Fault Prognosis with Supervisory Control of Discrete-Event Systems under Non-deterministic Observations

Rui Zhao*, Fuchun Liu**

*School of Computers, Guangdong University of Technology, Guangzhou 510006 China (Tel: +8613160809171; e-mail: *zhaorui118204@163.com, **fliu2011@163.com).*

Abstract: Fault prediction seeks to predict the future occurrences of faults in order to implement effective control measures. In the context of non-deterministic observations, this article investigates the fault prognosis of discrete-event systems using supervisory control. The concept of active prognosability is formalized to characterize a discrete-event system's ability to predict and prevent a fault under non-deterministic observations. A transformed automaton is created from the initial system to test the property by converting non-deterministic observations into deterministic observations. Based on this, a verifier is built, and the necessary and sufficient conditions to ensure such property are deduced. Furthermore, an algorithm for verifying active prognosability is proposed, the complexity of which is polynomial in the number of system states and events. Also, the conditions for the controllable sub-language's existence are established, and the matching supervisor is synthesized, allowing the system to run faultlessly and properly. Our results expand on previous research with regard to safe controllability through diagnosis and prognosis.

Keywords: discrete-event systems, fault prognosis, non-deterministic observations.

1. INTRODUCTION

Complex technological systems are prone to faults that can result in unintended damage or a violation of safety standards. For safety-critical systems, fault diagnosis and prognosis are required to avoid dangerous or unsafe situations.The research of fault diagnosis of discrete-event systems (DESs) has been widely concerned in the last 20 years, and many results have been obtained; see these literature (Sampath, 1995; Paoli, 2005; Liu, 2008, 2015, 2022; Moreira, 2011; Keroglou, 2018; Yin, 2019; Carvalho, 2021;Viana, 2019) for details. A. Paoli and S. Lafortune, in particular, introduced the concept of safe diagnosability (Paoli, 2005) to describe the situation in which fault detection must be performed prior to the execution of a given set of forbidden strings after a fault. Many subsequent works (Liu 2008, 2015, 2022) investigated this problem in greater depth using various discrete-event system (DES) frameworks.

In terms of system safety, fault diagnosis can only take preventative measures after a fault has occurred. The goal of fault prognosis is to investigate methodologies for predicting future occurrences of potential system faults. As a result, fault prediction is more appealing in actual large-scale complex industrial systems. The research problem has received a lot of attention in the field of DESs in recent years. S. Genc and S. Lafortune proposed the concept of fault prediction (Genc, 2009) and provided two algorithms based on diagnosers and verifiers to test the system's predictability. Fault prediction under other formalization models of DESs, such as fuzzy framework (Benmessahel, 2017) and stochastic framework (Chang, 2013; Cao, 2021), was also investigated as a result of the research. Given that sensor information in more complex systems may be dispersed across multiple sites, decentralized or distributed fault prognosis under various architectures was investigated (Takai, 2012; Yin, 2016b, 2019). Some local predictors may fail due to sensor failure or communication failure in the network configuration. As a result, dependability is an unavoidable topic. Some formalized notions and verification algorithms for reliable prognosis have been proposed (Yin, 2016a; Liu, 2018; Liao, 2022) .

Previous fault prognosis research methods assume that event observation is partial and deterministic. That is, if an event sequence occurs in the system, only a subset of the events can be observed, and the observed results are fixed. In real-world systems, however, event observation may be uncertain due to sensor failure, packet loss, or malicious attacks. In this regard, the authors of literature (Carvalho, 2012; Tomola, 2017; Yin, 2019; Xiao, 2022) proposed the concept of robustness to describe the situation in which some events may lose observation intermittently or permanently and provided some useful robust diagnosis and prognosis mechanisms.

This paper considers the fault prognosis of DESs in the context of non-deterministic observations. It is worth noting that indeterminacy is a more general uncertainty that includes temporary loss (Oliveira, 2022), intermittent loss, and permanent loss of sensor observation. Furthermore, it allows the observation symbols to differ from the original event set (Xu, 2009). In a broader sense, this uncertainty is more valuable in practical applications. As a result, many important research projects have been conducted on it, with a primary focus on control and supervisor synthesis. S. Xu and R. Kumar investigated discrete event control under non-deterministic partial observation and proposed a method to find a supervisor to enforce a given specification (Xu, 2009); T. Ushio and S. Takai designed an anti-permissive supervisor to control the DES modeled by a mealy automaton with a

non-deterministic output function (Ushio, 2016); and L. Zhou and S. Shu proposed two methods for determining the O-observability of DESs under non-deterministic observations (Zhou, 2019, 2020).

