Verification of Pattern-Pattern Diagnosability in Partially Observed Discrete Event Systems

Ziyue Ma, *Member, IEEE*, Yin Tong, *Member, IEEE*, and Carla Seatzu, *Senior Member, IEEE*

***Abstract*—This work studies a new notion of diagnosability called the *pattern-pattern diagnosability* in discrete event systems modeled by partially observable finite state automata. Suppose that in the system there are some sequences of events that are undesirable to happen which we call the *fault pattern*. We want to determine whether the occurrence of the fault pattern can be determined before some sequences — which we call the *critical pattern* and may cause fatal consequences after the fault pattern — are completed. Both fault and critical patterns are assumed to be regular and hence are described by the languages accepted by finite automata. We propose a novel notion of *pattern-pattern diagnosability* (PP-diagnosability) which requires that the occurrence of a fault pattern can always be detected before the completion of a critical pattern thereafter. The properties of PP-diagnosability, and the relations between PP-diagnosability and conventional diagnosability are studied. Then we propose a method to verify PP-diagnosability using a structure called the *pattern-pattern verifier*. The complexity of the proposed method is polynomial in the number of states of the plant and the two pattern automata.**

***Index Terms*—Discrete event systems, fault diagnosis, pattern diagnosability, automata**

I. Introduction

Fault diagnosis in discrete event systems [1], [2] has drawn considerable attention over the past decades. The aim of fault diagnosis is to infer whether or not a *faulty behavior* has occurred based on the observation of the sequence of events generated by a system. *Diagnosability* [3], [4] guarantees that the occurrence of any fault can be detected in a finite number of future steps after its occurrence.

The work on fault diagnosis in discrete event systems is initially proposed in [3], [4]. In [3] a fault is modeled by a particular unobservable event called the *fault event*. It is proved [3] that diagnosability can be verified by checking the non-existence of the so-called indeterminate cycles in the *diagnoser automaton* of the plant. Later, in [5], [6], a polynomial-time algorithm for the verification of diagnosability is proposed using *verifier automata*. Various notions of diagnosability have then been proposed and investigated in the last decades. References [7] and [8] studied the robust diagnosability in discrete event systems subject to intermittent and permanent sensor losses. In [9] a notion called *pattern diagnosis* is proposed where the fault is not modeled by a single event but a set of finite strings. Since in safety-critical systems the delay in the notion of diagnosability could be too long to take reconfiguration actions, a quantitative version of diagnosability, namely k*-diagnosability*, is developed [10], [11], which requires that the occurrence of faults be determined within at most k-observable events upon its occurrence. The work in [12] considers repairable faults, i.e., faults that may be automatically (and possibly stealthy) repaired. The notion of T-diagnosability in this case requires that the occurrence of a fault is not only detected in finite time but also before it is repaired. Other works include the diagnosability verification in decentralized [13], [14] and stochastic systems [15]. Besides finite state automata, diagnosability in Petri nets is also investigated [16]–[20].

By the conventional notion of diagnosability, a plant is considered diagnosable (resp., k-diagnosable) if the occurrence of the fault can be detected by observing a finite number of (resp., at most k) events thereafter. Such a condition only specifies the length of the observation (after the fault) but not the content of it. However, we believe that such a notion of diagnosability may be too strict in practice. In fact, in many cases observing an arbitrarily long sequence of events after a fault may be no harm: only some crucial sequences of events may lead to fatal consequences.

Based on this consideration, in this paper we pro-

This work was supported in part by the National Natural Science Foundation of China under Grant No. 62373313, Shaanxi Provincial Natural Science Foundation under Grant No. 2022JM-323, the Natural Science Foundation of Sichuan Province under Grant No. 2022NSFSC0955, and the Fundamental Research Funds for the Central Universities under Grant No. ZYTS23023.

Z. Ma is with the School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China (email: maziyue@xidian.edu.cn, maziyue@gmail.com).

Y. Tong is with the School of Information Science and Technology, Southwest Jiaotong University, Chengdu 611756, China (e-mail: yintong@swjtu.edu.cn).

C. Seatzu is with the Department of Electrical and Electronic Engineering, University of Cagliari, Cagliari 09123, Italy (email: carla.seatzu@unica.it).

pose a new notion of diagnosability called the *pattern-pattern diagnosability* (PP-diagnosability) that concerns the possibility of detecting the faulty behavior before the execution of some critical sequences of events (instead of a finite number of arbitrary observable events). Consider a plant modeled by a partially observed finite state automaton. The undesired behavior of the plant is modeled by a *fault pattern*. On the other hand, a *critical pattern* is (part of) the behavior of the plant whose completion (after the occurrence of the fault pattern) may cause fatal consequences. We assume that both fault and critical patterns are described by two regular languages K_f and K_c, respectively. A plant is said to be *pattern-pattern diagnosable* (PP-diagnosable) with respect to a fault pattern K_f and a critical pattern K_c (also called (K_f, K_c)-diagnosable for short) if the occurrence of the fault pattern can always be detected *before* the critical pattern is completed afterwards.

The concept of pattern diagnosability is first proposed in [9], [21] and is extended to Petri net models [22], [23]. In [9], [21] a diagnoser-automaton-based method is developed to verify pattern diagnosability. We point out that the pattern diagnosability in [9], [21] can be formulated as PP-diagnosability proposed in this paper with particular fault and critical patterns. The main differences are: (i) pattern diagnosability in [9], [21] requires that the fault pattern be detected in a finite number of future observations regardless its contents, while PP-diagnosability requires that the fault pattern be detected before a pre-defined critical pattern, (ii) in [9], [21] some assumptions are made on the fault pattern (e.g., the fault pattern is a bounded set, all sequences in the fault pattern contain at least one observable event, etc.), while PP-diagnosability defined in this paper does not rely on these assumptions. Another closely related notion called *safe diagnosability* is studied in [24]. Safe diagnosability is stronger than conventional diagnosability since, besides diagnosability condition, it also requires that the shortest continuation that guarantees the detection of the fault does not contain any predefined illegal strings. In PP-diagnosability, however, we are not caring of whether the fault can be detected in finite steps but whether the occurrence of the fault pattern is detected before some critical strings happen regardless the number of steps to detect it. Overall, with PP-diagnosability we are relaxing the diagnosability property to better fit practical applications.

The main contributions of this work are summarized as follows:

- First, the notion of PP-diagnosability is proposed and its relations with conventional notions of diagnosability are studied. We show that the conventional diagnosability [3] and pattern diagnosability [9], [21] can be formulated by PP-diagnosability with particular fault and critical patterns.
- We then study the properties of PP-diagnosability. We prove that PP-diagnosability is preserved if we shrink the critical pattern while fixing the fault pattern. Moreover, we present an interesting, counterintuitive example to show that PP-diagnosability is not preserved if we modify (either enlarge or shrink) the fault pattern while fixing the critical pattern. On the other hand, we observe that in general PP-diagnosability may not be preserved while modifying the fault pattern.
- Finally, we propose a method to verify PP-diagnosability for a given fault pattern K_f and a critical pattern K_c. We first augment K_f and K_c to synthesize two automata called the *pattern recognizers*. Then we propose a structure called *pattern-pattern verifier* (PP-verifier) using *parallel composition*. We prove that the PP-diagnosability is equivalent to the non-existence of PP-violating states in the PP-verifier, and hence the verification of PP-diagnosability can be done by a structural inspection on the corresponding PP-verifier.
- The structural complexity of our approach is $O(n^2 \cdot (n_f^2 + n_f \cdot n_c))$ where n is the number of states of the plant, while n_f and n_c are the number of states of the automata that model the fault and the critical pattern K_f and K_c, respectively. Besides the merit of generality, our method has computational advantages with respect to the methods in [9], [21] since we do not need to construct observer/diagnoser of the system.

