma_{11} = \{b, c, d, e\}$. Notice that the transitions labeled by the fault event σ_f were colored red and the ones labeled by unobservable events particular to each G_i were colored blue according to the corresponding Σ_i for better visualization. Even though the language generated by automata G_9 and G_{11} are diagnosable with respect to its own natural projections and fault event set, we may check if a single diagnoser is capable of fault diagnosing the language considering all the diagnosis basis presented. Let us consider the occurrence of trace $s = c\sigma_f bd$. Automaton G_9 will project this trace as $s_9 = \{cbd\}$ while automaton G_{11} projects as $s_{11} = \{bd\}$. Even without the permanent observance of events a and c respectively in automata G_9 and G_{11} the language diagnosability property remains unchanged since there is no faulty trace with the same projection as a normal trace.

4.4.3 Robust diagnosis of discrete event systems against intermittent loss of observations

The most complex and less explored problem among the ones addressed in this work is the robust diagnosis of discrete event systems against intermittent loss of observations. Even though it is a perfectly normal practical possibility easily described as sensor malfunctioning or communication failure between sensors and the diagnoser which can be caused by bad electrical linkage, defective components, circuit heating, measurement noise, data communication failure, etc, to model these situations is a

very hard task.

The work proposed by CARVALHO *et al.* [21] includes a modeling tool - the dilation operation - that facilitates a lot the comprehension and solution of the problem. Hence, in our transformation mechanism here presented we make use of the abovementioned operation.

Algorithm 4.4 has as input a system model G along with a set of possible intermittent loss of observation sets $\Sigma_{ilo_i} \in \Sigma_{ilo}$. The objective of the algorithm is to build automata that generates the dilated languages of the original system model based on different possible combinations of sensors subject to intermittent failure and consequently events subject to intermittent loss of observations.

Algorithm 4.4 Transformation mechanism from the problem of robust diagnosis of DESs against intermittent loss of observations to GRD

Inputs:

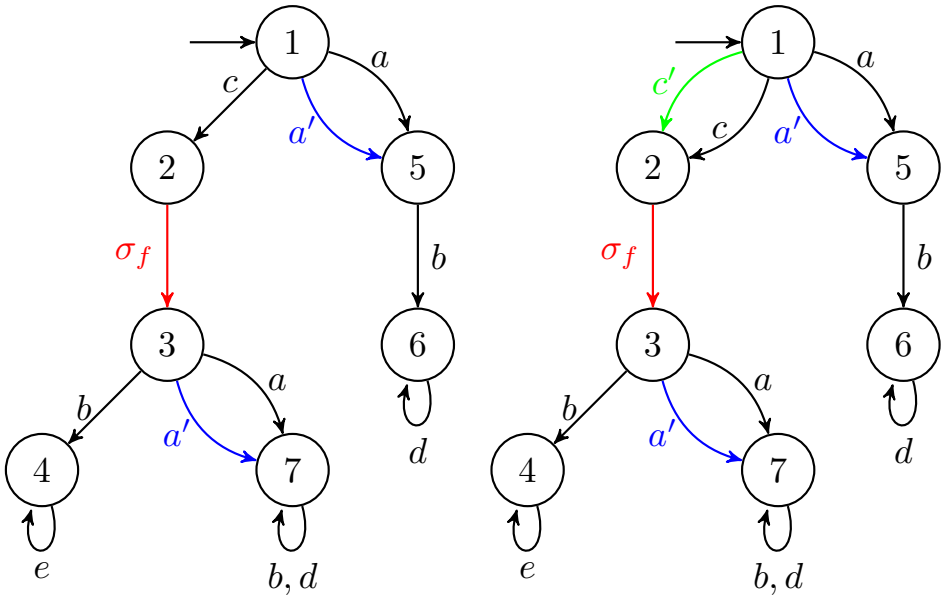
- $G = (X, \Sigma, f, \Gamma, x_0, X_m)$
- $\Sigma_{ilo} = \{\Sigma_{ilo_1}, \Sigma_{ilo_2}, \ldots, \Sigma_{ilo_m}\}$

