



UNICA

UNIVERSITÀ
DEGLI STUDI
DI CAGLIARI



西安电子科技大学
XIDIAN UNIVERSITY

DOUBLE Ph.D. DEGREE IN

Ingegneria elettronica ed informatica (Università degli Studi di Cagliari)

and

Electronic Science and Technology (Xidian University)

Cycle XXXVI

TITLE OF THE Ph.D. THESIS

Fault Diagnosis of Discrete Event Systems Under Sensor Attack

Scientific Disciplinary Sector(s)

ING-INF/04-Automatica

Ph.D. Student:	Kang Tenglong
Supervisor (UniCa)	Prof. Carla Seatzu
Supervisor (XDU)	Prof. Zhiwu Li

Final exam. Academic Year 2022/2023

Thesis defence session: May 2025

Chapter I Introduction	1
1.1 Overview of Fault Diagnosis Methods for DESS	2
1.2 Cyber Attacks	3
1.2.1 Background.....	3
1.2.1 Related Works on Cyber Attacks in DESS	4
1.3 Contribution of the Thesis	7
Chapter II Preliminaries	11
2.1 Language and Finite State Automata	11
2.2 Observation Models and Observer Constructions	14
2.3 Fault Diagnosis of DES	15
2.3.1 Language Diagnosability.....	15
2.3.2 Online Diagnosis	16
2.3.3 Diagnosability Verification.....	17
Chapter III Fault Diagnosis Under Attack: Using Diagnoser	23
3.1 Introduction.....	23
3.2 Attack model	25
3.3 Active Attacks on Fault Diagnosis	26
3.1 Problem Statement	27
3.2 Attacker's Stealthiness.....	29
3.4 Stealthy Joint Diagnoser	30
3.4.1 Attacker Diagnoser and Operator Diagnoser	30

3.4.2 Joint Diagnoser and Refining Process	34
3.4.3 Verification of existence of harmful attackers.....	37
3.5 Attack Synthesis.....	42
3.5.1 Introduction.....	42
3.5.2 Synthesis of Attackers	43
3.5.3 Synthesis of Successful Attackers	46
3.6 Case Study: A Power System.....	47
3.7 Summary.....	49
Chapter IV Fault Diagnosis Under Attack: Using Verifier	51
4.1 Motivation and Contribution.....	51
4.2 Problem Statement	52
4.3 Attack Model.....	53
4.4 Faulty and Normal Automaton Under Attack	54
4.5 Verification Under Attack Using Verifiers.....	59
4.6 Case Study	63
4.6.1 A Power System.....	63
4.6.2 A Networked Computer Systems.....	64
4.7 Summary	66
Chapter V Conclusion and Future Work	69
5.1 Concluding Remarks.....	69
5.2 Future Works	70

Chapter I Introduction

The safety and reliability of dynamic systems has been increasing as technology evolves. The occurrence of a fault causes deviation from the normal or desired behavior of a system or one of its components, which may undermine the reliability and safety in this system. Experience in monitoring dynamic systems shows that there is a large spectrum of faulty situations in complex processes, which drives significant interest in developing on-board diagnostics to detect component faults in practical applications, such as heating, ventilation, and air conditioning systems [1], transportation systems [2-3], manufacturing systems [4-5], and communication networks [6].

Faults could arise from various sources including design flaws, equipment failures, and operator errors. For example, in a heating, ventilation, and air conditioning system, faulty behavior is modeled, including: (i) the behavior of the controller when it fails "on" or "off", and similarly for the pump; (ii) the valve getting stuck in the "open" or "closed" position. Common to these applications of interest is the fact that sensor information is limited and hence *fault diagnosis* would require inferencing from limited sensor data using a suitable dynamic system model under consideration.

This thesis addresses the fault diagnosis problem with the framework of *discrete event systems* (DES) [7-8] that can transition between its states in response to discrete events, such as completing a job or the activation of a sensor. More specifically, a DES is an event-driven model-based formalism that can provide a time-abstract model for complex dynamic systems, including telecommunication networks [9], manufacturing systems [10-11], power systems [12], sensor networked systems [13], smart buildings [14], aircraft control [15], and IoT privacy [16]. For the purpose of fault diagnosis, DES models naturally capture faults as abrupt changes (events) in the system, which facilitates the analysis of faulty behaviors of the system. There are many approaches and tools available for fault diagnosis of DES such as Petri nets [17-19], Markov processes [20-21], Process Algebra [22], and Automata theory [23-25]. Within the context of Automata theory, the underlying structure of a DES is that of a deterministic state transition system, where transitions are labeled by *events* of interest.

1.1 Overview of Fault Diagnosis Method for DESs

Fault diagnosis has been one of the key research topics in the DES community for over 20 years [23-37]. Starting with the first well recognized state-based modeling framework introduced in [23], followed by an even more widely adopted language-based framework proposed in [24], extensive research approaches has been presented to address a large variety of fault scenarios. In the DES fault diagnosis literature, a fault is usually modeled as an unobservable event. Fault diagnosis in DES aims to determine whether fault events have occurred based on the current observation.

Fault diagnosis can be performed offline (when some experiment is performed and data are collected from the experiment and analyzed to verify if some fault has actually occurred), or online (when the fault is detected while the system is running). Diagnosability of a DES, as formalized in [24], is a crucial property of system behavior that guarantees the detection of any fault occurrence within a defined and bounded delay. Offline diagnosability verification can be performed using a systematic process, called *a diagnoser*, which, based on a model of the physical system, provides a complete characterization of the diagnosis problem under the considered model, where every state of the diagnoser is a possible diagnosis and every possible diagnosis is a state of the diagnoser. Moreover, the diagnoser presents an efficient online diagnosis algorithm, i.e., updates the diagnosis after a new observation only requires the firing of a single transition.

However, the construction of the entire diagnoser may be unwieldy since in the worst case its size is exponential in the number of states of the system. To overcome the potential state explosion problem for testing diagnosability using diagnosers, the so-called *verifier*, introduced in [38-39] and later on revisited in [40-42], is developed to provide a worst-case polynomial test in the number of states of the system for diagnosability, without constructing diagnosers. The negation of diagnosability motivates the idea behind a verifier, i.e., a system is diagnosable if and only if there is no pair of arbitrarily long words that have the same observable projection, such that the fault event occurs in the first word but not in the second. Based on this, the construction of a verifier can be achieved by performing a suitable product of two copies of the automaton modeling the system to be diagnosed.

Before we move to the next section, it is essential to reference some studies on robust diagnosability. The notion of robust diagnosability is first introduced in [43] within the context of codiagnosability, ensuring that the failure of communication between some local diagnoser and the plant does not preclude the fault from being diagnosed. Later, a new notion of robust diagnosability in which the system is described by a specific class of automata

was proposed in [4]. Robust diagnosability against permanent loss of observations has been explored in [44-46]; the former assumes that observation loss occurs before any occurrence of the events subject to loss of observation, whereas the latter assumes that such loss can happen at any time, even after an event that is subject to loss of observation has occurred; however, once the loss of observation occurs, no other event can have its observation lost. Lastly, a generalized formulation of robust diagnosability has been introduced in [47-48]. However, all aforementioned disruptions are endogenous in systems, which may be caused by component aging degradation, measurement noise, etc. The situation is usually worse when exogenous disruptions are taken into account, especially when produced by cyber attackers which are sufficiently intelligent to actively choose their actions for malicious purposes. Faults can be treated when production stops, but attacks need to be circumvented as soon as they are detected to prevent the attacker from taking full control of the process.

1.2 Cyber Attacks

This section will provide an overview of the context of cyber attacks, as well as a review of existing research and efforts addressing this critical area.

1.2.1 Background

With the continuous advancement of information and communications technology, the adoption of communication networks in various systems is expanding. Communication networks provide an effective and practical way of connecting devices across different domains, which enables remote monitoring and control of numerous processes. However, some challenges may be identified in this scenario, particularly the risk of cyber attacks. Despite these concerns, networked control systems have been extensively implemented in industries such as manufacturing, energy, healthcare, transportation. These systems, which integrate physical processes with sensors, actuators, and computing and communication technologies, are so-called *cyber physical systems* (CPS) [49-53].

A Cyber-physical system (CPS) is a mechanism controlled or monitored by computer-based algorithms. Examples of CPS include smart grid, autonomous automobile systems, medical monitoring, process control systems, distributed robotics, and automatic pilot avionics, etc. A general architecture of a CPS is depicted in Fig. 1.1. The communication network is used to collect data from the physical processes via sensors and issue control commands to the actuators. Based on such data, the desired behavior is maintained in the system over the network. This has been raising a major security concern, since different types of cyber

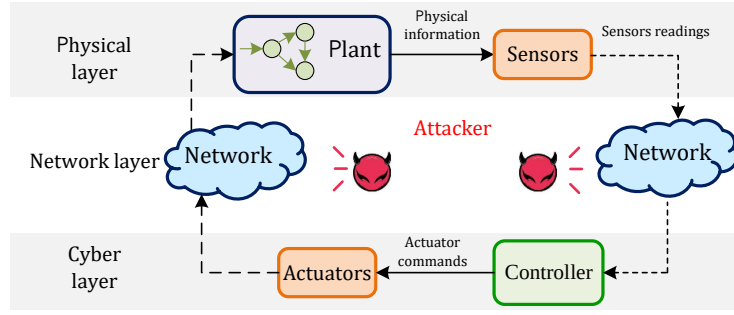


Figure 1.1 General architecture of a CPS.

attacks can tamper the data collection processes and interfere safety critical decision making processes, which may cause irreparable damages to the physical systems being controlled and to users who depend on those systems [54-56].

1.2.2 Related Works on Cyber Attacks in DESs

A CPS can essentially be modeled as a dynamic system containing a continuous state-space, whose physical behavior is usually represented by differential equations. However, depending on the level of observation, a CPS can also be seen as a discrete state-space system that evolves according to events which are asynchronous over time. In this thesis, the CPS is modeled as a discrete event system (DES). Cyber attacks in DESs are defined as actions that exploit the system's vulnerabilities and result in some kind of damage [56]. In the context of DES, cyber attacks can be carried out in three possible manners: (i) attacks in the vulnerable sensor channels, also called "sensor attacks", which affects the plant data; (ii) attacks in the vulnerable control command channels, named "actuator attacks", which affects the data sent to the plant by the controller; (iii) attacks on both sensor and control command channels, known as "sensor-actuator attacks".

There has been a growing number of works addressing the cyber attacks in the security domain of a DES, which provides motivations for our research introduced in this thesis and also provide future research topics that the community may consider. Currently, a major effort on DES cyber security comes from the problem of *state estimation under attacks*, which explore the ability to perform state estimation in the presence of malicious attackers. Based on this, two types of problems are investigated: (i) *attack synthesis*, which provides formulations for successful active attacks from an attacker's point of view; (ii) *attack-tolerant control*, which presents methods to detect the occurrence of active attacks as well as methods to ensure the achievement of the system specification in such situations from the defense viewpoint. A survey of cyber attacks and cyber defence can be found in [57].

(1) State Estimation Under Attack

State estimation is one of the most fundamental problems in the systems and control theory. In many practical systems, it is not always possible to access the complete system information due to measurement noises/uncertainties and the existence of adversaries. Consequently, how to estimate the state of the system is crucial when one needs to make decisions based on the limited and incomplete information.

State estimation in partially discrete event systems (DESSs) has drawn considerable attention over the past decades. The solutions to many important problems in the field of DESSs, such as fault diagnosis [24, 58], opacity [59-60], detectability [61], and supervisory control [62-63], rely on the state estimation of a system. In the setting of partially observed DES, an operator monitors the evolution of a plant in an attempt to estimate the current state of the system. Let the operator be any agent that reads information recorded by a sensor.

Considering that the communication channels is vulnerable, an attacker can perform active attacks on sensor observations in order to alter the state estimation of the operator. In [64], the authors propose a method based on a structure called a joint estimator, which contains all possible attacks in a given attack scenario and the corresponding state estimation. The aim of the proposed solution is to determine whether the attacker leads the plant to a forbidden state. Later, the author proceeds to show how to trim the joint estimator to obtain the supremal stealthy joint subestimator that contains stealthy attacks only [65]. On the other hand, a property called detectability, is proposed in [66], which is used to assess whether the available events are sufficient to accurately determine the system's current state in the case of the presence of active attacks.

(2) Attack Synthesis

An important cybersecurity strategy that has emerged in the literature is to solve *active attack synthesis problems*, i.e., formulations for expected active attacks from the perspective of an attacker. The problem generally assume that the attacker is fully aware of the system model. Although the assumption may seem too strong, the goal is to effectively support the construction of defense methods to protect systems against attacks, given the proposed attack synthesis itself. As for the terms regarding the characteristics of attacks, an attacker is said to be *stealthy* if it remains undetected (at all time instances, including future times) by the controller or any other monitoring mechanism. The other general terms used to describe attackers are *harmful* attackers that inflict damage to the system components.

Usually, an attacker that achieve the two terms are said to be *successful*.

In [67], a game-like automaton called insertion-deletion attack structure is proposed, which captures the interaction between two players (system and attacker). Based on it, the existence of stealthy and harmful (i.e., successful) sensor attacker can be verified, and different successful sensor attack synthesis strategies can also be extracted. This approach was also considered for stochastic DES in [68]. The actuator attack synthesis problem is considered in [69], where it is assumed that the attacker can modify each control command issued by the supervisor under a specified subset of vulnerable controllable events. The synthesis of a successful actuator attack is addressed by reducing it to the standard supervisor synthesis problem using a method with exponential complexity. Using a similar technique, both actuator and sensor attacker synthesis are considered in [70]. By providing a successful attack synthesis, it is possible to make use of such synthesis to achieve an attack-tolerant control against attacks to prevent a target system from being harmed by a specific type of attacks.

(3) Attack-tolerant Control

Before focusing on attack-tolerant control against attacks, the primary objective is recognize the occurrences of active attacks. There are two different approaches on *attack detection* [9, 64, 67, 71-73]. The first approach is to treat each attack as a fault that disrupts event orders in the system by either removing existing events or inserting fictitious events. In this case research focuses firstly on how to decide the existence of an attack in the system based on a finite number of observations, similar to diagnosability verification in fault diagnosis, and secondly on how to ensure sufficient control abilities to mitigate the attack effect, such as shutting down the entire system. In [72], the threat of both sensor attacks and actuator attacks are considered, and a method for detecting such attacks in time to prevent unsafe system behavior is proposed. The second approach is to develop a specific intelligent attack model that checks the stealthiness of an attacker. Under such attack model, the research focuses on analyzing how the attack causes damage to the system without being detected. In [64, 67], an attack detection mechanism is established by determining whether the system operator perceives abnormal system observations.

Instead of merely detecting attacks, most research on cyber-security in DES emphasizes the synthesis of a resilient supervisor against attacks [67, 74-79]. In [75], it is shown that the existence and synthesis of a resilient supervisor can be addressed using classical supervisory control theory without requiring an expansion of the original state space. In [79], a method is introduced for synthesizing a supremal controllable and normal resilient supervisor to

counteract specific bounded sensor attacks. When both actuator attack and sensor attack are considered, algorithms for designing a resilient supervisor are proposed in [80]. In [76], a game theoretical approach, without imposing a normality condition, is employed to develop a supervisor that remains resilient against any sensor attack.

1.3 Contribution of the Thesis

In this thesis, we study fault diagnosis problem under sensor attacks in DES. Note that the sensor attacker is a new agent that interferes in the fault diagnosis problem, which makes this problem different from the fault diagnosis in the presence of imperfect sensors/channels, in which observation delays and losses occur without the action of a malicious agent. We focus on a type of sensor attacks that can intercept information about the operation of the system and insert fake information into the communication network channel between sensors and an operator. The operator is an agent that reads information recorded by a sensor but cannot influence the system operation. The sketch of this thesis is shown in Fig. 1.2, and its organization and main contributions are summarized as follows:

- In Chapter II we give a background on automata and introduce some notations that we will use in the rest of the thesis.
- In Chapter III, a framework for fault diagnosis under attack is established, where a *stealthy joint diagnoser* (SJD for short) is constructed, which (i) describes all possible stealthy attacks (i.e., undiscovered by the operator) in a given attack scenario; (ii) records the joint diagnosis state, i.e., the diagnosis state of the attacker consistent with the original observation and the diagnosis state of the operator consistent with the corrupted observation. The SJD is used for fault diagnosis under attack. From the attacker's point of view, we present two levels of stealthy attackers:
 - one only temporarily degrades the diagnosis state of the operator (called a *weakly harmful attacker*), thereby affecting correct online diagnosis.
 - the other permanently causes damage to the diagnosis state of the operator (called a *harmful attacker*), thereby violating robust diagnosability against attacks.

Necessary and sufficient conditions for the existence of the two levels of attackers are presented. Notice that the construction of the SJD is, in the worst case, exponential with respect to the number of states of G .

- Based on the previous result, Chapter IV aims to find a more efficient arena for diagnosability verification under attack. We construct a deterministic verifier under attack, whose basic idea is to perform a parallel composition between an attack faulty word and an attack normal word, with synchronization on the defined *attack alphabet*. We prove that from the attack faulty and normal words, a pair of ambiguous words that lead to a violation of diagnosability, i.e., a faulty word generated by a system and a normal word perceived by a system operator, can be found. Finally, such a verifier provides a necessary and sufficient condition for robust diagnosability against attacks, which can be verified with polynomial complexity in the size of the system.
- Chapter V concludes the thesis and discuss the potential directions for future work.

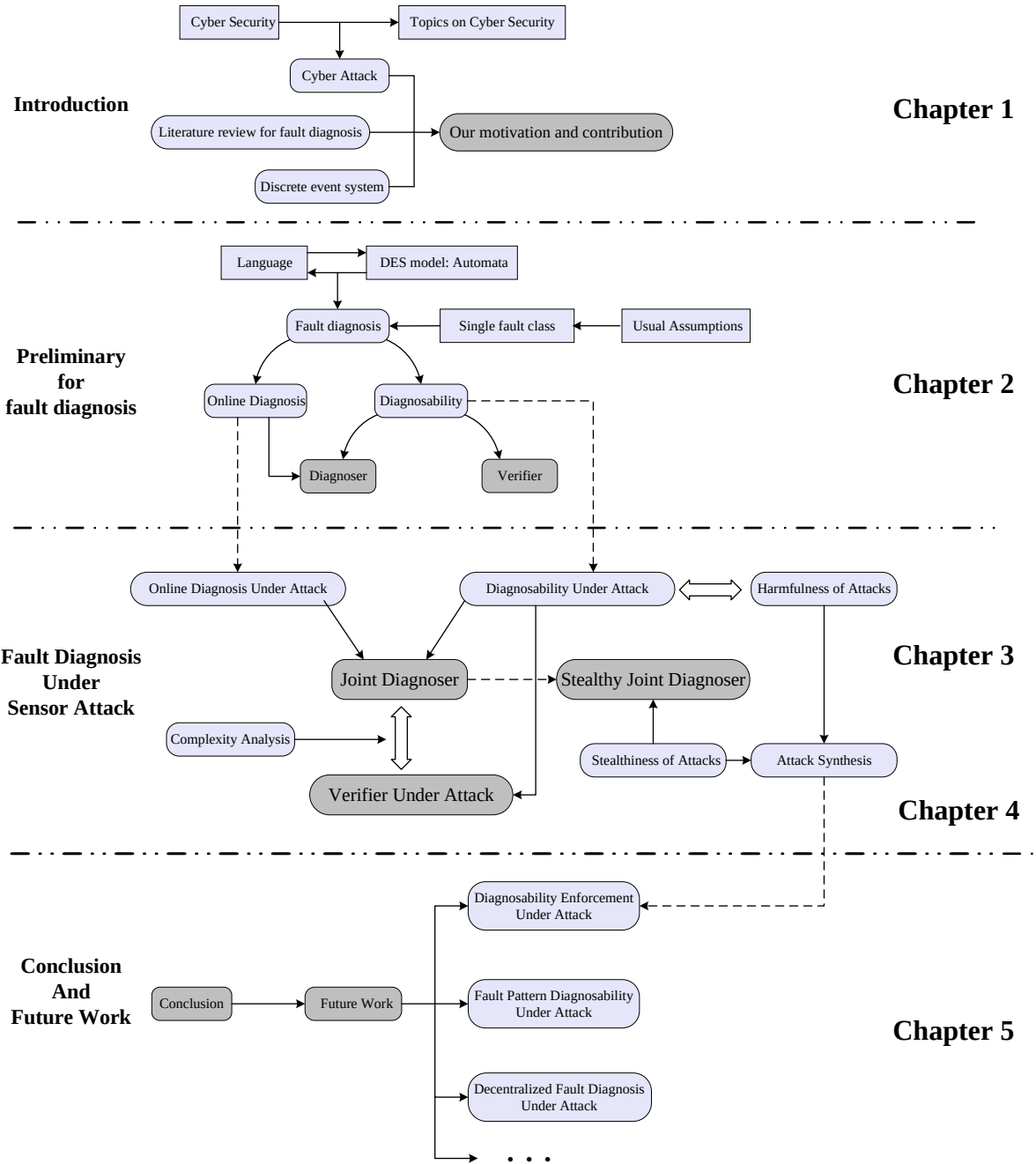


Figure 1.2 Sketch of the thesis.

Chapter II Preliminaries

In this chapter, we introduce notations that will be used throughout this thesis to support discussions on fault diagnosis and related issues in discrete event systems (DES). Given that the thesis primarily deal with DES using regular languages and finite automata, we restrict attention to aspects pertaining to these models. However, the interested reader could explore related extensions to other DES models, including Petri nets. More detailed discussions about the introductory material in this chapter can be found in [7].

2.1 Languages and Finite State Automaton

Let E be a finite alphabet of events. We use E^* to denote the *Kleene closure* of E , consisting of all words over E with finite lengths (including the empty word ε). The *length* of a word $\sigma \in E^*$ is the number of letters in it, denoted by $|\sigma|$; by convention, $|\varepsilon| = 0$. Given two words $u, \sigma \in E^*$, u is a *prefix* of σ , written as $u \leq \sigma$, if there exists $v \in E^*$ such that $uv = \sigma$, where uv denotes the *concatenation* of u and v . The set of the prefixes of σ is denoted by $\bar{\sigma}$. A language $L \subseteq E^*$ is a subset of E^* . The *prefix closure* of L is defined as $\bar{L} = \{u \in E^* \mid (\exists \sigma \in L) u \leq \sigma\}$. If $L = \bar{L}$, then L is called *prefix closed*. Given a word $\sigma \in L$, the *postlanguage of L after σ* is defined as $L/\sigma = \{t \in E^* \mid \sigma t \in L\}$. A language L is said to be *live* if for all $\sigma \in L$ there exists an event $e \in E$ such that $\sigma e \in L$. Given two languages $L_1, L_2 \subseteq E^*$, let $L_1 L_2 = \{\sigma_1 \sigma_2 \mid \sigma_1 \in L_1, \sigma_2 \in L_2\}$ denote their concatenation; moreover, the concatenation of a word σ and a language L is defined as $\sigma L = \{\sigma\}L$.

Example 2.1. Consider an alphabet $E = \{a, b, c\}$. Word $\sigma = abcc$ defined on E has length $|\sigma| = 4$. Its prefixes are ε, a, ab, abc , and $abcc$. The language $L_1 = \{\sigma \in E^* \mid |\sigma| \geq 3\}$ consists of all words of length greater than 3. If $L_2 = \{\varepsilon, a\}$ and $L_3 = \{a, b, ab\}$, then $L_2 L_3 = \{\varepsilon \cdot a\} \cup \{\varepsilon \cdot b\} \cup \{\varepsilon \cdot ab\} \cup \{a \cdot a\} \cup \{a \cdot b\} \cup \{a \cdot ab\} = \{a, b, aa, ab, aab\}$. The word ab can be obtained in two different ways: either concatenating ε with ab or concatenating a with b . The language L_2 is prefix closed, while the language L_3 is not due to $\varepsilon \notin L_3$. $\diamond$

In this thesis, we model a plant using a *deterministic finite state automaton* (DFA for short) $G = (X, E, \eta, \Gamma, x_0, X_m)$, where X is the finite set of states, E is the finite set of events, $\eta : X \times E \rightarrow X$ is the partial transition function, $\Gamma : X \rightarrow 2^E$ is the active event set, $x_0 \in X$ is the initial state, and X_m is the set of marked states. The set of *active events* at state x is defined as $\Gamma(x) = \{e \in E \mid \eta(x, e)!\}$, where “!” means “is defined”. The transition

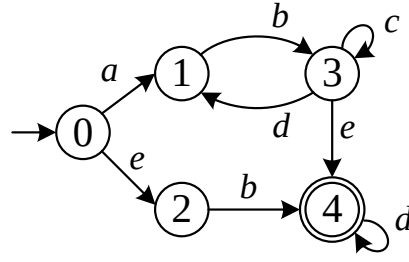


Figure 2.1 A DFA G .

function can be extended to $\eta : X \times E^* \rightarrow X$ by recursively defining $\eta(x, \varepsilon) = x$ and $\eta(x, \sigma e) = \eta(\eta(x, \sigma), e)$ for $\sigma \in E^*$ and $e \in E$. For simplicity, the active event set and the set of marked states are omitted unless otherwise specified. An automaton can be described by a graph in which each state corresponds to a node and is represented by a circle: in particular, the initial state is represented by a circle with an input arrow, and a final state by a double circle. The *language generated by G* , denoted as $L(G)$ or, simply L , is defined as $L(G) = \{\sigma \in E^* \mid \eta(x_0, \sigma) \neq \emptyset\}$. The *marked language* of G is defined as $L_m(G) = \{\sigma \in E^* \mid \eta(x_0, \sigma) \in X_m\}$. Automaton G_1 is said to be a sub-automaton of G_2 if $\eta_1(x_{02}, \sigma) = \eta_2(x_{01}, \sigma)$ for all $\sigma \in L(G_1)$. Let $\mathbb{N}$ denote the set of non-negative integers.

Example 2.2. Fig. 2.1 shows the graphical structure of a DFA $G = (X, E, \eta, \Gamma, x_0, X_m)$ with $X = \{0, 1, 2, 3, 4\}$, $E = \{a, b, c, d, e\}$, $x_0 = 0$, $X_m = \{4\}$. Its transition relation is given by Table 2.1. In this table, the value 1 at the intersection between row 0 and column a denotes that $\eta(0, a) = 1$. An empty box, such as the one at the intersection between row 0 and column c , denotes that the corresponding transition is not defined.

Table 2.1 Transition relation of the DFA in Fig. 2.1

η	a	b	c	d	e
0	1				2
1		3			
2		4			
3			3	1	4
4				4	

Word abd is generated by G because $\eta^*(0, abd) = 1$, but is not accepted because state 3 is not marked. Conversely, the word eb is accepted since $\eta^*(0, eb) = 4$ and 4 is marked. Hence, it is $abd \in L(G)$ and $eb \in L_m(G)$. $\diamond$

The behavior of an automaton is given by all its possible evolutions, characterized by its runs.

Definition 2.1. Given a DFA $G = (X, E, \eta, x_0, X_m)$, we define run of length k a sequence of states and transitions

$$x_1 \xrightarrow{e_1} x_2 \xrightarrow{e_2} \cdots x_{(l-1)} \xrightarrow{e_{l-1}} x_l$$

where: for all $i = 1, \dots, l$ it holds that $x_i \in X$ and for all $i = 1, \dots, l$ it holds that $x_{i+1} = \eta(x_i, e_i)$, i.e., the occurrence of event e_i from state x_i leads to state x_{i+1} . In this case, we say that the run is associated with the word $\sigma = e_1 e_2 \cdots e_{l-1}$. A run is said to be a cycle if $x_l = x_1$. $\diamond$

Example 2.3. A possible run of the DFA in Fig. 2.1 is

$$0 \xrightarrow{a} 1 \xrightarrow{b} 3 \xrightarrow{e} 4 \xrightarrow{d} 4.$$

This run has length 4: it starts from state 0 and produces word $\sigma = abed$ reaching state 4. $\diamond$

In order to analyze automata, we need to have a set of operations on a single automaton in order to modify appropriately its state transition diagram according, for instance, to some language operation that we wish to perform.