II. Preliminaries

A. Deterministic Finite Automaton

A *deterministic finite automaton* (*automaton* for short) is a four-tuple $G = (Q, E, \delta, q_0, Q_m)$ where Q is a set of *states*; E is a set of *events*; $\delta : Q \times E \to Q$ is the partial transition function; $q_0 \in Q$ is the initial state; $Q_m \subseteq Q$ is the set of final (marked) states.

We use E^* to denote the *Kleene closure* of E, consisting of all finite sequences composed by the events in E (including the *empty sequence* ε). The transition function δ is extended to $\delta^* : Q \times E^* \to Q$ by recursively defining $\delta^*(q, \varepsilon) = q$ and $\delta^*(q, se) = \delta(\delta^*(q, s), e)$, where $s \in E^*$ and $e \in E$. The *generated language* of G is defined as $L(G) = \{s \in E^* \mid \delta^*(q_0, s) \in Q\}$. The *accepted language* of G is defined as $L_m(G) = \{s \in E^* \mid \delta^*(q_0, s) \in Q_m\}$.

The *accessible part* of an automaton $G = (Q, E, \delta, q_0, Q_m)$, denoted as $Ac(G)$, is the automaton

$G' = (Q', E, \delta', q_0, Q'_m)$ obtained from G by removing all unreachable states and their corresponding transitions. Precisely speaking, $Q' = \{q \in Q \mid (\exists s \in L(G))\ \delta^*(q_0, s) = q\}$, $Q'_m = \{q_m \in Q_m \mid (\exists s \in L(G))\ \delta^*(q_0, s) = q_m\} = Q' \cap Q_m$, and δ' is the restriction of δ to $Q' \times E \to Q'$.

A sequence $\bar{s} \in E^*$ is a *prefix* of a sequence $s \in E^*$ if there exists $s' \in E^*$ such that $s = \bar{s}s'$. In particular, prefix $\bar{s}$ is *strict* if $s' \neq \varepsilon$. The *prefix closure* of $s \in E^*$ is the set $\mathbf{Pr}(s) = \{s' \in E^* \mid (\exists s'' \in E^*)\ s's'' = s\}$. The length of a sequence $s \in E^*$, denoted by $|s|$, is the number of events in s.

A language $L \subseteq E^*$ is called *right-closed* (resp., *left-closed*) if for all $s \in L$ and $s' \in E^*$, it holds that $ss' \in L$ (resp., $s's \in L$).

Definition 1: [25] Given two automata $G' = (Q', E', \delta',\ q'_0, Q'_m)$ and $G'' = (Q'', E'', \delta'', q''_0, Q''_m)$, the *product* of G' and G'' is the automaton

$$G' \times G'' = Ac(Q' \times Q'', E' \cup E'', \delta, (q'_0, q''_0), Q'_m \times Q''_m),$$

where δ is defined as: $\delta((q'_i, q''_j), e) = (\delta'(q'_i, e), \delta''(q''_j, e))$ if both $\delta'(q'_i, e)$ and $\delta''(q''_j, e)$ are defined; otherwise $\delta((q'_i, q''_j), e)$ is undefined. ♢

For G' and G'' defined on the same alphabet, i.e., $E' = E''$, $L(G' \times G'') = L(G') \cap L(G'')$ and $L_m(G' \times G'') = L_m(G') \cap L_m(G'')$ hold.

B. Partial Observation and Diagnosability

Given an automaton $G = (Q, E, \delta, q_0, Q_m)$, the event set E is partitioned into the set of *observable events* E_o and the set of *unobservable events* E_{uo}, i.e., $E = E_o \cup E_{uo}$. Given a sequence $s \in E^*$, the *natural projection* $P : E^* \to E_o^*$ is defined as:

$$\begin{cases} P(\varepsilon) = \varepsilon; \\ P(e) = e,\ if\ e \in E_o; \\ P(e) = \varepsilon,\ if\ e \in E_{uo}; \\ P(se) = P(s)P(e),\ \text{for } s \in E^*, e \in E. \end{cases} \tag{1}$$

When a sequence $s = e_1 e_2 \dots e_k \in L(G)$ occurs in a plant, the corresponding observation is $w = P(s) = P(e_1)P(e_2)\dots P(e_k)$. Two sequences s', s'' are said to be *observation-equivalent* if $P(s') = P(s'')$. The inverse projection $P^{-1} : E_o^* \to 2^{L(G)}$ is defined as:

$$P^{-1}(w) = \{s \in L(G) \mid P(s) = w\},$$

i.e., $P^{-1}(w)$ consists of all the sequences in $L(G)$ whose observations are w.

In the literature on fault diagnosis in discrete event systems, the unobservable event set E_{uo} is further partitioned into the set of *regular unobservable events* E_r and the set of *fault events* E_f. Some works consider different fault types, i.e., $E_f = \{f_1, f_2, ..., f_n\}$. In this paper we consider a single type of faults, i.e., $E_f = \{f\}$ for simplicity, but all the proposed results may be easily extended to the case of multiple types of faults.

Definition 2: [3] Given a live plant $G = (Q, E, \delta, q_0, Q_m)$ where $E = E_o \cup E_{uo}$ and $E_{uo} = E_r \cup \{f\}$, G is *diagnosable* with respect to fault f if for all sequences $\sigma f \in L(G)$, there exists $k_\sigma \in \mathbb{N}$ such that for all $\sigma f \sigma' \in L(G)$,

$$|\sigma'| \geq k_\sigma \quad \Rightarrow \quad \forall \hat{\sigma} \in P^{-1}(P(\sigma f \sigma')) : f \in \hat{\sigma}.$$

System G is *k-diagnosable* if there exists a universal $k \in \mathbb{N}$ such that for all $\sigma f \in L(G)$, for all $\sigma f \sigma' \in L(G)$,

$$|\sigma'| \geq k \quad \Rightarrow \quad \forall \hat{\sigma} \in P^{-1}(P(\sigma f \sigma')) : f \in \hat{\sigma}.$$

♢

In plain words, diagnosability is a property such that if the fault has occurred after σ, then after a sufficiently long continuation σ' (with $|\sigma'| \geq k$) all sequences $\hat{\sigma}$ that produce the same observation as that of $\sigma f \sigma'$ must contain the fault. This implies that the occurrence of fault f can be detected within a finite delay of observations. In the framework of finite state automata, diagnosability and k-diagnosability are mathematically equivalent. The upper bound on the length of the confused path in a verifier automaton is n^2 where n is the number of states in the plant. Hence, a plant G is diagnosable if and only if it is (n^2)-diagnosable.

III. Problem Formulation

A. Fault and Critical Patterns

The following example illustrates the motivation of defining a new notion of diagnosability apart from the original ones.