Output: $\mathbb{G} = \{G_{dil_1}, G_{dil_2}, \ldots, G_{dil_m}\}$

1: Define $\Sigma'_{ilo_i} = \{\sigma' : \sigma \in \Sigma_{ilo_i}\}$ for each $\Sigma_{ilo_i} \in \Sigma_{ilo}$
2: Compute $\Sigma_{dil_i} = \Sigma \cup \Sigma'_{ilo_i}$ for each Σ'_{ilo_i}
3: Build an automaton modeled by $G_{dil_i} = (X, \Sigma_{dil_i}, f_{dil_i}, \Gamma_{dil_i} x_o)$ where $\Gamma_{dil_i}(x) = D[\Gamma(x)]$, $x \in X$, and $\forall \sigma_{dil_i} \in \Gamma_{dil_i}(x) : \sigma_{dil_i} \in D(\sigma)$, $f_{dil_i}(x, \sigma_{dil_i}) = f(x, \sigma)$, $\sigma \in \Gamma(x)$ for each Σ_{dil_i}
4: Compute the class of automata $\mathbb{G} = \{G_{dil_1}, G_{dil_2}, \ldots, G_{dil_m}\}$

Since we make use of dilation, the procedure gets very simple and similar to what was presented for the problem of diagnosing a DES subject to permanent loss of observations. We may start by renaming the events in each Σ_{ilo_i} and defining corresponding Σ'_{ilo_i} according to Step 1. Then, we define the event set Σ_{dil} in order to include in the same event set the events from Σ and the ones just renamed in Step 1. We are now ready for building automata G_{dil_i} where the dilation operation is applied on the feasible event set of G in a sense of detecting the occurrence of events in G that are particularly subject to intermittent loss of observations in each G_{dil_i} to be constructed. For each case, the transition labeled by the occurrence of this event is duplicated and labeled by the renaming of the referred event. A class of automata $\mathbb{G} = \{G_{dil_1}, G_{dil_2}, \ldots, G_{dil_m}\}$ is then computed in Step 4 which becomes an input to the generalized robust diagnosability definition.

We may get a clear view of the algorithm development by observing an example. Consider the automaton G from Figure 3.3 (a). Assume $\Sigma_o = \{a, b, c, d, e\}$, $\Sigma_{ilo_1} = \{a\}$ and $\Sigma_{ilo_2} = \{a, c\}$. Step 1 results in the definition of sets $\Sigma'_{ilo_1} = \{a'\}$ and $\Sigma'_{ilo_2} = \{a', c'\}$. In Step 2 $\Sigma_{dil_1} = \{a, b, c, d, e, \sigma_f, a'\}$ and $\Sigma_{dil_2} = \{a, b, c, d, e, \sigma_f, a', c'\}$ are defined. Then, in Step 3, both G_{dil_1} and G_{dil_2} are constructed according to Figure



(a) G_{dil_1} (b) G_{dil_2}

Figure 4.11: Automata G_{dil_1} and G_{dil_2} constructed according to Algorithm 4.4 where the red transitions are labeled by the faulty event; the blue transitions are labeled by a'; and the green transitions are labeled by c'.

4.11. Note how the events subject to intermittent loss of observations are depicted in green and blue and the fault event is in red.

Consider the occurrence of trace $s = c\sigma_f ad$ in G. In case events a' and c' were not modeled in automata G_{dil_1} and G_{dil_2}, for $\Sigma_{ilo_1} = \{a\}$ the occurrence would lead the diagnoser to correctly detect a fault occurrence since the event c is observable. However, for $\Sigma_{ilo_2} = \{a, c\}$ the diagnoser would be uncertain whether the system was in state 6 or state 7, since the observed sequence would be $s_o = \{d\}$. The dilation operation allows the sensors monitoring a and c to fail without compromising the language diagnosability property since the possibility of failure is modeled through the renamed events a' and c'.