Definition 2.2. Given an automaton $G = (X, e, \eta, x_0, X_m)$, the accessible part, which excludes all states that cannot be reached from the initial state, is defined as $Ac(G) = (X_{Ac}, E, \eta_{Ac}, x_0, X_{m,Ac})$, where $X_{Ac} = \{x \in X \mid (\exists \sigma \in E^*) [\eta(x_0, \sigma) = x]\}$, $\eta_{Ac} : \eta|_{X_{Ac} \times E \rightarrow X_{Ac}}$ is the restriction of η to domain $X_{Ac} \times E$, and $X_{m,Ac} = X_m \cap X_{Ac}$. $\diamond$

Definition 2.3. Given an automaton $G = (X, e, \eta, x_0, X_m)$, the coaccessible part of G removes those states from which it is not possible to reach any marked state, being defined as $CoAc(G) = (X_{CoAc}, \Sigma, \eta_{CoAc}, x_0, CoAc, X_m)$, where $X_{CoAc} = \{x \in X \mid (\exists \sigma \in E^*) [\eta(x, \sigma) \in X_m]\}$, $\eta_{CoAc} = \eta|_{X_{CoAc} \times E \rightarrow X_{CoAc}}$ is the restriction of η to domain $X_{CoAc} \times E$, and $x_0, CoAc = x_0$ if $x_0 \in X_{CoAc}$; undefined, otherwise. $\diamond$

We also need to define operations that allow to compose two or more automata, so that models of complete systems could be built from models of individual components.

Definition 2.4. Given two automata $G_1 = (X_1, E_1, \eta_1, x_{01})$ and $G_2 = (X_2, E_2, \eta_2, x_{02})$, their parallel composition synchronizes the common event and allows the private event to happen without constraints, being defined as $G_1 \parallel G_2 = Ac(X_1 \times X_2, E_1 \cup E_2, \eta, (x_{01}, x_{02}))$, where $\eta[(x_1, x_2), e] = (x'_1, x'_2)$ if $\eta_1(x_1, e) = x'_1$ and $\eta_2(x_2, e) = x'_2$; $\eta[(x_1, x_2), e] =$

(x'_1, x_2) if $\eta_1(x_1, e) = x'_1$ and $e \notin E_2$; $\eta[(x_1, x_2), e] = (x_1, x'_2)$ if $\eta_2(x_2, e) = x'_2$ and $e \notin E_1$; undefined, otherwise. $\diamond$

Definition 2.5. Given two automata $G_1 = (X_1, E_1, \eta_1, x_{01})$ and $G_2 = (X_2, E_2, \eta_2, x_{02})$, the product of G_1 and G_2 models their synchronized behavior, being defined as $G_1 \times G_2 = Ac(X_1 \times X_2, E_1 \cup E_2, \eta, (x_{01}, x_{02}))$, where $\eta[(x_1, x_2), e] = (x'_1, x'_2)$ if $\eta_1(x_1, e) = x'_1$ and $\eta_2(x_2, e) = x'_2$; undefined, otherwise. $\diamond$

The main difference between the two operations is how the private events, i.e., the events not in $E_1 \cap E_2$, are handled. In the product, the transitions of the two automata must always be synchronized on a shared event, $e \in E_1 \cap E_2$. In other words, an event in the product occurs iff it occurs in both automata. In the synchronization, the two automata are still synchronized on the shared events but the private events can independently be executed whenever possible.

2.2 Observation Models and Observer Constructions

Given two sets $E_l, E_s \subseteq E$ with $E_s \subseteq E_l$, the natural projection $P : E_l^* \rightarrow E_s^*$ is defined as

$$P(\varepsilon) = \varepsilon \text{ and } P(\sigma e) = \begin{cases} P(\sigma)e, & \text{if } e \in E_s; \\ P(\sigma), & \text{if } e \in E_l \setminus E_s. \end{cases} \quad (2-1)$$

The projection can be extended to a language L by applying $P_o(\sigma)$ to all words $\sigma \in L$. The inverse projection $P^{-1} : E_s^* \rightarrow 2^{E_l^*}$ is defined as, for $t \in E_s^*$, $P^{-1}(t) = \{\sigma \in E_l^* \mid P_o(\sigma) = t\}$.

A plant G is partially observable in general. The set of observable events is denoted by $E_o \subseteq E$, while that of unobservable events is denoted by $E_{uo} = E \setminus E_o$. In particular, the observation projection $P_o : E^* \rightarrow E_o^*$ erases the unobservable events from a word in E^* . In practice, an unobservable event can be made observable by adding a sensor that records its occurrence. Given a word $\sigma \in E^*$, its observable projection $s = P_o(\sigma)$ is the output of the natural projection $P_o : E^* \rightarrow E_o^*$.

Example 2.4. Let $E_o = \{a, b, d, e\}$ and $E_{uo} = \{c, f\}$. Assuming $L = \{a, abc, abdf, acfe\}$, we have $P_o(L) = \{a, ab, abd, ae\}$. Note that $P_o^{-1}(P_o(L)) = P_o^{-1}(\{a, ab, abd, ae\}) \supset L$. $\diamond$

Given a system $G = (X, E, \eta, x_0)$ and an observation $s \in P_o(L(G))$, the state estimate associated with s is defined as

$$\mathcal{C}(s) := \{x \in X \mid \exists \sigma \in E^* [\eta(x_0, \sigma) = x, P_o(\sigma) = s]\}$$

i.e., the set of states in which the system can be given the observation s . The problem of calculating $\mathcal{C}(w)$ is typically referred to as the *state estimation problem*. To solve it, the *observer* automaton of G needs to be constructed. The procedure to build the observer is detailed in [7]. In simple words, the observer of G itself is an automaton. We denote it as $Obs(G) = (B, E_o, \eta_o, b_0)$, where $B \subseteq 2^X$. Each state $b \in B$ of the observer is a set $b \subseteq X$ of states, and coincides with the set of states consistent with the observation leading to state b starting from b_0 . More precisely, given an observation $s \in P_o(L(G))$ of G , its state estimate is given by $\mathcal{C}(s) = \eta_o(b_0, s)$. By construction, it holds that $L(Obs(G)) = P_o(L(G))$.

Example 2.5. Fig. 2.2(a) shows a plant G , where $E_o = \{a, b, d, e\}$ and $E_{uo} = \{c, f\}$. Its observer is shown in Fig. 2.2(b). If $s = abd$ is observed, looking at the observer, we immediately compute $\mathcal{C}(s) = \{4\}$. $\diamond$

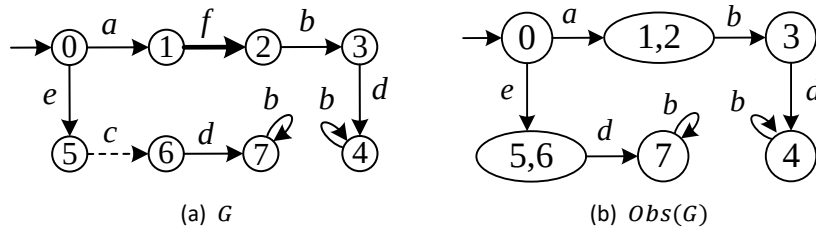


Figure 2.2 (a) A plant G and (b) its observer $Obs(G)$.

2.3 Fault Diagnosis of DES

We present, in this section, two important notions related to fault diagnosis: (i) diagnosability, which is a system property that ensures that all fault occurrences can be diagnosed; (ii) online diagnosis, which corresponds to the actual detection of the fault occurrence.

2.3.1 Language Diagnosability

In the literature of fault diagnosis in DESs, the unobservable event set E_{uo} is further partitioned into the set of *regular unobservable events* E_{ruo} and the set of *fault events* E_f . In this thesis, we consider a single class of fault for simplicity, but all the proposed results can be extended in a straightforward way to the case of multiple fault classes. Let $\Psi(E_f) = \{\sigma f \in L \mid \sigma \in E^*, f \in E_f\}$ denote the set of all finite words in L that end with a fault event f . With a slight abuse of notation, $E_f \in \sigma$ denotes that $\bar{\sigma} \cap \Psi(E_f) \neq \emptyset$. A word

σ is said to be *faulty* (resp., *normal*) if $E_f \in \sigma$ (resp., $E_f \notin \sigma$). The language containing all normal words is denoted by $L_N \subseteq L$. The following usual assumptions are also required:

A1) The language of G is live;

A2) There is no cycle of unobservable events in G .

Assumption A1) is made for the sake of simplicity. Assumption A2) ensures that the system G does not generate sequences of unobservable events whose length can be infinite.

Let us consider the DES modelled by the DFA $G = (X, E, \eta, x_0)$ depicted in Fig. 2.2 (a), where $E = \{a, b, c, d, e, f\}$, with $f \in E_f$ being the fault event whose occurrence must be diagnosed. The language generated by automaton G is $L(G) = \overline{(afb + ec)bd^*}$. Assume that $E_o = \{a, b, d, e\}$ and $E_{uo} = \{c, f\}$. We find that the occurrence of the fault event f can be detected since there does not exist any normal word σ_N that has the same projection as $\sigma_Y = afbdb^n$, i.e., $P_o(\sigma_N) = P_o(\sigma_Y) = abdb^n$. In plain words, the detection of the fault event requires that $P_o(\sigma_Y)$ should become different from $P_o(\sigma_N)$ after a finite number of event occurrences once the fault occurs. The above discussions lead to the following definition of language diagnosability [24]. In words, diagnosability guarantees that the occurrence of a fault event can be detected with certainty in a finite number of transitions after its occurrence.

Definition 2.6. [24] A live language L is diagnosable w.r.t. $P_o : E^* \rightarrow E_o^*$ and E_f if

$$(\exists n \in \mathbb{N})(\forall \sigma \in \Psi(E_f))(\forall t \in L/\sigma) [|t| \geq n \Rightarrow \mathbf{C}_D]$$

where $\mathbf{C}_D$ is the diagnosability condition, defined as

$$(\nexists \sigma' \in L_N) [P_o(\sigma t) = P_o(\sigma')].$$

◊

2.3.2 Online Diagnosis

Assuming that the language generated by a DFA G is diagnosable with respect to P and E_f , a problem of interest is how to diagnose the occurrence of f in real time, i.e., how to perform online diagnosis of DES. Based on the observation $s \in E_o^*$, it is desirable to online determine if a fault has already occurred or not. To this end, one wishes to build a *diagnosis function* $\gamma : E_o^* \rightarrow \{N, F, U\}$ associating to each observation a diagnosis state, such that

$$\gamma(s) = \begin{cases} N, & \text{if } \forall \sigma \in P^{-1}(s), E_f \notin \sigma; \\ F, & \text{if } \forall \sigma \in P^{-1}(s), E_f \in \sigma; \\ U, & \text{otherwise.} \end{cases}$$

An observation s is called: *normal* if $\gamma(s) = N$ since no word producing s contains a fault; *faulty* if $\gamma(s) = F$ since all words producing s contain a fault; *ambiguous* otherwise.

A standard approach to compute the diagnosis function is by using a *diagnoser* that captures diagnostic information in its state, shown in [24]. The idea behind the construction of the diagnoser automaton is to label the states of G with labels $F(N)$ in order to indicate if the fault event has (respectively, has not) occurred, and, based on the occurrences of observable events only, estimate the current state of G together with the information regarding the fault event occurrence. Formally, the diagnoser $Diag(G) = (X_d, E_o, \eta_d, x_{d,0})$ is defined as

$$Diag(G) = Obs(G_\ell) = Obs(G \parallel A_\ell) \quad (2-2)$$

where G_ℓ is the *fault recognizer* obtained by the parallel composition of G and A_ℓ , in which the *label automaton* is $A_\ell = (X_{A_\ell}, E_f, \eta_{A_\ell}, x_{0,A_\ell})$, where $X_{A_\ell} = \{N, F\}$, $x_{0,A_\ell} = \{N\}$, $\eta_{A_\ell}(N, f) = F$ and $\eta_{A_\ell}(F, f) = F$ with $E_f = \{f\}$, shown in Fig. 2.3. It is worth remarking that since $E_f \subset E$, this operation is language preserving, and so $L(G_\ell) = L(G)$.

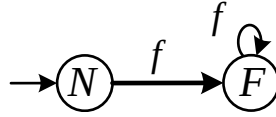


Figure 2.3 Label automaton A_ℓ on alphabet E_f .

State $x_d \in X_d$ is in the form $x_d = \{(x_1, \ell_1), \dots, (x_n, \ell_n)\}$, where $x_i \in X$ and $\ell_i \in \{N, F\}$ for $i = 1, \dots, n$. The diagnoser allows one to associate every state to a diagnosis value $\gamma(x_d) = \gamma(s)$, where $x_d = \eta_d(x_{d,0}, s)$. The diagnoser state x_d is *negative* if $\gamma(x_d) = N$; *positive* if $\gamma(x_d) = F$; *uncertain* if $\gamma(x_d) = U$.

Remark 2.1. Notice that, due to Assumption A1, although the language generated by G is live, the language generated by $Diag(G)$ may or may not be live. Indeed, from Eq. (2-2), we have $L(Diag(G)) = P(L(G_\ell)) = P_o(L(G))$, and so, all suffixes formed with sequences in E_{u0}^* disappear when the language projection is computed.

Remark 2.2. Since the diagnoser construction is based on the computation of an observer automaton, it has exponential worst case computational complexity, i.e., $O(|X_d|) = 2^{2|X|}$, where $|X|$ denotes the cardinality of the set of states of G .

2.3.3 Diagnosability Verification

This section considers the problem of diagnosability verification, i.e., how to verify if the language generated by G is diagnosable with respect to P and E_f . Now we present two

tests for diagnosability verification. The first one builds on the above-defined diagnoser, while the second diagnosability verification test employs the so-called verifier automaton [38-42].

(1) Diagnoser-based Approach

We first show that the diagnoser, which provides a solution to the diagnosis problem, can also be a useful tool to solve the diagnosability problem.

Definition 2.7. Given a diagnoser $Diag(G)$, a cycle

$$x_{d,1} \xrightarrow{e_1} x_{d,2} \xrightarrow{e_2} \dots \xrightarrow{e_{l-1}} x_{d,l} \xrightarrow{e_l} x_{d,1}$$

is called an *uncertain cycle* if all its states are uncertain, i.e., $\gamma(x_{d,i}) = U$ for $i = 1, 2, \dots, l$.

As a preliminary result, we can now state a sufficient condition for diagnosability.

Proposition 2.1. A DFA G is diagnosable if its diagnoser does not contain uncertain cycles.

Example 2.6. Consider the diagnoser $Diag(G)$ shown in Fig. 2.4 for G in Fig. 2.2(a). Since it does not contain uncertain cycles, we conclude that G is diagnosable. $\diamond$

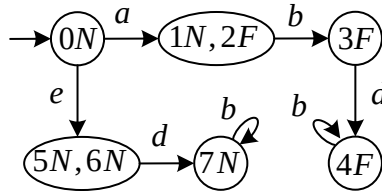


Figure 2.4 Diagnoser $Diag(G)$ for G in Fig. 2.2(a).

This sufficient condition for diagnosability is not necessary as shown in Example 2.7.

Example 2.7. Fig. 2.5(a) shows a plant G , where $E_o = \{a, b\}$ and $E_{uo} = E_o = \{f\}$. In its diagnoser in Fig. 2.5(b), there exists in the diagnoser a cycle of uncertain states $\{0N, 2F\} \xrightarrow{a} \{1N, 2F\} \xrightarrow{b} \{0N, 2F\}$. However one can verify by inspection that this DFA is diagnosable. In fact, after a fault occurs state 2 is reached. Since the only active event in state 2 is a , one will eventually observe two consecutive a^2 which cannot be generated by the fault-free system and the fault is detected. To be more precise, we observe that state 2 can only be reached by a faulty string $\sigma = vf$ and two choices are possible for such a v .

(a) If $v = (ab)^k$, after two more events occur the generated string becomes $\sigma = (ab)^k f aa$ whose projection is $s = P_o(\sigma) = (ab)^k aa$.

(b) If $v = (ab)^k a$, after one more event occurs the generated string becomes $\sigma = (ab)^k a f a$ whose projection is again $s = P_o(\sigma) = (ab)^k a a$.

In both cases, as soon as one observes two consecutive a^2 the fault is detected. $\diamond$

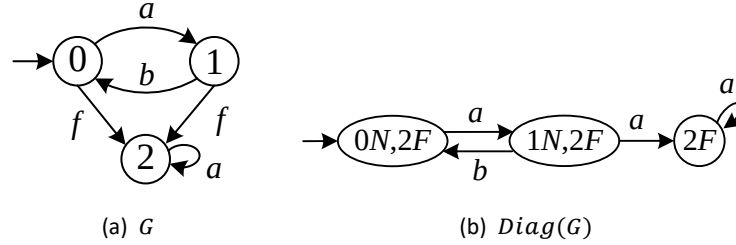


Figure 2.5 (a) A plant G and (b) its diagnoser $Diag(G)$.

To derive a general necessary and sufficient condition for diagnosability, we need to introduce the notion of *indeterminate cycles* that cause a violation of diagnosability. If there is a cycle formed with uncertain states where the diagnoser can remain forever, this cycle is indeterminate; on the other hand if somehow it always leaves this cycle of uncertain states, this cycle is not indeterminate. Therefore, it is important to distinguish between uncertain cycles that are indeterminate and uncertain cycles that are not indeterminate. This requires, besides the analysis of cycles in the diagnoser, the search for cycles in G . These cycles in G capture the existence of two words, with identical projections, where one word is normal while the second one contains an arbitrarily long suffix after the fault event, thereby providing a counter-example to diagnosability.

Theorem 2.1. [24] A DFA G is diagnosable with respect to projection P and E_f if and only if its diagnoser $Diag(G)$ has no indeterminate cycles.

Example 2.8. Fig. 2.6(a) shows a plant G , where $E_o = \{b, d, e\}$, $E_{uo} = \{c, f\}$ and $E_f = \{f\}$. Since its diagnoser in Fig. 2.6(b) contains an indeterminate cycle $\{3F, 6N\} \xrightarrow{b} \{3F, 6N\}$, we conclude that G is not diagnosable. $\diamond$

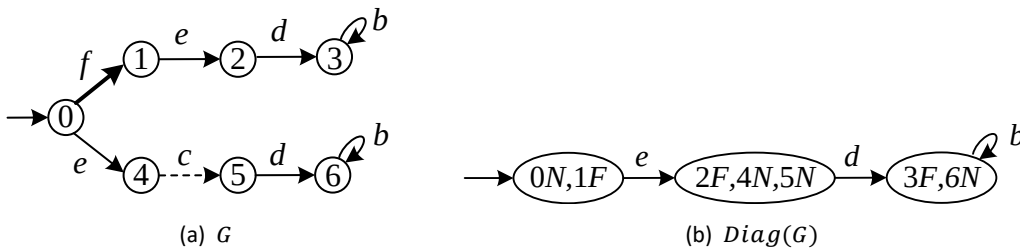


Figure 2.6 (a) A plant G and (b) its diagnoser $Diag(G)$.

(2) Verifier-based Approach

Another way to verify diagnosability is through the so-called verifier automaton, initially introduced in [38-39] and later revisited in [40-41]. The basic idea is to seek ambiguities (words in the language generated by the system automaton with identical projections with respect to E_o^*) between normal and faulty words. If such ambiguity persists indefinitely for some faulty word, the language is not diagnosable. Conversely, if for all faulty words, the ambiguity stops after a finite number of observable events, the language is diagnosable.

In this section, we present the diagnosability verification test proposed in [41]. The reason for choosing this approach is that the verifier in [41] is deterministic and its construction relies solely on standard automata operations such as parallel composition and renaming of transition labels. Let us begin by defining automaton V as follows:

$$V = G_N || G_F \quad (2-3)$$

where $G_N = (X_N, E_N, \eta_N, x_{0_N})$ and $G_F = (X_F, E, \eta_Y, x_{0_N})$ are subautomata of automaton G_ℓ , whose generated language is formed only with normal and faulty words, respectively, and can be obtained as follows:

- G_N is formed by removing the transitions of G_ℓ labeled by fault event f and taking the accessible part of the resulting automaton; thus, $E_N = E \setminus E_f$.
- G_F is obtained by first marking all states x^F of G_ℓ , then taking the coaccessible part of the resulting automaton and, finally, unmarking the marked states.

We refer to automaton G_N as the *system normal behaviour automaton*, and to G_F as the *system faulty behaviour automaton*. Note that automata G_N and G_F share the same set of events, and so all of their events are synchronized, which does not allow us to distinguish between normal and faulty words in the search for possible ambiguities. Since ambiguity is caused by unobservable events, it is essential to find a method to differentiate normal from faulty words. In [41], such a distinction is achieved by renaming the unobservable events that occurs in G_N . The renaming function R is defined as follows. Let $E'_R = \{e_R \mid e \in E_{uo} \setminus E_f\}$, $E_R = E_o \cup E'_R$ and define $E_N = E \setminus E_f$. Then, function $R : E_N \rightarrow E_R$, where

$$R(e) = \begin{cases} e, & \text{if } e \in E_o \\ e_R, & \text{if } e \in E_{uo} \setminus E_f \end{cases}$$

Notice that $R(\sigma e) = R(\sigma)R(e)$, for all $\sigma \in E^*$ and $e \in E$. Function R just renames the labels of the events in $E_{uo} \setminus E_f$. The following verifier automaton is constructed:

$$G_V = G_N^R \parallel G_F, \quad (2-4)$$

where $G_N^R = (X_N, E_R, \eta_N^R, x_{0N})$, with $\eta_N^R(x, R(e)) = \eta_N(x, e)$ for all events e with $\eta_N(x, e) \neq !$. We refer to automaton G_N^R as the *system normal behaviour renamed automaton*. Notice that $G_V = (X_V, E_V, \eta_V, (0N, 0N))$, where, $E_V = E \cup E_R$ is synchronized only by the observable events, leaving the unobservable events (renamed or not) to occur independently. Finally, diagnosability verification can be carried out using verifier G_V .

Theorem 2.2. [41] *A plant G is diagnosable if and only if the verifier G_V does not contain a cycle $cl = x_v^k e_k x_v^{k+1} e_{k+1} \dots x_v^l e_l x_v^k$ with $l \geq k > 0$, being of the form $x_v^k = (x_f^k, \bar{x}_n^k) = ((x^k, \ell^k), (\bar{x}^k, N))$, satisfying the following condition:*

$$(\exists j \in \{k, k+1, \dots, l\}) [\ell^j = F \wedge e_j \in E]. \quad (2-5)$$

Example 2.9. Consider again the plant in Fig. 2.2(a). Fig. 2.7(c) shows its verifier automaton G_V , which is the parallel composition of G_F in Fig. 2.7(a) and G_N^R in Fig. 2.7(b). Notice that since G_V has no cycles satisfying condition (2-5), we can conclude, as in the verification of language diagnosability using diagnosers in Example. 2.6, that G is diagnosable. $\diamond$

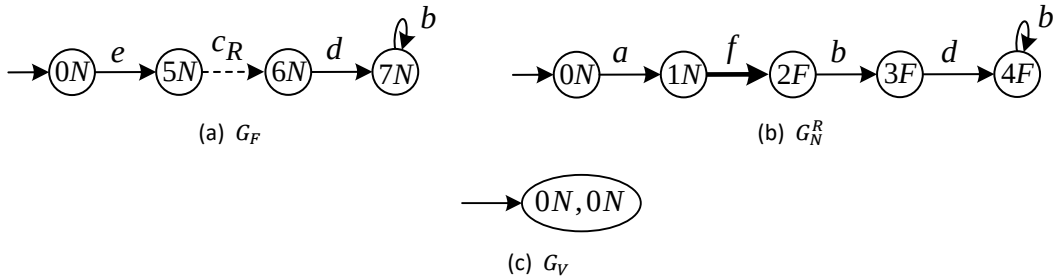


Figure 2.7 (a) G_N^R , (b) G_F , and (c) G_V for G in Fig. 2.2(a).

Chapter III Fault Diagnosis Under Attack: Using Diagnoser

3.1 Introduction

Over the past decade, there have been some works on *robust fault diagnosis* against imperfect sensors/channels, in which observation delays and losses occur without the action of a malicious agent; see [44-47, 81-85]. Such disruptions are endogenous in systems, which may be caused by component aging degradation, measurement noise, etc. For a survey of the work in this domain, we refer to [85]. The situation is usually worse when exogenous disruptions are taken into account, especially when produced by malicious attackers, which are sufficiently intelligent to actively choose their actions for malicious purposes. In our framework, the attacker is a new agent that interferes in the fault diagnosis problem, which makes this problem more challenging than the fault diagnosis under imperfect sensors/channels.

Indeed, the increasing number of networked components in systems possibly introduce vulnerabilities to cyber-attacks. In this chapter, we consider a special class of cyber-attacks, called *sensor deception attacks*, that may obtain access to and compromise the transmitted signals between sensors attached to a system and an operator that monitors the system (see Fig. 3.1 for a sketch). Hence, the operator may be misled to take incorrect actions based on the corrupted information, which is our considered attack mechanism.

Recent works have explored sensor attacks in the context of attack detection [72], state estimation [64-65], and supervisory control [67]. However, very few works have been devoted to diagnosability verification under attack. Two recent contributions in this context are [86-87] and [88-89]. In these works, from *defense viewpoint*, the notion of *robust diagnosability against attacks* is proposed, which requires that the system language can remain diagnosable even in the case of attacks.

In this chapter, we develop a novel framework for diagnosis and diagnosability verification under attack. We take the perspective of the attacker. If an attacker may make a system no longer diagnosable, it is said to be *harmful*. This occurs when the attacker is certain of the fault occurrence, while due to the attack, the operator is always not. In this context, we say that the attacker permanently conceals the fault occurrence to the operator. In the relaxed case, if the attacker might temporarily conceal the fault occurrence at some points after attacks occur, it is said to be *weakly harmful*. On the other hand, *stealthiness* is an-

other crucial property of an attacker, ensuring that its attacks remain undiscovered by the operator during system execution. To check stealthiness, we establish an attack detection mechanism by determining whether the operator perceives abnormal system observations.

From the *attack viewpoint*, we aim to verify the existence of attackers that can achieve harmfulness and stealthiness. To this end, a bipartite diagnoser called *a joint diagnoser (JD)* is constructed, which captures all possible attacks in a given attack scenario. We prove that the joint diagnoser shows the joint diagnosis state for both the attacker (based on the original observation) and the operator (based on the corrupted observation). As a result, the joint diagnoser provides a necessary and sufficient condition for the existence of a harmful attacker. In particular, we show that in our approach, diagnosability under attack can be verified by studying certain cycles in the joint diagnoser, without the analysis of an indeterminate cycle as was the case for the classical diagnoser-based approach in [24].

The joint diagnoser also allows to determine if an attacker may actively make stealthy choices in its attack strategies. To capture stealthy attacks only, we present a refined joint diagnoser, i.e., *stealthy joint diagnoser (SJD)* that is used to check the existence of a stealthy and harmful attacker. This is a distinguishing feature of our work as compared with prior works [86-89]. In [86-87], the attacker's stealthiness is not considered. In [88-89], attack detection relies on the fault diagnosis technique by passively modeling an attack as a fault behavior under certain assumptions. Considering the distinction between faults and attacks, as faults can be treated when production stops but attacks need to be circumvented as soon as they are detected to prevent the attacker from taking full control of the process, we aim to employ a technique to detect attacks differently from faults.