Motivated by the studies listed above, this article examines the problem of fault prognosis in the case of non-deterministic observations and presents supervisory control as a solution. This is because, in general, once a fault is detected in advance, the following step is to activate the control laws to prevent it from occurring, i.e. to ensure pre-specified safety objectives. The fault detection filter's output will be utilized to construct a fault control system. The article initially determines if the existence of a probable fault may be predicted in the future. Consider utilizing the supervisory control method to prevent it from happening if it is predictable. Some new concepts and strategies are presented to accomplish this.

This article made the following contributions. First, under non-deterministic observations, the concept of active prognosability of DESs is redefined. Second, the necessary and sufficient conditions for testing the property are derived using a verifier. Finally, a supervisor is synthesized using the system's predictive information to adjust the system's behavior and keep it away from faults.

The findings of this paper involve work-related studies. (Zhao, 2021; Paoli, 2011; Watanabe, 2017, 2021). The concept of active predictability of DESs was first presented in the work (Zhao, 2021), and a verification algorithm is provided. However, the supervisor's design is not mentioned in the paper, and the definition of the concept is inaccurate. This article redefines the notion and develops a safety controller to keep the system from behaving wrong. The topic of fault tolerant control based on online defect diagnosis or prognosis has been discussed in the literature (Paoli, 2011; Watanabe, 2017, 2021). A. Paoli et al. used the diagnostic controller to meet the requirements and proposed the concept of safe controllability (Paoli, 2011) to describe the ability to direct the system away from prohibited zones following a fault. Ana T. Y. Watanabe et al. introduced a new concept of safe controllability by prognosis in the research (Watanabe, 2017), and designed a prognosis controller to predict faults and dynamically switch supervisors to steer system behavior in time. They then combined safe controllability by prognosis and diagnosis to create the new concept of DP-safe controllability (Watanabe, 2021) and provided the necessary and sufficient conditions to ensure the property. All of these studies attempt to combine supervisory control with fault diagnosis or prediction in order to create appropriate controllers to prevent the execution of illegal strings following faults. The issue discussed in this paper is derived from the literature (Watanabe, 2021) and can be viewed as a subset of the problem. However, there are two distinctions. The first distinction is that the article discusses this problem in the context of uncertain observation, whereas the research results in the preceding literature are in the context of deterministic observation; the second distinction is that the goal in incorporating supervisory control into fault prognosis is to control the system's behavior away from faults rather than illegal strings following faults.

The rest of the paper is structured as follows. Section 2 provides the necessary background information on DESs. Section 3 formalizes the concept of active prognosability. Section 4 describes an efficient method for verifying this property. Section 5 examines the topic of supervisory control based on fault prognosis information, and the concluding section presents a conclusion.

2. PRELIMINARIES

A DES can be described by an automaton, denoted as G.

$$G = (X_g, \Sigma, f_g, \Gamma_g, x_0, X_m), \tag{1}$$

where X_g is the finite state space, Σ is the set of events, $f_g : X_g \times \Sigma \to X_g$ is the transition function, $\Gamma_g : X_g \to 2^{\Sigma}$ is the active event function, $x_0 \in X$ is the initial state, and $X_m \subseteq X_g$ is the set of marked states. Σ^* denotes the set of all strings formed by events in Σ, including the empty string ε. The transition function f_g is also extended to $f_g : X_g \times \Sigma^* \to X_g$ in the usual way: for any $x \in X, s \in \Sigma^*, and \ \sigma \in \Sigma$, $f_g(x, \varepsilon) = x$ and $f_g(x, s\sigma) = f_g(f_g(x, s), \sigma)$.

