Fault pattern diagnosis of discrete-event systems by means of logical verifiers

Ye Liang, Dimitri Lefebvre, Zhiwu Li

Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-6 (2022) 551–556

Fault Pattern Diagnosis of Discrete-Event Systems by Means of Logical Verifiers

Ye Liang * **Dimitri Lefebvre** ** **Zhiwu Li** ***

* *School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China (e-mail: liangye@stu.xidian.edu.cn)*

** *GREAH Laboratory, Normandy University, 75 rue Bellot, 76600 Le Havre, France (e-mail: dimitri.lefebvre@univ-lehavre.fr)*

*** *Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau and School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China (e-mail: zhwli@xidian.edu.cn)*

Abstract: In this paper, a diagnosis problem of discrete event systems (DESs) is considered, including fault pattern detection and diagnosability. A fault pattern in a DES is modeled by an automaton that represents the occurrence of complex faults, i.e., the language of the automaton is the objective to be diagnosed. To solve the problem of fault pattern detection, two verifiers are provided. The former, based on state isolation, can perform state estimation in an efficient manner recursively such that at any point during recursion, the states can be isolated. The latter, inspired by the notion of synchronous product, allows us to concisely synthesize and analyze the system information. Comparing these two verifiers, it is found that the two structures are equivalent from the perspective of pattern detection and diagnosability. On the basis of aforementioned verifier structures, we establish their respective diagnosers, and develop algorithms for fault pattern diagnosability.

Keywords: Discrete event systems, Fault diagnosis, Fault patterns, Fault detection, Diagnosability.

1. INTRODUCTION

In the context of discrete event systems (DESs) Cassandras and Lafortune (2009), fault diagnosis is a crucial and challenging task to ensure reliability Lin (1994); Sampath et al. (1995), which involves two objectives, fault detection and fault diagnosability, where fault detection aims at detecting whether the faults are recognized from a given sequence of observations and fault diagnosability focuses on determinating whether any predetermined failure can be distinguished within a finite delay after its occurrence.

To this end, a systemic procedure for fault diagnosis with its formal formulation, namely the construction of a diagnoser, is proposed in Sampath et al. (1995), where the diagnoser verifies off-line the necessary and sufficient conditions for diagnosability; it is also used to perform diagnosis by on-line observations of system behavior. Once a diagnoser in Sampath et al. (1995) has been built, diagnosability can be tested in polynomial time with respect to the size of state space of the diagnoser. However, the state space of a diagnoser is in the worst case exponential with respect to the size of a system model. To overcome the potential state explosion problem by using diagnoser, a so-called "twin machine" technique Jiang et al. (2001); Yoo and Lafortune (2002) is introduced to provide a worst-case polynomial test in the number of system states for diagnosability, without constructing a diagnoser. An algorithm with linear complexity proposed in Hadjicostis (2019) converts the fault detection problem into state isolation, with aim of determining whether the observations allow us to isolate the states to be within a particular state set which represents the failures.

All the aforementioned works assume that the faults are permanent, i.e., once a fault occurs, the system remains indefinitely faulty. Experience with monitoring of dynamic systems shows that there is a larger spectrum of faulty situations in practical systems, such as multiple faults, intermittent faults Contant et al. (2004), temporary faults Biswas (2012), fault repetitions Jiang et al. (2003). Therefore, a more general method which involves fault pattern detection and fault pattern diagnosability, is needed to solve the diagnosis problem Jeron et al. (2006); Ye and Dague (2012); Yan et al. (2010); Gougam et al. (2017), where fault pattern detection aims at deciding whether a fault pattern is recognized from a given sequence of observations, and diagnosability focuses on determinating whether a fault pattern can be distinguished with certainly within a finite delay. This also is the core of this work.

A fault pattern is modeled by an automaton that represents the occurrence of complex faults, whose language is the objective to be diagnosed. Pattern diagnosis is introduced by Jeron et al. (2006). Differently from this former work, our paper presents an algorithm to detect the fault patterns with state isolation and to detail how state isolation can be used for diagnosability, which provides another possible vehicle for fault diagnosis. The advantage of the

10.1016/j.ifacol.2022.07.186

algorithm proposed in this paper is that state estimation can be performed recursively in an efficient manner and, at any point during the recursion, the state can be isolated.

The problem of diagnosis with fault patterns have received widespread attention in a broad class of frameworks, such as the work in Gougam et al. (2017), which focuses on the diagnosibility analysis of Petri nets so that the problem of pattern diagnosis can be worked out by using synchronous product. In Gougam et al. (2014), the study on discernability, as opposed to diagnosability, is considered to detect the occurrence of the exclusive behavior, thus expanding the application of patterns. In Ye and Dague (2012); Yan et al. (2010), the authors put forward the methods of constructing local pattern diagnosers for distributed systems. In Jiang and Kumar (2004), the Linear-time Temporal Logic formulas are used to specify failures in the system such that the problem of testing diagnosability is reduced to that of model checking. The work in Pencolé and Subias (2017) investigates the problem of fault pattern diagnosis for bounded labeled Petri nets, which relies on a matching relation between the system and the pattern that turns the pattern diagnosis problem into a model checking problem. In Boussif et al. (2021), the authors review the main definitions of diagnosability with regard to intermittent faults, and discuss appropriate verification techniques.

The aim of this paper is to investigate fault pattern diagnosis of DESs that includes both fault pattern detection and fault pattern diagnosability. Specifically, we concentrate on the case of labeled finite automata and its fault pattern diagnosis. In order to solve the problem of fault pattern detection, we propose two verifiers. The former is based on state isolation, which captures the ability to determinate, following a sequence of observations, whether a state falls within a given set of states of interest, thereby achieving failure diagnosis. Also, in the foregoing verifier, one possibility is that there may exist some non-reachable states beginning from the initial state. With slight differences, we establish another verifier inspired by the concept of synchronous product, which can concisely analyze the system information. Comparing these two verifiers, we found that the two structures are equivalent from the perspective of pattern diagnosis. By synthesizing diagnosers of the two verifiers, we develop the algorithms for fault pattern diagnosability.