4.4.4 Verification of robust diagnosability for partially observed discrete event systems

The problem of verification of robust diagnosability for partially observed discrete event systems addressed in [23], seems, at first, to be very simple and ordinary comparing to more complex assessments. However, unlike most of the works to the best of our knowledge, TAKAI [23] proposes a new source of uncertainty to the diagnosability equation by assuming that an exact model for a system is not necessarily given; instead, a set of possible models sharing the same event set is the most accurate information about a system. Each possible model G has an associated subautomaton G_k that models the normal behavior of the system. Another difference

in this work: the diagnosability is not based solely on the language generated by the system. We do not have here any information regarding the occurrence of a faulty event so that we can search for traces containing it. By the observation of the normal behavior, diagnosability must be tested regarding the traces that belong to the language $L(G)$ but exceeds the traces conforming the normal language $K \subseteq L(G)$. Hence, even though it is an ordinary problem, the notation and methodology here used demanded a little more sophistication from the transformation mechanism. Let

Algorithm 4.5 Transformation mechanism from the problem of verification of robust diagnosability for partially observed DESs to GRD

Inputs:

- $\mathbb{G} = \{G_1, G_2, \ldots, G_n\}$ where $G_i = (X_i, \Sigma, f_i, \Gamma_i, x_0)$, $i \in I_m$
- $\mathbb{G}_N = \{G_{N_1}, G_{N_2}, \ldots, G_{N_n}\}$ where $G_{N_i} = (X_{N_i}, \Sigma, f_{N_i}, \Gamma_{N_i}, x_0)$, $i \in I_m$

Output: $\mathbb{G}_T = \{G_{T_1}, G_{T_2}, \ldots, G_{T_m}\}$

1: Function **TAKAI2012TOGRD**(G,GN):

1.1: Compute $G_{comp} = Comp(G_N)$

1.2: Compute $G_p = G||G_{comp} = (X_p, \Sigma, f_p, \Gamma_p, x_0)$ where $x_p = (x, x_N)$ for $x_p \in X_p, x \in X$ and $x_N \in X_N$

1.3: Compute $X_c = \{x_{c_1}, x_{c_2}, \ldots, x_{c_k}\}$ such that $x_{c_i} = x_1$, $x_1 \in X$, in $f_p(x_{p_1}, \sigma) = (x_{p_2})$ where $x_{p_1} = (x_1, x_{N_1})$, $x_1, x_{N_1} \neq D \wedge x_{p_2} = (x_2, x_{N_2})$, $x_{N_2} = D$ for each state $x_i \in X$ and $x_{d_2}, \ldots, x_{d_k}\}$ for each $x_{c_i} \in X_c$.

1.4: Compute $X_d = \{x_{d_1}, x_{p_2}\}$ where $x_{p_1} = (x_1, x_{N_1})$, $x_1, x_{N_1} \neq D \wedge x_{p_2} = (x_2, x_{N_2})$, $x_{N_2} = D$ for each state $x_i \in X$ and $x_{N_i} \in X_N$. Let $x_c = x_1$ denote a state that satisfies this condition and define $X_c = \{x_{c_1}, x_{c_2}, \ldots, x_{c_k}\}$.

1.5: Define $G_T = (X \cup X_d, \Sigma \cup \{\sigma_f\}, f_T, \Gamma_T, x_0)$ where $f_T(x_T, \sigma) = f_N(x_N, \sigma)$

1.6: Add the transition $f_T(x_{c_i}, \sigma_f) = x_{d_i}$, $i = 1, 2, \ldots, k$, for each $x_{c_i} \in X_c$ and $x_{d_i} \in X_d$

1.7: Given that $f_p(x_{p_1}, \sigma) = (x_{p_2})$, if $x_{p_2} = (\cdot, D) \wedge D \notin x_{p_1} \wedge x_1 = x_{c_i} \in X_c$, add the transition $f_T(x_{d_i}, \sigma) = x_2$, $i = 1, 2, \ldots, k$, $x_2 \in x_{p_2}$, and define a event set X_s with all the states $x_2 \in X$ satisfying this condition.