Example 1: Consider the plan of a robot in Figure 1 *without* the dashed blue zone. The corresponding automaton model is shown in the same figure *without* the self-looped event d at state 5. In this system events a, b, c, d, e, f are observable while event u is unobservable. The robot is supposed to transport a part from zone 0 to zone 6 along the upper track via zones 3 and 5. Let us consider a fault such that the robot erroneously goes to zone 2 and picks a wrong part. Hence, the fault pattern consists of all sequences ending with au. Our safety requirement is that such a fault behavior au should be detected before the wrong part is processed through the bottom track routine, i.e., before the execution of a critical sequence def. On the other hand, processing the wrong part through the upper track is acceptable: any

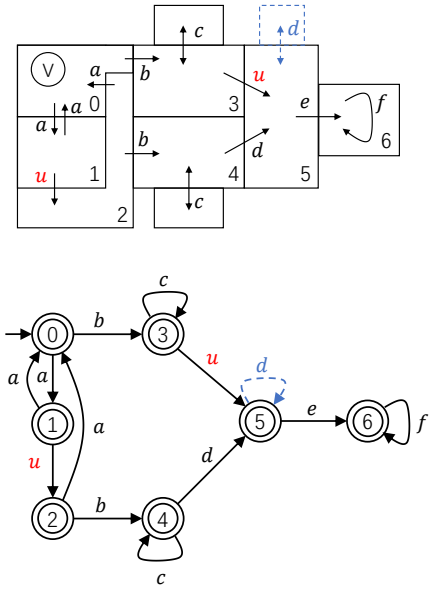

Fig. 1. A plan of a zone-controlled AGV and its corresponding automaton model with unobservable event u.

sequence corresponding to the upper track routine (such as $auabuef$) causes no harm.

One can readily verify that such a system is not pattern-diagnosable according to the definition in [9]. In fact, after the fault pattern occurs, the faulty sequence $auabc^k$ and the non-faulty sequence $aabc^k$ have the same observation $P(auabc^k) = P(aabc^k) = aabc^k$, where k can be arbitrarily large. Therefore, there does not exist an integer $k \in \mathbb{N}$ such that the occurrence of fault sequence au can be determined by observing at most k observable events. However, the occurrence of fault pattern au can always be detected before the critical sequence def is completed. Hence, observing $aabc^k$ for arbitrary large k does not imply that the safety requirement is violated, since the wrong part is never transported to zone 6. In fact, the occurrence of fault pattern au can be determined as soon as we observe event d, which always happens before the completion of critical sequence def. ♢

Motivated by Example 1, we formulate the notions of fault and critical patterns. Formally speaking, given a plant G with event set E, a *fault pattern* is a language $K_f \subseteq E^*$ which describes the undesirable behavior to be diagnosed. On the other hand, a *critical pattern* is a language $K_c \subseteq E^*$ which describes the behaviors that may cause fatal consequences after the occurrence of the fault pattern. In this work we consider regular patterns such that both the fault pattern K_f and the critical pattern K_c are accepted by automata H_f and H_c, respectively. In other words, $L_m(H_f) = K_f$ and $L_m(H_c) = K_c$. For simplicity, in the sequel, sequences in K_f and in K_c are called "fault sequences" and "critical sequences",

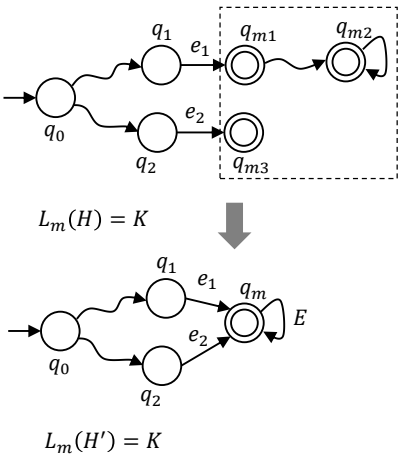

Fig. 2. Pretreatment of K_f, K_c.

respectively.

Moreover, we assume that both K_f and K_c are right-closed. The fault pattern K_f is right-closed since we assume that the fault is not self-repairable (for self-repairable fault, see e.g. [12])) once occurred. On the other hand, notice that if a faulty sequence $s_f \in K_f$ can be detected before the occurrence of a critical sequence $s_c \in K_c$, then s_f can also be detected before any continuation of s_c, which implies that the critical pattern K_c is also right-closed.

Remark 1: For the aim of fault diagnosis, we are not interested in the accepted language of a plant G. Hence, all states in G are considered final, i.e., $Q_m = Q$, as shown in Example 1. ♢

B. Pattern-Pattern Diagnosability

Since both K_f and K_c are right-closed, all successor states of the final states in H_f and H_c (if they exist) are also final. All final states in H_f and H_c can thus be fused into a single final state, respectively, as shown in Figure 2. For simplicity, in the sequel we omit the step of the final-state-fusion and suppose that H_f and H_c consist of a single final state with a self-loop labeled by E. Moreover, we assume that H_c and H_f are *complete*, i.e., $L(H_f) = L(H_c) = E^*$. If H_c and/or H_f are not complete, they can be transformed into their equivalent complete automata without affecting their accepted languages (see Ch.2 in [25]).

Example 2 (Ex. 1 cont.): In Example 1, the fault pattern $K_f = \{s \in E^* \mid (s', s'' \in E^*)\, s = s'aus''\}$ is the marked language of automaton H_f in Figure 3. The critical pattern $K_c = \{s \in E^* \mid (s_0, s_1, s_2, s_3 \in E^*)\, s = s_0 d s_1 e s_2 f s_3\}$ is the marked language of automaton H_c in the same figure. ♢

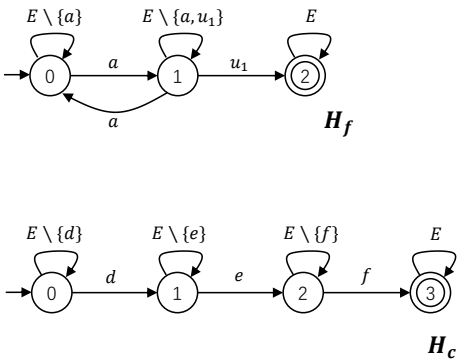

Fig. 3. H_f and H_c in which $E = \{a, b, c, d, f, g\}$.

Like most literature on diagnosis, in this paper we assume that the plant G is *live*, i.e., at each state of G there exists at least one output arc. If the plant is not live, it can be transformed into an equivalent live plant by adding self-loops of unobservable events at the dead states [26]. Now we are ready to define the notion of pattern-pattern diagnosability (PP-diagnosability) as follows.

Definition 3: [PP-diagnosability] Given a plant $G = (Q, E, \delta, q_0, Q_m)$ where $E = E_o \cup E_{uo}$, a fault pattern $K_f \subseteq E^*$, and a critical pattern $K_c \subseteq E^* \setminus \{\varepsilon\}$[1], G is *pattern-pattern diagnosable* (PP-diagnosable) with respect to (K_f, K_c) if for all sequences $s_f \in L(G) \cap K_f$, the following condition holds:

$$\begin{aligned}&(\forall s_c \in K_c)\ [s_f s_c \in L(G)] \Rightarrow \\ &(\exists \bar{s} \in \mathbf{Pr}(s_f s_c), \bar{s} \neq s_f s_c)\ P^{-1}(P(\bar{s})) \subseteq K_f.\end{aligned} \tag{2}$$

♢

The notion of PP-diagnosability can be interpreted as follows. For any fault sequence s_f that can be generated by the plant and that belongs to the fault pattern K_f, before a critical sequence s_c in the critical pattern is completed thereafter, there necessarily exists $\bar{s}$ that is a *strict* prefix of $s_f s_c$ such that all sequences that look like $P(\bar{s})$ belong to the fault pattern K_f. This means that the occurrence of fault sequence s_f can always be detected when observing $P(\bar{s})$, before the critical sequence s_c is completed afterwards. Note that since K_f is right-closed, the last term "$P^{-1}(P(\bar{s})) \subseteq K_f$" in Eq. (2) means that there exists at least one string $\hat{s} \in P^{-1}(P(\bar{s}))$ that can be written as $\hat{s} = uv$ where $u \in K_f$" (v can be the empty string). To simplify the presentation, in the sequel "G is PP-diagnosable with respect to fault pattern K_f and critical pattern K_c" is also expressed as "G is (K_f, K_c)-*diagnosable*".