2. PRELIMINARIES

In this paper, we use $\mathbb{N}$ and $\mathbb{N}_0$ to denote the set of strictly positive integers and the set of non-negative integers respectively.

2.1 Finite state automaton

In the following, we review the model of finite automaton and its related notions.

Definition 1. A deterministic finite automaton (DFA) is a four-tuple $G_0 = (L, \Sigma, e, l_0)$, where L is the set of states, Σ is the set of events, l_0 is the initial state, and $e : L \times \Sigma \to L$ is the transition function: $l' = e(l, \sigma)$ means that there is a transition labeled with event σ from the state l to l'.

For a finite automaton, the set Σ can be partitioned into two disjoint subsets $\Sigma = \Sigma_o \cup \Sigma_{uo}$, where Σ_o and Σ_{uo} represent the sets of observable events and unobservable events, respectively. The set of all strings defined on Σ is denoted by Σ^*, including the empty string λ. A string $w = \sigma_0\sigma_1 \dots \sigma_n$ over Σ is a sequence of events, where $\sigma_i \in \Sigma$ for $i = 0, 1, \dots, n$. The concatenation of two strings $w', w'' \in \Sigma^*$ is a new string $w = w'w'' \in \Sigma^*$, where w' is followed by w''. To model unobservable events, we use a special label, namely the empty label, denoted by ε. We also use the symbol λ to denote a special string, called the empty string, that contains no symbol. In order to be more general, we introduce the notion of output labels. Let $Q_\varepsilon = Q \cup \{\varepsilon\}$ be the set of output labels, where Q is the set of observable labels. The labeling function is defined as $Lab : \Sigma \to Q_\varepsilon$, where $Lab(\sigma) \in Q$ if $\sigma \in \Sigma_o$ and $Lab(\sigma) = \varepsilon$ if $\sigma \in \Sigma_{uo}$. A labeled finite automaton is defined as follows.

Definition 2. A labeled finite automaton (LFA) is a three-tuple $G = (G_0, Q_\varepsilon, Lab)$, where $G_0 = (L, \Sigma, e, l_0)$ is a DFA, Q_ε is a set of output labels, and $Lab : \Sigma \to Q_\varepsilon$ is a labeling function.

Given a state $l \in L$, the set of active events at l is defined as $\Lambda(l) = \{\sigma \in \Sigma | \exists l' \in L : l' = e(l, \sigma)\}$. The transition function e is extended to a string $w \in \Sigma^*$ in the usual way. We use $|w|$ to denote the length of w. We use $w' \leq w$ to denote that the string w' is a prefix of w, i.e., there exists w'' such that $w = w'w''$. The language generated by LFA G is denoted by $\mathcal{L}(G)$. For each string $w \in \mathcal{L}(G)$, we use $\mathcal{L}(G)/w$ to denote the post-language of $\mathcal{L}(G)$ after w, defined as $\mathcal{L}(G)/w = \{w' \in \Sigma^* \mid ww' \in \mathcal{L}(G)\}$. A run beginning from the initial state l_0 has the form

$$\rho = l_0 \xrightarrow{\sigma_0} l_1 \xrightarrow{\sigma_1} \cdots l_n \xrightarrow{\sigma_n} l_{n+1},$$

where l_i, $l_{i+1} \in L, \sigma_i \in \Sigma, l_{i+1} = e(l_i, \sigma_i)$, $i = 0, 1, \dots, n$. The run ρ is associated to the string $w = \sigma_0 \dots \sigma_n$. A cycle is a run such that $l_{n+1} = l_0$. Given an LFA G, a projection function $\mathscr{P} : \Sigma^* \to Q^*$ can be defined as follows. For $w \in \Sigma^*$ and $\sigma \in \Sigma$,

$$\mathscr{P}(w\sigma) = \begin{cases} \mathscr{P}(w)q & \text{if } Lab(\sigma) = q \\ \mathscr{P}(w) & \text{if } Lab(\sigma) = \varepsilon, \end{cases}$$

and $\mathscr{P}(\lambda) = \lambda$. In simple words, labeling function Lab replaces the events by their corresponding labels, including unobservable events, while projection function $\mathscr{P}$ removes all the unobservable events and tracks only the observable events.

We use $\mathcal{L}_Q(G)$ to denote the observable language, defined by $\mathcal{L}_Q(G) = \{w_q \in Q^* | \mathscr{P}(w) = w_q, w \in \mathcal{L}(G)\}$. Given a sequence of observations w_q, the inverse projection $\mathscr{P}^{-1} : Q^* \to \Sigma^*$ is defined by $\mathscr{P}^{-1}(w_q) = \{w \in \mathcal{L}(G) | \mathscr{P}(w) = w_q\}$. Given an LFA G, we use $R_G(l, w_q)$ to denote the set of states reached from l by the execution of the strings w such that $\mathscr{P}(w) = w_q$, which is defined by

$$R_G(l, w_q) = \{l' \in L | \exists w \in \mathcal{L}(G) : l' = e(l, w), \mathscr{P}(w) = w_q\}.$$

2.2 Fault patterns

Definition 3. A fault pattern is a five-tuple DFA $\Omega = (S, \Sigma, e_\Omega, s_0, s_\Omega)$, with the set of states S, the single final state $s_\Omega \in S$, the initial state s_0, and the transition function $e_\Omega : S \times \Sigma \to S$. The fault pattern Ω satisfies a complete condition, i.e., for all $s \in S$, $\Lambda(s) = \Sigma$ holds and the final state is stable, i.e., for all $\sigma \in \Sigma$, $e_\Omega(s_\Omega, \sigma) = s_\Omega$.