Control restrictions are reflected by partitioning as $\Sigma = \Sigma_c \bigcup \Sigma_{uc}$. Σ_c is the set of controllable events; these are the events that can be prevented from happening by a supervisor. Σ_{uc} is the set of uncontrollable events; these events cannot be prevented from happening by a supervisor. Let $\Sigma_f = \{\sigma_f\} \subseteq \Sigma$ denote the collection of fault events to be predicted. Fault events are generally regarded as uncontrollable occurrences, i.e., $\Sigma_f \subseteq \Sigma_{uc}$. ϕ indicates an empty set.

The language generated by G represents the behavior of the system and it is defined as $L(G) = \{s \in \Sigma^* : f_g(x_0, s)!\}$, where ! means "is defined". Without ambiguity, it is often written as L. Here, it is assumed that $\varepsilon \in L$.

Given a string s originating from x_0, the length of s (number of events including repetitions) is denoted by $\|s\|$. Denote L/s as the post-language of L after s, i.e., $L/s = \{t \in \Sigma^* : st \in L\}$, and denote $\overline{s}$ as the set of all prefixes of s. The strict prefix of s is denoted by $\hat{s} = \overline{s} - \{s\}$. Let E_1 and E_2 be the sets of strings or events, $E_1 - E_2$ means to remove all strings or events in E_2 from E_1.

For the language L, $\overline{L}$ is its prefix closure, which is defined as $\overline{L} = \{t \in \overline{s} : s \in L\}$. If for all $s \in L$, there is an event $\sigma \in \Sigma$ such that $s\sigma \in L$, L is said to be live. In this work, L is assumed to be live.

The string containing fault events is referred to as the fault string in the language L, while the string containing no fault events is referred to as the normal string.

$L_f = L \bigcap \Sigma^* \Sigma_f$ represents a set of strings ending with a fault event. $L_{\sim f} = L \bigcap (\Sigma - \Sigma_f)^*$ denotes the collection of all normal strings. It is noted that $L_{\sim f}$ can be divided into two subsets, namely, $L_{\sim f} = L^{'}_{\sim f} \bigcup L^{''}_{\sim f}$, where

$$L^{'}_{\sim f} := \{s \in L_{\sim f} : (\exists t \in L/s)[\sigma_f \in t]\} \quad (2)$$

and

$$L^{''}_{\sim f} := \{s \in L_{\sim f} : (\forall t \in L/s)[\sigma_f \notin t]\}. \quad (3)$$

Intuitively, fault events are present in the post-language of the strings in $L^{'}_{\sim f}$ but are absent from the post-language of the strings in $L^{''}_{\sim f}$. Therefore, $L^{'}_{\sim f}$ can be thought of as a description of the system's temporary normal behavior before the system fault happens, whereas $L^{''}_{\sim f}$ is an expression of the system's permanent normal behavior.

Let L_1 be sub-language of L and $L_2 \subseteq \Sigma^*$, the quotient operation for L_1 and L_2 is defined as

$$L_1 \setminus L_2 := \{s \in \Sigma^* : (\exists t \in L_2)[st \in L_1]\}. \quad (4)$$

A sequence of transitions in G denoted by $x_1 \xrightarrow{\sigma_1} x_2 \xrightarrow{\sigma_2} \ldots x_{n-1} \xrightarrow{\sigma_{n-1}} x_n$, where $\sigma_i \in \Sigma$ and $x_{i+1} = f_g(x_i, \sigma_i)$ for all $i = 1, 2, ..., n-1$, is called a path. When $x_1 = x_n$, the path forms cycle.

The co-accessible part of G is denoted by $Coac(G) := (X_{coac}, \Sigma, f_{coac}, \Gamma_{coac}, x_{0,coac})$, where $X_{coac} = \{x \in X_g : (\exists s \in \Sigma^*)[f_g(x,s) \in X_m]\}$, $f_{coac} = f_g|_{X_{coac} \times \Sigma \to X_{coac}}$, $\Gamma_{coac} = \Gamma_g|_{X_{coac} \to 2^{\Sigma}}$, and $x_{0,coac} = x_0$ if $x_0 \in X_{coac}$ or undefined if $x_0 \notin X_{coac}$. Let G_1 and G_2 be two automata. The product and parallel composition of automata G_1 and G_2 are denoted as $G = G_1 \times G_2$ and $G = G_1 \| G_2$ (Cassandras, 2008), respectively.