[1]We require that the fault be detected before a critical sequence in K_c occurs, which is not valid if $\varepsilon \in K_c$.

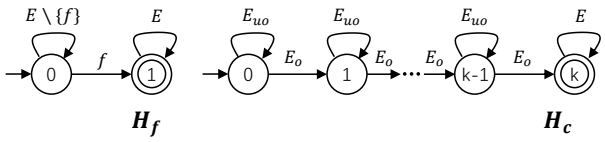

Fig. 4. H_f and H_c for conventional k-diagnosability.

Remark 2: In this paper we impose no assumption on the observability of fault pattern K_f and critical pattern K_c: both patterns can be arbitrary regular languages over alphabet E. This is different from the conventional fault event diagnosis (which usually assumes that the fault event is unobservable) and the pattern diagnosability [9] (which assumes that the fault pattern is bounded and each sequence in it contains at least one observable event). Moreover, unlike the other works in the literature, we do not require that the plant does not contain unobservable cycles (i.e., cycles with all events unobservable). ♢

At the end of this section, we discuss the relation of PP-diagnosability and conventional diagnosabilities. First, we point out that the conventional notion of k-diagnosability [10] can be formulated in terms of PP-diagnosability with particular K_f and K_c. Precisely speaking, k-diagnosability is equivalent to (K_f, K_c)-diagnosability with fault pattern $K_f = (E \setminus \{f\})^* f E^*$ and critical pattern $K_c = (E_{uo}^* E_o)^k E^*$. Such patterns can be accepted by the corresponding automata H_f and H_c in Figure 4. Since in the framework of automata, diagnosability is equivalent to (n^2)-diagnosability where n is the number of plant states, PP-diagnosability is more general than the conventional notion of diagnosability. Second, pattern diagnosability in [9] can also be formulated as PP-diagnosability with a particular critical pattern K_c that is the accepted language of automaton H_c in Figure 4 with $k = 2^n$. The reason is: by [9] a pattern is diagnosable if and only if there does not exist an indeterminated cycle in the observer of the plant, and the length of an indeterminated acyclic path in the observer is bounded by 2^n.

IV. Properties of Pattern-Pattern Diagnosability

A property of the conventional diagnosability is that k-diagnosability implies k'-diagnosability for any $k' > k$. In plain words, if a fault can be detected within a delay k, then it can also be detected within a delay larger than k, i.e., diagnosability is preserved when the delay k increases. In this section, we study under which condition the property of PP-diagnosability is preserved. The following proposition shows that for a plant G that is PP-diagnosable with respect to (K_f, K_c), if we fix the

fault pattern K_f and modify the critical pattern K_c, then G is PP-diagnosable if the new critical pattern K_c' is a subset of K_c.

Proposition 1: For a given plant $G = (Q, E, \delta, q_0, Q_m)$, (K_f, K_c)-diagnosability implies (K_f, K_c')-diagnosability if $K_c' \subseteq K_c$.

Proof: We prove this proposition by contrapositive. Suppose that G is not (K_f, K_c')-diagnosable. By Definition 3, there exists a sequence $s = s_f s_c \in L(G)$ such that (i) $s_f \in K_f$, $s_c \in K_c'$, and (ii) there exists a sequence $\hat{s} \in P^{-1}(P(s_f s_c))$ such that $\hat{s} \notin K_f$. Since $K_c' \subseteq K_c$, s_c also belongs to K_c. Hence, sequence $s_f s_c \in L(G)$ satisfies: (i) $s_f \in K_f$, $s_c \in K_c$, and (ii) $\hat{s} \in P^{-1}(P(s_f s_c))$ such that $\hat{s} \notin K_f$. Therefore, G is not (K_f, K_c)-diagnosable. □

Proposition 1 is intuitive: if a fault sequence $s \in K_f$ can be detected before the completion of any sequence in K_c thereafter, then s can also be detected before the completion of any sequence in K_c'.

Now, let us consider the preserveness of PP-diagnosability when the fault pattern changes. One may conjecture that PP-diagnosability is preserved if we fix the critical pattern K_c and reduce the fault pattern K_f, a result analogous to Proposition 1. Precisely speaking, one may conjecture that (K_f, K_c)-diagnosability implies (K_f', K_c)-diagnosability if $K_f' \subseteq K_f$. However, such a conjecture is false.

Fact 1: (K_f, K_c)-diagnosability does not necessarily imply (K_f', K_c)-diagnosability if $K_f' \subsetneq K_f$.

Proof: We prove this fact by the following example. Consider the plant G in Figure 5 with alphabet $E = \{u, a, b\}$ in which event u is unobservable and a, b are observable. Consider two fault patterns $K_f = \{aE^*\} \cup \{uaE^*\}$, $K_f' = \{aE^*\}$ and one critical pattern $K_c = \{\{b, u\}^* a \{a, u\}^* bE^*\}$. One can verify that G is (K_f, K_c)-diagnosable. When fault sequence aa occurs, we observe $P(aa) = aa$ whose consistent plant sequences are: aa, uaa both of which are faulty. This also holds for fault sequence uaa. Hence, one can detect the fault by observing aa, before the completion of critical pattern ab. However, G is not (K_f', K_c)-diagnosable since in such a case uaa is not faulty. Since for faulty sequence aa there exists a non-faulty sequence uaa such that $P(uaa) = P(aa) = aa$, it is not possible to detect the faulty sequence aa before the next event b occurs — at this moment the critical pattern ab has been completed. □

The reason behind Fact 1 is that, when K_f is reduced to K_f', some faulty sequences in the plant may become non-faulty so that some originally fault-definite sequences may become fault-ambiguous. On the other hand, it is not difficult to understand that (K_f, K_c)-diagnosability does not necessarily imply (K_f', K_c)-diagnosability if $K_f' \supsetneq K_f$, since fault-ambiguous sequences may come from the sequences in $K_f' \setminus K_f$. Hence, PP-diagnosability may not be preserved if we fix the critical pattern K_c and vary the fault pattern K_f.

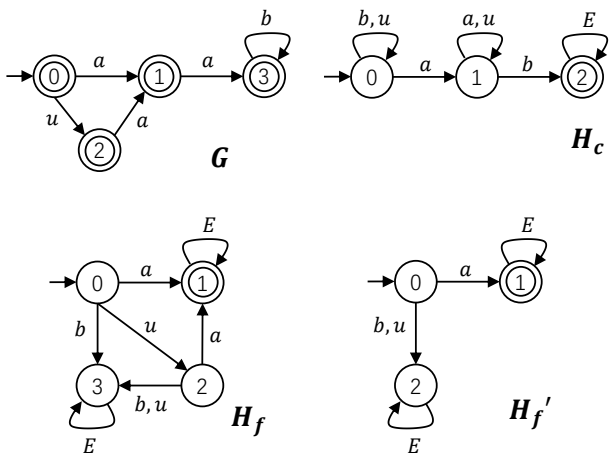

Fig. 5. Example for Fact 1.

V. Verification of PP-Diagnosability

In this section we propose a method to verify PP-diagnosability for a plant G with respect to fault and critical patterns K_f and K_c. The flowchart of our approach is depicted in Figure 6.