Based on the complete condition of Ω, we know that $\mathcal{L}(\Omega) = \Sigma^*$. We use $\mathcal{L}_\Omega(\Omega)$ to denote the accepted language of Ω, defined as $\mathcal{L}_\Omega(\Omega) = \{w \in \Sigma^* | e_\Omega(s_0, w) = s_\Omega\}$, and introduce $\mathcal{L}_\Omega(G) = \mathcal{L}(G) \cap \mathcal{L}_\Omega(\Omega)$.

Example 1. A fault pattern Ω is shown in Fig. 1 with the occurrence of events f_1 and f_2, where F is the final state. The accepted language of Ω is $\mathcal{L}_\Omega(\Omega) = \Sigma^* f_1 \Sigma^* \cup \Sigma^* f_2 \Sigma^*$.

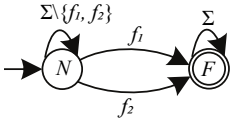

Fig. 1. An example of fault pattern Ω.

3. FAULT PATTERN DIAGNOSIS OF AUTOMATA

In this section, we introduce related notions of fault pattern diagnosis, including fault pattern detection and fault pattern diagnosability.

Definition 4. The detection function $Detect_\Omega$: $\mathcal{L}_Q(G) \to \{Yes, No, Ambiguous\}$ is defined as

- $Detect_\Omega(w_q) = Yes$ if $\mathcal{P}^{-1}(w_q) \subseteq \mathcal{L}_\Omega(G)$,
- $Detect_\Omega(w_q) = No$ if $\mathcal{P}^{-1}(w_q) \cap \mathcal{L}_\Omega(G) = \emptyset$,
- $Detect_\Omega(w_q) = Ambiguous$, otherwise.

Definition 5. Given an LFA G and a pattern Ω, G is diagnosable with regard to pattern Ω if

$$(\exists k \in \mathbb{N})\ (\forall w \in \mathcal{L}_\Omega(G))\ (\forall w' \in \mathcal{L}(G)/w)$$
$$[|w'| \geq k] \Rightarrow [\mathcal{P}^{-1}(\mathcal{P}(ww')) \subseteq \mathcal{L}_\Omega(G)].$$

We make the following assumption for diagnosability:

(H) Given an LFA G, the observable language $\mathcal{L}_Q(G)$ is live, i.e., for all observations $w_q \in \mathcal{L}_Q(G)$, there always exists a label $q \in Q$ such that $w_q q \in \mathcal{L}_Q(G)$.

3.1 Fault pattern detection based on verifiers

Two verifier structures are proposed in the following for fault pattern detection, one based on state isolation and the other inspired by synchronous product.

Definition 6. Given an LFA $G = (G_0, Q_\varepsilon, Lab)$ with $G_0 = (L, \Sigma, e, l_0)$ and a pattern $\Omega = (S, \Sigma, e_\Omega, s_0, s_\Omega)$, the verifier of G with respect to Ω is defined as an LFA

$$V_G(\Omega) = (L_V, \Sigma, e_V, l_0^V, L_{s_\Omega}, Q_\varepsilon, Lab),$$

with $s \in S$ are $|S|$ copies of L, $L_V \subseteq \cup_{s \in S} L_s$ is the set of states, $n = |L|$, $L_s = \{l_0^s, l_1^s, \ldots, l_n^s\}$, $l_i^s = (l_i, s)$, $l_i \in L$, $s \in S$, $i = 0, ..., n$, Σ is the set of system events, e_V is the transition function defined for $s, s' \in S$, $l_i^s \in L_s$ and $l_j^{s'} \in L_{s'}$ by $e_V(l_i^s, w) = l_j^{s'}$ if $e(l_i, w) = l_j$ and $e_\Omega(s, w) = s'$, $l_0^V = l_0^{s_0} = (l_0, s_0)$ is the initial state, $L_{s_\Omega} = \{l_0^{s_\Omega}, l_1^{s_\Omega}, \ldots, l_n^{s_\Omega}\}$ is the set of final states, Q_ε is the set of output labels, and $Lab : \Sigma \to Q_\varepsilon$ is the labeling function.

Proposition 1 below shows that the verifier $V_G(\Omega)$ can be used to provide the answer of detection function.

Proposition 1. Given an LFA G, a pattern Ω, its corresponding verifier $V_G(\Omega)$, and a sequence of observations $w_q \in \mathcal{L}_Q(G)$, the detection function satisfies

(a) $Detect_\Omega(w_q) = Yes$ iff $R_{V_G(\Omega)}(l_0^V, w_q) \subseteq L_{s_\Omega}$,
(b) $Detect_\Omega(w_q) = No$ iff $R_{V_G(\Omega)}(l_0^V, w_q) \cap L_{s_\Omega} = \emptyset$,
(c) $Detec_\Omega(w_q) = Ambiguous$, otherwise.

Proof. Consider $w_q \in \mathcal{L}_Q(G)$ and a sequence $w = \sigma_0 \sigma_1 \ldots \sigma_n \in \mathcal{P}^{-1}(w_q)$. Runs in Ω and $V_G(\Omega)$ that begin respectively from the initial states s_0 and l_0^V are associated to w:

$$\rho_\Omega : s(0) \xrightarrow{\sigma_0} s(1) \xrightarrow{\sigma_1} \cdots s(n) \xrightarrow{\sigma_n} s(n+1),$$

$$\rho_V : l_V(0) \xrightarrow{\sigma_0} l_V(1) \xrightarrow{\sigma_1} \cdots l_V(n) \xrightarrow{\sigma_n} l_V(n+1).$$

To prove (a, $\Rightarrow$), assume that $Detect_\Omega(w_q) = Yes$. Then, we have $w \in \mathcal{L}_\Omega(G)$ and $e_\Omega(s(0), w) = s_\Omega$. By considering the run ρ_Ω, there exists one or more indices $n_1, ..., n_k$ such that $s(0) = ... = s(n_1) = s_0$, $s(n_h + 1) = ... = s(n_{h+1})$, $h = 1, ..., k-1$, and $s(n_k + 1) = ... = s(n+1) = s_\Omega$. Considering now the run ρ_V and according to Definition 6, $l_V(0), ..., l_V(n_1) \in L_{s_0}$, $l_V(n_h + 1), ..., l_V(n_{h+1}) \in L_{s(n_{h+1})}$, $h = 1, ..., k-1$, and $l_V(n_k + 1), ..., l_V(n+1) \in L_{s_\Omega}$. In particular, the last state of the run ρ_V belongs to L_{s_Ω}. Thus $R_{V_G(\Omega)}(l_0^V, w_q) \subseteq L_{s_\Omega}$ holds.