Sensor errors, packet losses, or other unforeseen damages frequently result in non-deterministic observations in real DESs. A brand-new observation mapping (Zhou, 2019) is presented to describe the situation. Let Δ represent a set of all possible observations or output symbols. When there is no output signal or the observation result cannot be obtained, it is represented by ε. Or to put it another way, $\varepsilon \in \Delta$. As a result, Δ in this case is typically a non-empty collection. Define the observation mapping as

$$M : X_g \times \Sigma \to 2^{\Delta}. \quad (5)$$

The mapping can be extended as usual way to $M : X_g \times \Sigma^* \to 2^{\Delta^*}$.

$M(x_0, s)$ is abbreviated as $M(s)$ for convenience. If a string $s = s'\sigma \in L, M(s) = M(s')M(f_g(x_0, s'), \sigma)$.

The inverse projection of M is $M^{-1}(y) = \{s \in L : y \in M(s)\}$.

To avoid unnecessary complexity, this article assumes that there is no loop with the observation label ε in G.

3. ACTIVE PROGNOSABILITY

In this part, the concept of active prognosability within the context of non-deterministic observations will be introduced.

Let's first recapitulate the idea of a safe controllable string by prognosis, which was first presented in the work (Watanabe, 2021) by Ana T. Y. Watanabe et al.

Definition 1 (Watanabe, 2021): Consider a diagnosable language (Sampath, 1995) L. The occurrence of event σ_f in a string $s \in L_f$ is safe controllable by prognosis w.r.t. P_o (a general projection for filtering out unobservable events) and Φ (an illegal language subset) if the following conditions hold:

(C1): the occurrence of event σ_f in s is prognosable (Genc, 2009) w.r.t. P_o;

(C2): consider $s = tu$ and $t = r\sigma$, with $r \in \Sigma^*$ and $\sigma \in \Sigma_o$ (Σ_o represents a set of observable events), and such that the occurrence of event σ_f in s is prognosable w.r.t. P_o and t, but unpredictable w.r.t. P_o and r. Then, $\forall \omega \in L/t$ such that $\omega = uv\zeta$, with $\zeta \in \Phi$, $\exists \sigma_c \in \Sigma_c$ such that $\sigma_c \in \omega$.

Intuitively, a string s that ends with a fault σ_f is safe controllable by prognosis, which means that the fault event σ_f in the string s is prognosable and there is a controllable event σ_c to prevent the execution of any illegal event sequence following the fault.

It is worth noting that the controllable event σ_c can occur before, after, or in a prefix of the illegal substring, as considered in the literature (Watanabe, 2021). The authors of (Watanabe, 2021) specifically stated that if the controllable event σ_c occurs before event σ_f (in substring u), as shown in Fig.1, then the fault can be avoided by disabling the controllable event. They did not, however, specifically address this issue within the framework of non-deterministic observations. This paper focuses solely on the issue.

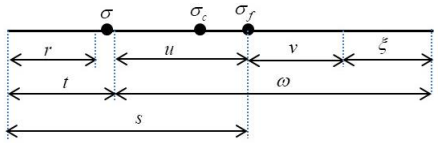

Fig. 1. Controllable event σ_c in s.

To address the problem, the concept of active prognosability will be proposed.

Definition 2 : Consider the language L generated by G. The occurrence of event σ_f in a string $s \in L_f$ is prognosable and preventable w.r.t. M and Σ_c if

$$(\exists t, u \in \Sigma^*)(\exists \sigma_c \in \Sigma_c)(tu \in \hat{s} \wedge \|u\| \geq 1 \wedge \sigma_c \in u)[\mathbb{R}(t, u, \sigma_c) = 1],$$

where the function $\mathbb{R}(t, u, \sigma_c)$ is defined as follows:

$$\mathbb{R}(t,u,\sigma_c)=\begin{cases}1, if\ (\forall t'u'\in L_{\sim f})(M(t')\bigcap M(t)\neq\phi)\wedge \\ \|u'\|\geq 1\wedge\sigma_c\in u')\Rightarrow(t'u'\in L_{\sim f}^{'}), \\ 0,\ otherwise.\end{cases} \quad (6)$$

The occurrence of an error event in a string s is predictable and preventable, which means that it is possible to infer the occurrence of the fault based on the non-faulty prefix t of s and the suffix of the t or other strings with the same observations as t contains a controllable event that can be posed to the system to prevent the fault from occurring.