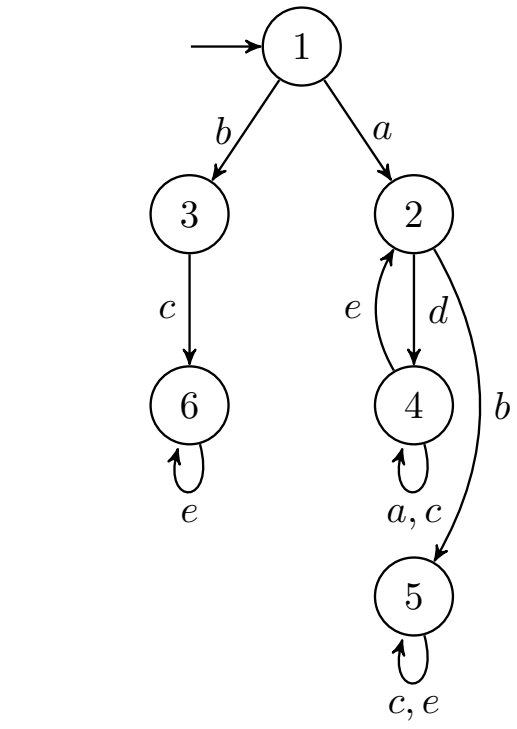
1.8: Add the transition $f_T(x_s, \sigma) = f(x_s, \sigma)$ for each state $x_s \in X_s$ and for every event $\sigma \in \Sigma$ such that $\Gamma(x_s) \neq \Gamma_N(x_s) \wedge x_s \notin X_c$

1.9: Return $G_T = (X \cup X_d, \Sigma \cup \{\sigma_f\}, f_T, \Gamma_T, x_0)$

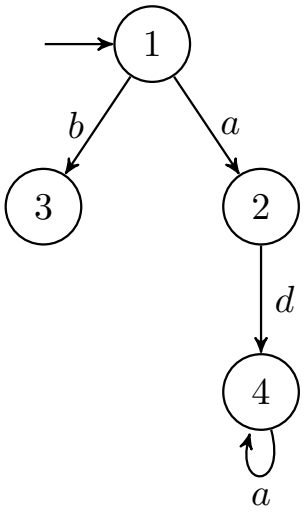
2: **for** i = 1 **to** m **do** $G_{T_i} =$ **TAKAI2012TOGRD**(G, G_N)

3: Compute the class of automata $\mathbb{G}_T = \{G_{T_1}, G_{T_2}, \ldots, G_{T_m}\}$

us consider the Algorithm 4.5 where a set of possible models for a system G is given by the class $\mathbb{G}$ and the associated set of possible normal behavior models for the referred system is given by the class $\mathbb{G}_N$. Due to the complexity of this algorithm we have decided to first define a function for a particular case where we have only one model G and one associated normal behavior model G_N. The function is applied to each one of the possible models and the output of the algorithm is also a class of automata, here called as $\mathbb{G}_T$.