To prove (a, $\Leftarrow$), assume that $R_{V_G(\Omega)}(l_0^V, w_q) \subseteq L_{s_\Omega}$. For each w such that $\mathcal{P}(w) = w_q$ there exists one or more indices $n_1, ..., n_k$ and $k-1$ states $s(n_{h+1}) \in S$, $h = 1, ..., k-1$, such that $l_V(0), ..., l_V(n_1) \in L_{s_0}$, $l_V(n_h + 1), ..., l_V(n_{h+1}) \in L_{s(n_{h+1})}$, $h = 1, ..., k-1$, $l_V(n_k + 1), ..., l_V(n+1) \in L_{s_\Omega}$. By Definition 6, we have $s(0) = ... = s(n_1) = s_0$, $s(n_h + 1) = ... = s(n_{h+1})$, $h = 1, ..., k-1$ and $s(n_k + 1) = ... = s(n+1) = s_\Omega$. Consequently, the run ρ_Ω ends in s_Ω and $w = \sigma_0 \sigma_1 ... \sigma_n \in \mathcal{L}_\Omega(G)$. According to Definition 4, we have $Detect_\Omega(w_q) = Yes$.

To prove (b, $\Rightarrow$), assume that $Detect_\Omega(w_q) = No$. Then, for each $w \in \mathcal{P}^{-1}(w_q)$, $w \notin \mathcal{L}_\Omega(G)$ and consequently, $e_\Omega(s(0), w) \neq s_\Omega$. Considering the run ρ_Ω, we have $s(n+1) \neq s_\Omega$ and considering the run ρ_V, we have also $l_V(n+1) \notin L_{s_\Omega}$. Thus $R_{V_G(\Omega)}(l_0^V, w_q) \cap L_{s_\Omega} = \emptyset$.

To prove (b, $\Leftarrow$), assume that $R_{V_G(\Omega)}(l_0^V, w_q) \cap L_{s_\Omega} = \emptyset$. For each w such that $\mathcal{P}(w) = w_q$, we have $l_V(n+1) \notin L_{s_\Omega}$ in run ρ_V. Consequently, $s(n+1) \notin s_\Omega$ in run ρ_Ω. As far as s_Ω is stable by e_Ω, all states $s(k)$, $k = 1, ..., n+1$ satisfy $s(k) \neq s_\Omega$ and according to Definition 4, we have $Detect_\Omega(w_q) = No$.

The proof of (c) results implicitly from (a) and (b). ■

Given an LFA G and a pattern Ω with $|L| = n$, $|S| = m$. The number of reachable states of verifier $V_G(\Omega)$ is $n \times m$ at most. We conclude that the complexity of fault pattern detection with the verifier $V_G(\Omega)$ is $O(n \times m)$.

Example 2. Consider the fault pattern Ω in Fig. 1, and an LFA G in Fig. 2(a) with the set of unobservable events $\Sigma_{uo} = \{f_1, f_2\}$, and the single observable label $Q = \{q\}$. According to Definition 6, the verifier $V_G(\Omega)$ of G and Ω is obtained in Fig. 2(b), where the set of states is $L_V = \{0N, 5N, 6N, 0F, 1F, 2F, 3F, 4F, 5F, 6F\}$, and the transition function is defined as $e_V(0N, a) = 5N$ since $e(0, a) = 5$ and $e_\Omega(N, a) = N$, and so on. Given, for example, an output label sequence qq, the diagnosis answer is Ambiguous, since $R_{V_G(\Omega)}(0N, qq) = \{6N, 0F, 2F\}$, where $0F, 2F \in L_{s_\Omega}$ and $6N \notin L_{s_\Omega}$. Note that there are some non-reachable states in $V_G(\Omega)$ from the initial state $0N$ with dashed lines in Fig. 2(b).

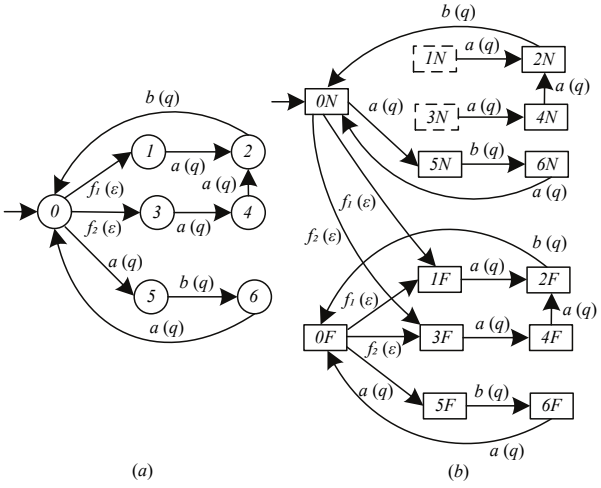

Fig. 2. (a) LFA G and (b) verifier $V_G(\Omega)$.

Definition 7 introduces a synchronous product-based verifier, whose resulting structure allows us to acquire the system information concisely.