Definition 3 : Assume L is the language produced by G. If the occurrence of event σ_f is prognosable and preventable w.r.t. M and Σ_c for each string $s\in L_f$, then L is said to be actively prognosable w.r.t. M, Σ_f, and Σ_c.

Roughly speaking, language L is considered actively prognosable when the occurrence of fault events in each string is predictable and preventable.

The following example is provided to demonstrate the concept.

Example 1: Consider the DES represented by the automaton G_1 shown in Fig.2, with the initial state $x_0=1$, the event set $\Sigma=\{\alpha,\beta,\gamma,\sigma_f\}$, and $\Sigma_c=\{\alpha\}$. σ_f represents a fault event, i.e. $\Sigma_f=\{\sigma_f\}$. σ/E is the label of an event, where $\sigma\in\Sigma$ represents an event and E denotes possible observations of the event. In this case, $\Delta=\{e_1,e_2,\varepsilon\}$ and $E\subseteq\Delta$.

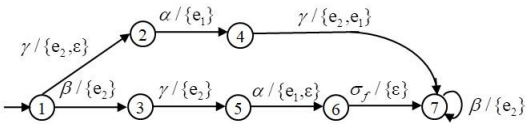

Fig. 2. Automaton G_1 of Example 1.

$L_f=\{\beta\gamma\alpha\sigma_f\}$, as shown in Fig.2. Let $s=\beta\gamma\alpha\sigma_f$, Choose $t=\beta\gamma$ and $u=\alpha$, then $tu\in\hat{s}$, $\|u\|\geq 1$, and $\alpha\in u$. $M(t)=M(1,\beta\gamma)=\{e_2\}\{e_2\}=\{e_2e_2\}$. It is worth noting that in the system if there is $t'u'\in L_{\sim f}$ and $M(t')\bigcap M(t)\neq\phi$, then $t'=t$. In this case, if $u'=u$, it is obvious that $\|u'\|\geq 1$ and $\alpha\in u'$. In terms of $t'u'$, there exists a $\omega\in L/t'u'$ such that $\sigma_f\in\omega$. As a result, $t'u'\in L_{\sim f}^{'}$. According to Definition 2, the occurrence of event σ_f in s is prognosable and preventable w.r.t. M and Σ_c. According to Definition 3, L is actively prognosable with respect to M, Σ_f, and Σ_c.

4. VERIFICATION OF ACTIVE PROGNOSABILITY

In this section, a method will be presented for verifying the active prognosability of L. The approach is divided into four steps. First, a transformed automaton $\tilde{G}$ is built to convert non-deterministic observations into deterministic observations. Second, automata $\tilde{G}_P$ and $\tilde{G}_f$ are constructed to describe $\tilde{G}_p$'s permanent normal and fault behavior, respectively. Third, based on $\tilde{G}_P$ and $\tilde{G}_f$, a testing automaton $V_{\tilde{G}}$ is established, and the necessary and sufficient conditions for active prognosability are derived. Finally, a verification algorithm is provided.

4.1 Transformed automaton $\tilde{G}$

Let G be an automaton with M describing non-deterministic observations. Its transformed automaton, also known as the dilated automaton, is defined as

$$\tilde{G}=(X_g,\tilde{\Sigma},\tilde{f}_g,\tilde{\Gamma}_g,x_0,X_m), \quad (7)$$

where X_g, x_0, and X_m are the same as those of G. The event set is extended from Σ by adding an observation label to each event, which is defined as

$$\tilde{\Sigma}=\{(\sigma,e)\in\Sigma\times\Delta:(\exists x\in X_g)[f_g(x,\sigma)!\wedge e\in M(x,\sigma)]\}. \quad (8)$$

It is worth noting that $\tilde{\Sigma}_f=\{(\sigma_f,\varepsilon)\}$, $\tilde{\Sigma}_c=\{(\sigma_c,e)\in\tilde{\Sigma}:\sigma_c\in\Sigma_c\}$, and $\tilde{\Sigma}_f,\tilde{\Sigma}_c\subseteq\tilde{\Sigma}$.