Furthermore, we study the attack synthesis problem with respect to fault diagnosability. The methodology developed to synthesize a suitable attack strategy is inspired by the work in [64-65, 67]. In [67], a game-like automaton called insertion-deletion attack structure is proposed, from which different stealthy attack strategies that induce a plant into a forbidden state can be derived. In [64-65], the authors propose a structure called a stealthy joint estimator to capture how an attacker can select the possible suitable choices to affect the state estimate of an operator. As in these works, we use the proposed SJD to model the game-like interaction between the system and the attacker. Successful attack strategies can be extracted from the SJD. Our focus on attacks on diagnosability differentiate our results from [64-65, 67].

3.2 Attack Model

We consider an attacker that can manipulate the data transmitted over the network by intercepting a subset of the sensor network channels. More precisely, it may implement two types of sensor network attacks:

- *Sensor Erasure attack (SE-attack)*: erase some readings generated by the plant.
- *Sensor Insertion attack (SI-attack)*: insert some fake readings that have not occurred in the plant.

Considering a system modeled as an automaton, we denote by $E_{era} \subseteq E_o$ (resp., $E_{ins} \subseteq E_o$) the set of events subject to SE-attacks (resp., SI-attacks), i.e., the occurrence of events in E_{era} (resp., E_{ins}) can be erased (resp., inserted). To be more general, no relation is imposed between the sets E_{era} and E_{ins} . We define $E_{com} = E_{era} \cup E_{ins}$ as the *compromised event set*. Fig. 3.1 illustrates the architecture of a fault diagnosis system under attack, where the shadowed block denotes a sensor attacker that intervenes in the communication channels between the sensor and the system operator. If a plant generates a word $\sigma \in E^*$, the attacker observes an *original observation* $s = P_o(\sigma)$ and then produces a *corrupted observation*, denoted as s' . The operator diagnoses the system based on the received corrupted observation s' , i.e., the diagnosis state $\gamma(s')$ is computed.

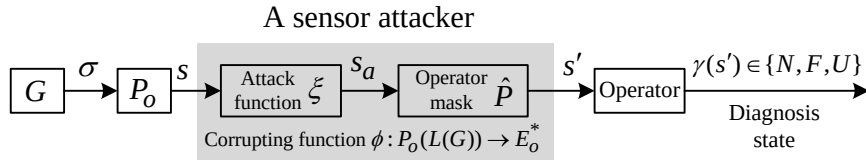


Figure 3.1 A fault diagnosis system under attack.

To identify the insertion or erasure of an event, we define $E_+ = \{e_+ \mid e \in E_{ins}\}$ as the set of *inserted events* and $E_- = \{e_- \mid e \in E_{era}\}$ as the set of *erased events*. The occurrence of event e_+ (resp., e_-) denotes the fact that the attacker inserts event $e \in E_{ins}$ that has not occurred in the plant (resp., event $e \in E_{era}$ has been erased by the attacker). Note that E_o, E_+ , and E_- are disjoint sets. We also define $E_a = E_o \cup E_+ \cup E_-$ as the *attack alphabet*.

Definition 3.1. Given a plant G with set $E_{com} = E_{era} \cup E_{ins}$, an attack function $\xi : P_o(L(G)) \rightarrow E_a^*$ satisfies:

- 1) $\xi(\varepsilon) \in E_+^*$;

2) $\forall se \in P_o(L(G))$ with $s \in E_o^*$ and $e \in E_o$:

$$\begin{cases} \xi(se) \in \xi(s)\{e\}E_+^*, & \text{if } e \in E_o \setminus E_{era}; \\ \xi(se) \in \xi(s)\{e_-, e\}E_+^*, & \text{if } e \in E_{era}. \end{cases} \quad \diamond$$

Statement 1) in Definition 3.1 implies that an arbitrary word $t \in E_+^*$ may be initially inserted. By Statement 2), the attacker cannot erase e if e is outside of E_{era} , but can insert an arbitrary word $t \in E_+^*$ after e . If an event $e \in E_{era}$ occurs, the attacker may either erase e or leave it intact, and then insert any arbitrary word $t \in E_+^*$. Note that the attack function ξ is deterministic. We define the set of all possible attack functions depending on sets E_{era} and E_{ins} as $\Xi(E_{era}, E_{ins})$ (abbreviated as Ξ).

An attack function ξ determines an *attack language*, denoted as $L(\xi, G) = \{\xi(s) \mid s \in P_o(L(G))\}$ (abbreviated as $L(\xi)$). Let $s_a \in L(\xi)$ be an *attack word*. Given an attack word s_a , to capture the corrupted observation s' perceived by the operator, an *operator mask* $\hat{P} : E_a^* \rightarrow E_o^*$ is defined as

$$\hat{P}(\varepsilon) = \varepsilon, \hat{P}(s_a e') = \begin{cases} \hat{P}(s_a)e, & \text{if } e' = e \in E_o \vee e' = e_+ \in E_+; \\ \hat{P}(s_a), & \text{if } e' = e_- \in E_- . \end{cases}$$

The operator mask can be extended to the language $L(\xi)$ by applying $\hat{P}(s_a)$ to all words $s_a \in L(\xi)$. We now define two diagnosis functions on E_a as follows. The *attacker diagnosis function* $\gamma_{att} : E_a^* \rightarrow \{N, F, U\}$ is defined as $\gamma_{att}(s_a) = \gamma(s)$, where $s_a = \xi(s)$; the *operator diagnosis function* $\gamma_{opr} : E_a^* \rightarrow \{N, F, U\}$ is defined as $\gamma_{opr}(s_a) = \gamma(s')$, where $s' = \hat{P}(s_a)$.

Definition 3.2. A corrupting function $\phi : P_o(L(G)) \rightarrow E_o^*$ is defined as $\phi(s) = \hat{P}(\xi(s))$, where $\xi : P_o(L(G)) \rightarrow E_a^*$ and $\hat{P} : E_a^* \rightarrow E_o^*$ are the attack function and the operator mask, respectively. $\diamond$

In this thesis, a sensor attacker is characterized by a corrupting function that takes an original observation s as input and produces a corrupted observation $s' = \phi(s) = \hat{P}(\xi(s))$ as output, as shown in Fig. 3.1. We define $L(\phi, G) = \hat{P}(L(\xi, G))$, abbreviated as $L(\phi)$, as the *corrupted language* induced by function ϕ . We also define the set of all possible corrupting functions that depend on E_{era} and E_{ins} as $\Phi(E_{era}, E_{ins})$ abbreviated as Φ .

3.3 Active Attacks on Fault Diagnosis

We start this section with a motivating example to show that, in the case of sensor attacks, the attacker may actively degrade the estimation performance.

Example 3.1. Recall the plant and its diagnoser in Example 2.5, which are again illustrated in Figs. 3.2(a) and 3.2(b), respectively. In the nominal setting, if word $\sigma = afbdb^k$ ($k \in \mathbb{N}$) occurs, then the diagnoser $Diag(G)$ goes from the initial state $\{0N\}$ to $\{4F\}$.

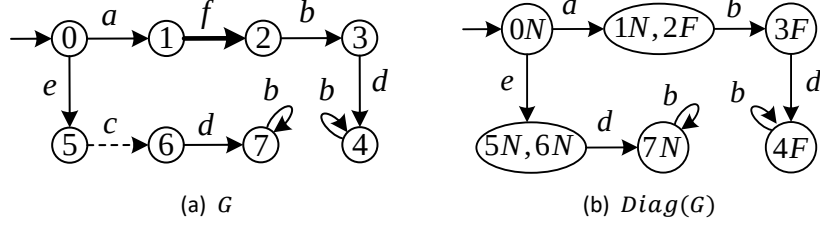


Figure 3.2 (a) A plant G and (b) its diagnoser $Diag(G)$.

Now assume that the sensors that record the occurrence of a and b are subject to SE-attacks, and the sensor that records the occurrence of e is subject to SI-attacks, i.e., $E_{era} = \{a, b\}$ and $E_{ins} = \{e\}$. We consider an attacker ϕ with the following attack function

$$\xi(s) = \begin{cases} \varepsilon, & \text{for } s = \varepsilon; \\ a_b_e_db^k, & \text{for } s = abdb^k, k \in \mathbb{N}. \end{cases}$$

The attacker first erases event a and b , and then inserts event e while observing $abdb^k$. Under such an attack, the first event occurrence to be recognized by $Diag(G)$ is the inserted event e , which takes the diagnoser state to $\{5N, 6N\}$. When the next event of σ , i.e., event d occurs, the diagnoser moves to state $\{7N\}$, where it stays as long as event b continues to occur, therefore displaying wrong information regarding the fault occurrence. This false negative shows that the diagnosis state based on the original observation $s = abdb^k$ may not coincide with the diagnosis state based on the corrupted observation $s' = \hat{P}(\xi(abdb^k)) = edb^k$. $\diamond$

3.3.1 Problem Statement

Before formalizing the above discussion, in addition to A1 and A2, we make the following assumption: A3) The language of G is diagnosable in the nominal setting, i.e., when no attacks occur. Example 3.1 motivates the notion of *weakly harmful attackers*.

Definition 3.3. An attacker ϕ for a language L is said to be *weakly harmful* if there exists an observation $s \in P_o(L)$ with $\gamma(s) = \{F\}$ which may be corrupted into another observation $s' = \phi(s)$ and $\gamma(s') \in \{N, U\}$. $\diamond$

As defined, a weakly harmful attacker can degrade the diagnoser performance in such

a way that a faulty observation s that allows the detection of a fault is altered into a normal or ambiguous observation s' received by the operator, which corresponds to the absence of the fault or to the uncertain situation, respectively. Such an attacker introduces a delay in the detection of the fault, i.e., it makes a fault temporarily hidden from the operator. Now we formalize the *Weakly Harmful Attacker Existence Problem*.

Problem 3.1. *Given a language L , determine whether there exists a weakly harmful attacker for language L .* $\diamond$

Further, a weakly harmful attacker may lead to a violation of diagnosability. We now give a language diagnosability condition that takes into account a sensor attacker ϕ .

Definition 3.4. *A language L is robustly diagnosable against a sensor attacker ϕ with respect to (w.r.t.) $P_o : E^* \rightarrow E_o^*$ and E_f if*

$$(\exists n \in \mathbb{N})(\forall \sigma \in \Psi(E_f))(\forall t \in L/\sigma) [|t| \geq n \Rightarrow C_{AD}]$$

where condition C_{AD} is given as

$$(\nexists \sigma' \in L_N) [\phi(P_o(\sigma t)) = P_o(\sigma')].$$

This definition can be expressed as follows. Assume that the plant generates a word $\sigma \in \Psi(E_f)$ and the evolution continues. After a finite number of steps n (that depends on σ), any attacker $\phi \in \Phi(E_{ins}, E_{era})$ observes $P_o(\sigma t)$ with $|t| \geq n$ and produces the corrupted observation $\phi(P_o(\sigma t))$, which is a faulty observation consistent with no normal word $\sigma' \in L_N$. Note that Definition 3.4 is stated for a particular attacker. Now we extend Definition 3.4 to the case of the set of all possible attackers $\Phi(E_{era}, E_{ins})$ as follows.

Definition 3.5. *A language L is robustly diagnosable against $\Phi(E_{era}, E_{ins})$ w.r.t. $P_o : E^* \rightarrow E_o^*$ and E_f if*

$$(\exists n \in \mathbb{N})(\forall \sigma \in \Psi(E_f))(\forall t \in L/\sigma) [|t| \geq n \Rightarrow \mathbf{C}'_{AD}]$$

where condition $\mathbf{C}'_{AD}$ is given as

$$(\forall \phi \in \Phi(E_{era}, E_{ins}))(\nexists \sigma' \in L_N) [\phi(P_o(\sigma t)) = P_o(\sigma')].$$

With the above concepts, we introduce the following fault diagnosis problem.

Problem 3.2. (*Diagnosability Under Attack*) Given a language L , determine whether L is robustly diagnosable against attacks. $\diamond$

This problem can be investigated by exploring the existence of *harmful attackers*. The negation of Definition 3.4 involves the formal notion of harmful attackers as follows.

Definition 3.6. A sensor attacker $\phi \in \Phi(E_{era}, E_{ins})$ for a language L is said to be harmful if

$$(\forall n \in \mathbb{N})(\exists \sigma \in \Psi(E_f))(\exists t \in L/\sigma) [|t| \geq n \wedge C_{NAD}]$$

where condition C_{NAD} is defined as

$$(\exists \sigma' \in L_N) [\phi(P_o(\sigma t)) = P_o(\sigma')].$$

$\diamond$

This definition implies that there exists a faulty word σ which can be extended with an arbitrarily long word t in such a way that all observations $P_o(\sigma t)$ are corrupted by the attacker into a normal or ambiguous observation $\phi(P_o(\sigma t))$, which is identical to the observation produced by a normal word σ' . The harmful attacker introduces an infinite delay in the detection of the fault, i.e., permanently conceals the fault occurrence to the operator, thereby resulting in a violation of diagnosability. Hence, Problem 3.2 is equivalent to the *Harmful Attacker Existence Problem*.

Example 3.2. Consider the attacker described in Example 3.1. We focus on the following words in $L(G)$: $\sigma t = afbdb^k$, and $\sigma' = ecdb^k$. For the faulty word $afbdb^k$, the attacker ϕ observes $s = abdb^k$ and produces $s' = \phi(abdb^k) = \hat{P}(\xi(abdb^k)) = \hat{P}(a_b_e_db^k) = edb^k$, which is a normal observation consistent with $\sigma' = ecdb^k$. This implies that condition C_{NAD} in Definition 3.6 holds. Hence, ϕ is a harmful attacker. $\diamond$

3.3.2 Attacker's Stealthiness

Due to the effect of sensor attacks on the system diagnosability, another issue of interest is attack detection, in particular, the detection of attacks that violate diagnosability. To this end, we introduce an attack detection mechanism [64-65, 67]. It is assumed that the attacker and the operator have full knowledge of the plant, while the operator does not realize the attacker's presence at the beginning. If an attacker can always keep its attacks undiscovered by the operator during system execution, it is said to be *stealthy*, defined as follows.

Definition 3.7. A sensor attacker ϕ for a language L is said to be *stealthy* if $L(\phi) = \hat{P}(L(\xi)) \subseteq P_o(L(G))$. $\diamond$

The stealthiness of an attacker is guaranteed when any corrupted observation perceived by the operator is contained in the observed language of G . Note that Definition 3.7 requires the attacker not only to be undiscovered at the point when attacks occur but also to remain stealthy no matter how the system evolves in the future. Consider the stealthiness of the attacker ϕ in Example 3.2. Since $L(\phi) \subseteq P_o(L(G))$, we conclude that ϕ is stealthy.

We now define two sets of words $s_a \in E_a^*$ as follows. Given a plant G , the set of *stealthy words* on E_a is defined as $\mathcal{W}_s = \{s_a \in E_a^* \mid \hat{P}(s_a) \in P_o(L(G))\}$, while the set of *exposing words* on E_a is defined as $\mathcal{W}_e = \{s_a e_a \in E_a^* \mid s_a \in \mathcal{W}_s, e_a \in E_a, s_a e_a \notin \mathcal{W}_s\}$. A stealthy word s_a produces a corrupted observation $s' = \hat{P}(s_a) \in P_o(L(G))$, which does not reveal the attacker's presence. On the contrary, an exposing word exits from the set of stealthy words at the final step, implying that the attacker is discovered.

From an attacker's viewpoint, the goals of affecting fault diagnosis and keeping stealthy are separated. In this regard, two types of *successful attackers* that can achieve both goals are: *stealthy weakly harmful attackers* and *stealthy harmful attackers*.

Problem 3.3. Given a language L , determine whether there exists a *stealthy weakly harmful attacker* for L . $\diamond$

Problem 3.4. Given a language L , determine whether there exists a *stealthy harmful* for L . $\diamond$

3.4 Stealthy Joint Diagnoser

This section introduces an information structure called a *joint diagnoser* which describes for all possible attack words, the diagnosis state corresponding to the original observation, and the diagnosis state corresponding to the corrupted observation.

3.4.1 Attacker Diagnoser and Operator Diagnoser

Before constructing a joint diagnoser, two diagnosers, i.e., *Attacker Diagnoser* and *Operator Diagnoser* are developed. The former describes for all attack words s_a that can be generated under attack which is the diagnosis state computed by the attacker. The latter describes for all words $s_a \in \mathcal{W}_s \cup \mathcal{W}_e$ which is the diagnosis state computed by the operator. We first take the perspective of the attacker to capture its ability to corrupt system observations. Given a diagnoser $Diag(G)$, the attacker diagnoser $Diag_{att}(G)$ is defined as follows.

Definition 3.8. Given a nominal diagnoser $Diag(G) = (X_d, E_o, \eta_d, x_{d,0})$ with sets E_{ins} and E_{era} , the attacker diagnoser is $Diag_{att}(G) = (X_d, E_a, \eta_{att}, x_{d,0})$, where the transition function η_{att} is defined as follows: for all $x_d \in X_d$,

$$\begin{cases} \eta_{att}(x_d, e) := \eta_d(x_d, e), & \forall e \in E; \\ \eta_{att}(x_d, e_+) := x_d, & \forall e \in E_{ins}; \\ \eta_{att}(x_d, e_-) := \eta_d(x_d, e), & \forall e \in E_{era} \cap \Gamma(x_d). \end{cases} \quad \diamond$$

From this definition, $Diag_{att}(G)$ has the same set of states of $Diag(G)$. An event $e \in E$ in $Diag_{att}(G)$ corresponds to an event actually occurring in the plant without attacks; thus, for all $x_d \in X_d$, for all $e \in E$, it is $\eta_{att}(x_d, e) = \eta_d(x_d, e)$. The attacker can insert any word in E_{ins}^* between the occurrences of two observable events in $Diag(G)$. On the other hand, the attacker can also erase the observation of an observable event $e \in E_{era}$ after its occurrence in $Diag(G)$, with the erasure being represented by event e_- .

Now let us take the perspective of the operator to capture its ability to perceive attack words. Based on $Diag(G)$, the operator diagnoser $Diag_{opr}(G)$ is defined as follows.

Definition 3.9. Given a nominal diagnoser $Diag(G)$ with sets E_{ins} and E_{era} , the operator diagnoser is $Diag_{opr}(G) = (X_d \cup d_\phi, E_a, \eta_{opr}, x_{d,0})$, where the transition function η_{opr} , for all $x_d \in X_d$, is defined as

$$\begin{cases} \eta_{opr}(x_d, e) := \eta_d(x_d, e), & \forall e \in E; \\ \eta_{opr}(x_d, e_-) := x_d, & \forall e \in E_{era}; \\ \eta_{opr}(x_d, e_+) := \eta_d(x_d, e), & \forall e \in E_{ins} \cap \Gamma(x_d); \\ \eta_{opr}(x_d, e) := d_\phi, & \forall e \in E_o \setminus \Gamma(x_d); \\ \eta_{opr}(x_d, e_+) := d_\phi, & \forall e \in E_{ins} \setminus \Gamma(x_d). \end{cases} \quad \diamond$$

As defined, $Diag_{opr}(G)$ also has the same set of states of G_N except the dump state d_ϕ . The dump state d_ϕ is reachable by all undefined transitions. An event $e \in E$ in $Diag_{opr}(G)$ corresponds to an event perceived by the operator without attacks; thus, for all $x_d \in X_d$, for all $e \in E$, $\eta_{opr}(x_d, e) = \eta_d(x_d, e)$ holds. If the attacker erases an event $e \in E_{era}$, the operator cannot observe the occurrence of e and thus $\eta_{opr}(x_d, e_-) = x_d$. If the attacker inserts an event e_+ , the operator cannot distinguish between the inserted event e_+ and the corresponding event $e \in E_o$, thus updating its state in the same way, i.e., $\eta_{opr}(x_d, e_+) = \eta_d(x_d, e)$.

Example 3.3. Consider G in Fig. 3.2(a). The corresponding $Diag_{att}(G)$ and $Diag_{opr}(G)$ are shown in Fig. 3.3(a) and (b), respectively. To make the construction more illustrative, additional transitions of events in E_+ and E_- are colored in red and blue, respectively. At each state of $Diag_{att}(G)$, all transitions labeled with the inserted event e_+ are in self-loop, since the attacker knows that the inserted event is fake. There exists a transition labeled with the erased event a_- from state $\{0N\}$ to $\{1N, 2F\}$ in $Diag_{att}(G)$ since the attacker knows that the event a has occurred. At each state of $Diag_{opr}(G)$, all transitions labeled with the erased events a_- and b_- are in self-loop, since the operator cannot perceive their occurrences after erasure. There exists a transition labeled with the inserted event e_+ from state $\{0N\}$ to $\{5N, 6N\}$ in $Diag_{opr}(G)$ since the operator cannot distinguish between e_+ and the corresponding event e .

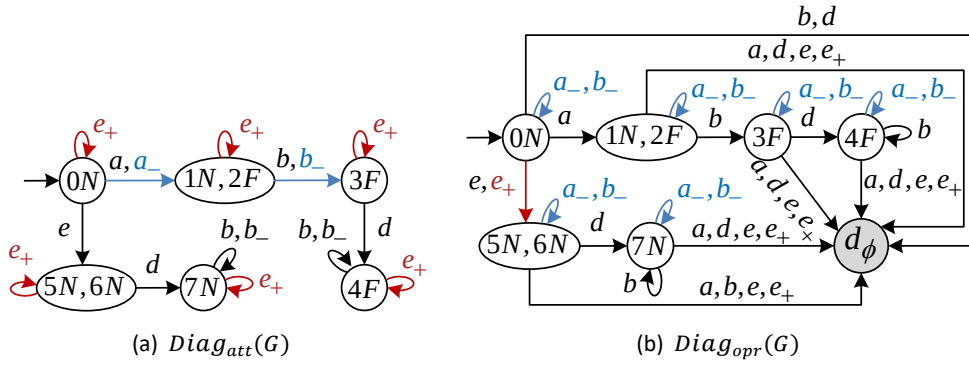


Figure 3.3 (a) $Diag_{att}(G)$ and (b) $Diag_{opr}(G)$ for G in Fig. 3.2(a).

Now we give formal characterizations of $Diag_{att}(G)$ and $Diag_{opr}(G)$ that will be used in the sequel.

Proposition 3.1. Given a nominal diagnoser $Diag(G) = (X_d, E_o, \eta_d, x_{d,0})$, let $Diag_{att}(G) = (X_d, E_a, \eta_{att}, x_{d,0})$ be the attacker diagnoser. It holds that

- 1) $s_a \in L(Diag_{att}(G)) \Leftrightarrow (\exists \xi \in \Xi)(\exists s \in L(Diag(G)) [s_a = \xi(s)]$;
- 2) $(\forall \xi \in \Xi)(\forall s \in L(Diag(G))) [\eta_{att}(x_{d,0}, \xi(s)) = \eta_d(x_{d,0}, s)]$

Proof: The proof is carried out by recursively considering all possible attack functions. First, we focus on the “null attack function” that produces no insertions or erasures. Clearly, given a word $s_a \in E_o^*$, the predicate $s_a \in L(Diag(G)) \Leftrightarrow s_a \in L(Diag_{att}(G))$ holds with $\eta_{att}(x_{d,0}, s_a) = \eta_d(x_{d,0}, s_a)$. Now suppose that the statements hold for a given attack function ξ . Consider an attack function ξ' that contains just one more insertion action than ξ after some observation s_1 . This means that there exists $s_1 \in L(Diag(G))$ with $\xi(s_1) = s_{a,1}$

and $e \in E_{ins}$ such that for all $s = s_1 s_2 \in L(Diag(G))$, it holds that $\xi(s) = s_{a,1} s_{a,2}$ and $\xi'(s) = s_{a,1} e_+ s_{a,2}$, while for all other words $s \in L(Diag(G))$, we have $\xi(s) = \xi'(s)$. Hence, it comes that $\xi'(s) \in L(Diag_{att}(G))$ and $\eta_{att}(x_{d,0}, \xi'(s)) = \eta_{att}(x_{d,0}, \xi(s)) = \eta_d(x_{d,0}, s)$, since all events $e_+ \in E_+$ are self-looped at each state of $Diag_{att}(G)$.

Consider an attack function ξ' that contains just one more erasure action than ξ . This means that there exists $s_1 \in L(Diag(G))$ with $\xi(s_1) = s_{a,1}$ and $e \in E_{era}$ such that for all $s = s_1 e s_2 \in L(Diag(G))$, it holds that $\xi(s) = s_{a,1} e s_{a,2}$ and $\xi'(s) = s_{a,1} e_- s_{a,2}$, while for all other words $s \in L(Diag(G))$, we have $\xi(s) = \xi'(s)$. Clearly, it comes that $\xi'(s) \in L(Diag_{opr}(G))$ and $\eta_{opr}(x_{d,0}, \xi'(s)) = \eta_{opr}(x_{d,0}, \xi(s)) = \eta_d(x_{d,0}, s)$, since for all events $e \in E_{era}$ in $Diag_{opr}(G)$, there exists a parallel event $e \in E_-$. As this inductive procedure generates the set Ξ of all attack functions, the result follows. $\square$

By Statement 1) in Proposition 3.1, the language of the attacker diagnoser consists of all possible attack words s_a , which may correspond to an original observation s produced by G (also perceived by the attacker). According to Statement 2), the diagnosis state estimation of $Diag_{att}(G)$ based on s_a coincides with that of $Diag(G)$ based on s , where $s_a = \xi(s)$.

Proposition 3.2. *Given a nominal diagnoser $Diag(G) = (X_d, E_o, \eta_d, x_{d,0})$, let $Diag_{opr}(G) = (X_d \cup d_\emptyset, E_a, \eta_{opr}, x_{d,0})$ be the operator diagnoser. It holds that*

- 1) $L(Diag_{opr}(G)) = \mathcal{W}_s \cup \mathcal{W}_e$;
- 2) $s_a \in \mathcal{W}_s \subseteq L(Diag_{opr}(G)) \Rightarrow (\exists s' \in L(Diag(G)))[s' = \hat{P}(s_a)]$;
- 3) $\forall s_a \in L(Diag_{opr}(G)) : \text{if } s_a \in \mathcal{W}_s, \eta_{opr}(x_{d,0}, s_a) = \eta_d(x_{d,0}, \hat{P}(s_a)); \text{ if } s_a \in \mathcal{W}_e, \eta_{opr}(x_{d,0}, s_a) = d_\emptyset$.

Proof: Statement 1). By construction, $Diag_{opr}(G)$ includes all words in $\mathcal{W}_s \cup \mathcal{W}_e$. We need to prove that all the words $s_a \in L(Diag_{opr}(G))$ either belong to $\mathcal{W}_s$ or $\mathcal{W}_e$. Consider a word $s_a \in L(Diag_{opr}(G))$ that reaches a state $x_d \in X_d$ and only contains the events in E_o , implying that no attack has been performed. At each state all events e_- are in self-loop, which corresponds to the generation of s_a . By the definition of $\hat{P}$, it holds that $\hat{P}(s_a) \in P_o(L(G))$, i.e., $s_a \in \mathcal{W}_s$. If the word s_a is generated by executing a transition $\eta_{opr}(x'_d, e_+) = x''_d$ with $e \in E_{ins}$, it is also possible to execute the “parallel” transition $\eta_{att}(x'_d, e) = x''_d$ and thus $\hat{P}(s_a) \in P_o(L(G))$, i.e., $s_a \in \mathcal{W}_s$. Then, if the word s_a yields the dump state $d_\emptyset$, then $\hat{P}(s_a) \notin P_o(L(G))$, i.e., $s_a \in \mathcal{W}_e$, which completes the proof of Statement 1). Similar to Proposition 3.1, Statements 2) and 3) can be proved by induction. $\square$

By Statement 1) in Proposition 3.2, all words in $\mathcal{W}_s$ and $\mathcal{W}_e$ can be generated by the operator diagnoser $Diag_{opr}(G)$. By Statement 2), a word in $\mathcal{W}_s$ generated by $Diag_{opr}(G)$ is perceived by the operator as a corrupted observation $s' \in P_o(L(G))$. Statement 3) implies

that: (i) the diagnosis state estimation of $Diag_{opr}(G)$ based on a stealthy word $s_a \in \mathcal{W}_s$ coincides with that of $Diag(G)$ based on s' , where $s' = \hat{P}(s_a)$; (ii) all exposing words $s_a \in \mathcal{W}_e$ yield the dump state $d_\emptyset$.