Definition 7. Given an LFA $G = (G_0, Q_\varepsilon, Lab)$, with $G_0 = (L, \Sigma, e, l_0)$ and a pattern $\Omega = (S, \Sigma, e_\Omega, s_0, s_\Omega)$, the synchronous product of G and Ω is an LFA

$$G_\Omega = (L_{G_\Omega}, \Sigma, e_{G_\Omega}, l_0^{G_\Omega}, L_F^{G_\Omega}, Q_\varepsilon, Lab),$$

where $L_{G_\Omega} \subseteq L \times S$ is the set of states, Σ is the set of events, $l_0^{G_\Omega} = (l_0, s_0)$ is the initial state, $L_F^{G_\Omega} = L \times s_\Omega$ is the set of final states, and $e_{G_\Omega} : (L \times S) \times \Sigma \to (L \times S)$ is the transition function, satisfying $(l^2, s^2) = e_{G_\Omega}((l^1, s^1), \sigma)$ if there exists $\sigma \in \Sigma$ such that $l^2 = e(l^1, \sigma)$ and $s^2 = e_\Omega(s^1, \sigma)$.

According to Ogheneovo (2018), two automata are equivalent if and only if they generate and accept the same languages. Proposition 2 below compares verifiers $V_G(\Omega)$ and G_Ω.

Proposition 2. Given an LFA G and a pattern Ω, the verifiers G_Ω and $V_G(\Omega)$ are equivalent, i.e., $\mathcal{L}(G_\Omega) = \mathcal{L}(V_G(\Omega))$ and $\mathcal{L}_\Omega(G_\Omega) = \mathcal{L}_\Omega(V_G(\Omega))$.

Proof. Consider a string $w = \sigma_0 \dots \sigma_n \in \Sigma^*$. Runs in G, Ω, G_Ω and $V_G(\Omega)$, associated to w, that begin respectively from the initial states l_0, s_0, $l_0^{G_\Omega} = (l_0, s_0)$, and $l_0^V = l_0^{s_0}$ are referred to as (when such runs exist):

$$\rho_G : l(0) \xrightarrow{\sigma_0} l(1) \xrightarrow{\sigma_1} \cdots l(n) \xrightarrow{\sigma_n} l(n+1), \quad (1)$$

$$\rho_\Omega : s(0) \xrightarrow{\sigma_0} s(1) \xrightarrow{\sigma_1} \cdots s(n) \xrightarrow{\sigma_n} s(n+1), \quad (2)$$

$$\rho_{G_\Omega} : (l(0), s(0)) \xrightarrow{\sigma_0} \cdots \xrightarrow{\sigma_n} (l(n+1), s(n+1)), \quad (3)$$

$$\rho_V : l_V(0) \xrightarrow{\sigma_0} l_V(1) \xrightarrow{\sigma_1} \cdots l_V(n) \xrightarrow{\sigma_n} l_V(n+1). \quad (4)$$

Let us first prove $\mathcal{L}(G_\Omega) \subseteq \mathcal{L}(V_G(\Omega))$. Assume $w = \sigma_0\sigma_1 \dots \sigma_n \in \mathcal{L}(G_\Omega)$. There exists a run ρ_{G_Ω} of the form (3) in G_Ω such that $(l(0), s(0)) = l_0^{G_\Omega}$. We know that $w \in \mathcal{L}(G)$ and $w \in \mathcal{L}(\Omega)$. According to Definition 7, there exists also a run ρ_G of the form (1) with $l(0) = l_0$ in G and a run ρ_Ω of the form (2) with $s(0) = s_0$ in Ω and zeros or more indices $n_1, \dots, n_k$ such that $s(0) = ... = s(n_1) = s_0$, $s(n_h + 1) = ... = s(n_{h+1})$, $h = 1, ..., k$. By Definition 6, we can find a run ρ_V in $V_G(\Omega)$ of the form (4) with $l_V(0) = l_0^{s_0}$, $l_V(0), ..., l_V(n_1) \in L_{s_0}$, $l_V(n_h + 1), ..., l_V(n_{h+1}) \in L_{s(n_{h+1})}$, $h = 1, ..., k$. Thus $w \in \mathcal{L}(V_G(\Omega))$.

In addition, if $w \in \mathcal{L}_\Omega(G_\Omega)$, then $e_{G_\Omega}(l_0^{G_\Omega}, w) \in L_F^{G_\Omega}$. Then $s(n_k + 1) = ... = s(n + 1) = s_\Omega$ and $l_V(n_k + 1), ..., l_V(n + 1) \in L_{s_\Omega}$. Thus, $w \in \mathcal{L}_\Omega(V_G(\Omega))$ is true.

Let us now prove $\mathcal{L}(G_\Omega) \supseteq \mathcal{L}(V_G(\Omega))$. Assume $w = \sigma_0\sigma_1 \dots \sigma_n \in \mathcal{L}(V_G(\Omega))$. There exists a run ρ_V of the form (4) in $V_G(\Omega)$ such that $l_V(0) = l_0^V = l_0^{s_0}$. We know that $w \in \mathcal{L}(G)$ and $w \in \mathcal{L}(\Omega)$. By Definition 6, there exists also a run ρ_G of the form (1) with $l(0) = l_0$ in G and a run ρ_Ω of the form (2) with $s(0) = s_0$ in Ω and zeros or more indices $n_1, \dots, n_k$ such that $s(0) = ... = s(n_1) = s_0$, $s(n_h + 1) = ... = s(n_{h+1})$, $h = 1, ..., k$. According to Definition 7, we can find a run ρ_{G_Ω} of the form (2) with $(l(0), s_0), ..., (l(n_1), s_0) \in L \times s_0$, $(l(n_h + 1), s(n_{h+1})), ..., (l(n_{h+1}), s(n_{h+1})) \in L \times s(n_{h+1})$, $h = 1, ..., k$. Thus $w \in \mathcal{L}(G_\Omega)$.

If $w \in \mathcal{L}_\Omega(V_G(\Omega))$, then $e_V(l_0^{s_0}, w) \in L_{s_\Omega}$. We have $s(n_k + 1) = ... = s(n + 1) = s_\Omega$ and $(l(n_k + 1), s_\Omega), ..., (l(n + 1), s_\Omega) \in L \times s_\Omega$. Thus, $w \in \mathcal{L}_\Omega(G_\Omega)$. ■

Example 3. Consider the fault pattern Ω in Fig. 1, and G in Fig. 2(a). The synchronous product G_Ω is obtained in Fig. 3. By removing the non-reachable states of $V_G(\Omega)$ and according to Proposition 2, we know that the two structures G_Ω and $V_G(\Omega)$ are identical.