- First, the fault pattern automaton H_f is modified to $\hat{H}_f$ by removing the self-loop at its final state. An automaton H_{fc} is then computed which is the concatenation[2] of $\hat{H}_f$ and H_c (the critical pattern automaton).
- Then, two *pattern recognizers* are constructed which encode the information associated with the patterns: the *fault-pattern recognizer* H_{fr} is the product of G and H_f, while the *critical-pattern recognizer* H_{cr} is the product of G and H_{fc}.
- Then, we perform a (backward) *step shift* on the critical-pattern recognizer H_{cr} to obtain a new automaton $H_{cr,-1}$ in which all strict prefixes $\bar{s}$ in Eq. (2) are captured, namely they lead to final states in $H_{cr,-1}$.
- Next, a new automaton denoted as $V_{pp} = H_{fr} ||_o H_{cr,-1}$ and called *pattern-pattern verifier* (PP-verifier) is synthesized from H_{fr} (the fault-pattern recognizer) and $H_{cr,-1}$ (the *critical-pattern*

[2]The concatenation of two automata G_1 and G_2 is a new automaton G consisting of both G_1 and G_2 followed by adding to each final state of G_1 an ε-transition to the initial state of G_2. There hold: $L(G) = L(G_1) \cdot L(G_2)$ and $L_m(G) = L_m(G_1) \cdot L_m(G_2)$. See [25].

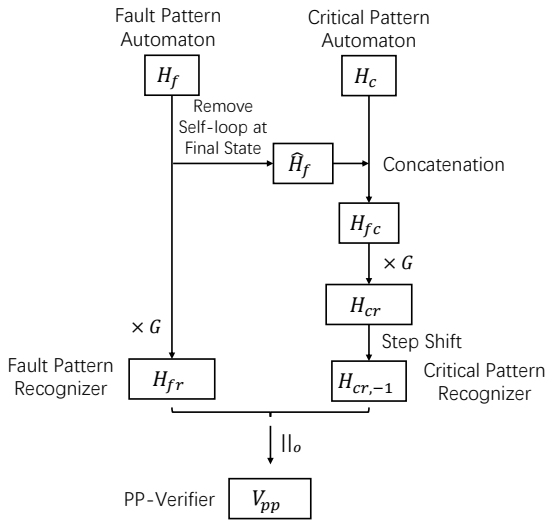

Fig. 6. The flowchart of the proposed approach for verification of PP-diagnosability.

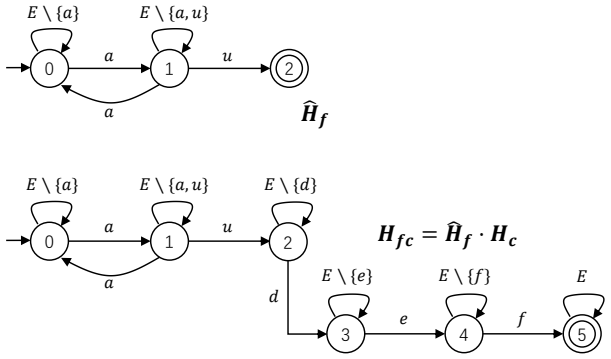

Fig. 7. Automaton $\hat{H}_f$ augmented from H_f and $H_{fc} = \hat{H}_f \cdot H_c$, where H_f, H_c are from Fig. 3.

recognizer after step-shift). Operator $||_o$ will be formally defined in the sequel.

- Finally, the verification of PP-diagnosability is done by checking the existence of a particular type of states called *pattern violating states* in the PP-verifier.

A. Augmentation on Fault Pattern Automaton H_f

To capture the completion of the critical pattern K_c after the occurrence of the fault pattern K_f, we concatenate the two patterns K_f and K_c to obtain a new pattern $K_f \cdot K_c$. This step can be done by simply concatenating the corresponding two automata $\hat{H}_f$ and H_c. In brief, automaton H_{fc} is obtained by adding an unobservable ε-transition from the final state of $\hat{H}_f$ to the initial state

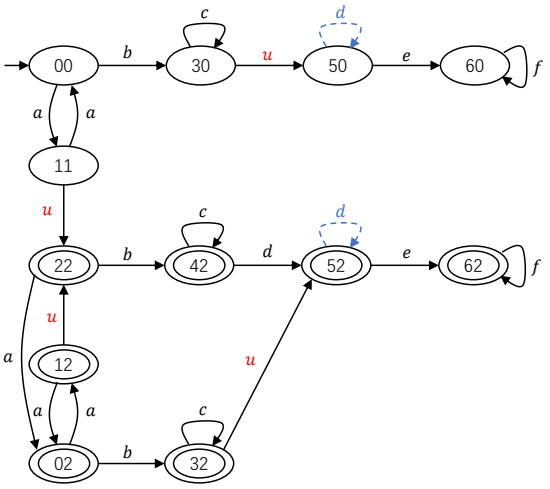

Fig. 8. Fault pattern recognizer H_{fr}.

of H_c.[3] Since there exists a unique final state in H_f, in $\hat{H}_f$ there also exists a unique final state that has no output arcs.

On the other hand, if K_c is left-closed, automaton H_{fc} can be straightforwardly obtained by fusing the final state of $\hat{H}_f$ and the initial state of H_c. For instance, for the fault and the critical patterns in Figure 3, the automaton $H_{fc} = \hat{H}_f \cdot H_c$ is depicted in Figure 7.

B. Pattern Recognizers

Now we combine the fault pattern automaton H_f and the automaton H_{fc} obtained above with the behavior of the plant G. We construct two automata called the *pattern recognizers*, which are the products (see Definition 1) of G with H_f and H_{fc}, respectively. The fault-pattern recognizer is denoted by $H_{fr} = G \times H_f$ (where the subscript "fr" denotes "fault-pattern recognizer"), while the critical-pattern recognizer is denoted by $H_{cr} = G \times H_{fc}$ (where the subscript "cr" denotes "critical-pattern recognizer"). Then, we have the following result.

Proposition 2: Given a plant G, a fault pattern K_f, and a critical pattern K_c, the following statements hold:

$$\begin{aligned} s = uv \in L(G), u \in K_f \;&\Leftrightarrow\; s \in L_m(H_{fr}) \\ s = uv \in L(G), u \in K_f, v \in K_c \;&\Leftrightarrow\; s \in L_m(H_{cr}) \end{aligned} \tag{3}$$

Proof: This proposition directly follows from the definition of the product operator and the construction of H_{fr} and H_{cr}. □

Example 3 (Ex. 2 cont.): For the plant G in Figure 1 *without* the dashed blue self-loop at state 5, the fault

[3]Automaton H_{fc} is not deterministic due to the introduction of ε-transition. However, we do not need to determinize H_{fc} since the method and the results in the sequel can be applied to non-deterministic finite automata.

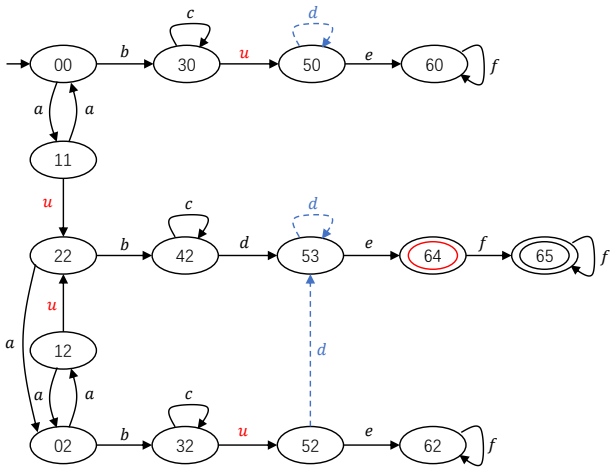

Fig. 9. Critical pattern recognizer H_{cr} (with state 64 unmarked) and corresponding $H_{cr,-1}$ (with state 64 marked).

pattern $K_f = L_m(H_f)$ in Figure 3, and the critical pattern $K_c = L_m(H_c)$ in Figure 3, the corresponding pattern recognizers H_{fr}, H_{cr} are shown in Figures 8 and 9, respectively, *without* the dashed blue transitions. There are 7 final states in H_{fr} while there is only one final state 65 in H_{cr}. For readability, state "(q'_i, q''_j)" is denoted as "ij" in the figure. ♢