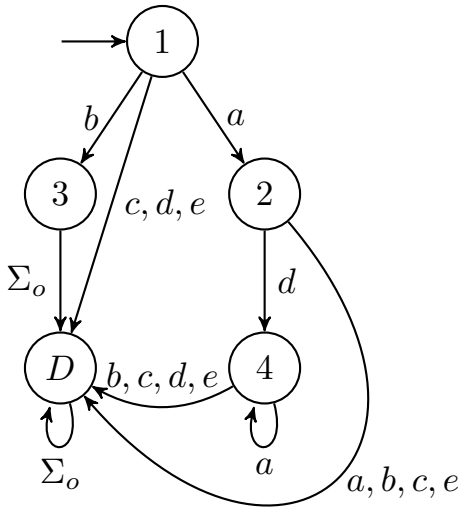


(a) G

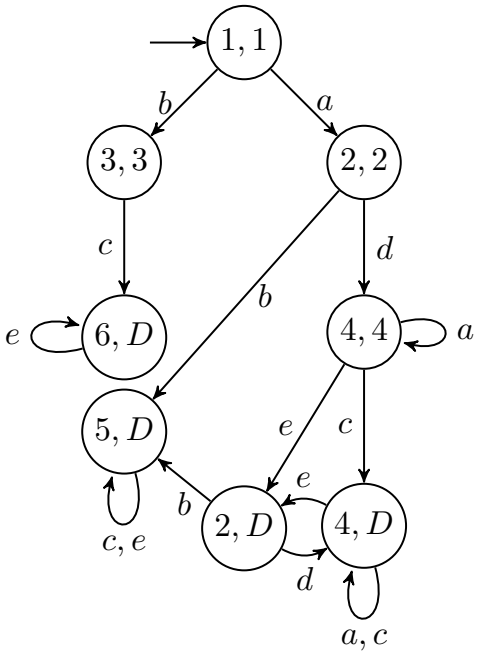


(b) G_N

Figure 4.12: State transition diagrams for the example of Algorithm 4.5.



(a) G_{comp}



(b) $G_p = G||G_{comp}$

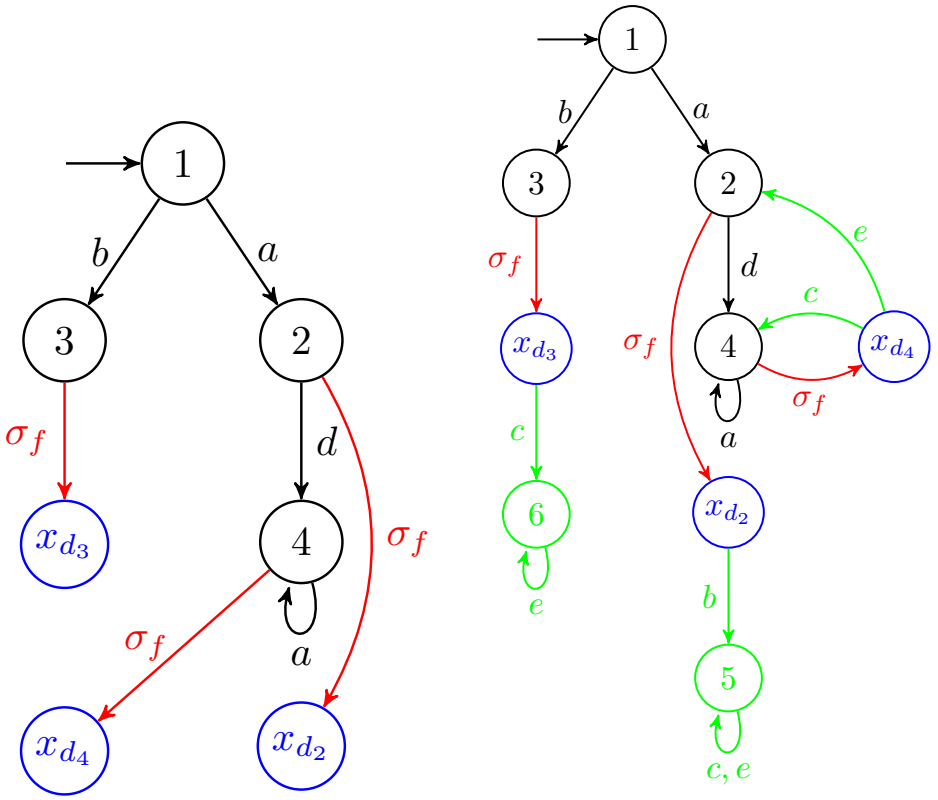
Figure 4.13: Complement of G_N and the parallel composition G_p from Algorithm 4.5.