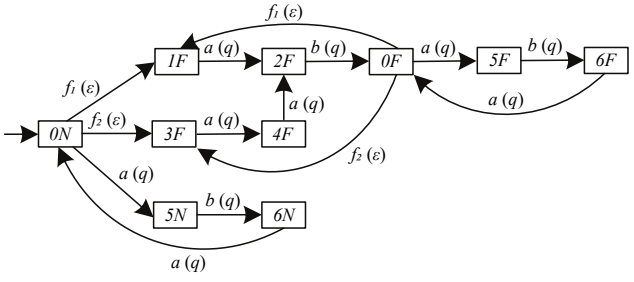

Fig. 3. Verifier G_Ω based on synchronous product.

For the seek of brevity, results will be discussed for $V_G(\Omega)$ only, and the analysis with G_Ω will not be detailed here.

3.2 Diagnosability of fault patterns based on diagnosers

In this section, a diagnoser structure is introduced for fault pattern diagnosability checking based on state isolation.

Definition 8. Given a verifier $V_G(\Omega) = (L_V, \Sigma, e_V, l_0^V, L_F^V, Q_\varepsilon, Lab)$, the diagnoser $D_G(\Omega)$ of G with respect to Ω is the determinisation of $V_G(\Omega)$

$$D_G(\Omega) = (L_D, e_D, l_0^D, Q, Lab),$$

where $L_D \subseteq 2^{L_V}$ is the set of states, the initial state is $l_0^D = R_{V_G(\Omega)}(l_0^V, \lambda)$, and e_D is the transition function defined for all $l_D, l'_D \in L_D$ and $q \in Q$, by $l'_D = e_D(l_D, q)$ with $l'_D = \bigcup_{l_V \in l_D} R_{V_G(\Omega)}(l_V, q)$.

Observe that it is also possible to define a diagnoser $d_G(\Omega)$ as the determinisation of G_Ω, which is the same construction as $D_G(\Omega)$. For the seek of brevity, the construction detail and the results will be discussed for $D_G(\Omega)$ only, and the analysis of $d_G(\Omega)$ will not be pursued here.

Definition 9. Given a system G, pattern Ω, and its diagnoser $D_G(\Omega)$, a state $l_D \in L_D$ (recall that $L_D \subseteq 2^{L_V}$ and $L_V = \cup_{s \in S} L_s$) is said to be indeterminate if $l_D \cap L_{s_\Omega} \neq \emptyset$ and $l_D \cap (L_V \setminus L_{s_\Omega}) \neq \emptyset$. A cyclic run, for short a cycle, in $D_G(\Omega)$ is called an indeterminate cycle if each state of the cycle is indeterminate.

Lemmas 1 to 3 provide preliminary results for diagnosability property. Lemma 1 states a monotonicity property of L_V with respect to the subset L_{s_Ω}: if a given state belongs to L_{s_Ω}, all its possible successors also belong to L_{s_Ω}.

Lemma 1. Given a verifier $V_G(\Omega) = (L_V, \Sigma, e_V, l_0^V, Q_\varepsilon, Lab)$ with $L_V = \cup_{s \in S} L_s$, for all $l_V \in L_{s_\Omega}$, for all $w \in \Sigma^*$, if $e_V(l_V, w)$ is defined, then $e_V(l_V, w) \in L_{s_\Omega}$.

Proof. Consider a state $l_V \in L_{s_\Omega}$ and a string $w = \sigma_0 \sigma_1 \dots \sigma_{k-1}$, such that $e_V(l_0^V, w) = l_V$. Consider also a string $w' = \sigma_k \dots \sigma_n$, the concatenation $ww' = \sigma_0 \sigma_1 \dots \sigma_{k-1} \sigma_k \dots \sigma_n$ and the state l'_V such that $e_V(l_0^V, ww') = e_V(l_V, w') = l'_V$. The runs in Ω and $V_G(\Omega)$, associated to ww', that begin respectively from the initial states s_0, and $l_0^{s_0}$ are referred to as:

$$\rho_\Omega : \ s(0) \xrightarrow{\sigma_0} s(1) \cdots \xrightarrow{\sigma_{k-1}} s(k) \xrightarrow{\sigma_k} \cdots \xrightarrow{\sigma_n} s(n+1),$$

$$\rho_V : \ l_V(0) \xrightarrow{\sigma_0} l_V(1) \cdots \xrightarrow{\sigma_{k-1}} l_V(k) \xrightarrow{\sigma_k} \cdots \xrightarrow{\sigma_n} l_V(n+1).$$

Since $l_V \in L_{s_\Omega}$ and $l_V(k) = l_V$, we have $l_V(k) \in L_{s_\Omega}$ and $s(k) = s_\Omega$. Then, according to Definition 3, s_Ω is a stable state and $s(k+1) = \cdots = s(n+1) = s_\Omega$. According to Definition 7, for the run ρ_V of the verifier $V_G(\Omega)$, we have $l_V(k), \dots, l_V(n+1) \in L_{s_\Omega}$. In particular, $l'_V = l_V(n+1)$ satisfies $l'_V \in L_{s_\Omega}$. ■

Lemma 2 states that for any indeterminate cycle, there necessarily exist two cycles in the verifier $V_G(\Omega)$: one composed only of final states and the second composed only of states that are not final states, both of which lead to the same sequence of observations as $D_G(\Omega)$.