Proposition 2 indicates that, by doing the two products $G \times H_f$ and $G \times H_{fc}$, $L_m(H_{fr})$ consists of all sequences $u \in L(G) \cap K_f$, and $L_m(H_{cr})$ consists of all sequences $uv \in L(G) \cap (K_f \cdot K_c)$ where $u \in L(G) \cap K_f$ and $v \in K_c$. However, since PP-diagnosability requires that the occurrence of the fault pattern be detected *before* the completion of the critical pattern, we are not interested in such $v \in K_c$ but those sequences v' that are the longest strict prefixes of critical sequences v which satisfies $|v| - |v'| = 1$. To this aim, we will next revise (which we call the *step-shift*) H_{cr} to a new automaton $H_{cr,-1}$ such that $L_m(H_{cr,-1})$ consists of all sequences that are strict prefixes of $L_m(H_{cr})$, i.e., $L_m(H_{cr,-1})$ consists of all sequences $uv \in L(G)$ such that (i) $u \in K_f$ and (ii) there exists an event $e \in E$ such that $uve \in L(G)$ and $ve \in K_c$. We also emphasize that, so far, the observability of events is not involved: when doing the product to synthesize H_{fr} and H_{cr}, all events in the corresponding automata involved are treated as observable. The observation structure will be encoded when constructing the pattern-pattern verifier in the next subsection.

C. Step Shift for Critical Pattern Recognizer H_{cr}

Suppose that in a system a critical sequence $v \in K_c$ may occur after a faulty sequence $u \in K_f$. To ensure PP-diagnosability, after the occurrence of u, we must determine the fault at least one event before completing v. To this aim, we modify the critical pattern recognizer H_{cr} to a new automaton $H_{cr,-1}$ in which all such sequences are captured. In terms of automata, this means that the augmented critical pattern recognizer $H_{cr,-1}$ differs from the critical pattern recognizer H_c for the fact that all the above sequences s also lead to final states (since K_c is right-closed).

Definition 4: Given a language $L \subseteq E^*$, we define the backward 1-step shift language L_{-1} as:

$$L_{-1} = L \cup \{s \in E^* \mid (\exists e \in E)\ se \in L\}.$$

♢

In plain words, L_{-1} is enlarged from L by adding the longest strict prefix of each sequence in L. To perform the 1-step shift on a given language L is not difficult. In fact, for $L = L_m(G)$ where $G = (Q, E, \delta, q_0, Q_m)$, L_{-1} is the accepted language of the automaton $G' = (Q, E, \delta, q_0, Q'_m)$ where $Q'_m = Q_m \cup \hat{Q}$ and

$$\hat{Q} = \{q \in Q \mid (\exists e \in E)\ \delta(q, e) = q' \in Q_m\}.$$

The states in $\hat{Q}$ are those in G which are coreachable to the original final states of G via only one event. By redefining the final states in automaton H_{cr} as described above, we obtain automaton $H_{cr,-1}$ whose marked language is $L_m(H_{cr,-1}) = L_{-1}$ where $L = L_m(H_{cr})$. Clearly, $L_m(H_{cr,-1})$ contains all sequences $s = uv \in L(G)$ such that $u \in K_f$ and there exists an event $e \in E$ such that $uve \in L(G)$ and $ve \in K_c$. Intuitively speaking, after the occurrence of a faulty sequence, the execution of a sequence in the shortest prefix sublanguage of $L_m(H_{cr,-1})$ is the last chance to successfully diagnose the fault and issue an alarm: if the fault cannot be determined at this moment, then a critical pattern may be completed after the execution of the next event.

Example 4 (Ex. 3 cont.): Consider the critical pattern recognizer H_{cr} in Figure 9 in which state 65 is the only final state. By marking state 64 as final (marked with a red circle) which is one-step co-reachable to the original final state 65, we obtain the automaton $H_{cr,-1}$ in the same figure. Clearly, $L_m(H_{cr,-1})$ is the language that is one-step shifted from $L_m(H_{cr})$. ♢

D. Pattern-Pattern Verifier

To check the condition in Eq. (2), it is sufficient to check for each sequence $s \in L(G)$ if there exists another sequence $s' \in L(G)$ that looks like s such that s contains the critical pattern after the fault pattern while s' does not contain the fault pattern. By Proposition 2, all sequences s that contain a fault subsequence (i.e., $s =$

$uv, u \in K_f$) belong to $L_m(H_{fr})$, and all sequences s' that contain a critical subsequence after a fault sequence (i.e., $s' = uv, u \in K_f$, $\exists e \in E$ such that $ve \in K_c$) belong to $L_m(H_{cr,-1})$. Hence, it is sufficient to examine all observation-equivalent sequences in H_{fr} and H_{cr}. In the literature, a tool widely used to check observation-equivalent sequences in partially observed automata is the so-called *verifier automaton* [6] (which is also called the *twin-plant*). Here, we slightly modify the approach to construct the verifier and call the resulting automaton *pattern-pattern verifier*.

Definition 5: Given two automata $G' = (Q', E, \delta', q'_0, Q'_m)$, $G'' = (Q'', E, \delta'', q''_0, Q''_m)$ where $E = E_o \cup E_{uo}$, the *verifier* composed by G' and G'' is the *nondeterministic finite automaton* $V_{(G',G'')} = G'||_o G'' = Ac(V, E, \delta_v, v_0)$ where (i) $V \subseteq Q' \times Q''$; (ii) the initial verifier state is $v_0 = (q'_0, q''_0)$; (iii) the transition relation δ_v is defined as follows:

- for $e \in E_o$:

$$\delta_v((q'_i, q''_j), e) = (\delta'(q'_i, e), \delta''(q''_j, e))$$

  if both $\delta'(q'_i, e)$ and $\delta''(q''_j, e)$ are defined;
- for $e \in E_{uo}$:

$$\delta_v((q'_i, q''_j), e) = \begin{cases} (\delta'(q'_i, e), q''_j), & \text{if } \delta'(q'_i, e) \text{ is defined} \\ (q'_i, \delta''(q''_j, e)), & \text{if } \delta''(q''_j, e) \text{ is defined} \end{cases}$$

♢

Definition 6: Given a plant G, its fault pattern recognizer H_{fr} and its critical pattern recognizer after step-shift $H_{cr,-1}$, the corresponding verifier $V_{pp} = H_{fr}||_o H_{cr,-1}$ is called the *pattern-pattern verifier* (PP-verifier). ♢

In plain words, the initial state is the pair (q'_0, q''_0) that consists of the two initial states in G' and G''. An observable event can occur if and only if both G' and G'' must be able to execute the event at q'_i and at q''_j, respectively. On the other hand, an unobservable event can occur if and only if either or both G' and G'' are able to execute it at q'_i and at q''_j. Differently from the original construction [6], here for an unobservable event e we do not impose the transition $\delta_v((q'_i, q''_j), e) = (\delta'(q'_i, e), \delta''(q''_j, e))$ at a verifier state (q'_i, q''_j) if both $\delta'(q'_i, e)$ and $\delta''(q''_j, e)$ are defined. This simplification does not cause essential difference with respect to the original construction in [6] (since in the case above state $(\delta'(q'_i, e), \delta''(q''_j, e))$ can still be created by $(q'_i, q''_j) - (q'_i, \delta''(q''_j, e)) - (\delta'(q'_i, e), \delta''(q''_j, e)))$ but reduces the interleaving of unobservable transitions in the verifier. Note that when doing $||_o$ we do not specify the set of final states in the resulting verifier, which is also different from the classical concurrent composition and product operations on automata.