3.4.2 Joint Diagnoser and Refining Process

Definition 3.10. A joint diagnoser (JD for short) $J-Diag(G)$ is defined as $J-Diag(G) = (R, E_a, \eta_a, r_0) = Diag_{att}(G) \parallel Diag_{opr}(G)$. $\diamond$

As defined, each state of $J-Diag(G)$ is a pair $r = (x_d, \bar{x}_d)$. We define the set of exposing states as $R_e = \{(x_d, \bar{x}_d) \in R \mid \bar{x}_d = d_\emptyset\}$ reached by exposing words (those in $\mathcal{W}_e$), and the set of stealthy states as $R_s = R \setminus R_e$ reached by stealthy words (those in $\mathcal{W}_s$). An attacker that aims to remain stealthy should never produce an attack word $s_a \in \mathcal{W}_e$ yielding an exposing state in R_e in $J-Diag(G)$. However, there may exist stealthy states in R_s from which, following some future evolution of the plant G , an exposing state will necessarily be reached regardless of all future attacker's attempts (including insertions and erasures) to remain stealthy. This leads to the notion of *weakly exposing region* denoted as $R_{we} \supseteq R_e$, which is a superset of R_e , i.e., $R_{we} \supseteq R_e$. Dually, we define the *strongly stealthy region* as $R_{ss} = R \setminus R_{we} \subseteq R_s$. Its physical interpretation is that from any state in R_{ss} , the attacker can take appropriate actions to avoid its exposure.

To compute R_{ss} , we consider function $g : 2^{R_s} \rightarrow 2^{R_s}$ defined for all $R'_s \subseteq R_s$, introduced in [65], as follows:

$$g(R'_s) = g_1(R'_s) \cup g_2(R'_s) \quad (3-1)$$

where

$$g_1(R'_s) = \{r \in R'_s \mid (\exists e \in E_o)[\eta_a(r, e) \in R_s \setminus R'_s \Rightarrow e \in E_{era} \wedge \eta_a(r, e_-) \in R'_s]\}, \quad (3-2)$$

and

$$g_2(R'_s) = \{r \in R'_s \setminus g_1(R'_s) \mid (\exists \sigma_a \in E_+^*)[(\eta_a(r, \sigma_a) \in g_1(R'_s) \wedge (\forall \sigma'_a \leq \sigma_a) \eta_a(r, \sigma'_a) \in R'_s)]\}. \quad (3-3)$$

The set $g(R'_s) \subseteq R'_s$ includes the states from which a suitable operation (including erasure and/or insertion) of an attacker can prevent G_T from leaving R'_s . More precisely, set

$g_1(R'_s)$ is a subset of R'_s , from which if there exists an observable event e whose occurrence leads $J\text{-}Diag(G)$ outside R'_s , then e can be erased by the attacker. Set $g_2(R'_s)$ is a subset of R'_s , from which it is possible to reach a state in $g_1(R'_s)$ through the insertion of a word $\sigma_a \in E_+^*$, and each state visited during the generation of σ_a belongs to R'_s . The strongly stealthy region can be iteratively found by the following fixed-point computation:

$$R_{ss} = \bigcap_{k \geq 0} g^k(R_s). \quad (3-4)$$

That is, R_{ss} is finally derived until function g reaches a fixed point, i.e., $g(R_{ss}) = R_{ss}$.

Example 3.4. Continue with Example 3.3. Part of the joint diagnoser for G is shown in Fig. 3.4, where the exposing states in R_{we} are highlighted in gray, while the stealthy states in R_{we} are highlighted in brown. For instance, a stealthy state $(\{0N\}, \{5N, 6N\})$ is in R_{we} , since from this state, there exists an unerased event $e \notin E_{era}$ yielding an exposing state $(\{5N6N\}, \{x_{d,\emptyset}\})$, and meanwhile the inserted event $e \in E_{ins}$ also reaches an exposing state $(\{0N\}, \{x_{d,\emptyset}\})$. In plain words, once such a stealthy state is reached, all attempts of an attacker to prevent it from reaching a subsequent exposing state will fail. $\diamond$

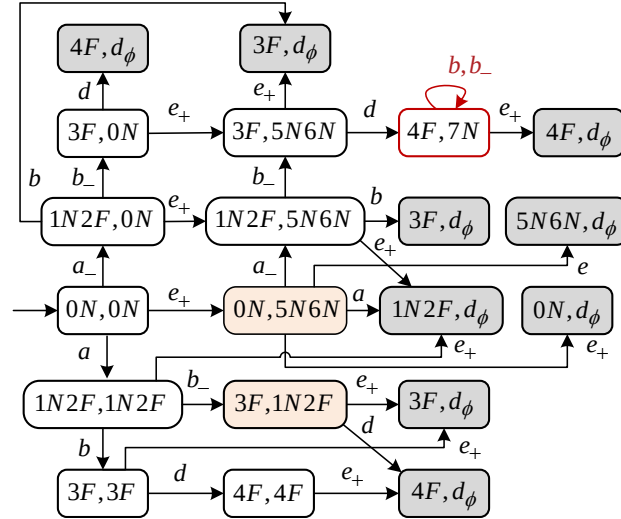


Figure 3.4 Part of $J\text{-}Diag(G)$ in Example 3.4.

To characterize the set of all possible stealthy attackers, we can refine $J\text{-}Diag(G)$ to ensure that no attack word reaching a state in R_{we} is produced.

Definition 3.11. Let $J\text{-}Diag(G) = (R, E_a, \eta_a, r_0)$ be a joint diagnoser. The stealthy joint diagnoser (SJD, for short) is defined as $SJ\text{-}Diag(G) = Ac(R_{ss}, E_a, \eta_{sa}, r_0)$, where $R_{ss} = R \setminus R_{we}$ and $\eta_{sa} = \eta_a|_{R_{ss} \times E_a \rightarrow R_{ss}}$. $\diamond$

The resulting SJD is obtained by removing all states in R_{we} from a JD and taking its accessible part. Now we give a formal characterization of $SJ\text{-}Diag(G)$ as follows.

Theorem 3.1. *Given a plant G , let $Diag(G) = (X_d, E_o, \eta_d, x_{d,0})$ be the nominal diagnoser and $SJ\text{-}Diag(G) = (R_{ss}, E_a, \eta_{sa}, r_0)$ be the SJD. It holds that:*

$$(\forall \xi \in \Xi)(\forall s \in L(Diag(G))) \\ [\eta_{sa}(r_0, \xi(s)) = (x_d, \bar{x}_d) \Leftrightarrow x_d = \eta_d(x_{d,0}, s) \wedge \bar{x}_d = \eta_d(x_{d,0}, \hat{P}(\xi(s))) \neq d_\emptyset]$$

Proof: ($\Leftarrow$) Assume that $\eta_d(x_{d,0}, s) = x_d$ and $\eta_d(x_{d,0}, \hat{P}(\xi(s))) = \bar{x}_d$. According to Propositions 3.1 and 3.2, and by $\bar{x}_d \neq d_\emptyset$, it holds that $\eta_d(x_{d,0}, s) = \eta_{att}(x_{d,0}, \xi(s))$ and $\eta_d(x_{d,0}, \hat{P}(\xi(s))) = \eta_{opr}(x_{d,0}, \xi(s))$, i.e., $\eta_{att}(x_{d,0}, \xi(s)) = x_d$ and $\eta_{opr}(x_{d,0}, \xi(s)) = \bar{x}_d$. By $J\text{-}Diag(G) = Diag_{att}(G) \parallel Diag_{opr}(G)$, and by the construction of $SJ\text{-}Diag(G)$, we have $\eta_{sa}(r_0, \xi(s)) = (x_d, \bar{x}_d)$.

($\Rightarrow$) Assume that $\eta_{sa}(r_0, \xi(s)) = (x_d, \bar{x}_d)$. According to $J\text{-}Diag(G) = Diag_{att}(G) \parallel Diag_{opr}(G)$, and by the construction of $SJ\text{-}Diag(G)$, it holds that $\eta_{att}(x_{d,0}, \xi(s)) = x_d$ and $\eta_{opr}(x_{d,0}, \xi(s)) = \bar{x}_d \neq d_\emptyset$. By Propositions 3.1 and 3.2, we have $\eta_d(x_{d,0}, s) = \eta_{att}(x_{d,0}, \xi(s))$ and $\eta_d(x_{d,0}, \hat{P}(\xi(s))) = \eta_{opr}(x_{d,0}, \xi(s))$, i.e., $\eta_d(x_{d,0}, s) = x_d$ and $\eta_d(x_{d,0}, \hat{P}(\xi(s))) = \bar{x}_d$. $\square$

Hence, a state pair $r_{ss} = (x_d, \bar{x}_d)$ in the SJD reached by an attack word $s_a = \xi(s)$ describes the *joint diagnosis state estimate*, where x_d represents the *correct diagnosis state estimate* of the attacker for the original observation s , and $\bar{x}_d$ represents the *corrupted diagnosis state estimate* of the operator based on the corrupted observation $s' = \hat{P}(s_a)$.

Finally, we conclude this section by the complexity analysis of the proposed approach. Let $Diag(G)$ be a nominal diagnoser with $|X_d|$ states. By construction, the attacker diagnoser $Diag_{att}(G)$ shares the same set of states of $Diag(G)$; so does the operator diagnoser $Diag_{opr}(G)$ except for the dump state $d_\emptyset$. Since the JD $J\text{-}Diag(G)$ is the parallel composition of $Diag_{att}(G)$ and $Diag_{opr}(G)$, its maximum number of states is $|X_d| \times |X_d + 1|$. Moreover, determining if a state of $J\text{-}Diag(G)$ is in the strongly stealthy region has linear complexity in the size of $J\text{-}Diag(G)$. Hence, the complexity of building an SJD is $O(|X_d|^2)$, which is polynomial in the number of states of the nominal diagnoser. However, it is well known that the construction of the diagnoser is worst-case exponential with respect to the number of states in the system. As a result, the overall computational complexity of an SJD is exponential with respect to the number of states of G .

3.4.3 Verification of existence of harmful attackers

We show that the proposed JD (resp., the SJD) leads to the solution of Problems 3.1 and 3.2 (resp., Problems 3.3 and 3.4). The following definition is first required.

Definition 3.12. *Given an attack word s_a , a diagnosis pair function $d : E_a^* \rightarrow \{N, F, U\} \times \{N, F, U\}$ associating to $s_a \in E_a^*$ a diagnosis state is defined as $d(s_a) = (\gamma_{att}(s_a), \gamma_{opr}(s_a))$, where γ_{att} and γ_{opr} are the attacker and operator diagnosis functions, respectively. $\diamond$*

From the definitions of γ_{att} and γ_{opr} , $d(s_a) = (\gamma_{att}(s_a), \gamma_{opr}(s_a)) = (\gamma(s), \gamma(s'))$ holds, where $s_a = \xi(s)$ and $s' = \hat{P}(s_a)$. For any ξ , the diagnosis pair function can be computed by using the SJD. Let $r_{ss} = (x_d, \bar{x}_d) = \eta_{sa}(r_0, s_a)$. By Theorem 3.1, the SJD allows one to associate every state to a diagnosis state pair $d(r_{ss}) = d(s_a)$, i.e., $\gamma(x_d) = \gamma(s)$ and $\gamma(\bar{x}_d) = \gamma(s')$ are the diagnosis state of the attacker and operator, respectively.

The set of all diagnosis state pairs is defined as $D = \{N, F, U\} \times \{N, F, U\}$, which can be partitioned into $D = D_c \cup D_w \cup D_h$, where $D_c = \{(N, N), (U, U), (F, F)\}$, $D_w = \{(N, U), (N, F), (U, N), (U, F)\}$, and $D_h = \{(F, N), (F, U)\}$. The motivation for this partition will be clear later.

Definition 3.13. *Let $SJ-Diag(G)$ be an SJD. A state r_{ss} is correct if $d(r_{ss}) \in D_c$; wrong non-harmful if $d(r_{ss}) \in D_w$; harmful if $d(r_{ss}) \in D_h$. Denote the set of correct states, the set of wrong non-harmful states, and the set of harmful states by R_{sc} , R_{sw} , and R_{sh} , respectively. $\diamond$*

When the SJD is in a correct state, the operator correctly computes the diagnosis state regardless of the fact that an attack has occurred or not. When the SJD reaches a wrong non-harmful state, the operator computes a wrong diagnosis state based on the corrupted observation due to an attack, which is inconsistent with the diagnosis state based on the original observation. Note that, in such a case, the fault diagnosis is manipulated due to the attack but does not pose a real danger. For example, there exists a special case where an attacker induces the operator to think that a fault has occurred, while the system is actually under nominal behavior. The false positive may make the operator decide to take some unnecessary actions, e.g., shut down the system and start it again, which may increase the operational expenses of operating a system.

Finally, harmful states of the SJD correspond to the detection of the fault based on the original observation, and no detection based on corrupted observation. Its physical interpretation is that the attacker itself has already confirmed that the fault has occurred, but it induces the operator to be unable to claim the fault occurrence. Intuitively, the presence of a harmful state is related to Problem 3.3.

Theorem 3.2. *Given a plant G , there exists a stealthy weakly harmful attacker if and only if the $SJ\text{-}Diag(G)$ contains a harmful state, i.e., $R_{sh} \neq \emptyset$.*

Proof: (If) Assume that there exists a harmful state r_{ss} in $SJ\text{-}Diag(G)$ such that $d(r_{ss}) \in D_h$, where $r_{ss} = \eta_{sa}(r_0, s_a)$. By Theorem 3.1, associated with r_{ss} is an attack word s_a such that $d(s_a) = d(r_{ss}) \in D_h$. Hence, there exists a sensor attacker ϕ that alters the observation s into s' such that $(\gamma(s), \gamma(s')) = (\gamma_{att}(s_a), \gamma_{opr}(s_a)) = d(s_a)$, i.e., $(\gamma(s), \gamma(s')) \in \{(F, N), (F, U)\}$. By Definition 3.3, the sensor attacker ϕ is weakly harmful. By the construction of $SJ\text{-}Diag(G)$, ϕ is also stealthy.

(Only if) Assume that there exists a stealthy weakly harmful attacker ϕ . By Definition 3.3, there exists an observation s such that s is corrupted into s' satisfying $(\gamma(s), \gamma(s')) \in \{(F, N), (F, U)\}$. By Theorem 3.1, there necessarily exists a state r_{ss} in $SJ\text{-}Diag(G)$ by executing attack word s_a such that $d(r_{ss}) = (\gamma(x_d), \gamma(\bar{x}_d)) = (\gamma(s), \gamma(s')) \in \{(F, N), (F, U)\}$, i.e., $d(r_{ss}) \in D_h$. Hence, it holds that $R_{sh} \neq \emptyset$. $\square$

In many practical cases, the attacker may only be interested in the impact of fault diagnosis, while it is willing to accept the risk of being discovered. Based on this consideration, we give the following result to solve Problem 3.1 (that is a relaxation of Problem 3.3).

Corollary 3.1. *Given a plant G , there exists a weakly harmful attacker if and only if the $JD\text{-}Diag(G)$ contains a harmful state.*

This corollary can be considered as a relaxation of Theorem 3.2; its proof is similar to that of Theorem 3.2. An example is provided to illustrate the above results.

Example 3.5. *Continue with Example 3.4. The corresponding SJ is omitted since it is part of the JD in Fig. 3.4 without states in R_{we} . Let $s_{a,1} = a_e_+$ be an attack word that yields the wrong non-harmful state $(\{1N, 2F\}, \{5N, 6N\})$. This implies that the diagnosis state of the attacker based on $s = a$ is $\boxed{U}$, while the diagnosis state of the operator based on $s' = e$ is $\boxed{N}$. At this point, the attacker has doubted if the fault has occurred or not; however, the operator is certain that the fault has not occurred.*

Let the evolution continue. Another word $s_{a,2} = a_e_+b_d$ yields the harmful state $(\{4F\}, \{7N\})$. At this point, the attacker is certain that the fault has occurred based on $s = abd$; however, the operator persists in its opinion that the fault has not occurred based on the corrupted observation $s' = ed$. By Theorem 3.2, there exists a stealthy weakly harmful attacker. It produces the attack word $a_e_+b_d$ by first erasing the occurrence of event a , then inserting e_+ , and finally erasing event b while observing abd .

By contrast, let us focus on $w = ab_-$ that yields another harmful state $(\{3F\}, \{1N, 2F\})$. By Corollary 3.1, there exists a weakly harmful attacker that produces the attack word $s_a = ab_-$. However, the attack word $s_a = ab_-$ yields a state $\{3F, 1N2F\} \in R_{we}$ as discussed in Example 3.4. Therefore, the weakly harmful attacker is not stealthy. This conclusion can also be reached in terms of language by $\phi(abd) = ad \notin P_o(L(G))$. $\diamond$

As discussed before, when the SJD of G reaches a harmful state, the attacker currently conceals to the operator the fact that a fault has occurred in G . If it further remains indefinitely in a cycle formed with harmful states, it is possible that the attacker may permanently conceal the fault occurrence to the operator. In addition, the cycle should not be exclusively caused by the inserted events. The reason is that by Definition 3.6, the existence of a stealthy harmful attacker implies a sufficiently long faulty word, but along the cycle exclusively formed with the inserted events, the number of events generated after the fault does not increase. Motivated by this, we provide the following result to solve Problem 3.4.

Theorem 3.3. *Given a plant G , there exists a stealthy harmful attacker if and only if there exists a reachable cycle*

$$cl = r_{ss,1} \xrightarrow{e_1} r_{ss,2} \xrightarrow{e_2} \dots \xrightarrow{e_{l-1}} r_{ss,l} \xrightarrow{e_1} r_{ss,1}$$

in the SJD $SJ\text{-}Diag(G)$, satisfying condition:

$$(\exists j \in \{1, 2, \dots, l\}) [r_{ss,j} \in R_{sh} \wedge e_j \in E_o \cup E_-]. \quad (3-5)$$

Proof: (If) Assume that in $SJ\text{-}Diag(G)$ there exists a cycle $r_{ss,1} \xrightarrow{e_1} r_{ss,2} \xrightarrow{e_2} \dots \xrightarrow{e_{l-1}} r_{ss,l} \xrightarrow{e_1} r_{ss,1}$ satisfying condition (3-5). Let $\eta_{sa}(r_{ss,0}, s_a) = r_{ss,1}$. Since $r_{ss,j} = (x_{d,j}, \bar{x}_{d,j}) \in R_{sh}$ for some $j \in \{1, 2, \dots, l\}$, from the construction of $SJ\text{-}Diag(G)$, it can be seen that $r_{ss,j} \in R_{sh}$ for all $j \in \{1, 2, \dots, l\}$.

The cycle cl may correspond to a cycle $x_{d,1} \xrightarrow{e'_1} x_{d,2} \xrightarrow{e'_2} \dots \xrightarrow{e'_{l-1}} x_{d,l} \xrightarrow{e'_1} x_{d,1}$ in $Diag_{att}(G)$, where $x_{d,j} = (x_j, F)$ for all $j \in \{1, 2, \dots, l\}$. Let $\eta_{att}(x_{d,0}, s'_a) = x_{d,1}$. By Proposition 3.1, it is $s_a e_1 e_2 \dots e_l = \xi(s' e'_1 e'_2 \dots e'_l)$. By the assumption that there exists an event $e_j \in E_o \cup E_-$ in cl , associated with $s' e'_1 e'_2 \dots e'_l$ is an arbitrarily long faulty word $\sigma_Y = \sigma t \in L(G)$, where $|t| \geq n$ for all $n \in \mathbb{N}$, such that $P_o(\sigma) = s'$ and $P_o(t) = e'_1 e'_2 \dots e'_l$.

The cycle cl may also correspond to a cycle $\bar{x}_{d,1} \xrightarrow{e''_1} \bar{x}_{d,2} \xrightarrow{e''_2} \dots \xrightarrow{e''_{l-1}} \bar{x}_{d,l} \xrightarrow{e''_1} \bar{x}_{d,1}$ in $Diag_{opr}(G)$. The following two cases could have occurred:

Case i): $\bar{x}_{d,1} = \bar{x}_{d,2} = \dots = \bar{x}_{d,l}$. This means that $e_j \in E_-$ for all $j \in \{1, 2, \dots, l\}$, since

the occurrence of event $e_- \in E_-$ does not update state $\bar{x}_{d,j}$. Let $\eta_{opr}(\bar{x}_{d,0}, s''_a) = \bar{x}_{d,1}$. By Proposition 3.2, it holds that $s''_a = \hat{P}(s_a e_1 e_2 \cdots e_l)$. Associated with s''_a is a bounded normal word $\sigma_{N,1} \in L(G)$ such that $P_o(\sigma_{N,1}) = s''_a$. Hence, it follows that $P_o(\sigma_{N,1}) = s''_a = \hat{P}(s_a e_1 e_2 \cdots e_l) = \hat{P}(\xi(s'_1 e'_1 e'_2 \cdots e'_l)) = \hat{P}(\xi(P_o(\sigma_t))) = \phi(P_o(\sigma_Y))$. According to Definition 3.6, there exists a harmful attacker for G .

Case ii): In states $\bar{x}_{d,1}, \bar{x}_{d,2}, \dots, \bar{x}_{d,l}$, there exist $n, m \in \{1, \dots, l\}$ such that $\bar{x}_{d,n} \neq \bar{x}_{d,m}$. This means that $e_j \in E_o \cup E_+$ for some $j \in \{1, 2, \dots, l\}$. Let $\eta_{opr}(x_{d,0}, s''_a) = \bar{x}_{d,1}$. By Proposition 3.2, it holds that $s''_a e'_1 e'_2 \cdots e'_l = \hat{P}(s_a e_1 e_2 \cdots e_l)$. If $\bar{x}_{d,j} = (x_j, N)$ for all $j \in \{1, 2, \dots, l\}$, i.e., $\bar{x}_{d,j}$ are negative states, there exists an unbounded normal word $\sigma_{N,2} \in L(G)$ such that $P_o(\sigma_{N,2}) = s''_a e'_1 e'_2 \cdots e'_l$. Similar to Case i), it holds that $P_o(\sigma_{N,2}) = \phi(P_o(\sigma_Y))$. If $\bar{x}_{d,j} = (x_j, U)$ for all $j \in \{1, 2, \dots, l\}$, i.e., $\bar{x}_{d,j}$ are uncertain states, as proved in [24], there also exists an unbounded normal word $\sigma_{N,3}$ such that $P_o(\sigma_{N,3}) = s''_a e'_1 e'_2 \cdots e'_l$. In addition, it holds that $\phi(P_o(\sigma_Y)) = P_o(\sigma_{N,3})$. Otherwise, the set $\bar{x}_{d,j}$ contains both normal and uncertain states, there also exists an unbounded normal word $\sigma_{N,4}$ such that $\phi(P_o(\sigma_Y)) = P_o(\sigma_{N,4}) = s''_a e'_1 e'_2 \cdots e'_l$. By Definition 3.6, the existence of σ_Y and $\sigma_{N,2}$ or $\sigma_{N,3}$ or $\sigma_{N,4}$ indicates the existence of a harmful attacker.

In summary, for both Cases i) and ii), since the SJD contains only stealthy attacks, the harmful attacker is also stealthy.

(Only if) Assuming in fact that there exists a stealthy harmful attacker, the following situation must occur: (a) G can generate a word σ that ends with a fault event; there exists a normal word σ' satisfying $\phi(P_o(\sigma)) = P_o(\sigma')$; (b) the word σ can be extended indefinitely to a word $\sigma_k = \sigma e_1 e_2 \cdots e_k$ (for $k \geq 1$); there also exists a normal word $\sigma'_k \in L_N$ such that $\phi(P_o(\sigma_k)) = P_o(\sigma'_k)$ (for $k \geq 1$). Let the attack word $s_a = \xi(P_o(\sigma))$ reach a state $r_{ss,1}$ in $SJ-Diag(G)$. The following two cases could have occurred:

Case i): $r_{ss,1}$ is a harmful state with $d(r_{ss,1}) \in \{(F, N), (F, U)\}$. Starting from $r_{ss,1}$, as k grows, by Assumption A2, the attack word $s_{a,k} = \xi(P(\sigma_k))$ (for $k \geq 1$) also has an unbounded length and leads to a harmful state. Since the number of states of the SJD is finite, this is only possible if there exists a cycle of harmful states. Since $\sigma_k = u f e_1 e_2 \cdots e_k$ is of unbounded length, at least one event $e \in E_o \cup E_-$ is contained along the harmful cycle.

Case ii): $r_{ss,1}$ is a non-harmful state with $d(r_{ss,1}) \in \{(U, N), (U, U)\}$: Starting from $r_{ss,1}$, as k grows, by Assumption A2, the attack word $s_{a,k} = \xi(P(\sigma_k))$ (for $k \geq 1$) of unbounded length is generated. However, according to Assumption A3 that $Diag(G)$ does not contain indeterminate cycles, every uncertain state will reach a positive state in $Diag(G)$ following $s_k = P_o(\sigma_k)$; correspondingly, the SJD will reach a harmful state by executing

$s_{a,k} = \xi(P(\sigma_k))$ (for $k \geq 1$). Thus, the proof of Case ii) reduces to the case of i).

For both Cases i) and ii), the presence of a stealthy harmful attacker implies that the SJD must contain cycles satisfying condition (3-5), which completes the proof. $\square$

Different from the classical diagnosability verification by searching indeterminate cycles in a nominal diagnoser, the test of cycles in the SJD is self-contained. Namely, it does not require the examination of corresponding plant cycles. The reason is that once a harmful state is reached, a fault has certainly occurred previously (i.e., the diagnosis state of the attacker is $\mathbb{F}$). As one continues along the cycle satisfying condition (3-5), the number of events generated after the fault increases indefinitely. Finally, ignoring the attacker's stealthiness, the following result only guarantees its strong harmfulness, i.e., the impact on diagnosability, and serves as a solution to Problem 3.2 (that is a relaxation of Problem 3.4), whose proof is similar to that of Theorem 3.3 and thus omitted.

Corollary 3.2. *Given a plant G , there exists a harmful attacker if and only if the $JDJ\text{-}Diag(G)$ contains a cycle satisfying condition (3-5).*

Example 3.6. *Let us revisit the SJD of G in Fig. 3.2(a). Since $SJ\text{-}Diag(G)$ has a reachable cycle $(4F, 7N) \xrightarrow{b} (4F, 7N)$ satisfying condition (3-5) (highlighted by red lines), by Theorem 3.3, there exists a stealthy harmful attacker for G . From $SJ\text{-}Diag(G)$, associated with the above cycle is an attack word $s_a = a_-b_-e_+db^k$, which is exactly produced by the stealthy harmful attacker ϕ described in Example 3.2. In addition, an inspection of $SJ\text{-}Diag(G)$ implies the attack actions of the attacker ϕ : first erasing event a and b and then inserting event e while observing $abdb^k$.*

A stealthy harmful attacker does not always exist. We consider a new plant H which is a copy of G in Fig. 3.2(a) but state 4 has a self-loop transition labeled with g . It can be verified that the $JDJ\text{-}Diag(H)$ has a reachable cycle $(4F, 7N) \xrightarrow{b} (4F, 7N)$ satisfying condition (3-5). By Corollary 3.2, there exists a harmful attacker ϕ for H . Again consider the attack word $s_a = a_-b_-e_+db^k$. In $J\text{-}Diag(H)$, it leads to state $(4F, 7N)$. Using Eq. (??), we have $(4F, 7N) \in R_{we}$. Hence, the harmful attacker ϕ that produces $a_-b_-e_+db^k$ is no longer stealthy for H . In essence, if H infinitely generates the unerased event $g \in E_o \setminus E_{era}$ at state 4, it comes that $\phi(abdg) = edg \in L(\phi, H)$, but $\notin P_o(L(H))$. Finally, there is no stealthy harmful attacker for H . $\diamond$

We conclude this section with the following comments. In the next chapter, a more efficient alternative using verifier automata will be explored. However, it is still worth developing a diagnoser for the fault diagnosis problem under attack due to the following statements.