Lemma 2. Given an indeterminate cycle $c_D = l_D(1) \dots l_D(n)$, there exist two cycles $c_V = l_V(1) \dots l_V(m)$ with $l_V(1), \dots, l_V(m) \in L_V \setminus L_{s_\Omega}$ and $c'_V = l'_V(1) \dots l'_V(m')$ with $l'_V(1), \dots, l'_V(m') \in L_{s_\Omega}$ in the verifier $V_G(\Omega)$ that lead to the same sequence of observations.

Proof. Each state $l_D(h)$, $h = 1, ..., n$, is indeterminate and contains two or more states of $V_G(\Omega)$ (one being in L_{s_Ω} and the other being not).

Consider, in particular, $l_V(1) \in l_D(1) \cap L_{s_\Omega}$. Then, all successors of $l_V(1)$ also belong to L_{s_Ω} (Lemma 1). There exist $m_1 \geq 0$ successors of $l_V(1)$ that belong to $l_D(1) \cap L_{s_\Omega}$ and $l_V(m_1 + 1) \in l_D(2) \cap L_{s_\Omega}$. The same reasoning is repeated for $l_D(2), \dots l_D(n)$. As far as $c_D = l_D(1) \dots l_D(n)$ defines a cyclic run in $D_G(\Omega)$, i.e., $l_D(n) = l_D(1)$, $c_V = l_V(1) \dots l_V(m_1) l_V(m_1 + 1) \dots l_V(m_n + 1)$ defines also a cyclic run in $V_G(\Omega)$, i.e., $l_V(1) = l_V(m_n + 1)$, and all states of this run belong to L_{s_Ω}.

Consider now $l'_V(m') \in l_D(n)$ and $l'_V(m') \notin L_{s_\Omega}$. Then, all predecessors of $l'_V(m')$ also belong to $L_V \setminus L_{s_\Omega}$. Then, by a reasoning similar to the previous one, it becomes possible to design a cyclic run $c'_V = l'_V(1) \dots l'_V(m'_1) l'_V(m'_1 + 1) \dots l'_V(m'_n + 1)$ in $V_G(\Omega)$, such that all states of this run belong to $L_V \setminus L_{s_\Omega}$.

The observations resulting from the execution of this cyclic runs c_V and c'_V are identical because both coincide with cycle c_D from the perspective of an external observer. ■

Lemma 3 states that all strings consistent with a given sequence of observations, driving the diagnoser state to a subset of the set of final states, are necessary in the accepted language of the verifier.

Lemma 3. Given a diagnoser $D_G(\Omega)$ and a sequence of observations w_q, let $l_D = e_D(l_0^D, w_q)$. If $l_D \subseteq 2^{L_{s_\Omega}}$, then $\mathscr{P}^{-1}(w_q) \subseteq \mathcal{L}_\Omega(G)$.

Proof. Consider a string w with $\mathscr{P}(w) = \mathscr{P}(w_q)$. Let $e_D(l_0^D, w_q) = l_D$ and $l_D = \{l_{j_1}^{s_\Omega}, \dots, l_{j_k}^{s_\Omega}\} \subseteq 2^{L_{s_\Omega}}$. Since $l_0^D = R_{V_G(\Omega)}(l_0^V, \lambda)$ and $l_D = e_D(l_0^D, w_q) = \{l_1^{s_\Omega}, \dots, l_k^{s_\Omega}\}$, for each $m = 1, \dots, k$, there exists the string w_m in the verifier $V_G(\Omega)$ generating the run ρ_m from initial state l_0^V

$$\rho_m : l_V(0) \xrightarrow{\sigma_0} l_V(1) \xrightarrow{\sigma_1} \cdots \xrightarrow{\sigma_{m-1}} l_V(m)$$

such that $l_V(m) = l_{j_m}^{s_\Omega}$. According to Definition 8, we have $\mathscr{P}(w_m) = w_q$. Since for all $m = 1, \dots, k$, $l_{j_m}^{s_\Omega} \in L_{s_\Omega}$, according to Definition 6, we have $w_m \in \mathcal{L}_\Omega(G)$ and for the state $l_D = \{l_{j_1}^{s_\Omega}, \dots, l_{j_k}^{s_\Omega}\}$, $\mathscr{P}^{-1}(w_q) \subseteq \mathcal{L}_\Omega(G)$. ■

Proposition 3 provides sufficient condition to check diagnosability with a given fault pattern.

Proposition 3. Let G be an LFA that satisfies assumption H. G is diagnosable with regard to Ω if there is no indeterminate cycle in its corresponding diagnoser $D_G(\Omega)$.

Proof. Let $w \in \mathcal{L}_\Omega(G)$ of length k_0 and $w' \in \mathcal{L}(G)/w$ of arbitrarily large length k. Then, consider $w_q = \mathscr{P}(ww')$ with $w_q = q_0 \dots q_n$ in $D_G(\Omega)$, $w'' \in \mathscr{P}^{-1}(w_q)$ with $w'' = \sigma''_0 \dots \sigma''_m$ and the runs respectively associated to w_q in $D_G(\Omega)$ and to ww' and w'' in $V_G(\Omega)$

$$\rho_D : \ l_D(0) \xrightarrow{q_0} l_D(1) \xrightarrow{q_1} \cdots \xrightarrow{q_n} l_D(n+1),$$

$$\rho_V : l_V(0) \xrightarrow{\sigma_0} \cdots l_V(k_0 + 1) \xrightarrow{\sigma'_1} \cdots \xrightarrow{\sigma'_k} l_V(k_0 + k + 1),$$

$$\rho''_V : \ l''_V(0) \xrightarrow{\sigma''_0} l''_V(1) \xrightarrow{\sigma''_1} \cdots \xrightarrow{\sigma''_m} l''_V(m+1).$$

Thanks to Assumption H, $\mathcal{L}(G)$ is live, and the length k of w' can be as large as necessary. In addition, silent cycles do not exist in G. Increasing the length of w' will necessarily result (sooner or later) in an observable jump where the state of the observer will change. Thus, the length n of w_q will also increase. For a similar reason the lengths m of all strings consistent with w'' also increase. Consequently, one can find k large enough such that (i) there exists $n' < n+1$ such that $l_D(n+1) = l_D(n')$ (the last part of ρ_D is cyclic since $G_D(\Omega)$ has a finite number of states), (ii) $l_D(n+1)$ is determinate since $D_G(\Omega)$ has no indeterminate cycle and thanks to Lemma 3. Observe that $l_V(k_0+k+1) \in L_{s_\Omega} \cap l_D(n+1)$. Then, for all $l_V \in l_D(n+1)$, we have $l_V \in L_{s_\Omega}$ and in particular, $l''_V(m+1) \in L_{s_\Omega}$.