Note that each state in V_{pp} is a pair (q', q'') whose first component q' is a state in H_{fr} and the second is a state in $H_{cr,-1}$. In the following we define *PP-violating states* in a PP-verifier. Then, we are ready to show that the non-existence of PP-violating states in the corresponding verifier is sufficient and necessary for PP-diagnosability, which is the main result of this section.

Definition 7: Given a PP-verifier $V_{pp} = H_{fr}||_o H_{cr,-1}$ where $H_{fr} = (Q_{fr}, E, \delta_{fr}, q_{0,fr}, Q_{m,fr})$ and $H_{cr,-1} = (Q_{cr}, E, \delta_{cr}, q_{0,cr}, \hat{Q}_{m,cr})$, a state (q', q'') in V_{pp} is called a *PP-violating state* if state q' is *not* final in H_{fr} (i.e., $q' \notin Q_{m,fr}$) and state q'' is final in $H_{cr,-1}$ (i.e., $q'' \in \hat{Q}_{m,cr}$). ♢

Example 5 (Ex. 4 cont.): Again consider the plant G in Figure 1 *without* the dashed blue self-loop at state q_5, the fault pattern $K_f = L_m(H_f)$ in Figure 3, and the critical pattern $K_c = L_m(H_c)$ in Figure 3. The corresponding PP-verifier $V_{pp} = H_{fr}||_o H_{cr,-1}$ that has 37 states is shown in Figure 10 *without* all dashed blue states and transitions. For readability, state "(q'_i, q''_j)" is denoted as "i, j" in the figure. ♢

Theorem 1: Given a plant $G = (Q, E, \delta, q_0, Q_m)$ where $E = E_o \cup E_{uo}$, a fault pattern $K_f \subseteq E^*$, and a critical pattern $K_c \subseteq E^*$, let $H_{fr} = (Q_{fr}, E, \delta_{fr}, q_{0,fr}, Q_{m,fr})$ and $H_{cr,-1} = (Q_{cr}, E, \delta_{cr}, q_{0,cr}, \hat{Q}_{m,cr})$ be the corresponding fault pattern recognizer and the recognizer after step-shift. Plant G is PP-diagnosable with respect to (K_f, K_c) if and only if the corresponding PP-verifier $V_{pp} = H_{fr}||_o H_{cr,-1}$ does not contain any PP-violating state.

Proof: By the construction of PP-verifier, there exists a state (q', q'') in V_{pp} if and only if there exist two sequences $s, s' \in L(H_{fr}) \cap L(H_{cr,-1})$ such that $\delta_{fr}(q_{0,fr}, s) = q'$, $\delta_{cr}(q_{0,cr}, s) = q''$, and $P(s) = P(s')$, where δ_{fr} and $q_{0,fr}$ (resp., δ_{cr} and $q_{0,cr}$) are the transition function and the initial state of H_{fr} (resp., $H_{cr,-1}$).

We now prove the "only if" part, and the "if" part follows an analogous argument. Suppose that in V_{pp} there exists a state (q', q'') such that q' is not final in H_{fr} and q'' is final in $H_{cr,-1}$. By Proposition 2, there exists two sequences $s, s' \in L(G)$ that satisfy the following conditions: (i) $P(s) = P(s')$; (ii) $s = uv \in L(G)$ with $u \in K_f, v \in K_{c,-1}$ (since $s \in L_m(H_{cr,-1})$ and by Proposition 2), and there exists $e \in E$ such that $se \in L(G)$ and $ve \in K_c$; (iii) s' does not have a prefix in K_f (since $s' \notin L_m(H_{fr})$ and by Proposition 2). Therefore, G is not PP-diagnosable with respect to (K_f, K_c). □

Example 6 (Ex. 5 cont.): In the PP-verifier in Figure 10 (*without* all dashed blue states and transitions) the final sub-states of H_{fr} and $H_{cr,-1}$ components are underlined.

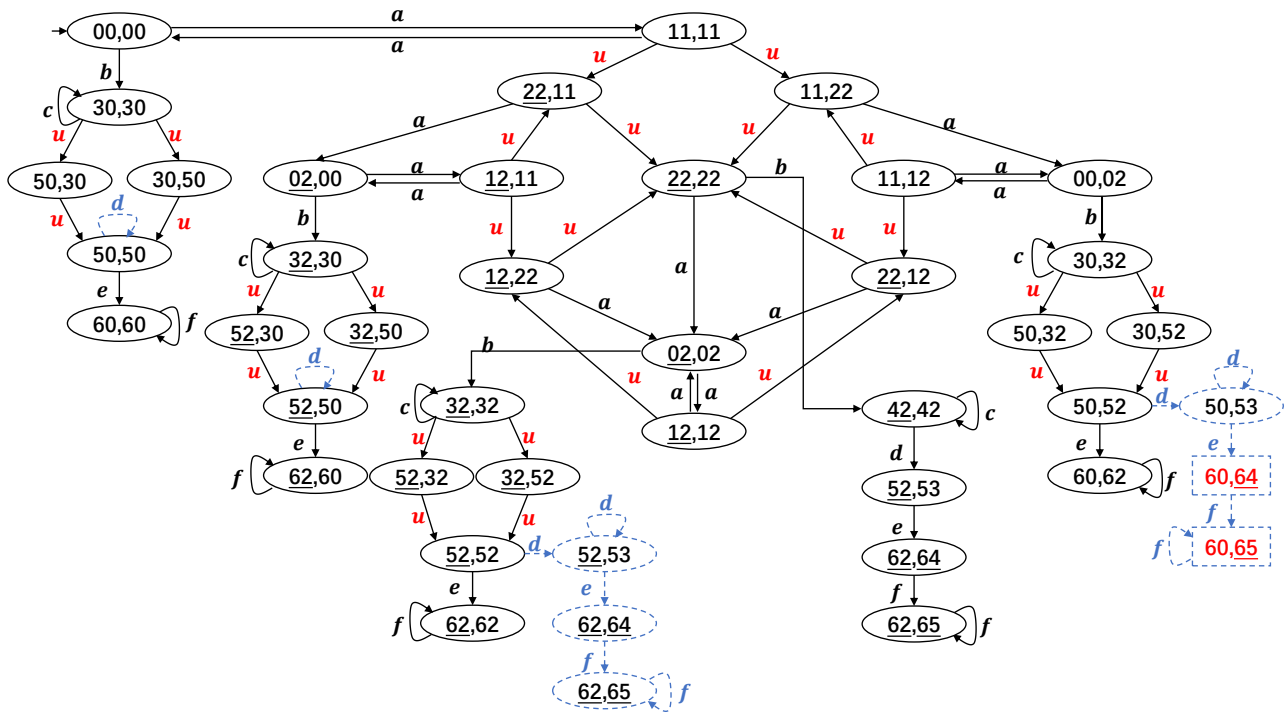

Fig. 10. The pattern-pattern verifier composed by pattern recognizers H_{fr} and $H_{cr,-1}$ in Figures 8 and 9.

One can readily verify that there is no PP-violating state in V_{pp}. According to Theorem 1, G is PP-diagnosable with respect to (K_f, K_c). In fact, after the fault pattern E^*auE^* occurs, to complete a critical sequence ending with def, the plant necessarily follows the trajectory 4-5-6-6 and executes d first when reaching state 5. By observing such an event d we immediately know that G must be going on the trajectory 2-4-5, which indicates that the fault pattern must have occurred.