- The diagnoser approach has the advantage of being used for both online diagnosis and offline diagnosability verification; however, the verifier automata can only be used for offline verification. This is due to the fact that the diagnoser can provide a complete characterization of the diagnosis problem under the considered model. Unlike the verifier under attack (that will be investigated in Chapter IV), the joint diagnoser can show how a diagnosis state is affected by an attacker during the system evolution.
- Besides the fault diagnosis under attack, we expect to establish an attack detection mechanism. This is done by determining whether the operator perceives abnormal observations of words. In this sense, the diagnoser can provide an intuitive solution. However, to achieve this, the observer of a verifier may be additionally needed.

Moreover, here is an encouraging result as far as the state sizes of diagnosers are concerned, from Clavijo et al. [90] and adopted by Viana et al. [91]. It is claimed, based on a rigorous statistical analysis, that the size of the states of diagnosers and verifiers are respectively $\Theta(n^{0.77 \log k + 0.63})$ and $\Theta(n^2)$, on the average, where k (resp. n) is the number of events (resp. states) of a plant automaton. This means that, for systems with 60 events or less, the number of states of diagnosers is likely to be smaller than that of the corresponding verifier.

3.5 Attack Synthesis

3.5.1 Introduction

In this section, we aim to develop a successful attack synthesis algorithm. We limit our presentation on attack synthesis against the diagnosability verification. From the attacker's point of view, besides the damage infliction to system diagnosability, it desires to maintain itself as undiscovered by the system operator, i.e., achieves the stealthiness. To ensure stealthiness, an intelligent attacker must consider the race between inserted events and internal system events. In essence, when synthesizing successful attacks, an attack event must be inserted at the desired time. In order to capture this race, we define a *pre-empting state* of the SJD as those states corresponding to the case where the internal system events are pre-empted by the inserted events. Based on this, a successful attack synthesis algorithm is finally proposed. By providing a general synthesis approach, our future goal is to understand what kinds of attacks systems are not robust to and how to reduce their impact on diagnosis estimation performance.

3.5.2 Synthesis of Attackers

We define a subset of the states of $SJ\text{-}Diag(G)$, called pre-empting states, which will be used in the procedure to extract an attack function from $SJ\text{-}Diag(G)$.

Definition 3.14. Given a JD $J\text{-}Diag(G) = (X, E_a, \eta_a, x_0)$ with the corresponding SJD $SJ\text{-}Diag(G) = (R_{ss}, E_a, \eta_{sa}, r_0)$, the set of pre-empting states of $SJ\text{-}Diag(G)$ is defined as $R_p = \{r_{ss} \in R_{ss} \mid (\exists e \in E_o) [\eta_a(r_{ss}, e) \in R \setminus R_{ss} \wedge (e \notin E_{era} \vee \eta_a(r_{ss}, e_-) \in R \setminus R_{ss})]\}$; the set of non-pre-empting states of $SJ\text{-}Diag(G)$ is defined as $R_{np} = R_{ss} \setminus R_p$.

A state $r_{ss} \in R_{ss}$ is pre-empting if i) there exists an event $e \in E_o$ whose occurrence (even if erased) causes $J\text{-}Diag(G)$ to lead outside R_{ss} ; ii) however, e can be definitely pre-empted by inserting an appropriate word of events in E_+ to reach a state in the strongly stealthy state. This state captures the races between inserted events and internal system events. Specifically, at a pre-empting state, the inserted word must be performed before the execution of event σ . Recall the plant in Fig. 3.2(a). Its entire joint diagnoser is shown in Fig. 3.5. We can see that state $(\{3F\}, \{0N\})$ is a pre-empting state (highlighted in green), where event d must be pre-empted by inserting a fake event e_+ to reach a state $(\{3F\}, \{5N, 6N\}) \in R_{ss}$; otherwise, the occurrence of $d \in E_o \setminus \Sigma_{era}$ will take $(\{3F\}, \{0N\})$ to an exposing state $(\{4F\}, \{d_\phi\})$.

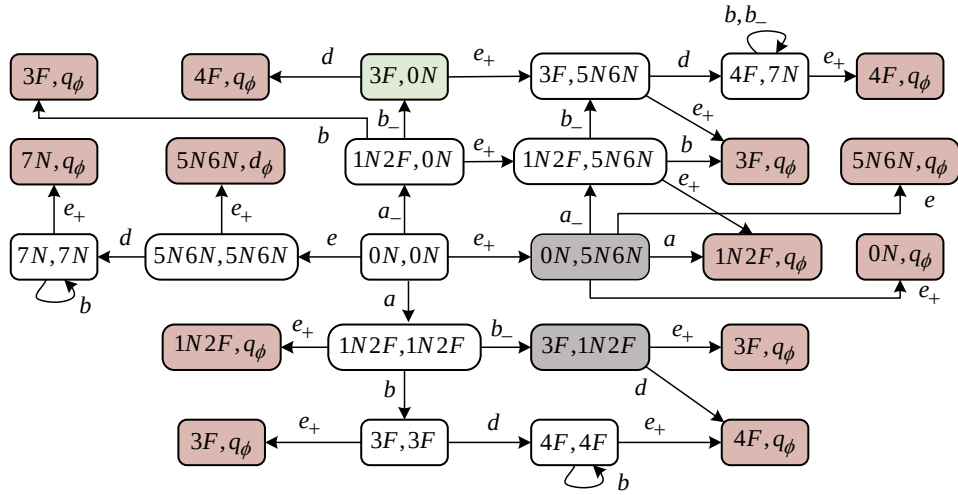


Figure 3.5 $J\text{-}Diag(G)$ for G in Fig. 3.2(a).

In order to synthesize an attacker, the following definition is also required.

Definition 3.15. Let $r_{ss} \in R_{ss}$ be a state and R_{np} be the set of non-pre-empting states in the SJD $SJ\text{-}Diag(G) = (R_{ss}, E_a, \eta_{sa}, r_0)$.

(1) The set of events in E_a enabled at r_{ss} is defined as $\Gamma_{SJ}(r_{ss}) = \{\sigma \in E_a \mid \eta_{sa}(r_{ss}, \sigma)!\}$.

Algorithm 3.1 Synthesis of the corrupting function ϕ

Require: $G = (X, E, \eta, x_0)$ and $SJ\text{-}Diag(G) = (R_{ss}, E_a, \eta_{sa}, r_0)$

Ensure: corrupting function $\phi \in \Phi$

```

1:  $\phi \leftarrow Synth(SJD)$ 
2: procedure  $SYNTH(SJD)$ 
3:   initialization:  $s \leftarrow \varepsilon$ 
4:   select a word  $s_{a,+} \in L_+(SJ\text{-}Diag(G), r_0)$ 
5:    $\xi(s) := s_{a,+}$ ;  $\phi(s) := \hat{P}(\xi(s))$ 
6:    $r_{ss} := \eta_{sa}(r_0, s_{a,+})$ 
7:   while a new observable event  $e \in E_o$  is produced by  $G$  do
8:      $I := \emptyset$ 
9:     if  $e \in \Gamma_{SJ}(r_{ss})$  then  $I := I \cup \{e\}$ 
10:    if  $e_- \in \Gamma_{SJ}(r_{ss})$  then  $I := I \cup \{e_-\}$ 
11:    select an event  $e' \in I$  and a word  $s_{a,+} \in L_+(SJ\text{-}Diag(G), \eta_{sa}(r_{ss}, e'))$ 
12:    let  $s_a := e' s_{a,+}$  and  $s'_a \in \{e', s_a\}$ 
13:    let  $r_{ss} := \eta_{sa}(r_{ss}, e')$ 
14:    if  $r_{ss} \in R_{np}$  then  $\xi(se) \in \xi(s) s'_a$  and  $r_{ss} := \eta_{sa}(r_{ss}, s'_a)$ 
15:    if  $r_{ss} \in X_p$  then  $\xi(se) := \xi(s) s_a$  and  $r_{ss} := \eta_{sa}(r_{ss}, s_a)$ 
16:     $\phi(se) := \hat{P}(\xi(se))$ 
17:     $s \leftarrow se$ 
18:  end while
19: end procedure

```

(2) The set of words consisting of events in E_+ that originate from r_{ss} and lead to a non-pre-empting state is defined as $L_+(SJ\text{-}Diag(G), r_{ss}) = \{s_{a,+} \in E_+^* \mid \eta_{sa}(r_{ss}, s_{a,+}) \in R_{np}\}$.

The following objective is to show how an attacker may determine an attack function from an SJD. In [65], an algorithmic procedure is presented for this purpose. The main idea is to select attack words that do not end in any state that has an inserted event as a successor transition, of course, including the pre-empting state. Compared with [65], it is now expected to synthesize the attack function producing an attack word that only does not end in pre-empting states. This means that an attacker cannot stay in the pre-empting state when determining its attack function. Now, we provide the following attacker synthesis algorithm.

This algorithm may recursively associate each observation generated by G with a corresponding attack word. We now briefly explain how it works. When no event is generated by G , the attacker may insert a suitable word $s_{a,+}$ in the set $L_+(SJ\text{-}Diag(G), r_0)$ (Steps 3 and 4). In Step 5, $\xi(\varepsilon)$ is computed, and the new current state r_{ss} of $SJ\text{-}Diag(G)$ is updated. Then, we wait for G to generate a new observable event e (Step 7). A new set I initialized at the empty set is defined. If e is enabled at r_{ss} , e is added to I . If e_- is enabled at r_{ss} , e_- is added to I (Steps 8 to 10). In Step 11, an event $e' \in I$ and the insertion of a

word $s_{a,+} \in L_+(SJ\text{-}Diag(G), \eta_{sa}(r_{ss}, e'))$ are selected. If the current state r_{ss} with the occurrence of e' is a non-pre-empting state, an attack word s'_a is produced, which is either e' or the concatenation of e' and $s_{a,+} \in L_+(SJ\text{-}Diag(G), \eta_{sa}(r_{ss}, e'))$ (Step 14). Otherwise, another attack word s_a is produced as the concatenation of e' and $s_{a,+}$ (Step 15). The corresponding corrupting function is computed in Step 16. Step 17 updates the observation s to se . This procedure goes to Step 7 when a new observable event is generated by G .

Proposition 3.3. *Given a plant G , function ϕ is stealthy if and only if, for all $w \in P_o(L(G))$, the corrupted observation $\phi(s)$ can be computed by Algorithm 3.1.*

According to the property of the SJD, the above results trivially hold, which means that the corrupting function synthesized by Algorithm 3.1 must be stealthy. Using the notion of pre-empting states, we provide a characterization of the SJD as follows, which is the generalization of Theorem 3.1.

Theorem 3.4. *Given a plant G , let $Diag(G) = (X_d, E_o, \eta_d, x_{d,0})$ be the diagnoser and $SJ\text{-}Diag(G) = (R_{ss}, E_a, \eta_{sa}, r_0)$ be the SJD. The following implication holds:*

$$\begin{aligned} & (\exists \xi(s) \in E_a^*) [\eta_{sa}(x_0, \xi(s)) = r_{ss} = (x_d, \bar{x}_d) \wedge r_{ss} \in R_{np}] \\ \Leftrightarrow & (\exists w \in P_o(L(G))), (\exists \xi \text{ synthesized by Algorithm 3.1}) \\ & [x_d = \eta_d(x_{d,0}, s_a) \wedge \bar{x}_d = \eta_d(x_{d,0}, \hat{P}(\xi(s))) = \eta_d(x_{d,0}, \phi(s))]. \end{aligned}$$

Proof: (If) Assume that there exists an observation $w \in P_o(L(G))$ and an attack function ξ such that $x_d = \eta_d(x_{d,0}, s_a)$ and $\bar{x}_d = \eta_d(x_{d,0}, \hat{P}(\xi(s)))$, where the attack function ξ is synthesized by Algorithm 3.1. By Proposition 3.3, the attack word $\xi(s)$ is a stealthy word, i.e., $\xi(s) \in \mathcal{W}_s$. By Propositions 3.1 and 3.2, we have $\eta_{att}(x_{d,0}, s_a) = \eta_d(x_{d,0}, s_a)$ and $\eta_{opr}(x_{d,0}, s_a) = \eta_d(x_{d,0}, \hat{P}(s_a))$, i.e., $x_d = \eta_{att}(x_{d,0}, s_a)$ and $\bar{x}_d = \eta_{opr}(x_{d,0}, s_a)$. By $J\text{-}Diag(G) = Diag_{att}(G) \parallel Diag_{opr}(G)$, and by the definition of $SJ\text{-}Diag(G)$, we have $\eta_{sa}(x_0, s_a) = r_{ss} = (x_d, \bar{x}_d)$. Since $\xi(s)$ is computed by selecting s_a , which ends in r_{ss} , by Algorithm 3.1, $r_{ss} \in R_{np}$.

(Only if) Assume that there exists an attack word $\xi(s) \in E_a^*$ in $SJ\text{-}Diag(G)$ such that $\eta_{sa}(x_0, \xi(s)) = r_{ss} = (x_d, \bar{x}_d)$ and $r_{ss} \in R_{np}$. By $J\text{-}Diag(G) = Diag_{att}(G) \parallel Diag_{opr}(G)$, and by the definition of $SJ\text{-}Diag(G)$, it holds that $\eta_{att}(x_{d,0}, \xi(s)) = x_d$ and $\eta_{opr}(x_{d,0}, \xi(s)) = \bar{x}_d$. By Propositions 3.1 and 3.2, we have $\eta_{att}(x_{d,0}, \xi(s)) = \eta_d(x_{d,0}, s_a)$ and $\eta_{opr}(x_{d,0}, \xi(s)) = \eta_d(x_{d,0}, \hat{P}(\xi(s)))$, i.e., $x_d = \eta_d(x_{d,0}, s_a)$ and $\bar{x}_d = \eta_d(x_{d,0}, \hat{P}(\xi(s)))$. $\square$

According to the above result, a state pair $r_{ss} = (x_d, \bar{x}_d)$ in the joint diagnoser reached by an attack word s_a that synthesized by Algorithm 3.1 represents the joint diagnosis state

estimation, where x_d describes the original diagnosis state estimation of the attacker based on the original observation s , where $s_a = \xi(s)$, and $\bar{x}_d$ describes the corrupted diagnosis state estimation of the operator based on the corrupted observation $s' = \hat{P}(s_a)$.

3.5.3 Synthesis of Successful Attackers

We first provide the following proposition to ensure that a successful attacker can be synthesized from the SJD.

Proposition 3.4. *Let G be a plant and $SJ\text{-}Diag(G) = (R_{ss}, E_a, \eta_{sa}, x_0)$ be the SJD. A successful corrupting function ϕ can be computed if and only if $R_{sh} \cap R_{np} \neq \emptyset$.*

Proof: (If) Suppose that a state $r_{ss} \in R_{sh} \cap R_{np}$ in $SJ\text{-}Diag(G)$ satisfies $r_{ss} = \eta_{sa}(x_0, s_a)$ and $s_a = \xi(s)$, where $w \in P_o(L(G))$. By $r_{ss} \in R_{np}$, the attack word s_a does not end in a pre-empting state. According to Algorithm 3.1 and Proposition 3.3, we see that ϕ is stealthy. Since $r_{ss} = (x_d, \bar{x}_d) \in R_{sh}$, by Theorem 3.1, we have $r_{ss} = \eta_{sa}(x_0, s_a) = (x_d, \bar{x}_d)$ such that $\eta_d(x_{d,0}, s_a) = x_d$ and $\eta_d(x_{d,0}, s') = \bar{x}_d$ with $s' = \phi(s)$, where the attacker ϕ alters the observation w into s' such that $(\gamma(w), \gamma(s')) = (\gamma_{att}(s_a), \gamma_{opr}(s_a)) = r(s_a) = r(r_{ss}) \in R_h$, i.e., $(\gamma(w), \gamma(s')) \in \{(F, N), (F, U)\}$. According to Definition 3.3, the corrupting function ϕ is also harmful.

(Only if) Suppose that ϕ is a successful corrupting function that can be computed. Since function ϕ is stealthy, by Algorithm 3.1 and Proposition 3.3, the attack word $s_a = \xi(s)$ does not end in a pre-empting state of $SJ\text{-}Diag(G)$, i.e., $r_{ss} = \eta_{sa}(x_0, \xi(s)) \in R_{np}$. On the other hand, as function ϕ is harmful, according to Definition 3.3, there exists an observation $w \in P_o(L(G))$ that can be changed into an observation $s' = \phi(s) \in P_o(L(G))$, and $(\gamma(w), \gamma(s')) \in \{(F, N), (F, U)\}$. By $w, s' \in P_o(L(G))$, it holds that $\eta_d(x_{d,0}, s_a) = x_d$ and $\eta_d(x_{d,0}, s') = \bar{x}_d$. Since $SJ\text{-}Diag(G)$ contains all possible stealthy attacks, by Theorem 3.1, there must exist a state $r_{ss} = (x_d, \bar{x}_d)$ in $SJ\text{-}Diag(G)$ with the occurrence of $s_a = \xi(s)$ such that $\eta_d(x_{d,0}, s_a) = x_d$ and $\eta_d(x_{d,0}, s') = \bar{x}_d$. By $r(r_{ss}) = r(s_a) = (\gamma_{att}(s_a), \gamma_{opr}(s_a)) = (\gamma(w), \gamma(s')) \in \{(F, N), (F, U)\}$, i.e., $r(r_{ss}) \in R_h$, we see that the state r_{ss} is harmful. It holds that $r_{ss} \in R_{sh} \cap R_{np}$, i.e., $R_{sh} \cap R_{np} \neq \emptyset$. $\square$

This proposition implies that, from the synthesis viewpoint, only a successful attacker that determines an attack word yielding a non-pre-empting harmful state can make sense. We now give Algorithm 3.2 to synthesize a successful attacker from an SJD. In Algorithm 3.2, the condition $r_{ss} \in R_{sh} \cap R_{np}$ (Step 3) for selecting a state r guarantees the harmfulness of the stealthy corrupting function synthesized by Algorithm 3.1.

Algorithm 3.2 Synthesis of the successful corrupting function ϕ

Require: $G = (X, E, \eta, x_0)$ and $SJ\text{-}Diag(G) = (R_{ss}, E_a, \eta_{sa}, r_0)$

Ensure: successful corrupting function ϕ

```

1:  $\phi \leftarrow Synthsuc(SJD)$ 
2: procedure SYNTHSUC( $SJD$ )
3:   if  $R_{sh} \cap R_{np}$  then choose a harmful state  $r_{ss} \in R_{sh} \cap R_{np}$  with  $r_{ss} = \eta_{sa}(x_0, s_a)$ 
4:   compute the observation  $s$  such that  $s_a = \xi(s)$ 
5:   while there is an observation  $s^* \in \bar{s} \cap P_o(L(G))$  do
6:     compute the corrupted observation  $\phi(s^*)$  by Algorithm 3.1
7:   end while
8: end procedure

```

Example 3.7. Following Algorithm 3.2, choose a harmful state $(4F, 7N) \in R_{sh} \cap R_{np}$. Associated with the state is an attack word $s_a = a_b_e_+d \in L(SJ\text{-}Diag(G))$. It corresponds to an observation $w = abd$. For all $s^* \in \bar{s} \cap P_o(L(G))$, the attack function ξ is computed by Algorithm 3.1: $\xi(\varepsilon) = \varepsilon$, $\xi(a) = a_$, $\xi(ab) = a_b_e_+$, and $\xi(abd) = a_b_e_+d$ —and the corrupting function is synthesized accordingly: $\phi(\varepsilon) = \varepsilon$, $\phi(a) = \varepsilon$, $\phi(ab) = e$, and $\phi(abd) = ed$. The key point is that the attack word $a_b_$ is not selected in the synthesis procedure, i.e., it holds that $\xi(ab) = a_b_e_+$ rather than $\xi(ab) = a_b_$. This means that the attacker implements its attacks by first erasing the occurrence of event a , then erasing event b , and finally inserting e_+ while observing abd . $\diamond$

3.6 Case Study: A Power System

In this section, we apply the above results to a protection relay and a circuit breaker in a power system modified from [88], as shown in Fig. 3.6. The system works as follows. If there is a downed power line or other accident that has occurred in the power system causing an over current event on the power line, then the protection relay will be triggered. When the protection relay is triggered and closed, the circuit breaker will open and cut the faulty power line from the power system. After the circuit breaker is open for a short period of time, it will automatically try to reclose. If the over current disappears after the reclosure, then the fault is temporal and the line returns to normal operation. If the over current persists after the reclosure, then the fault is permanent and the circuit breaker will return to and remain open until the repair is made to the power line.

We model the system by the automaton G shown in Fig. 3.7(a). In the automaton G , the states are $X = \{1, 2, 3, 4, 5, 6, 7, 8\}$ and the events are listed in Table 3.1. Events OC , ZC , AC , and R are observable, since the current can be measured with sensors. Hence,

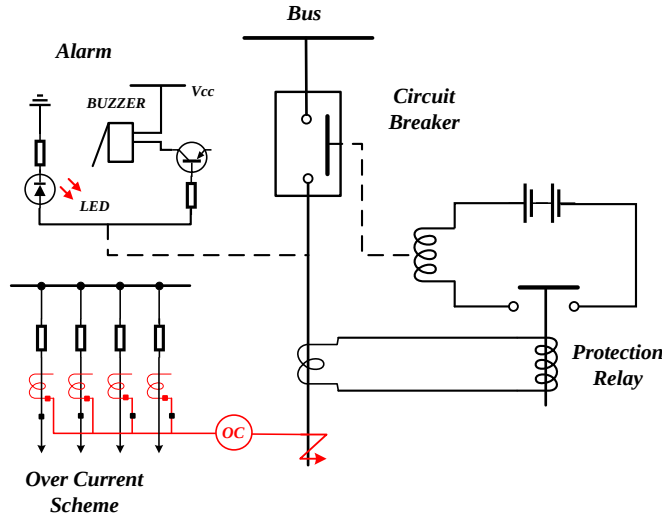


Figure 3.6 Protection relay and circuit breaker.

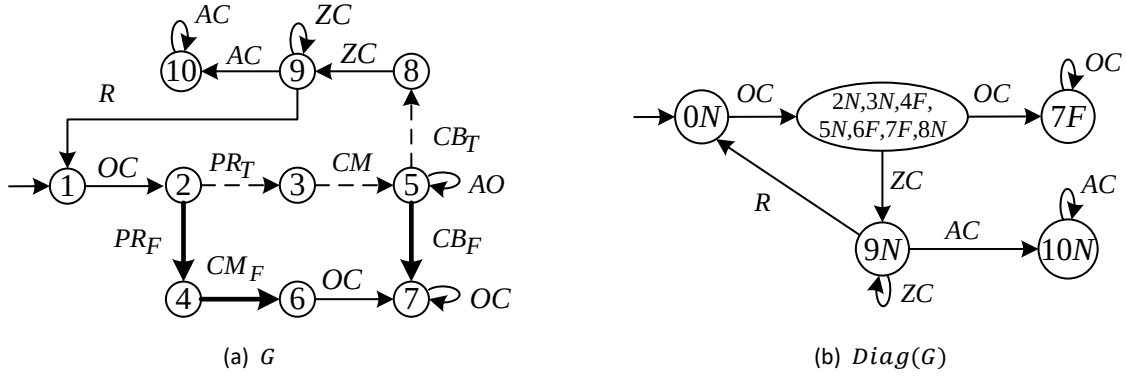


Figure 3.7 (a) Automaton G for protection relay and circuit breaker, and (b) the diagnoser of G .

$E_o = \{OC, ZC, AC, R\}$. It is expected to diagnose faults in the system. Clearly, there are three fault events: PR_F , CB_F and CM_F . Hence, $E_f = \{PR_F, CM_F, CB_F\}$. It can be checked that without sensor attacks, G is diagnosable with respect to P and E_f . Intuitively, after the occurrence of either PR_F , CB_F , or CM_F the operator will observe event OC , while under normal operation, it will observe event ZC .

Now we present SE-attacks and SI-attacks on the protection relay and the circuit breaker. Let the set of vulnerable sensor events to SE-attacks and SI-attacks be $E_{era} = \{OC\}$ and $E_{ins} = \{ZC\}$, respectively. Hence, $E_- = \{OC_-\}$ and $E_+ = \{ZC_+\}$. Assume that there exists a sensor attacker ϕ with the following attack function:

$$\xi(s) = \begin{cases} \varepsilon, & \text{for } s = \varepsilon; \\ OCOC_-ZC_+(OC_-ZC_+)^k, & \text{for } s = OC^2OC^k, k \in \mathbb{N}. \end{cases}$$

Table 3.1 Events in the protection relay and circuit breaker system

Event label	Meaning
OC	Over current
ZC	Zero current
PR_T	Protection relay triggered
PR_F	Protection relay failed
CM	Comand control sent
CM_F	Comand control failed
CB_T	Circuit breaker triggered
CB_F	Circuit breaker failed
R	Circuit breaker reclosed
AO	Alarme triggered
AC	Alarme closed

That is, the attacker lets the first over current sensor reading intact. After this, every time an over current sensor reading is observed, the attacker chooses to erase it and insert a zero current reading. As a result, the over current sensor readings are tampered with zero current readings by the attacker ϕ . Intuitively, G is not robustly diagnosable against ϕ , i.e., there exists a harmful attacker for G , since after the execution of faulty operations $OCPR_FOCOC^k$, the attacker observes $P(OCPR_FCM_FOCOC^k) = OC^2OC^k$ and implements its attack (i.e., $\phi(OC^2OC^k) = \hat{P}(\xi(OC^2OC^k)) = \hat{P}(OCOC_ZC_+(OC_ZC_+)^k) = OCZCZC^k$), which has the same projection with the normal operations $OCPR_TCMCB_TZCZC^k$. In addition, it is not difficult to see that ϕ satisfies stealthiness, i.e., ϕ is a stealthy harmful attacker.

The above result can also be verified with the SJD $SJ-Diag(G)$ for G , part of which is shown in Fig. 3.8, where we only focus on cycles formed with a harmful state $7F9N$, and omit other parts with loss of generality for the purpose of verification. An inspection of $SJ-Diag(G)$ implies the attack actions of the attacker ϕ : periodically first erasing event OC and then inserting event e .

3.7 Summary

In this chapter, we have investigated the fault diagnosis in the case of attacks that corrupt sensor readings by inserting or erasing event observations. We formally formulate the problem of diagnosability under attack. From the attacker's point of view, not only its impact on the diagnosability but also its stealthiness should be taken into account. To this end, we propose a stealthy joint diagnoser (SJD), from which necessary and sufficient conditions

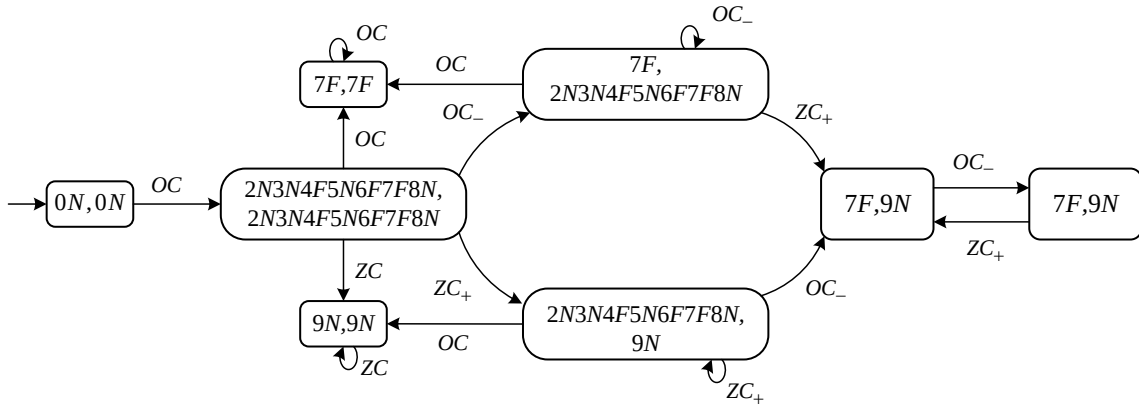


Figure 3.8 Part of SJD $Sj\text{-}Diag(G)$ for G in Fig. 3.7(a).

for the existence of certain attackers related to diagnosability are presented. As the construction of the joint diagnoser has exponential complexity, it is interesting to explore the use of verifier automata for diagnosability verification under attack in Chapter IV.