Then, we conclude that there exists $k \in \mathbb{N}_0$ such that for all $w' \in \mathcal{L}(G)/w$, $|w'| \geq k \Rightarrow \mathscr{P}^{-1}(\mathscr{P}(ww')) \subseteq \mathcal{L}_\Omega(G)$. Thus, system G is diagnosable with regard to fault pattern Ω. ■

Example 4. Consider an LFA G in Fig. 2(a), and its verifier $V_G(\Omega)$ in Fig. 2(b). According to Definition 8, we obtain the diagnoser $D_G(\Omega)$ shown in Fig. 4. According to Proposition 3, we know that system G is not diagnosable with regard to pattern Ω since there exists an indeterminate cycle $l_4^D l_5^D l_6^D$ in $D_G(\Omega)$.

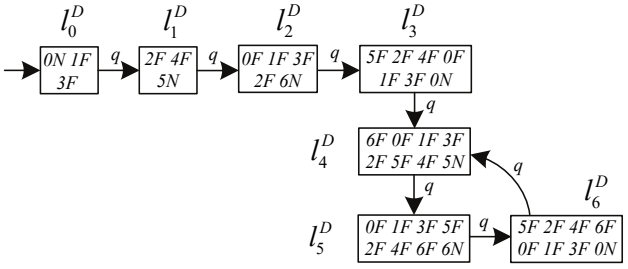

Fig. 4. Diagnoser $D_G(\Omega)$.

4. CONCLUSION

This paper deals with fault pattern diagnosis of discrete event systems. We propose a verifier based on state isolation properties. The detection of fault patterns can be obtained by analyzing the structure of the verifier. On the basis of synchronous product, we introduce the other verifier for fault pattern detection. We prove that the verifier is equivalent to the one that can be obtained from the synchronous product. Then, the diagnosers are provided for fault pattern diagnosability. Since the diagnoser is of exponential complexity, in the future work, we will improve the structures to reduce the complexity. Furthermore, we will consider diagnosis issues for timed fault patterns.

REFERENCES

Biswas, S. (2012). Diagnosability of discrete event systems for temporary failures. *Computers and Electrical Engineering*, 38(6), 1534–1549.

Boussif, A., Ghazel, M., and Basilio, J.C. (2021). Intermittent fault diagnosability of discrete event systems: an overview of automaton-based approaches. *Discrete Event Dynamic Systems*, 31(1), 59–102.

Cassandras, C.G. and Lafortune, S. (2009). *Introduction to discrete event systems*. Springer Science and Business Media.

Contant, O., Lafortune, S., and Teneketzis, D. (2004). Diagnosis of intermittent faults. *Discrete Event Dynamic Systems*, 14(2), 171–202.

Gougam, H.E., Pencolé, Y., and Subias, A. (2017). Diagnosability analysis of patterns on bounded labeled prioritized petri nets. *Discrete Event Dynamic Systems*, 27(1), 143–180.

Gougam, H.E., Subias, A., and Pencolé, Y. (2014). Discriminability analysis of supervision patterns by net unfoldings. *IFAC Proceedings Volumes*, 47(2), 459–464.

Hadjicostis, C.N. (2019). *Estimation and Inference in Discrete Event Systems: A Model-Based Approach with Finite Automata*. Springer, Cham.

Jeron, T., Marchand, H., Pinchinat, S., and Cordier, M.O. (2006). Supervision patterns in discrete event systems diagnosis. In *Proceedings of 8th International Workshop on Discrete Event Systems*.

Jiang, S., Huang, Z., Chandra, V., and Kumar, R. (2001). A polynomial algorithm for testing diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 46(8), 1318–1321.

Jiang, S. and Kumar, R. (2004). Failure diagnosis of discrete-event systems with linear-time temporal logic specifications. *IEEE Transactions on Automatic Control*, 49(6), 934–945.

Jiang, S., Kumar, R., and Garcia, H.E. (2003). Diagnosis of repeated/intermittent failures in discrete event systems. *IEEE transactions on robotics and automation*, 19(2), 310–323.

Lin, F. (1994). Diagnosability of discrete event systems and its applications. *Discrete Event Dynamic Systems*, 4(2), 197–212.

Ogheneovo, E.E. (2018). A new algorithm for determining the equivalence of two finite-state automata. *Journal of Advances in Mathematics and Computer Science*, 1–10.

Pencolé, Y. and Subias, A. (2017). Diagnosis of supervision patterns on bounded labeled petri nets by model checking. In *Proceedings 28th International Workshop on Principles of Diagnosis (DX 2017)*, volume 4, 184–199.

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. (1995). Diagnosability of discrete-event systems. *IEEE Transactions on automatic control*, 40(9), 1555–1575.

Yan, Y., Ye, L., and Dague, P. (2010). Diagnosability for patterns in distributed discrete event systems. In *Proceedings of the 21st International Workshop on Principles of Diagnosis DX'10, Portland, OR Etats-Unis*, 345–352.

Ye, L. and Dague, P. (2012). A general algorithm for pattern diagnosability of distributed discrete event systems. In *Proceedings of the 24th IEEE International Conference on Tools with Artificial Intelligence*, 130–137.

Yoo, T.S. and Lafortune, S. (2002). Polynomial-time verification of diagnosability of partially observed discrete-event systems. *IEEE Transactions on automatic control*, 47(9), 1491–1495.