The proposed function **TAKAI2012TOGRD** receives G and G_N as inputs and proceeds as follows. Firstly, it computes the complement of automaton G_N so that the events are all active in the feasible event sets of the automaton. Then the synchronous parallel composition G_p is proposed between G and the complement of G_N. By doing so we can see clearly the transitions and states present in G that are not part of the normal behavior of the system. From there we start an analytical procedure which main goal is to construct a new model G_T based initially on the normal behavior of the system. We then detect what is not part of the normal behavior of the system through compositions with the model G and sort of complete automaton G_N with the failure behavior model. In order to make it compatible with the generalized robust diagnosability definition and methodology we introduce a faulty event σ_f to the new model G_T so that it clearly marks the transition from the normal behavior to the failure behavior of the system.

All this information becomes a lot more clear if we consider an example. Before we go for the example proposed by TAKAI [23], we may work with the state transition diagrams from Figure 4.12 where (a) models a system and (b) models the normal behavior of this system. This model is used in order to cover a variety of deviations on a state transition diagram that must be encompassed by the transformation mechanism. Back to the example, our idea here is to start from the model depicted in Figure 4.12 (b), include transitions labeled by a new introduced fault event σ_f so that normal and failure behaviors can be easily differentiated from each other and conclude with the completion of the new automaton following the behavior depicted in Figure 4.12 (a).

By following the steps of Algorithm 4.5 we may start computing automaton G_{comp} and then the parallel composition $G_p = G||G_{comp}$ as shown in Figure 4.13. Notice from Figure 4.13 (b) that there are states $x_{p_1} = (x_1, x_{N_1})$ that $x_{N_1} = D$ and that $x_1 \neq D$, as condition imposed in Step 1.4 of the Algorithm 4.5. Precisely, the state set $X_c = \{2, 3, 4\}$ and $X_d = \{x_{d_2}, x_{d_3}, z_{d_4}\}$, since $f((2,2), b) = (5, D)$, $f((3,3), c) = (6, D)$ and $f((4,4), e) = (2, D)$. Note through Figure 4.12 (b) that these are the states on the edge of the normal behavior model G_N from where transitions leave in the original model G.

The next step defines an automaton G_T with the same transition function as G_N and the particularity of having the states in X_d and the fault event σ_f as part of its state set and event set, respectively. New step adds the transitions $f_T(x_{d_i}, \sigma_f) = (x_{p_2})$ according to Step 1.4, as shown in Figure 4.14 (a). The new transitions labeled by the faulty event are colored red and the new states x_{d_i} are colored blue for better visualization. By concluding this step we have successfully set the boundaries between normal behavior and failure behavior models of automaton G. Hence, the conclusion of the remaining steps of function **TAKAI2012T0GRD**(G, G_N) leads to the completion of model G_T so that it generates the language generated by the original system G and is at the same time aplicable to the generalized robust diagnosability definition. The result is shown in Figure 4.14 (b), where the transitions added after the boundary setting from normal to failure behaviors are colored green. Note how G_T generates the same language as G and is perfectly compatible to serve as input for the GRD definition.



(a) G_T after Step 1.6 (b) G_T

Figure 4.14: Function **TAKAI2012T0GRD**(G, G_N) of Algorithm 4.5 where the blue states are the new one; the red transitions are labeled by the faulty event; and the green states and transitions are the ones added after the normal behavior boundary.

The remaining steps of Algorithm 4.5 are responsible for creating the class of automata $\mathbb{G}_T$ for $i \geq 1$. The main step is simple and consists of using the just presented function for each $G_i \in \mathbb{G}$.