For any $x\in X_g$ and $(\sigma,e)\in\tilde{\Sigma}$, the transition function $\tilde{f}_g$ is defined as

$$\tilde{f}_g(x,(\sigma,e))=\begin{cases}f_g(x,\sigma),\ if\ \ f_g(x,\sigma)!\wedge e\in M(x,\sigma)\\ undefined,\ otherwise.\end{cases} \quad (9)$$

The active event function Γ_g is also extended to $\tilde{\Gamma}_g$: $x_g\rightarrow 2^{\tilde{\Sigma}}$. $\tilde{L}$ represents the language generated by $\tilde{G}$ and it is assumed that $(\varepsilon,\varepsilon)=\varepsilon\in\tilde{L}$.

For $\tilde{G}$, two kinds of mappings (Zhou, 2020), namely, observation mapping and event mapping need to be defined, which are denoted by $\tilde{P}_\Delta$ and $\tilde{P}_\Sigma$, respectively. Let $\tilde{s}\in\tilde{\Sigma}^*$ and $(\sigma,e)\in\tilde{\Sigma}$.

Observation mapping $\tilde{P}_\Delta:\tilde{\Sigma}^*\rightarrow\Delta^*$ is defined as

$$\tilde{P}_\Delta(\varepsilon)=\varepsilon,\tilde{P}_\Delta(\tilde{s}(\sigma,e))=\begin{cases}\tilde{P}_\Delta(\tilde{s})e,\ if\ \ e\in\Delta-\{\varepsilon\}\\ \tilde{P}_\Delta(\tilde{s}),\ if\ e=\varepsilon.\end{cases} \quad (10)$$

Event mapping $\tilde{P}_\Sigma:\tilde{\Sigma}^*\rightarrow\Sigma^*$ can be recursively defined as

$$\tilde{P}_\Sigma(\varepsilon)=\varepsilon,\qquad \tilde{P}_\Sigma(\tilde{s}(\sigma,e))=\tilde{P}_\Sigma(\tilde{s})\sigma.$$

The inverse projection of $\tilde{P}_\Sigma$ is

$$\tilde{P}_\Sigma^{-1}(s)=\{\tilde{s}\in\tilde{L}:s=\tilde{P}_\Sigma(\tilde{s})\}. \quad (11)$$

The set of fault strings of $\tilde{L}$ is defined as

$$\tilde{L}_f=\{\tilde{s}\in\tilde{L}:(\exists s\in L_f)[\tilde{P}_\Sigma(\tilde{s})=s]\}. \quad (12)$$

Remark 1: Assume that $\tilde{s}\in\tilde{L}$, $s\in L$, and $\tilde{P}_\Sigma(\tilde{s})=s$. If $\|s\|=n$, then $\|\tilde{s}\|=n$. This is clear since $\tilde{s}$ can only be formed

by assigning observation labels to events in s.

Certain lemmas must be provided first before presenting the subsequent theorem 1.

Lemma 1: Let L be the language generated by G and $\tilde{L}$ be the language generated by $\tilde{G}$. The result is $\tilde{P}_\Sigma(\tilde{L}) = L$.

Proof: The proof is by induction on the length of the strings in the two languages.

1) The base case is for strings of length 0. It should be noted that $\varepsilon \in L$ and $\varepsilon \in \tilde{L}$ are assumed. Furthermore, according to the preceding definition, $\tilde{P}_\Sigma(\varepsilon) = \varepsilon$. As a result, the base case is correct.

2) The induction hypothesis is that $\forall \tilde{s}: \tilde{s} \in \tilde{L}, \|\tilde{s}\| \le n \Rightarrow \exists s \in L, \|s\| \le n : P_\Sigma(\tilde{s}) = s;\ \forall s: s \in L, \|s\| \le n \Rightarrow \exists \tilde{s} \in \tilde{L}, \|\tilde{s}\| \le n : \tilde{P}_\Sigma(\tilde{s}) = s$.