By providing a general synthesis approach, we plan to develop a suitable defense strategy against attacks (e.g., design supervisors that enforce system diagnosability and at the same time are robust to sensor attacks), which leaves the door open for future exploration.

The work of this chapter is published as:

T. Kang, C. Seatzu, Z. Li, and A. Giua, "Fault diagnosis of discrete event systems under attack," in *Proc. IEEE 62nd Conf. Decis. Control (CDC)*, Singapore, Singapore, Dec 2023, pp. 7923-7929.

T. Kang, C. Seatzu, Z. Li, and A. Giua, "A Joint Diagnoser Approach for Diagnosability of Discrete Event Systems Under Attack," *Automatica*, vol. 172, p. 112004, 2025.

Chapter IV Fault Diagnosis Under Attack: Using Verifier

4.1 Motivation and contribution

As mentioned in the previous chapter, the joint diagnoser may face the potential state explosion problem; in the worst case, the size of a joint diagnoser is exponential in the size of the system. Motivated by the above observations, in this chapter, we aim to find a more efficient arena for diagnosability verification under attack. In that context, we first review the notion of *robust diagnosability* that takes into account all possible effects of sensor attacks in a given attack scenario. To verify it, we develop a deterministic verifier under attack, which holds some similarities with those proposed in [41-42] and [45-46, 85] that respectively deal with the nominal case in the sense that no attack exists and the case of observations loss due to sensor and communication failures.

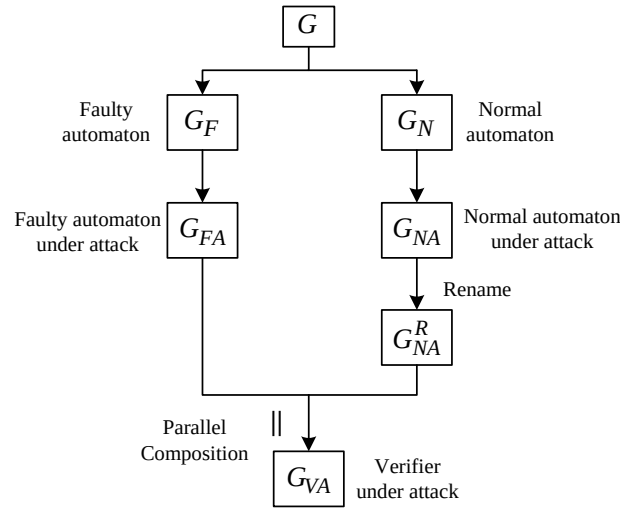


Figure 4.1 Scheme of the proposed methodology, where the involved notations will be formally defined later.

As depicted in Fig. 4.1, the construction of the verifier proceeds as follows.

- Construct the faulty automaton G_F and the normal automaton G_N of a plant G .
- Based on G_F , construct the faulty automaton under attack G_{FA} .
- Based on G_N , construct the normal automaton under attack G_{NA} .
- Rename the regular observable events in G_{NA} and obtain G_{NA}^R .
- Obtain the verifier under attack G_{VA} through the parallel composition of G_{FA} and G_{NA}^R .

The idea behind the verifier under attack is to perform a parallel composition between an attack faulty word and an attack normal word, with synchronization on the defined *attack alphabet*. We prove that from the attack faulty and normal words, a pair of ambiguous words that lead to a violation of diagnosability, i.e., a faulty word generated by a system and a normal word perceived by a system operator, can be found. Finally, such a verifier provides a necessary and sufficient condition for robust diagnosability against attacks, which can be verified with polynomial complexity in the size of the system.

4.2 Problem Statement

In this section, we first review the robust diagnosability condition against a set of sensor attackers introduced in Chapter III, as follows.

Definition 4.1. A language L is robustly diagnosable against $\Phi(E_{era}, E_{ins})$ w.r.t. $P_o : E^* \rightarrow E_o^*$ and E_f if

$$(\exists n \in \mathbb{N})(\forall \sigma \in \Psi(E_f))(\forall t \in L/\sigma) [|t| \geq n \Rightarrow \mathbf{C}'_{AD}]$$

where condition $\mathbf{C}'_{AD}$ is given as

$$(\forall \phi \in \Phi(E_{era}, E_{ins}))(\nexists \sigma' \in L_N) [\phi(P_o(\sigma t)) = P_o(\sigma')]. \quad \diamond$$

Example 4.1. Recall the plant and its diagnoser in Example 3.1, which are again illustrated in Figs. 4.2(a) and 4.2(b), respectively. As described above, the language of G , i.e., $L(G)$, is diagnosable in the nominal case.

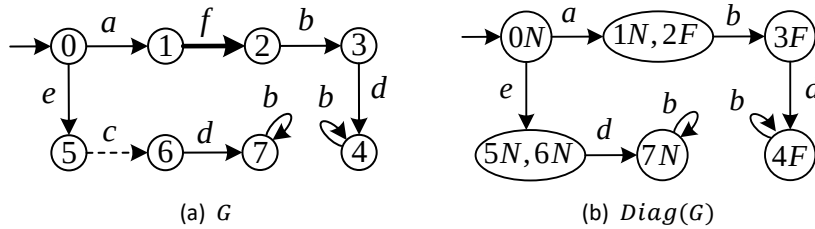


Figure 4.2 (a) A plant G and (b) its diagnoser $Diag(G)$.

Now assume that the sensor that records the occurrence of a is subject to SE-attacks, and the sensor that records the occurrence of e is subject to SI-attacks. That is, $E_{era} = \{a\}$ and $E_{ins} = \{e\}$. In the following, we show that L is not robustly diagnosable against the set of all possible attackers $\Phi(E_{era}, E_{ins})$.

First, we consider an attacker $\phi_1 \in \Phi(E_{era}, E_{ins})$ with the following attack function

$$\xi_1(s) = \begin{cases} e_+, & \text{for } s = \varepsilon; \\ e_+adb^k, & \text{for } s = adb^k, k \in \mathbb{N}. \end{cases}$$

That is, the attacker ϕ_1 inserts a fake event e before any word generated by G is observed. It can be seen that L is robustly diagnosable against ϕ_1 , since under attack all faulty words $afdc b^k$ ($k \geq 0$) produce the corresponding corrupted observations $s_1 = \phi_1(P(afdc b^k)) = \hat{P}(\xi_1(adb^k)) = \hat{P}(e_+adb^k) = eadb^k$ that cannot be produced by any normal word, implying that condition $\mathbf{C}_{AD}$ in Definition 4.1 is satisfied.

Next, consider another attacker $\phi_2 \in \Phi(E_{era}, E_{ins})$ with the following attack function

$$\xi_2(s) = \begin{cases} \varepsilon, & \text{for } s = \varepsilon; \\ a_b_e_adb^k, & \text{for } s = adb^k, k \in \mathbb{N} \end{cases}$$

That is, the attacker ϕ_2 first erases events a and b and then inserts event e while observing adb^k . We see that L is not robustly diagnosable against the attacker ϕ_2 . For all arbitrarily long faulty words $afbdb^k$, the attacker observes $P(afbdb^k) = abdb^k$ and produces the corrupted observation $\phi_2(abdb^k) = \hat{P}(\xi_2(abdb^k)) = edb^k$, which can be produced by a normal word $ecdb^k \in L$. In plain words, when the operator perceives edb^k , it is always not certain of the fault occurrence, as it should be in the nominal case. Finally, since there exists an attacker $\phi_2 \in \Phi(E_{era}, E_{ins})$ such that condition $\mathbf{C}'_{AD}$ in Definition 4.1 is not satisfied, we conclude that L is not robustly diagnosable against $\Phi(E_{era}, E_{ins})$. $\diamond$

4.3 Attack Model

A real sensor attacker can corrupt the recorded observed events by sensors, which makes sense in practice. In this chapter, we assume that a virtual attacker (represented in dashed lines in the bottom of Fig. 4.3), equivalent to a sensor attacker in mathematical terms, has the full observation of events in E . The motivation for such an assumption will be clear in the text. Inspired by the work in [77], we now extend attack function $\xi : P_o(L(G)) \rightarrow E_a^*$ on observation $s \in P_o(L(G))$ to virtual attack function $\xi' : L(G) \rightarrow E_w^*$ on strings $\sigma \in L(G)$, where $E_w = E \cup E_+ \cup E_- = E_a \cup E_{uo}$. A virtual attack function ξ' satisfies the definition of the attack function if $E_o = E$. Also, we denote by Ξ' the set of all possible virtual attack functions. Word $\sigma_a = \xi'(\sigma)$ is an *attack word*, which describes the action of the virtual attack function on the word $\sigma \in L(G)$.

Similarly, we assume that there exists a virtual operator that has a full observation of the events in E . In this regard, functions $\hat{P} : E_a^* \rightarrow E_o^*$ and $P_o : E^* \rightarrow E_o^*$ can be extended to new versions $\hat{P}' : E_w^* \rightarrow E^*$ and $P_o' : E_w^* \rightarrow E_a^*$, respectively. Word $\sigma' = \hat{P}'(\sigma_a)$ is called *the corrupted word*, which is perceived by the *virtual operator*.

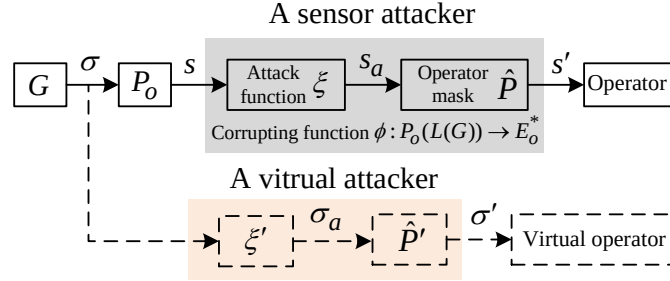


Figure 4.3 A fault diagnosis system under virtual attack.

The relations among these functions are shown in Fig. 4.4, where the part in black corresponds to the upper scheme in Fig. 4.3, while the part in red is related to the newly defined functions (the bottom scheme of Fig. 4.3). Given a word $\sigma \in L(G)$, it holds that $s_a = \xi(P(\sigma)) = P'(\xi'(\sigma))$. Given an attack word σ_a , $s' = P_o(\hat{P}'(\sigma_a)) = \hat{P}(P_o'(\sigma_a))$ holds.

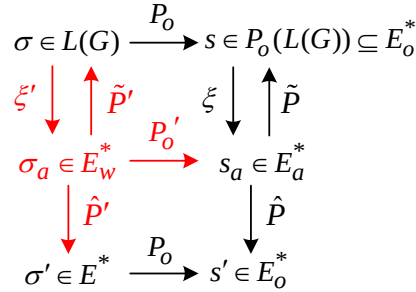


Figure 4.4 Relations between the attack functions and projections.

4.4 Faulty and Normal Automaton Under Attack

Diagnosability verification can be performed in the nominal setting based on verifier automata [38-39, 41]. The basic idea is to seek ambiguities (words in the language generated by G with identical projections with respect to E_o^*) between faulty and normal words. The verifier presented in [41] is deterministic and its construction has the least computational complexity among the verification algorithms. In [41], it is first necessary to build the label automaton $A_\ell = (X_{A_\ell}, E_f, \eta_{A_\ell}, x_{0,A_\ell})$, where $X_{A_\ell} = \{N, F\}$, $x_{0,A_\ell} = \{N\}$, $\eta_{A_\ell}(N, f) = F$ and $\eta_{A_\ell}(F, f) = F$ with $E_f = \{f\}$. Then, $G_\ell = G \parallel A_\ell = (X_\ell, E, \eta_\ell, x_{0_N})$ is com-

puted, where $L(G_\ell) = L(G)$. The following two automata are then obtained from G_ℓ :

- *faulty automaton* $G_F = (X_F, E, \eta_F, x_{0_N})$ that models the faulty behavior of G ; it is obtained by marking the states of G_ℓ that are labeled with F , and then taking the co-accessible part;
- *normal automaton* $G_N = (X_N, E \setminus E_f, \eta_N, x_{0_N})$ that models the normal behavior of G ; it is obtained by marking the states of G_ℓ that are labeled with N , and then taking the co-accessible part.

Next, let us consider the situation where some of the events of G are subject to sensor attacks. With a similar idea to the nominal setting, we introduce two automata: *faulty automaton under attack* and *normal automaton under attack*. We first take the perspective of the attacker to capture its ability to corrupt system words. Given a faulty automaton G_F , the faulty automaton under attack G_{FA} is proposed to capture all possible sensor attacks on faulty words generated by G .

Definition 4.2. Given a faulty automaton $G_F = (X_F, E, \eta_F, x_{0_N})$ with sets E_{ins} and E_{era} , the faulty automaton under attack is $G_{FA} = (X_F, E_w, \eta_{FA}, x_{0_N})$, where the transition function η_{FA} is defined as follows: for all $x_f \in X_F$,

$$\begin{cases} \eta_{FA}(x_f, e) := \eta_F(x_f, e), & \forall e \in E; \\ \eta_{FA}(x_f, e_+) := x_f, & \forall e \in E_{ins}; \\ \eta_{FA}(x_f, e_-) := \eta_F(x_f, e), & \forall e \in E_{era} \cap \Gamma(x_f). \end{cases} \quad \diamond$$

From this definition, G_{FA} has the same set of states of G_F . An event $e \in E$ in G_{FA} corresponds to an event actually occurring in the plant without attacks; thus, for all $x_f \in X_F$, for all $e \in E$, it is $\eta_{FA}(x_f, e) = \eta_F(x_f, e)$. The attacker can insert any word in E_{ins}^* between the occurrences of two events in G . On the other hand, the attacker can also erase the observation of an observable event $e \in E_{era}$ after its occurrence in G , with the erasure being represented by event e_- .

Example 4.2. Consider the plant G in Fig. 4.2(a). Its faulty automaton G_F is shown in Fig. 4.5(a). Let $E_{ins} = \{e\}$ and $E_{era} = \{a, b\}$. The corresponding faulty automaton under attack G_{FA} is shown in Fig. 4.5(b). To make the construction more illustrative, additional transitions of events in E_+ and E_- are colored in red and blue, respectively. We have self-loop transitions labeled with the inserted event e_+ at all the states of G_{FA} since the attacker knows that the

inserted event e_+ is fake. On the other hand, there exists a transition labeled with a_- from state $0N$ to $1N$ in G_{FA} since the attacker knows that the event a has occurred. $\diamond$

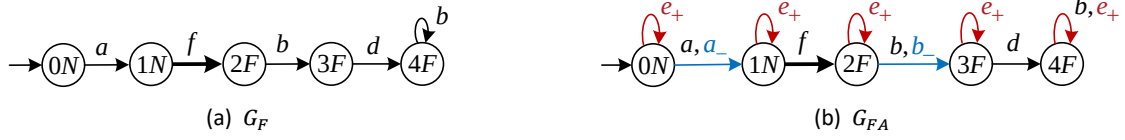


Figure 4.5 (a) G_F and (b) G_{FA} for G in Fig. 4.2(a).

Remark 4.1. We point out that the faulty automaton under attack assumes that events in E_+ can be inserted after the occurrence of each event since all events are observable for the virtual attacker. A real attacker, however, can only insert events after the occurrence of an observable event. As an example, given the G_{FA} in Fig. 4.5(b), we can see that a virtual attacker after observing event a may decide to insert e_+ at state $1N$ and the second e_+ at state $2F$ so that attack word is ae_+fe_+ . A real attacker, on the contrary, may decide to insert two e_+ after a , but since f is not observable, it cannot determine if the attack word produced is ae_+^2f or ae_+fe_+ or afe_+^2 . Nevertheless, all possible actions of the virtual attacker correspond to a possible scenario executed by the real attacker. Hence, the virtual attacker can destroy diagnosability if and only if the real attacker can. For this reason, with no loss of generality, we may only consider virtual attackers. $\diamond$

Now we give a formal characterization of G_{FA} as follows.

Proposition 4.1. Given a faulty automaton $G_F = (X_F, E, \eta_F, x_{0_N})$, let $G_{FA} = (X_F, E_w, \eta_{FA}, x_{0_N})$ be the faulty automaton under attack. It holds that

- 1) $\sigma_a \in L(G_{FA}) \Leftrightarrow (\exists \xi' \in \Xi')(\exists \sigma \in L(G_F)) [\sigma_a = \xi'(\sigma)]$;
- 2) $(\forall \xi' \in \Xi')(\forall \sigma \in L(G_F)) [\eta_{FA}(x_0, \xi'(\sigma)) = \eta_F(x_0, \sigma)]$.

Proof: We first consider Statement 1). $(\Leftarrow)$ By the definition of η_{FA} , all possible attacks implemented by a virtual attacker on faulty words of G have been included in G_{FA} . Therefore, given a word $\sigma \in L(G_F)$ and $\xi' \in \Xi'$, there exists a corresponding attack word $\sigma_a = \xi'(\sigma)$ such that $\sigma_a \in L(G_{FA})$.

$(\Rightarrow)$ Consider a word $\sigma_a \in L(G_{FA})$ that reaches a state x and only contains events in E , which represents an original evolution that has not been attacked. At each state of G_{FA} all events e_+ are in self-loop, which implies that σ_a can be corrupted by inserting a word of fake events in E_{ins} . In addition, if the word σ_a is generated by executing a transition $\eta_{FA}(x'_f, e) = x''_f$ with $e \in E_{era}$, it is also possible to execute the “parallel” transition $\eta_{FA}(x'_f, e_-) = x''_f$

corresponding to an attack that erases e . Therefore, there exist a function $\xi' \in \Xi'$ and a word $\sigma \in L(G_F)$ such that $\xi'(\sigma) = \sigma_a$.

For Statement 2), following Statement 1), the original word and all the attack ones reach the same state x , i.e., $\eta_{FA}(x_{0N}, \xi'(\sigma)) = \eta_F(x_{0N}, \sigma)$. $\square$

By Statement 1) in Proposition 4.1, the language of G_{FA} is the set of all attack faulty words of G_F , i.e., all words σ_a , which may correspond to a faulty word σ generated by G (also perceived by the virtual attacker). By Statement 2), the state reached in G_F by σ and the state reached in G_{FA} by $\sigma_a = \xi'(\sigma)$ coincide.

Now let us take the perspective of the operator to capture its ability to perceive attack words. Based on a normal automaton G_N , the normal automaton under attack G_{NA} is proposed to describe how any word $\sigma_a \in E_w^*$ is perceived by the operator as normal words generated by G .

Definition 4.3. Given a normal automaton $G_N = (X_N, E \setminus E_f, \eta_N, x_{0N})$ with sets E_{ins} and E_{era} , the normal automaton under attack is $G_{NA} = (X_N, E_w \setminus E_f, \eta_{NA}, x_{0N})$, where the transition function η_{NA} , for all $x_n \in X_N$, is defined as

$$\begin{cases} \eta_{NA}(x_n, e) := \eta_N(x_n, e), & \forall e \in E; \\ \eta_{NA}(x_n, e_-) := x_n, & \forall e \in E_{era}; \\ \eta_{NA}(x_n, e_+) := \eta_N(x_n, e), & \forall e \in E_{ins} \cap \Gamma(x_n). \end{cases} \quad \diamond$$

As defined, G_{NA} also has the same set of states of G_N . An event $e \in E$ in G_{NA} corresponds to an event perceived by the operator without attacks; thus, for all $x_n \in X_N$, for all $e \in E$, $\eta_{NA}(x_n, e) = \eta_N(x_n, e)$ holds. If the attacker erases an event $e \in E_{era}$, the operator cannot observe the occurrence of e and thus $\eta_{NA}(x_n, e_-) = x_n$. If the attacker inserts an event e_+ , the operator cannot distinguish between the inserted event e_+ and the corresponding event $e \in E_o$, thus updating its state in the same way, i.e., $\eta_{NA}(x_n, e_+) = \eta_N(x_n, e)$.

Example 4.3. Consider again the plant G in Fig. 4.2(a) with its normal automaton G_N in Fig. 4.6(a). The corresponding normal automaton under attack G_{NA} is shown in Fig. 4.6(b). We have self-loop transitions labeled with the erased events a_- and b_- at all the states of G_{NA} , since the events have been erased by the attacker and the operator receives no information regarding such events occurrences. On the other hand, there exists a transition labeled with e_+ from state $\{0N\}$ to $\{5N\}$ in G_{NA} since the operator cannot distinguish e_+ and the corresponding e . $\diamond$

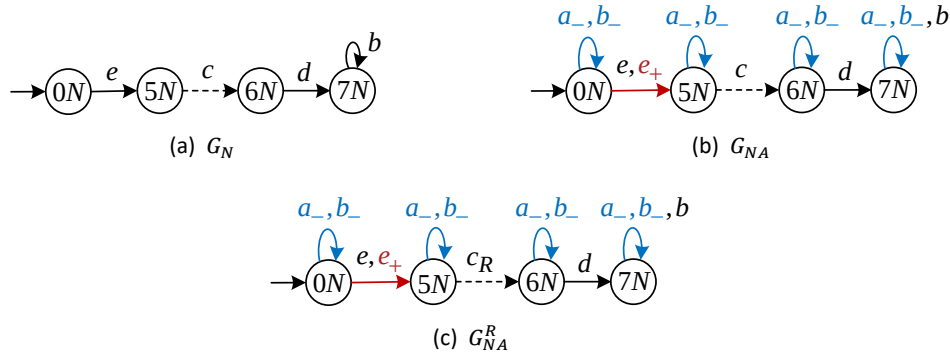


Figure 4.6 (a) G_N , (b) G_{NA} , and (c) G_{NA}^R for G in Fig. 4.2(a).

The following result provides a formal characterization of G_{NA} .

Proposition 4.2. *Given a normal automaton $G_N = (X_N, E \setminus E_f, \eta_N, x_{0N})$, let $G_{NA} = (X_N, E_w \setminus E_f, \eta_{NA}, x_{0N})$ be the normal automaton under attack. It holds that*

- 1) $\sigma_a \in L(G_{NA}) \Leftrightarrow (\exists \sigma' \in L(G_N)) [\sigma_a \in \hat{P}'^{-1}(\sigma')]$;
- 2) $(\forall \sigma_a \in L(G_{NA})) [\eta_{NA}(x_0, \sigma_a) = \eta_N(x_0, \hat{P}'(\sigma_a))]$.

Proof: Statement 1) is first considered. ($\Leftarrow$) By the definition of η_{NA} , G_{NA} describes how all words $\sigma \in E_w^*$ are perceived by the virtual operator. Therefore, given a word $\sigma' \in L(G_N)$, there exists a corresponding attack word $\sigma_a \in \hat{P}'^{-1}(\sigma')$ such that $\sigma_a \in L(G_{NA})$.

($\Rightarrow$) Consider a word $\sigma_a \in L(G_{NA})$ that reaches a state x and only contains the events in E , which implies that an original evolution has been perceived by the virtual operator. Second, at each state of G_{NA} , all events e_- are in self-loop, which corresponds to the generation of σ_a in G_{NA} . By the definition of $\hat{P}'$, it holds that $\sigma' = \hat{P}'(\sigma_a) \in L(G_N)$. In addition, if the word σ_a is generated in G_{NA} by executing a transition $\eta_{NA}(x', e_+) = x''$ with $e \in E_{ins}$, it is also possible to execute the “parallel” transition $\eta_{NA}(x', e) = x''$ and thus $\sigma' = \hat{P}'(\sigma_a) \in L(G_N)$. Therefore, there exists a word $\sigma' \in L(G_N)$ such that $\sigma' = \hat{P}'(\sigma_a)$, i.e., $\sigma_a \in \hat{P}'^{-1}(\sigma')$.

For Statement 2), we prove by induction on the length of σ_a . If $\sigma_a = \varepsilon$, the result holds with $\hat{P}'(\sigma_a) = \varepsilon$. Consider now a word $\sigma_a \in L(G_{NA})$ with length greater than one. Let $\sigma_a = \sigma'_a e_a$. Assume that the result holds for a generic σ'_a , i.e., $\eta_{NA}(x_0, \sigma'_a) = \eta_N(x_0, \hat{P}'(\sigma'_a))$. The following is to prove $\eta_{NA}(x_0, \sigma_a) = \eta_N(x_0, \hat{P}'(\sigma_a))$. Clearly, $\eta_{NA}(x_0, \sigma_a) = \eta_{NA}(\eta_{NA}(x_0, \sigma'_a), e_a)$. By the definition of $\hat{P}'$, $\eta_N(x_0, \hat{P}'(\sigma_a)) = \eta_N(x_0, \hat{P}'(\sigma'_a e_a)) = \eta_N(\eta_N(x_0, \hat{P}'(\sigma'_a)), e_a)$ if $e_a \in E \cup E_+$, and $\eta_N(x_0, \hat{P}'(\sigma_a)) = \eta_N(x_0, \hat{P}'(\sigma'_a e_a)) = \eta_N(\eta_N(x_0, \hat{P}'(\sigma'_a)), \varepsilon)$ if $e_a \in E_-$. By Definition 4.3, the transition function of G_{NA} starting for a generic state $x \in X$ is defined in the same way in case of e and e_+ , while it corresponds to a self-loop in the case

of $e_- \in E_-$. As a result, we conclude that $\eta_{NA}(x_0, \sigma'_a) = \eta_N(x_0, \hat{P}'(\sigma'_a))$. $\square$

According to Statement 1) in Proposition 4.2, $L(G_{NA})$ includes all normal words $\sigma \in E_w^*$ that can be perceived by the virtual operator as a normal word $\sigma' \in G_N$. By Statement 2), the state reached in G_{NA} by σ_a coincides with the state reached in G_N by $\sigma' = \hat{P}'(\sigma_a)$.

4.5 Verification Under Attack Using Verifiers

In the nominal setting [41], when constructing a verifier using the parallel composition of faulty automaton and normal automaton, only the observable events in E_o should be synchronized, while the events in $E_{uo} \setminus E_f$ cannot be. This is done by introducing a renaming function. Similarly, we now give the notion of a *renaming function under attack* as follows.

Definition 4.4. The renaming function $R : E_w \setminus E_f \rightarrow E_R$ is a bijective mapping, where

$$R(e) = \begin{cases} e, & \text{if } e \in E_a; \\ e_R, & \text{if } e \in E_{ruo}. \end{cases}$$

As defined, R only renames regular unobservable events in E_{ruo} . Hence it gives $E_R = E_a \cup E'_R$, where E'_R is defined as $E'_R = \{e_R \mid (\exists e \in E_{ruo}) e_R = R(e)\}$. The inverse renaming function is the mapping $R^{-1} : E_R \rightarrow E_w \setminus E_f$, where $R^{-1}(e_R) = e$ and $R^{-1}(e) = e$. The renaming function (resp., inverse renaming function) can be extended to words, as follows: $R(\sigma e) = R(\sigma)R(e)$ (resp., $R^{-1}(\sigma_R e_R) = R^{-1}(\sigma_R)R^{-1}(e_R)$), where $\sigma \in (E_w \setminus E_f)^*$ and $e \in E_w \setminus E_f$ (resp., $\sigma_R \in E_R^*$ and $e_R \in E_R$).