4.5 Final comments

This chapter concentrated all the contributions of this work. In Section 4.1 we have updated the generalized robust diagnosability definition by relaxing two commonly given assumptions. In Section 4.2 the most important part of the work was presented where a modified verification algorithm has improved the computational complexity of the initially presented verifier as it can be seen in Section 4.3. Besides, adjustments were made causing the necessary and sufficient theorem proving the correctness of the verifier to change, along with its proof. Finally and with the objective of showing how the generalized robust diagnosability definition is indeed capable of encompassing all problems addressed in this work we presented in Section 4.4, for each paper introduced in Chapter 3, transformation mechanisms given as simple algorithms in order to adjust the inputs from one definition to another.

Chapter 5

Conclusion and Future Work

The main objectives of this work were: *(i)* to update the already presented definition of generalized robust diagnosability by relaxing some assumptions, along with *(ii)* an improvement in the computational complexity of the verification algorithm presented to verify the language robust diagnosability of discrete event systems. In the process, not only the definition was updated and the computational complexity improved but the verifier got a little more general in a sense of performing diagnosability, codiagnosability and robust diagnosability, which demanded a new theorem to prove its correctness to be presented. In addition, a central goal of this work was *(iii)* to show whether or not the generalized robust diagnosability definition could be applied to different problems addressed concerning fault diagnosis of discrete event systems. Four main concerns and propositions were studied: diagnosability of DESs subject to permanent sensor failures, not necessarily considering the sensor to fail from the start of the system operation; robust diagnosis of DESs subject to permanent loss of observations, which considers that a sensor may fail prior to the first occurrence of the event it is monitoring and remains out of work through the operation of the system; robust diagnosis of DESs against intermittent loss of observations, the most complex and less explored problem among all the others where a sensor may fail at any time during the system operation and recover its functions as suddenly as it got broken and the system must be modeled and diagnosed with such a robustness capable of keeping the language diagnosability property. Finally, the verification of robust diagnosability for partially-observed DESs arises with the idea of uncertainties in the model of a system instead of in its language. This work was also complex to deal with since it does not consider a fault event to occur; instead, the normal behavior of the system is given, which demands more sophistication of a transformation mechanism to a language based diagnosability definition - which is the case of the GRD definition.

After deep comprehension of the papers here presented, wide discussions on the adaptations necessary to be made on the generalized robust diagnosability definition and its verifier so that it could safely encompass every single aspect of the problems presented, the results have corresponded the efforts and we may conclude that the generalized definition remains updated for the trending topics on fault diagnosis of discrete event systems.

Transformation mechanisms with irrelevant computational complexity were developed for each one of the fault diagnosis approaches, which were tested through the implementation of the algorithms presented in Python and using the scientific computing program DESLab. The verification algorithm has now an improved polynomial time computational complexity that summed to the complexity of the transformation algorithms still remain an advantageous methodology for the offline verification of language based diagnosability definitions .

During the development of the work here presented we have had preliminary discussions concerning a problem that, to be best of our knowledge, has not been explored so far regarding the possibility of a sensor not only to fail but also to file a wrong information. Very simple examples may be given to illustrate the concept, starting from temperature sensors of HVAC systems, that may inform the wrong temperature of a room for the user, to fingerprint password sensors to access a digital bank account, which may allow that an incompatible fingerprint, and therefore the wrong user, to accesses an account that does not belong to him. We understand that an interesting line of work outlines from the investigation of the relevance of this problem to discrete event systems modeling. Besides, it is important to understand if the generalized robust diagnosability definition could also be able to encompass this not yet explored diagnosability problem.

Bibliography

[1] RIFKIN, J. *The End of Work: Decline of the Global Labor Force and the Dawn of the Post-market Era*, v. 1. New York, NY, Warner Books, 1996.

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

[3] LIN, F. "Diagnosability of discrete event systems and its applications", *Discrete Event Dynamic Systems: Theory and Applications*, v. 4, pp. 197–212, 1994.

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

[5] YOO, T.-S., LAFORTUNE, S. "Polynomial-time verification of diagnosability of partially observed discrete-event systems", *IEEE Trans. on Automatic Control*, v. 47, n. 9, pp. 1491–1495, 2002.