Now, let us modify the plant in Figure 1 by adding the dashed blue event d at state 5. Still consider the fault and the critical patterns $K_f = L_m(H_f)$, $K_c = L_m(H_c)$ in Figure 3. In such a case, the corresponding pattern recognizers $H_{fr}, H_{cr}, H_{cr,-1}$ are those in Figures 8, 9 *with* the dashed transitions, and the corresponding PP-verifier V_{pp} that has 43 states is shown in Figure 10 *with* the blue dashed states and transitions. In such a case, in V_{pp} there exists a state $(q', q'') = (60, 64)$ (rectangular, on the right bottom) in which $q' = 60$ is not a final state in H_{fr} while $q'' = 64$ is a final state in H_{cr}. By Theorem 1, the modified plant G is not PP-diagnosable with respect to (K_f, K_c). In fact, when the plant executes $s = auabude$ which belongs to the faulty pattern, we observe $P(s) = aabde$ whose consistent sequences are:

- $auabude$, faulty, whose trajectory is 0-1-2-0-3-5-6;
- $aabude$, non-faulty, whose trajectory is 0-1-0-3-5-6.

Hence, at this moment the occurrence of a fault pattern cannot be determined. Then, by executing the next event f, sequence $auabudef$ is completed in which a critical sequence occurs after a fault sequence. Therefore, the modified G is not PP-diagnosable with respect to (K_f, K_c). ♢

At the end of this section we discuss the complexity of our approach. Suppose that plant G, fault pattern automaton H_f, critical pattern automaton H_c have n, n_f, n_c states, respectively. The augmentation from H_f to $\hat{H}_f$. Hence, $H_{fr} = H_f \times G$ has at most $n_f \cdot n$ states, and $H_{cr} = H_{fc} \times G = (\hat{H}_f \cdot H_{c,-1}) \times G$ has at most $(n_f + n_c) \cdot n$ states. The step-shift on H_{cr} do not add any state. Thus, the number of states in the PP-verifier $V_{pp} = H_{fr} ||_o H_{cr,-1}$ is bounded by $(n_f \cdot n) \cdot ((n_f + n_c) \cdot n)$ that is $n^2 \cdot (n_f^2 + n_f \cdot n_c)$. Therefore, the complexity of the proposed method is quadratic in the number of states of the plant G and the fault pattern automaton H_f, and is linear with the number of states of the critical pattern automaton H_c.

VI. Conclusions and Future Work

In this work we have proposed the notion of PP-diagnosability in partially observable discrete event systems and have studied its properties. We have shown that PP-diagnosability is preserved when the fault pattern is fixed while the critical pattern reduces. Then we have proposed a method to verify PP-diagnosability using a

structure called the pattern-pattern verifier. Precisely, we proved that a plant is PP-diagnosable if and only if its PP-verifier does not contain any PP-violating state. The complexity of our approach is polynomial in the number of states in the plant and the two pattern automata. In the future we will extend this work to PP-diagnosability on timed discrete event systems and to other models such as Petri nets.

REFERENCES

[1] S. Lafortune, F. Lin, and C. Hadjicostis, "On the history of diagnosability and opacity in discrete event systems," *Annual Reviews in Control*, vol. 45, pp. 257–266, 2018.

[2] J. Zaytoon and S. Lafortune, "Overview of fault diagnosis methods for discrete event systems," *Annual Reviews in Control*, vol. 37, no. 2, pp. 308–320, 2013.

[3] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete event systems," *IEEE Transactions on Automatic Control*, vol. 40, no. 9, pp. 1555–1575, 1995.

[4] F. Lin, "Diagnosability of discrete event systems and its applications," *Discrete Event Dynamic Systems*, vol. 4, no. 2, pp. 197–212, 1994.

[5] S. Jiang, Z. Huang, V. Chandra, and R. Kumar, "A polynomial algorithm for testing diagnosability of discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 46, no. 8, pp. 1318–1321, 2001.

[6] T. S. Yoo and S. Lafortune, "Polynomial-time verification of diagnosability of partially observed discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 47, no. 9, pp. 1491–1495, 2002.

[7] L. K. Carvalho, M. V. Moreira, J. C. Basilio, and S. Lafortune, "Robust diagnosis of discrete-event systems against permanent loss of observations," *Automatica*, vol. 49, no. 1, pp. 223–231, 2013.

[8] L. K. Carvalho, M. V. Moreira, and J. C. Basilio, "Diagnosability of intermittent sensor faults in discrete event systems," *Automatica*, vol. 79, no. C, pp. 315–325, 2017.

[9] S. Genc and S. Lafortune, "Diagnosis of patterns in partially-observed discrete-event systems," in *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006, pp. 422–427.

[10] F. Basile, P. Chiacchio, and G. D. Tommasi, "On K-diagnosability of Petri nets via integer linear programming," *Automatica*, vol. 48, no. 9, pp. 2047 – 2058, 2012.

[11] M. Cabasino, A. Giua, S. Lafortune, and C. Seatzu, "A new approach for diagnosability analysis of Petri nets using verifier nets," *IEEE Transactions on Automatic Control*, vol. 57, no. 12, pp. 3104–3117, 2012.

[12] E. Fabre, L. Hélouët, E. Lefaucheux, and H. Marchand, "Diagnosability of repairable faults," *Discrete Event Dynamic Systems*, vol. 28, no. 2, pp. 183–213, 2018.

[13] R. Kumar and S. Takai, "Decentralized prognosis of failures in discrete event systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 1, pp. 48–59, 2010.

[14] M. V. Moreira, T. C. Jesus, and J. C. Basilio, "Polynomial time verification of decentralized diagnosability of discrete event systems," *IEEE Transactions on Automatic Control*, vol. 56, no. 7, pp. 1679–1684, July 2011.

[15] X. Yin, J. Chen, Z. Li, and S. Li, "Robust fault diagnosis of stochastic discrete event systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 4237–4244, 2019.

[16] M. P. Cabasino, A. Giua, and C. Seatzu, "Diagnosability of discrete-event systems using labeled Petri nets," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 1, pp. 144–153, Jan 2014.

[17] X. Yin and S. Lafortune, "On the decidability and complexity of diagnosability for labeled Petri nets," *IEEE Transactions on Automatic Control*, vol. 62, no. 11, pp. 5931–5938, 2017.

[18] N. Ran, A. Giua, and C. Seatzu, "Enforcement of diagnosability in labeled Petri nets via optimal sensor selection," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2997–3004, July 2019.

[19] Z. Ma, X. Yin, and Z. Li, "Marking diagnosability verification in labeled Petri nets," *Automatica*, vol. 131, p. Article 109713, 2021.

[20] Y. Hu, Z. Ma, Z. Li, and A. Giua, "Diagnosability enforcement in labeled Petri nets using supervisory control," *Automatica*, vol. 131, p. 109776, 2021.

[21] T. Jéron, H. Marchand, S. Pinchinat, and M.-O. Cordier, "Supervision patterns in discrete event systems diagnosis," in *2006 8th International Workshop on Discrete Event Systems*. IEEE, 2006, pp. 262–268.

[22] Y. Pencolé and A. Subias, "Diagnosability of event patterns in safe labeled time Petri nets: a model-checking approach," *IEEE Transactions on Automation Science and Engineering*, 2021.

[23] D. Lefebvre and C. N. Hadjicostis, "Diagnosability of fault patterns with labeled stochastic Petri nets," *Information Sciences*, vol. 593, pp. 341–363, 2022.

[24] A. Paoli and S. Lafortune, "Safe diagnosability for fault-tolerant supervision of discrete-event systems," *Automatica*, vol. 41, no. 8, pp. 1335–1347, 2005.

[25] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer, 2021, 3rd edition.

[26] S. Genc and S. Lafortune, "Predictability of event occurrences in partially-observed discrete-event systems," *Automatica*, vol. 45, no. 2, pp. 301–311, 2009.