Definition 4.5. Given a normal automaton under attack $G_{NA} = (X_N, E_w \setminus E_f, \eta_{NA}, x_{0N})$, the renamed normal automaton under attack is $G_{NA}^R = (X_N, E_R, \eta_{NA^R}, x_{0N})$, where the transition function η_{NA^R} , for all $e \in E_w \setminus E_f$, is defined as $\eta_{NA^R}(x_N, R(e)) = \eta_{NA}(x_N, e)$. $\diamond$

Now the event sets of G_{FA} and G_{NA}^R are respectively given by E_w and E_R . From $E_w = E_a \cup E_{uo}$, $E_R = E_a \cup E'_R$, and $E_{uo} \cap E'_R = \emptyset$, we find that the unobservable events in E_{uo} and the renamed events in E'_R are respectively private events of G_{FA} and G_{NA}^R , while the events in E_a belong to both automata. In Example 4.3, Fig. 4.6(c) shows the renamed normal automaton under attack G_{NA}^R , where regular unobservable event $c \in E_{ruo}$ is renamed as c_R . With the above concepts, a verifier (that is deterministic) can be constructed through standard parallel composition, as shown in the following definition.

Definition 4.6. Given $G_{FA} = (X_F, E_w, \eta_{FA}, x_{0N})$ and $G_{NA}^R = (X_N, E_R, \eta_{NA^R}, x_{0N})$, the verifier under attack is defined as $G_{VA} = (X_{VA}, E_w \cup E_R, \eta_{VA}, x_{0,va}) = G_{FA} \parallel G_{NA}^R$. $\diamond$

Let us define now the following projections:

$$\begin{aligned} P_w &: (E_w \cup E_R)^* \rightarrow E_w^*, \\ P_R &: (E_w \cup E_R)^* \rightarrow E_R^*. \end{aligned}$$

A necessary and sufficient condition for robust diagnosability can be derived with the help of the following lemma.

Lemma 4.1. *Let $G_{VA} = G_{FA} \parallel G_{NA}^R$. Then, for every $\sigma_V \in L(G_{VA})$, there exists attack words $\sigma_a \in L(G_{FA})$ and $\sigma'_a \in L(G_{NA})$, such that $P'_o(\sigma_a) = P'_o(\sigma'_a)$.*

Proof: Let us consider a word $\sigma_V \in L(G_{VA})$, and let $\sigma_R = P_R(\sigma_V)$ and $\sigma_a = P_w(\sigma_V)$. By $G_{VA} = G_{FA} \parallel G_{NA}^R$, we have $L(G_{VA}) = P_w^{-1}[L(G_{FA})] \cap P_R^{-1}[L(G_{NA}^R)]$, and thus, $\sigma_R \in L(G_{NA}^R)$ and $\sigma_a \in L(G_{FA})$. Since G_{NA}^R is obtained from G_{NA} after renaming the events in $E_w \setminus E_f$ according to function R , there always exists a word σ'_a such that $P'_o(\sigma'_a) = P_w(\sigma_R)$. From $P_w(\sigma_R) = P_w[P_R(\sigma_V)] = P_R[P_w(\sigma_V)] = P_R(\sigma_a)$, it comes $P'_o(\sigma'_a) = P_R(\sigma_a)$. By $\sigma_a \in L(G_{FA}) \subseteq E_w^*$, $P_R(\sigma_a) = P'_o(\sigma_a)$ yields $P'_o(\sigma'_a) = P'_o(\sigma_a)$. $\square$

Theorem 4.1. *Let L be the language generated by G . Then, L is robustly diagnosable against $\Phi(E_{era}, E_{ins})$ if and only if the verifier under attack G_{VA} does not contain a cycle*

$$cl = x_{va}^k \xrightarrow{e_k} x_{va}^{k+1} \xrightarrow{e_{k+1}} \dots x_{va}^l \xrightarrow{e_l} x_{va}^k$$

with $l \geq k > 0$, being of the form $x_{va}^k = (x_f^k, \bar{x}_n^k) = ((x^k, \ell^k), (\bar{x}^k, N))$, satisfying the following condition:

$$(\exists j \in \{k, k+1, \dots, l\}) [\ell^j = F \wedge e_j \in E_- \cup E]. \quad (4-1)$$

Proof: (only if) By contrapositive, assume that there exists a cycle $cl = x_{va}^k \xrightarrow{e_k} x_{va}^{k+1} \xrightarrow{e_{k+1}} \dots x_{va}^l \xrightarrow{e_l} x_{va}^k$, where $l \geq k > 0$, in G_{VA} , satisfying condition (4-1). Since $x_f^j = (x^j, F)$ for some $j \in \{k, k+1, \dots, l\}$, from the constructions of A_ℓ and G_{VA} , it can be seen that $x_f^j = (x^j, F)$ for all $j \in \{k, k+1, \dots, l\}$. This implies that there exists a word $\sigma_V t_V \in L(G_{VA})$ such that $E_f \in \sigma_V$, and $t_V = (e_k e_{k+1} \dots e_l)^p$, where $p \in \mathbb{N}$ and $|t_V| > n$, for all $n \in \mathbb{N}$.

According to Lemma 4.1, there must exist an attack faulty word $\sigma_a = P_w(\sigma_V t_V) \in L(G_{FA})$ and an attack normal normal word $\sigma' \in L(G_{NA})$, such that $P'_o(\sigma_a) = P'_o(\sigma'_a)$. Let $\sigma_a = \tilde{\sigma} \tilde{t}$ with $\tilde{\sigma} = P_w(\sigma_V)$ and $\tilde{t} = P_w(t_V)$. Note that by the assumption that there exists

an event $e_j \in E \cup E_-$ for $j \in \{k, k+1, \dots, l\}$, the word $\tilde{t} = P_w(t_V)$ can also be of arbitrary length, which implies that $\sigma_a \in L(G_{FA})$ is an attack faulty word of arbitrary length after a fault event $f \in E_f$.

By Propositions 4.1 and 4.2, there exists a faulty word $\sigma t \in L(G_F) \subseteq L(G)$ of arbitrary length after a fault event $f \in E_f$, and a normal word $\sigma' \in L(G_N) \subseteq L(G)$, such that $\sigma_a = \xi'(\sigma t)$ and $\sigma' = \hat{P}'(\sigma'_a)$. Let $s_a = P'_o(\sigma'_a) = P'_o(\sigma_a)$. Now, from $s_a = P'_o(\sigma_a) = P'_o(\xi'(\sigma t)) = \xi(P_o(\sigma t))$ and $P_o(\sigma') = P_o(\hat{P}'(\sigma'_a)) = \hat{P}(P'_o(\sigma'_a)) = \hat{P}(s_a)$, we derive

$$\phi(P_o(\sigma t)) = \hat{P}(\xi(P_o(\sigma t))) = \hat{P}(s_a) = P_o(\sigma').$$

Hence, the robust diagnosability condition in Definition 3.4 is not satisfied, i.e., the ambiguous words σt and σ' violate the diagnosability.

(if) By contrapositive, without loss of generality, suppose now that L is not robustly diagnosable against a sensor attacker $\phi \in \Phi(E_{era}, E_{ins})$. Thus, there exist words $\sigma t \in L(G_F)$ with $E_f \in \sigma$ and $|t| > n$, for all $n \in \mathbb{N}$, and $\sigma' \in L(G_N)$ with $E_f \notin \sigma'$, such that $\phi(P_o(\sigma t)) = P_o(\sigma')$. By Propositions 4.1 and 4.2, there accordingly exist words $\tilde{\sigma}\tilde{t} \in L(G_{FA})$, where $E_f \in \tilde{\sigma}$ and $|\tilde{t}| > n$, for all $n \in \mathbb{N}$, and $\sigma'_a \in L(G_{NA})$ such that $\tilde{\sigma}\tilde{t} = \xi'(\sigma t)$ and $\sigma'_a \in \hat{P}'^{-1}(\sigma')$. By the definition of the corrupting function ϕ , we have $P'_o(\tilde{\sigma}\tilde{t}) = P'_o(\sigma'_a)$. We will show that G_{VA} has a cycle associated with words $\tilde{\sigma}\tilde{t}$ and σ'_a . To this end, the proof is split into two parts as follows:

1. we first show that there exists an arbitrarily long word $\sigma_V t_V \in L(G_{VA})$ such that $P_w(\sigma_V t_V) = \tilde{\sigma}\tilde{t}$ and $P_R(\sigma_V t_V) = \sigma_R$, where $\sigma_R = R(\sigma'_a)$;
2. we then prove that there exists a cycle cl , associated with the word $\sigma_V t_V$, satisfying condition (4-1).

For part 1), suppose that there exists a state $x_{va} = (x_f, x_n)$ in G_{VA} reachable from $x_{0,va}$ after the execution of a word $u \in L(G_{VA})$, where $P_w(u) \leq \tilde{\sigma}\tilde{t}$. Note that this state x_{va} always exists since u can be the empty word; if $u = \varepsilon$, it is $x_{va} = x_{0,va}$. Let $e \in E_w$ be an active event of x_f such that $P_w(u)e \leq \tilde{\sigma}\tilde{t}$. Now we consider the problem of finding a state of G_{VA} , called $\hat{x}_{va}$, reachable from x_{va} , that has $e \in E_w$ as an active event. Two cases are possible: e is an unobservable event, i.e., $e \in E_{uo}$; otherwise, $e \in E_a$. Let us first consider $e \in E_{uo}$. In this case, since a self-loop labeled with all the events in E_{uo} is adjoined to each state of G_{NA}^R in order to perform the parallel composition $G_{VA} = G_{FA} \parallel G_{NA}^R$, we conclude that e is already active at $x_{va} = (x_f, x_n)$. Consider now $e \in E_a$. In this case, e will be an active event of x_{va} if and only if it is active at the corresponding states of G_{NA}^R . Since

$P'_o(\sigma'_a) = P'_o(\tilde{\sigma}\tilde{t})$, e will be active at a state $\hat{x}_{va} = (x_f, \hat{x}_n)$ of G_{VA} , after the occurrence of a finite word of renamed events in E'_R . Therefore, it can be verified that there exists an arbitrarily long word $\sigma_V t_V$ associated with $\tilde{\sigma}\tilde{t}$ such that $\sigma_V t_V \in P_R^{-1}(\sigma_R) \cap P_w^{-1}(\tilde{\sigma}\tilde{t})$, implying $P_w(\sigma_V t_V) = \tilde{\sigma}\tilde{t}$ and $P_R(\sigma_V t_V) = \sigma_R$.

Now we prove part 2), i.e., associated with $\sigma_V t_V$ there exists a cycle cl in G_{VA} , with its states being in the form $x_{va} = (x_f, x_n)$, where $x_f = (x, \ell)$ with $\ell \in \{N, F\}$, such that $\ell = F$ and at least one of the events in the cycle belongs to $E \cup E_-$. Since G_{FA} is a finite state automaton, associated with $\tilde{\sigma}\tilde{t}$ of arbitrary length there is a cycle of faulty states cl_F in G_{FA} . Then, the cycle cl_F corresponds to a path in G_{VA} , where the events of cl_F are contained in this path. Since G_{VA} is a finite state automaton, this path must include a cycle cl . It can be seen from the construction of G_{VA} that cl is a cycle of states whose first components are faulty states, i.e., $\ell = F$. In addition, the word $\tilde{\sigma}\tilde{t}$ must not be exclusively formed by the inserted events in E_+ , since σt satisfying $\tilde{\sigma}\tilde{t} = \xi'(\sigma t)$ is of arbitrary length. The word $\sigma_V t_V$ must not be exclusively formed by the renamed events in E'_R , as $\tilde{\sigma}\tilde{t} = P_w(\sigma_V t_V)$ is also of arbitrary length. As a result, cl must not be connected with events in $E_+ \cup E'_R$ only. In other words, it necessarily contains at least one event in $(E_w \cup E_R) \setminus (E_+ \cup E'_R) = E \cup E_-$. $\square$

As is the case of the classical diagnosability verification [41], the search carried out in Theorem 4.1 seeks ambiguous words in $L(G)$ violating robust diagnosability, i.e., an arbitrarily long faulty word σt and a normal word σ' with the same observation under appropriate attacks. Assume that word $\sigma_V t_V$ forms a cycle satisfying condition (3-5) in Theorem 4.1. First, the cycle contains at least a state whose first element carries label F . In addition, the cycle should not be exclusively caused by the inserted events or the renamed events, since along the cycle exclusively formed with the inserted events or the renamed events, the number of events generated after the fault does not increase. From such a cycle, we have:

1. Its first entry captures the arbitrarily long faulty attack word $\sigma_a = P_w(\sigma_V t_V) \in L(G_{FA})$. By Proposition 4.1, σ_a can correspond to an arbitrarily long faulty word σt generated by G such that $\sigma_a = \xi'(\sigma t)$.
2. Its second entry captures the normal attack words $\sigma'_a = R^{-1}(P_R(\sigma_V t_V)) \in L(G_{NA})$. According to Proposition 4.2, σ'_a can correspond to a normal word σ' perceived by the virtual operator such that $\sigma' = \hat{P}'(\sigma'_a)$.

Finally, ambiguous words, i.e., the arbitrarily long faulty word σt and the normal word σ' that satisfy $\phi(P(\sigma t)) = P(\sigma')$ where $\phi \in \Phi(E_{era}, E_{ins})$, are found.

Example 4.4. Let us visit automaton G in Fig. 4.2(a), where $E_o = \{a, b, d, e\}$, $E_{uo} = \{c, f\}$, and $E_f = \{f\}$. Its faulty automaton under attack G_{FA} and renamed normal automaton under attack G_{NA}^R are respectively shown in Figs. 4.5(b) and 4.6(c). Then, the verifier under attack $G_{VA} = G_{FA} \parallel G_{NA}^R$ is reported in Fig. 4.7. Since there exists a cycle $cl = (4F, 7N) \xrightarrow{b} (4F, 7N)$ (highlighted with green lines) satisfying condition (4-1), by Theorem 4.1, L is not robustly diagnosable against $\Phi(E_{era}, E_{ins})$ in the case of $E_{era} = \{a, b\}$ and $E_{ins} = \{e\}$.

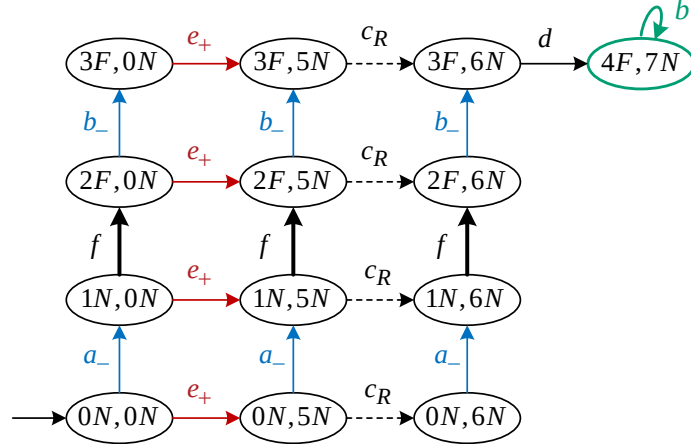


Figure 4.7 G_{VA} for G in Example 4.4, where the cycle satisfying condition (4-1) is highlighted with green lines.

From G_{VA} , associated with the above cycle is an attack word $\sigma_V t_V = a_- f b_- e_+ c_R d b^k$. Based on it, we can derive two words in E_w^* : an arbitrarily long attack faulty word $\sigma_a = \tilde{\sigma} \tilde{t} = P_w(\sigma_V t_V) = a_- f b_- e_+ d b^k$ in $L(G_{FA})$, and a renamed attack normal word $\sigma_R = P_R(\sigma_V t_V) = a_- b_- e_+ c_R d b^k$, $k \in \mathbb{N}$, in $L(G_{NA}^R)$. After applying R^{-1} to σ_R , we obtain an attack normal word $\sigma'_a = a_- b_- e_+ c d b^k \in L(G_{NA})$, for which $P'_o(\sigma'_a) = P'_o(\sigma_a) = a_- b_- e_+ d b^k$ holds. Then, according to Propositions 4.1 and 4.2, there exists a faulty word $\sigma t = a f b d b^k \in L(G_F)$ of arbitrary length after a fault event f , and a normal word $\sigma' = \hat{P}'(\sigma'_a) = e c d b^k \in L(G_N)$. As explained in the proof of Theorem 4.1, it holds that $P_o(\sigma') = \phi(P_o(\sigma t))$, i.e., $P(ecdb^k) = \phi(P(afbdb^k)) = edb^k$. This means that the attacker captured in G_{VA} that violates robust diagnosability is exactly ϕ_2 defined in Example 4.1; thus the diagnosability verification result agrees with that in Example 4.1. $\diamond$

4.6 Case Study

4.6.1 A Power System

This subsection again considers the power system introduced in Section 3.6, where it is shown that the system is not robustly diagnosable against the re-defined class of attackers,

i.e, $E_{era} = \{OC\}$ and $E_{ins} = \{ZC\}$. The result can also be verified with the verifier under attack, part of which is shown in Fig. 4.8, where we only focus on cycles, for example, $7F9N \xrightarrow{OC-ZC_+} 7F9N$, satisfying condition (4-1), as highlighted with green lines and omit other parts with loss of generality for the purpose of verification.

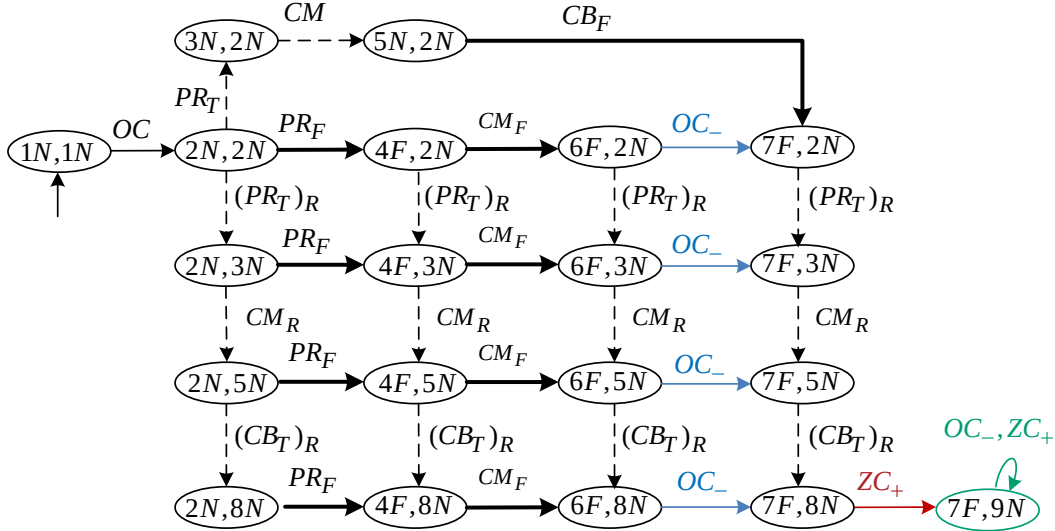
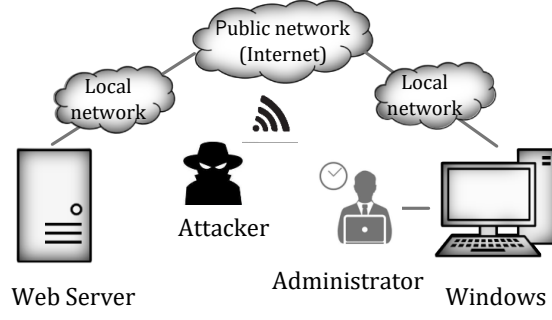


Figure 4.8 Part of verifier G_{SAV} for the power system in Section 3.6.

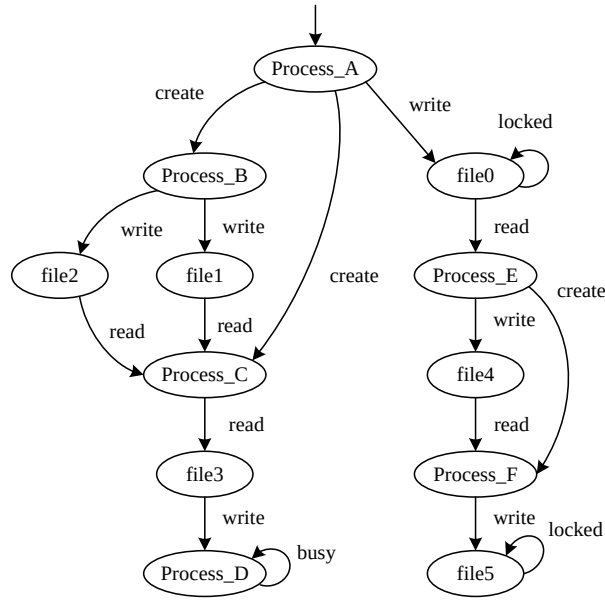
4.6.2 A Networked Computer System

In this subsection, we consider a networked computer system shown in Fig. 4.9(a) modified from [92]. It includes a Web server and a target Windows host on a local network, which are interconnected via a public communication network. Such networks have security holes that escape the notice of the system administrator. Hence, the vulnerabilities are potentially exploited by an attacker to launch its attacks.

The system is instrumented with security logs that record the events like user authentication, network traffic, email activity, and access to services or files. Consider an event log to be a known or possible fault. The system administrator's objective is to analyze the sequences of event logs that lead to a fault, which may be compromised by an intruder, and quickly take corrective actions. Dependency graph (DG) is a technique used in computer systems that records the commands issued via a log file. In a DG, there are two types of states, PROCESSES and FILES, and transitions between states define the dependencies of the command issued. We consider the sample event log in Table 4.1. Based on it, the DG is built in the form of the automaton G shown in Fig. 4.9(b). The event set is $E = \{busy, locked, create, read, write\}$. Suppose that the set of observable events is $E_o = \{create, busy, locked\}$. Assuming that event *read* is a fault event, we expect to diagnose its occurrence.



(a) A networked computer system



(b) G

Figure 4.9 (a) a networked computer system and (b) its dependency graph model G built from the sample event log in Table 4.1.

It is seen that without attacks, L is diagnosable. Intuitively, after the occurrence of *read* the operator will observe event *busy*, while under normal operation, it will observe event *locked*.

Now we are interested in a scenario where an attacker can create false and/or remove legitimate log information transmitted over the network. Let the set of event logs subject to SE-attacks and SI-attacks be $E_{era} = \{create, busy\}$ and $E_{ins} = \{locked\}$, respectively. Each of the compromised event logs corresponds to a real-world vulnerability. Hence, it gives $E_- = \{create_-, busy_-\}$ and $E_+ = \{locked_+\}$. Assume that there exists a network attacker ϕ

Table 4.1 The sample event log

Time	Log
0	process_O reads creates_A
1	process_A creates process_B
2	process_B writes file_1
3	process_B writes file_2
4	process_A writes file_0
5	file_0 is locked
6	process_A creates process_C
7	process_C reads file_1
8	process_C writes file_3
9	process_D reads file_3
10	process_C reads file_2
11	process_D is busy

with the following attack function ($k \in \mathbb{N}$):

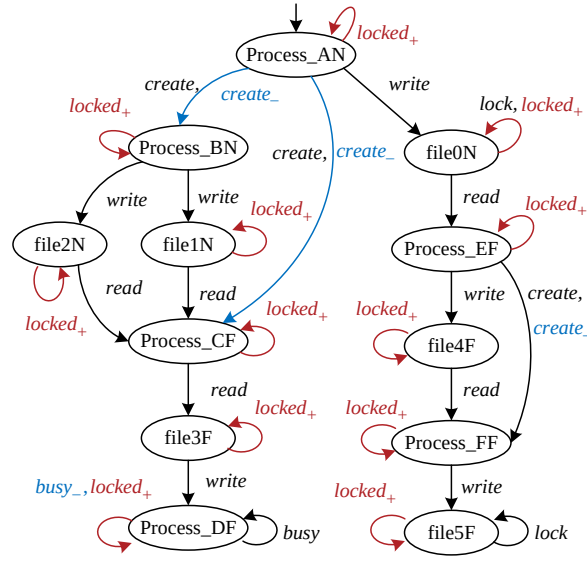
$$\xi(s) = \begin{cases} \varepsilon, & \text{for } s = \varepsilon; \\ \text{create_}(busy_locked_+)^k, & \text{for } s = \text{create}(busy)^k. \end{cases}$$

Its physical interpretation is that the attacker erases the event log *create*. After this, every time a *busy* event log is recorded, the attacker chooses to erase it and insert a *locked* event log. Hence, the *busy* event log is tampered with by *locked* event log due to the attacker ϕ .

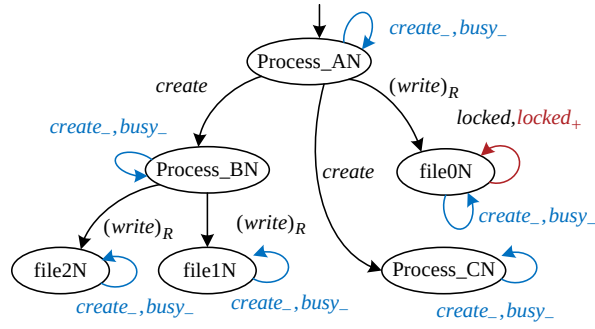
After the execution of faulty operations $(create)(write)(read)^2(write)(busy)^k$, the attacker observes $(create)(busy)^k$ and implements its attacks, i.e., $\phi((create)(busy)^k) = \hat{P}(\xi((create)(busy)^k)) = \hat{P}(create_ (busy_locked_+)^k) = (locked)^k$, which has the same projection with the normal operations $(write)(locked)^k$. Hence, L is not robustly diagnosable against ϕ . The same result can be obtained by inspecting the verifier under attack. To this end, we first construct the faulty automaton and normal automaton under attack, shown in Figs. 4.10(a) and 4.10(b). Then part of the overall verifier under attack is shown in Fig. 4.11 (other parts are omitted without loss of generality for the purpose of verification). It contains the cycles that satisfy condition (4-1), for example, (Process_DF, file0N) *busy_locked_+* (Process_DF, file0N). $\diamond$

4.7 Summary

In this chapter, we have considered again the problem of robust diagnosability of a DES in the presence of sensor attacks. It is assumed that a sensor attacker may corrupt the sen-



(a) G_{FA}



(b) G_{NA}^R

Figure 4.10 (a) G_{FA} and (b) G_{NA}^R .

sensor readings by inserting and/or erasing event observations. Constructions of the faulty automaton and the normal automaton under attack are separately presented. By synchronizing the two automata, the overall deterministic verifier under attack is constructed, which can seek ambiguities (words in the language of a plant with identical projections under appropriate attacks between faulty and normal words) that violate robust diagnosability. It is shown that the condition based on the verifier under attack can be verified in polynomial time.

The work of this chapter is submitted as:

T. Kang, C. Seatzu, Z. Li, and A. Giua, "Polynomial-Time Robust Diagnosability Verification in Discrete Event Systems Under Sensor Attack," *IEEE Transactions on Automatic*

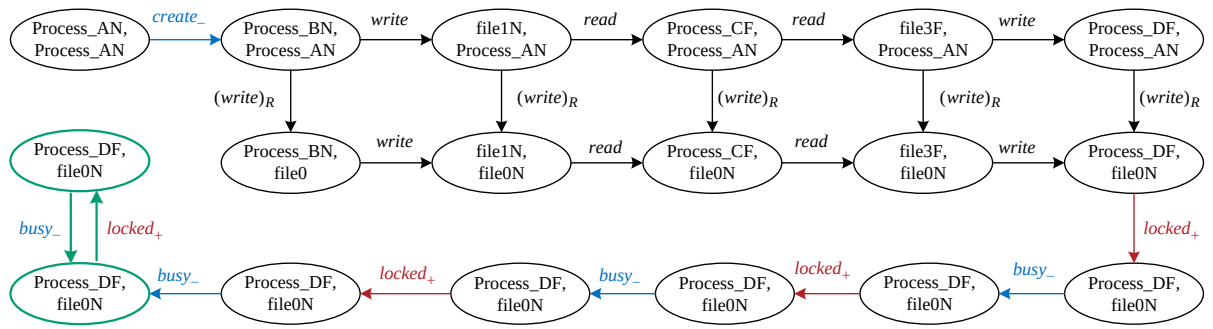


Figure 4.11 Part of verifier under attack G_{VA} for G in Fig. 4.9(b). Cycles satisfying condition (4-1) are highlighted with green lines.