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

[7] QIU, W., WEN, Q., KUMAR, R. "Decentralized diagnosis of event-driven systems for safely reacting to failures", *IEEE Trans. on Automation Science and Engineering*, v. 6, n. 2, pp. 362–366, April 2009. doi: 10.1109/TASE.2008.2009093.

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

[9] TRIPAKIS, S. "Fault diagnosis for timed automata". In: Springer-Verlag (Ed.), *Formal Techniques in Real Time and Fault Tolerant Systems*, v. 2469, *Lecture notes in Computer Sciences*, pp. 205–221, 2002.

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

[11] THORSLEY, D., TENEKETZIS, D. "Diagnosability of Stochastic Discrete-Event Systems", *IEEE Trans. on Automatic Control*, v. 50, n. 4, pp. 476–492, 2005.

[12] CONTANT, O., LAFORTUNE, S., TENEKETZIS, D. "Diagnosability of discrete event systems with modular structure", *Discrete Event Dynamic Systems: Theory And Applications*, v. 16, n. 1, pp. 9–37, 2006.

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

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

[15] KUMAR, R., TAKAI, S. "Inference-Based Ambiguity Management in Decentralized Decision-Making: Decentralized Diagnosis of Discrete-Event Systems", *IEEE Trans. on Automation Science and Engineering*, v. 6, n. 3, pp. 479–491, 2009.

[16] ROHLOFF, K. R. "Sensor Failure Tolerant Supervisory Control". In: *44th IEEE Conference on Decision and Control and European Control Conference*, pp. 3493–3498, Seville, Spain, 2005.

[17] SANCHEZ, A. M., MONTOYA, F. J. "Safe supervisory control under observability failure", *Discrete Event Dynamic Systems: Theory and Applications*, v. 16, n. 4, pp. 493–525, 2006.

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

[19] CARVALHO, L. K., BASILIO, J. C., MOREIRA, M. V. "Robust diagnosability of discrete event systems subject to intermittent sensor failures". In: *Preprints of the 10th International Workshop on Discrete Event Systems*, pp. 94–99, Berlin, Germany, 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, 2012.

[21] CARVALHO, L. K., BASILIO, J. C., MOREIRA, M. V. "Robust diagnosis of discrete event systems against intermittent loss of observations", *Automatica*, v. 48, n. 1, pp. 2068–2078, 2012.

[22] TAKAI, S. "Robust failure diagnosis of partially observed discrete event systems". In: *Preprints of 10th International Workshop on Discrete Event Systems*, pp. 215–220, Berlin, Germany, 2010.

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

[24] ATHANASOPOULOU, E., LINGXI, L., HADJICOSTIS, C. "Maximum Likelihood Failure Diagnosis in Finite State Machines Under Unreliable Observations", *IEEE Trans. on Automatic Control*, v. 55, n. 3, pp. 579 –593, 2010.

[25] THORSLEY, D., YOO, T.-S., GARCIA, H. "Diagnosability of Stochastic Discrete-Event Systems Under Unreliable Observations". In: *Proc. of the 2008 American Control Conference*, pp. 1158–1365, Seatle, WA, 2008.

[26] 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.

[27] CARVALHO, L. K., MOREIRA, M. V., BASILIO, J. C. "Generalized robust diagnosability of discrete event systems". In: *Proc. of 18th IFAC World Congress*, pp. 8737–8742, Milan, Italy, 2011.

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

[29] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., et al. *Introduction to algorithms*, v. 2. MIT press Cambridge, 2001.

[30] LIMA, S. T. S., BASILIO, J. C., LAFORTUNE, S., et al. "Robust diagnosis of discrete-event systems subject to permanent sensor failures". In: *Preprints of the 10th International Workshop on Discrete Event Systems*, pp. 100–107, Berlin, Germany, 2010.

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