Control, 2025.

Chapter V Conclusion and Future Work

5.1 Concluding Remarks

In this thesis, the problem of fault diagnosis under attack has been studied in the framework of DESs. We address two types of attacks: sensor erasure attack (SE-attack) and sensor insertion attack (SI-attack), which are the main concerns in the study on the security of discrete event systems. The former represents that an attacker hides correct information, while the latter represents that nonexisting (or fake) information is created by an attacker. A successful attacker can keep stealthy to a system operator until the goal of causing a damage to the system is achieved. The results are summarized as follows.

- At the formal language level, we propose a novel notion of *robust diagnosability against attacks*, which captures the ability of the system operator (for monitoring a system) to still diagnose the occurrences of faults in the case of attacks. In addition, we formalize the process of online diagnosis under attack, which aims to determine, based on the corrupted observation, if a fault has occurred during the system evolution.
- At the discrete event system model, we propose a uniform framework for online diagnosis and diagnosability verification under attack based on a *joint diagnoser* (JD for short). From attack point, an attacker that can permanently hide the occurrence of the fault to the operator, thereby resulting in a violation of robust diagnosability, is considered as a harmful attacker. Necessary and sufficient conditions for the existence of a harmful attacker are presented using the JD. Further, a refined joint diagnoser called a *stealthy joint diagnoser* (SJD for short), which contains stealthy attacks only, can be used to verify the existence of a stealthy harmful attacker, i.e., a successful attacker.
- The attack synthesis problem with respect to fault diagnosis is considered. From the perspective of the attacker, we propose an algorithmic procedure to synthesize successful attacks from the stealthy joint diagnoser (SJD). The SJD captures the game between the system and the attacker. Based on it, we can synthesize a successful attack function that leads the system to be non-diagnosable without detection by the system operator.

- Another issue of paramount importance is the complexity involved in a uniform framework. The construction of the SJD is, in the worst case, exponential with respect to the number of states of G . Based on this, we aim to find a more efficient arena for diagnosability verification under attack. The verification relies on the construction of a deterministic verifier under attack, which seeks for certain cycles in the parallel composition of the faulty and normal automata under attack. It is proved that robust diagnosability can be verified with polynomial complexity in the size of the system.

5.2 Future Works

For future work, several possible research directions emerge from the methodologies presented in this thesis:

- **(Diagnosability Enforcement Under Attack)** When an uncontrolled open-loop system does not satisfy a given property of interest, one is interested in enforcing the property via some enforcement mechanism. One of the most commonly-used enforcement mechanisms is to restrict the system behavior by supervisory control. The proposed approach allows the integration with other common event-based techniques (e.g. supervisory control). To move forward, we plan to further consider diagnosability enforcement if a system is no longer diagnosable due to the attack.
- **(More General Attacks)** We wish to relax the restrictions of the permitted moves of the attacker. It is assumed that there is no a priori knowledge about which events are susceptible to attackers or not. That is, the attacker may compromise all the events during the system evolution. As a result, we consider how to find an optimal attacker satisfying the given objective, in terms of the corrupted events (fewer events to corrupt).
- **(Different Information Structure)** In this thesis, we have assumed that the operator know a priori about which events may be affected by an attacker. However, such an assumption may be too strict in practice. Future works may consider a situation where the operator may have partial knowledge of the attacker control.
- **(Decentralized Fault Diagnosis Under Attack)** This thesis focus on centralized fault diagnosis, in which a centralized diagnoser is responsible for the diagnosis in a system. However, many large-sized and complex systems are physically decentralized, where diagnosis information collected by decentralized local sites will be sent to a centralized

site for analysis. Hence, it is necessary to extend the approach to the decentralized setting by considering the verification of codiagnosability under attack.

Bibliography

- [1] CHEN J, KUMAR R. Failure detection framework for stochastic discrete event systems with guaranteed error bounds[J]. IEEE Trans. Autom. Control, 2014, 60(6): 1542-1553.
- [2] DIENE O, MOREIRA M V, SILVA E A, et al. Diagnosability of hybrid systems[J]. IEEE Trans. Control Syst. Technol., 2019, 27(1): 386-393.
- [3] LYGEROS J, GODBOLE D N, BROUCKE M. A fault tolerant control architecture for automated highway systems[J]. IEEE Trans. Control Syst. Technol., 2000, 8(2): 205-219.
- [4] TAKAI S. Verification of robust diagnosability for partially observed discrete event systems[J]. Automatica, 2012, 48(8): 1913-1919.
- [5] PANDALAI D N, HOLLOWAY L E. Template languages for fault monitoring of timed discrete event processes[J]. IEEE Trans. Autom. Control, 2000, 45(5): 868-882.
- [6] QIU W, KUMAR R. Distributed diagnosis under bounded-delay communication of immediately forwarded local observations[J]. IEEE Trans. Syst. Man Cybern. Part A Syst. Humans, 2008, 38(3): 628-643.
- [7] CASSANDRAS C G, LAFORTUNE S. Introduction to discrete event systems[M]. 2nd ed. New York, NY, USA: Springer, 2008.
- [8] HADJICOSTIS C N. Estimation and inference in discrete event systems[M]. Springer, 2019.
- [9] AGARWAL M, BISWAS S, NANDI S. Discrete event system framework for fault diagnosis with measurement inconsistency: Case study of rogue dhcp attack[J]. IEEE/CAA J. Autom. Sinica, 2019, 6(3): 789-806.
- [10] ESCUDERO-ORNELAS I O, TIWARI D, FARNSWORTH M, et al. Modelling interdependencies in an electric motor manufacturing process using discrete event simulation[J]. J. Simul., 2024, 18(4): 604-625.
- [11] WHITE A, KARIMODDINI A, KARIMADINI M. Resilient fault diagnosis under imperfect observations-a need for industry 4.0 era[J]. IEEE/CAA J. Autom. Sinica, 2020, 7(5): 1279-1288.
- [12] JIANG Z, LI Z, WU N, et al. A petri net approach to fault diagnosis and restoration for power transmission systems to avoid the output interruption of substations[J]. J. Simul., 2018, 12(3): 2566-2576.
- [13] KHALEGHI M, BARKHORDARI YAZDI M, KARIMODDINI A, et al. An inferencing structure for mpo-based decentralized dynamic fault diagnosis[J]. IET Control Theory Appl., 2022, 16(2): 182-191.
- [14] CALDERARO V, HADJICOSTIS C N, PICCOLO A, et al. Failure identification in smart grids based on petri net modeling[J]. IEEE Trans. Ind. Electron., 2011, 58(10): 4613-4623.

- [15] WHITE A, KARIMODDINI A. Event-based diagnosis of flight maneuvers of a fixed-wing aircraft[J]. Reliab. Eng. Syst. Saf., 2020, 193: 106609.
- [16] SAADAOU I, LABED A, LI Z, et al. Language-based opacity verification in partially observed petri nets through linear constraints[J]. Mathematics, 2023, 11(18): 3880.
- [17] CABASINO M P, GIUA A, SEATZU C. Fault detection for discrete event systems using petri nets with unobservable transitions[J]. Automatica, 2010, 46(9): 1531-1539.
- [18] CABASINO M P, GIUA A, POCCI M, et al. Discrete event diagnosis using labeled petri nets. an application to manufacturing systems[J]. Control Eng. Pract., 2011, 19(9): 989-1001.
- [19] HU S, ZHANG J, WANG J, et al. Optimal sensor selection for diagnosability enforcement of discrete event systems using labeled petri net[J]. IET Control Theory Appl., 2024, 18(17): 2307-2321.
- [20] SOUALHI A, CLERC G, RAZIK H, et al. Hidden markov models for the prediction of impending faults. [J]. IEEE Trans. Ind. Electron., 2016, 63(5): 3271-3281.
- [21] CHENG P, HE S, STOJANOVIC V, et al. Fuzzy fault detection for markov jump systems with partly accessible hidden information: An event-triggered approach[J]. IEEE Trans. Cybern., 2021, 52(8): 7352-7361.
- [22] DRAGONI N, GASPARI M. An object based algebra for specifying a fault tolerant software architecture[J]. J. Log. Algebraic Program., 2005, 63(2): 271-297.
- [23] LIN F. Diagnosability of discrete event systems and its applications[J]. Discrete Event Dyn. Syst., 1994, 4: 197-212.
- [24] SAMPATH M, SENGUPTA R, LAFORTUNE S, et al. Diagnosability of discrete-event systems[J]. IEEE Trans. Autom. Control, 1995, 40(9): 1555-1575.
- [25] SAMPATH M, LAFORTUNE S, TENEKETZIS D. Active diagnosis of discrete-event systems[J]. IEEE Trans. Autom. Control, 1998, 43(7): 908-929.
- [26] LEFEBVRE D, DELHERM C. Diagnosis of DES with Petri net models[J]. IEEE Trans. Autom. Sci. Eng., 2007, 4(1): 114-118.
- [27] CABASINO M P, GIUA A, SEATZU C. Diagnosability of discrete-event systems using labeled Petri nets [J]. IEEE Trans. Autom. Sci. Eng., 2014, 11(1): 144-153.
- [28] ZAYTOON J, LAFORTUNE S. Overview of fault diagnosis methods for discrete event systems[J]. Annu. Rev. Control, 2013, 37(2): 308-320.
- [29] CABRAL F G, MOREIRA M V. Synchronous diagnosis of discrete-event systems.[J]. IEEE Trans. Autom. Sci. Eng., 2019, 17(2): 921-932.
- [30] LAFORTUNE S, LIN F, HADJICOSTIS C N. On the history of diagnosability and opacity in discrete event systems[J]. Annu. Rev. Control, 2018, 45: 257-266.
- [31] HU S, LI Z. A digital twin approach for enforcing diagnosability in petri nets[J]. IEEE Trans. Autom.

Sci. Eng., 2023: 1-13.

- [32] CONG X, FANTI M P, MANGINI A M, et al. Decentralized diagnosis by Petri nets and integer linear programming[J]. IEEE Trans. Syst. Man Cybern., 2017, 48(10): 1689-1700.
- [33] FANG Q, XIONG G, ZHOU M, et al. Process monitoring, diagnosis and control of additive manufacturing[J]. IEEE Trans. Autom. Sci. Eng., 2024, 21(1): 1041-1067.
- [34] WANG D, WANG X, LI Z. State-based fault diagnosis of discrete-event systems with partially observable outputs[J]. Inf. Sci., 2020, 529: 87-100.
- [35] LEFEBVRE D, HADJICOSTIS C N. Diagnosability of fault patterns with labeled stochastic Petri nets[J]. Inf. Sci., 2022, 593: 341-363.
- [36] CAO L, SHU S, LIN F, et al. Weak diagnosability of discrete-event systems[J]. IEEE Transactions on Control of Network Systems, 2021, 9(1): 184-196.
- [37] WANG X, WANG G, LI Z, et al. Event-based fault estimation and compensation for discrete-time systems via zonotopes[J]. Inf. Sci., 2023, 631: 1-14.
- [38] JIANG S, HUANG Z, CHANDRA V, et al. A polynomial algorithm for testing diagnosability of discrete-event systems[J]. IEEE Trans. Autom. Control, 2001, 46(8): 1318-1321.
- [39] YOO T S, LAFORTUNE S. Polynomial-time verification of diagnosability of partially observed discrete-event systems[J]. IEEE Trans. Autom. Control, 2002, 47(9): 1491-1495.
- [40] QIU W, KUMAR R. Decentralized failure diagnosis of discrete event systems[J]. IEEE Trans. Syst. Man Cybern. Part A: Syst. and Humans, 2006, 36(2): 384-395.
- [41] MOREIRA M V, JESUS T C, BASILIO J C. Polynomial time verification of decentralized diagnosability of discrete event systems[J]. IEEE Trans. Autom. Control, 2011, 56(7): 1491-1495.
- [42] MOREIRA M V, BASILIO J C, CABRAL F G. Polynomial time verification of decentralized diagnosability of discrete event systems” versus “decentralized failure diagnosis of discrete event systems” : A critical appraisal[J]. IEEE Trans. Autom. Control, 2015, 61(1): 178-181.
- [43] BASILIO J C, LAFORTUNE S. Robust codiagnosability of discrete event systems[C]//Proc. IEEE American Control Conference (ACC). Leiden, The Netherlands, 2009: 2202-2209.
- [44] KANAGAWA N, TAKAI S. Diagnosability of discrete event systems subject to permanent sensor failures[J]. Int. J. Control, 2015, 88(12): 2598-2610.
- [45] CARVALHO L K, BASILIO J C, MOREIRA M V. Robust diagnosis of discrete event systems against intermittent loss of observations[J]. Automatica, 2012, 48(9): 2068-2078.
- [46] CARVALHO L K, MOREIRA M V, BASILIO J C, et al. Robust diagnosis of discrete-event systems against permanent loss of observations[J]. Automatica, 2013, 49(1): 223-231.
- [47] CARVALHO L K, MOREIRA M V, BASILIO J C. Generalized robust diagnosability of discrete event systems.[C]//Proc. 18th IFAC World Congr. 2011: 8737-8742.

- [48] CARVALHO L K, MOREIRA M V, BASILIO J C. Comparative analysis of related notions of robust diagnosability of discrete-event systems[J]. *Annu. Rev. Control*, 2021, 51: 23-36.
- [49] BHARDWAJ J, KRISHNAN J P, MARIN D F L, et al. Cyber-physical systems for smart water networks: A review[J]. *IEEE Sensors J.*, 2021, 21(23): 26447-26469.
- [50] SUI T, MO Y, MARELLI D, et al. The vulnerability of cyber-physical system under stealthy attacks[J]. *IEEE Trans. Autom. Control*, 2020, 66(2): 637-650.
- [51] ZHANG K, KELIRIS C, PARISINI T, et al. Stealthy integrity attacks for a class of nonlinear cyber-physical systems[J]. *IEEE Trans. Autom. Control*, 2021, 67(12): 6723-6730.
- [52] LI X, YAN H, CUI K, et al. A novel hybrid yolo approach for precise paper defect detection with a dual-layer template and an attention mechanism[J]. *IEEE Sensors J.*, 2024, 24(7): 11651-11669.
- [53] CUI H, DONG X, DENG H, et al. Cyber attack detection process in sensor of DC micro-grids under electric vehicle based on Hilbert–Huang transform and deep learning[J]. *IEEE Sensors J.*, 2020, 21(14): 15885-15894.
- [54] EL-REWINI Z, SADATSHARAN K, SUGUNARAJ N, et al. Cybersecurity attacks in vehicular sensors[J]. *IEEE Sensors J.*, 2020, 20(22): 13752-13767.
- [55] YU Z, GAO H, CONG X, et al. A survey on Cyber–Physical Systems security[J]. *IEEE Internet Things J.*, 2023, 10(24): 21670-21686.
- [56] SÁNCHEZ H S, ROTONDO D, ESCOBET T, et al. Bibliographical review on cyber attacks from a control oriented perspective[J]. *Annu. Rev. Control*, 2019, 48: 103-128.
- [57] OLIVEIRA S, LEAL A B, TEIXEIRA M, et al. A classification of cybersecurity strategies in the context of discrete event systems[J]. *Annu. Rev. Control*, 2023, 56: 100907.
- [58] DEBOUK R, LAFORTUNE S, TENEKETZIS D. Coordinated decentralized protocols for failure diagnosis of discrete event systems[J]. *Discrete Event Dyn. Syst.*, 2000, 10(1): 33-86.
- [59] SABOORI A, HADJICOSTIS C N. Verification of initial-state opacity in security applications of discrete event systems[J]. *Inf. Sci.*, 2013, 246: 115-132.
- [60] TONG Y, LI Z, SEATZU C, et al. Verification of state-based opacity using petri nets[J]. *IEEE Trans. Autom. Control*, 2017, 62(6): 2823-2837.
- [61] SHU S, LIN F, YING H. Detectability of discrete event systems[J]. *IEEE Trans. Autom. Control*, 2007, 52(12): 2356-2359.
- [62] RAMADGE P J, WONHAM W M. Supervisory control of a class of discrete event processes[J]. *SIAM J. Control Optim.*, 1987, 25(1): 206-230.
- [63] KUMAR R, GARG V K. Modeling and control of logical discrete event systems[M]. Springer, 2012.
- [64] ZHANG Q, SEATZU C, LI Z, et al. Joint state estimation under attack of discrete event systems[J]. *IEEE Access*, 2021, 9: 168068-168079.

- [65] ZHANG Q, SEATZU C, LI Z, et al. Selection of a stealthy and harmful attack function in discrete event systems[J]. *Scientific Reports*, 2022, 12(1): 16302.
- [66] CHEN Q, SU R, LI Z. Attackable detectability of partially-observed discrete-event systems under sensor attack[C]//Proc. 16th Int. Workshop Discrete Event Syst.: Vol. 55. Prague, Czech Republic, 2022: 121-127.
- [67] MEIRA-GÓES R, KANG E, KWONG R H, et al. Synthesis of sensor deception attacks at the supervisory layer of Cyber-Physical Systems[J]. *Automatica*, 2020, 121: 109669.
- [68] R. M G, KWONG R, LAFORTUNE S. Synthesis of sensor deception attacks for systems modeled as probabilistic automata[C]//Proc. IEEE American Control Conference (ACC). Philadelphia, PA, USA, 2019: 5620-5626.
- [69] LIN L, THUIJSMAN S, ZHU Y, et al. Synthesis of supremal successful normal actuator attackers on normal supervisors[C]//Proc. IEEE American Control Conference (ACC). Philadelphia, PA, USA, 2019: 5614-5619.
- [70] LIN L, SU R. Synthesis of covert actuator and sensor attackers[J]. *Automatica*, 2021, 130: 109714.
- [71] FRITZ R, ZHANG P. Modeling and detection of cyber attacks on discrete event systems[C]//Proc. 14th Int. Workshop Discrete Event Syst. Sorrento, Italy, 2018: 285-290.
- [72] CARVALHO L K, WU Y C, KWONG R, et al. Detection and mitigation of classes of attacks in supervisory control systems[J]. *Automatica*, 2018, 97: 121-133.
- [73] ZHAO C, LIN H, LI Y, et al. Event-based state estimation against deception attacks: A detection based approach[J]. *IEEE Sensors J.*, 2023, 23(19): 23020-23029.
- [74] SU R. Supervisor synthesis to thwart cyber attack with bounded sensor reading alterations[J]. *Automatica*, 2018, 94: 35-44.
- [75] WAKAIKI M, TABUADA P, HESPANHA J P. Supervisory control of discrete-event systems under attacks [J]. *Dyn. Games and Appl.*, 2019, 9(4): 965-983.
- [76] MEIRA-GÓES R, MARCHAND H, LAFORTUNE S. Synthesis of supervisors robust against sensor deception attacks[J]. *IEEE Trans. Autom. Control*, 2021, 66(10): 4990-4997.
- [77] TONG Y, WANG Y, GIUA A. A polynomial approach to verifying the existence of a threatening sensor attacker[J]. *IEEE Control Syst. Lett*, 2022, 6: 2930-2935.
- [78] WANG Y, LI Y, YU Z, et al. Supervisory control of discrete-event systems under external attacks[J]. *Inf. Sci.*, 2021, 562: 398-413.
- [79] SU R. Supervisor synthesis to thwart cyber attack with bounded sensor reading alterations[J]. *Automatica*, 2018, 94: 35-44.
- [80] WANG Y, PAJIC M. Supervisory control of discrete event systems in the presence of sensor and actuator attacks[C]//Proc. IEEE 58th Conf. Decis. Control (CDC). Nice, France, 2019: 5350-5355.

- [81] LIMA S T S, BASILIO J C, LAFORTUNE S, et al. Robust diagnosis of discrete-event systems subject to permanent sensor failures[C]//Proc. 10th Int. Workshop Discrete Event Syst.: Vol. 43. Berlin, Germany, 2010: 90-97.
- [82] CARVALHO L K, MOREIRA M V, BASILIO J C. Diagnosability of intermittent sensor faults in discrete event systems[J]. Automatica, 2017, 79(1): 315-325.
- [83] NUNES C E, MOREIRA M V, ALVES M V, et al. Codiagnosability of networked discrete event systems subject to communication delays and intermittent loss of observation[J]. Discrete Event Dyn. Syst., 2018, 28: 215-246.
- [84] TOMOLA J H, CABRAL F G, CARVALHO L K, et al. Robust disjunctive-codiagnosability of discrete-event systems against permanent loss of observations[J]. IEEE Trans. Autom. Control, 2017, 62(11): 5808-5815.
- [85] TAKAI S. A general framework for diagnosis of discrete event systems subject to sensor failures[J]. Automatica, 2021, 129: 109669.
- [86] LI Y, HADJICOSTIS C N, WU N. Tamper-tolerant diagnosability under bounded or unbounded attacks [C]//Proc. 16th Int. Workshop Discrete Event Syst. Prague, Czech Republic, 2022: 52-57.
- [87] HADJICOSTIS C N, LAFORTUNE S, LIN F, et al. Cybersecurity and supervisory control: a tutorial on robust state estimation, attack synthesis, and resilient control[C]//Proc. IEEE 61th Conf. Decis. Control (CDC). Cancún, Mexico, 2022: 3020-3040.
- [88] LIN F, LAFORTUNE S, WANG C. Diagnosability of discrete event systems under sensor attacks[C]//Proc. 22nd IFAC World Congr. Yokohama, Japan, 2023: 3572-3578.
- [89] LIN F, LAFORTUNE S, WANG C. Diagnosability and attack detection for discrete event systems under sensor attacks[J]. Discrete Event Dyn. Syst., 2024, 34(3): 465-495.
- [90] CLAVIJO L B, BASILIO J C. Empirical studies in the size of diagnosers and verifiers for diagnosability analysis[J]. Discrete Event Dyn. Syst., 2017, 27(4): 701-739.
- [91] VIANA G S, BASILIO J C. Codiagnosability of discrete event systems revisited: A new necessary and sufficient condition and its applications[J]. Automatica, 2019, 101: 354-364.
- [92] SHEYNER O, WING J. Tools for generating and analyzing attack graphs[C]//Proc. 2nd Int. Symp. Formal Methods Components Objects. Leiden, The Netherlands, 2003: 344-371.

Acknowledgements

My heartfelt thanks go to my supervisors, Professor Seatzu Carla and Professor Alessandro Giua, in University of Cagliari in Italy. In Cagliari, Prof. Carla provided me with a clear research focus. During the early stages of my research, she spent considerable time discussing with me about my research methods. To my great impression, she usually pays close attention to the finer details. When reviewing my work, she always identified subtle yet critical typos, which helped improve my work significantly.

In my research, Professor Giua often optimizes my research path. In our academic discussions, I consistently gained valuable and constructive insights, which inspired much of my research. Many of the key issues in my work were resolved under his guidance. Incredibly, he always guided me towards the next steps in my research plan.

Also, I am deeply grateful to my supervisor, Professor Zhiwu Li, in Xidian University in China. When I was choosing my research direction in the first stage of my research, Prof. Li gave me so many useful suggestions. He devoted considerable time and patience to meticulously revising my papers, and his rigorous scientific research attitude has a profound impact on me. During challenging times, his encouragement and support were very invaluable.

Special thanks to Professor Christos G. Cassandras, Stéphane Lafortune, Christoforos N. Hadjicostis, João C. Basilio, Rong Su, Lilian K. Carvalho, Marcos V. Moreira, Xiang Yin, Yin Tong, and Ziyue Ma. To some extent, their research achievement have inspired my research.

I am deeply indebted to my parents, Yongli Kang and Caizheng Wang, and my girlfriend Ms. Wenrui Niu, for their unwavering encouragement, understanding, and patience during the ups and downs of this academic pursuit. Lastly, my sincerest thanks to my grandparents in heaven. Miss them forever.

Author Biography

1. Basic Information

Tenglong Kang (male) was born in Huining, Gansu Province, China, in Jun. 1999. He received Bachelor Degree in automation from Xidian University in 2018. From September 2018 to July 2019 he was a master student of School of Electro-Mechanical Engineering of Xidian University majored in Control Theory and Control Engineering. Since September 2019 he has been a Ph.D student of School of Electro-Mechanical Engineering of Xidian University majored in Control Theory and Control Engineering. During September 2020 to the present, he has been in the program of joint Ph.D degree between Xidian University and University of Cagliari, Italy.

2. Educational Background

- 2014.09 ~ 2018.07 Xidian University, B.A. in Automation, China.
- 2018.09 ~ 2019.07 Xidian University, M.A. in Automation, China.
- 2019.09 ~ present Xidian University, Ph.D student in Automation, China.
- 2021.10 ~ present University of Cagliari, Ph.D student in Automation, Italy.

3. Academic Publications During Ph.D

3.1 Published Publications

- [1] **T. Kang**, C. Seatzu, Z. Li, A. Giua. "A Joint Diagnoser Approach for Diagnosability of Discrete Event Systems Under Attack," *Automatica*, vol. 172, p. 112004, 2025. (SCI:001367916200001, JCR Q1)
- [2] **T. Kang**, C. Seatzu, Z. Li, A. Giua. "Fault Diagnosis of Discrete Event Systems Under Attack," in *Proc. IEEE 62nd Conf. Decis. Control (CDC)*, Singapore, Singapore, Dec. 2023, pp. 7923-7929. (EI:20240715547619)

3.2 Submitted Publications

- [1] **T. Kang**, C. Seatzu., Z. Li, and A. Giua. "Polynomial-Time Robust Diagnosability Verification in Discrete Event Systems Under Attack", submitted to *IEEE Transactions on Automatic Control*. 2024.

4. Research Experiences

4.1 Participation in Academic Programs

- [1] National Natural Science Foundation of China under Grant No. 61873342 and entitled "Deadlock control and analysis of labeled Petri nets", 2018.1-2021.12.
- [2] National Key Research and Development Program of China No. 2018YFB1700104 and entitled "Intelligent factory solutions for typical industries", 2019.6-2022.5.
- [3] The Science Technology Development Fund, MSAR under Grant No. 0029/2023/RIA1 and entitled "Opacity verification and enforcement in the framework of partially observed Petri nets".
- [4] Project SERICS (PE00000014) under the MUR National Recovery, and Resilience Plan funded by the European Union - NextGenerationEU.

4.2 Academic Reviewing Services

- [1] Reviewer for IEEE Systems, Man and Cybernetics: Systems.
- [2] Reviewer for Information Science.
- [3] Reviewer for The computer Journal.
- [4] Reviewer for 2023 IEEE International Conference on Automation Science and Engineering.
- [5] Reviewer for 2024 The 14th United Kingdom Automatic Control Council International Conference on Control.
- [6] Reviewer for 2024 IEEE International Conference on Systems, Man, and Cybernetics.

4.3 Conference talk

- [1] Invited Talk for the topic of "A New Approach to the Fault Diagnosis of Discrete Event Systems Under Attack" at Italy for 19th Conference on Detection of Intrusions and Malware and Vulnerability Assessment, 2022.
- [2] Invited Talk for the topic of "Fault Diagnosis of Discrete Event Systems Under Attack" at Singapore for IEEE 62nd Conference of Decision and Control (CDC), 2023.
- [3] Invited Talk for the topic of "Intelligent Attack Against Fault Diagnosis in Discrete-Event Systems Under Malicious Attack" at China for International Colloquium on Discrete Event Dynamic Systems, 2024.