3) The same is proved for strings of the form $\tilde{s}\tilde{\sigma}$.

a) Let $\tilde{s}\tilde{\sigma} \in \tilde{L}$. Since $\tilde{\sigma} \in \tilde{\Sigma}$, according to the induction hypothesis, there exists a $\sigma \in \Sigma$ such that $\sigma = \tilde{P}_\Sigma(\tilde{\sigma})$. For $\tilde{s} \in \tilde{L}$, similarly, there exists a $s \in L$ such that $s = \tilde{P}_\Sigma(\tilde{s})$. By the construction process of $\tilde{G}$, there is $s\sigma \in L$. As a consequence, $\forall \tilde{s}\tilde{\sigma} \in \tilde{L}$, there exists $s\sigma \in L$ such that $\tilde{P}_\Sigma(\tilde{s}\tilde{\sigma}) = \tilde{P}_\Sigma(\tilde{s})\tilde{P}_\Sigma(\tilde{\sigma}) = s\sigma$.

b) Assume $s\sigma \in L$. According to the induction hypothesis, there exists a $\tilde{s} \in \tilde{L}$ such that $\tilde{P}_\Sigma(\tilde{s}) = s$, and there exists a $\tilde{\sigma} \in \tilde{\Sigma}$ such that $\tilde{P}_\Sigma(\tilde{\sigma}) = \sigma$. It is clear from the definition of $\tilde{G}$ that $\forall \tilde{s}\tilde{\sigma} \in \tilde{L}$. Thus, for $s\sigma \in L$, there exists $\forall \tilde{s}\tilde{\sigma} \in \tilde{L}$ such that $\tilde{P}_\Sigma(\tilde{s}\tilde{\sigma}) = \tilde{P}_\Sigma(\tilde{s})\tilde{P}_\Sigma(\tilde{\sigma}) = s\sigma$.

This completes the proof of the induction steps.

Lemma 1 states that the language of the transformed automaton can be mapped to the language of the original automaton.

Lemma 2: Let s be a string of L and $\tilde{s}$ be a string of $\tilde{L}$. If $\tilde{P}_\Sigma(\tilde{s}) = s$, then $\tilde{P}_\Delta(\tilde{s}) = M(s)$.

Proof: Let string $s = \sigma_1\sigma_2...\sigma_n \in L$ and $\Delta_i = \{e_1^i, e_2^i, ..., e_{k_i}^i\}$ be the set of all possible observations of σ_i for all $i = 1, 2, ..., n$. To form $\tilde{s}$, choose each event in s and one of its possible observations. Assuming $\tilde{s} = (\sigma_1, e_1^1)(\sigma_2, e_1^2)...(\sigma_n, e_1^n)$, then $\tilde{s} \in \tilde{L}$ and $\tilde{P}_\Sigma(\tilde{s}) = s$. Furthermore, in this case, $\tilde{P}_\Delta(\tilde{s}) = e_1^1 e_1^2 ... e_1^n$, whereas $M(s) = \{e_1^1, e_2^1, ... e_{k_1}^1\}\{e_1^2, e_2^2, ... e_{k_2}^2\}... \{e_1^n, e_2^n, ... e_{k_n}^n\} = \{e_1^1 e_1^2 ... e_1^n, e_2^1 e_2^2 ... e_2^n, ...\}$. Clearly, $\tilde{P}_\Delta(\tilde{s}) = M(s)$.

Lemma 2 indicates that the string in L and the string in $\tilde{L}$ can be transformed to each other via mapping.

Lemma 3: Let L and $\tilde{L}$ be languages generated by G and $\tilde{G}$, respectively. The following conclusions follow.

1) $\forall t', t \in L$ and $M(t') \cap M(t) \ne \phi \Rightarrow \exists \tilde{t}', \tilde{t} \in \tilde{L}$ such that $\tilde{P}_\Sigma(\tilde{t}') = t', \tilde{P}_\Sigma(\tilde{t}) = t$ and $\tilde{P}_\Delta(\tilde{t}') = \tilde{P}_\Delta(\tilde{t})$.

2) $\forall \tilde{t}', \tilde{t} \in \tilde{L}$ and $\tilde{P}_\Delta(\tilde{t}') = \tilde{P}_\Delta(\tilde{t}) \Rightarrow \exists t', t \in L$ such that $\tilde{P}_\Sigma(\tilde{t}') = t', \tilde{P}_\Sigma(\tilde{t}) = t$ and $M(t') \cap M(t) \ne \phi$.

Proof:

1) Since $t', t \in L$ and $M(t') \cap M(t) \ne \phi$, let $l \in M(t') \cap M(t)$. Lemma 1 states that there exist $\tilde{t}', \tilde{t} \in \tilde{L}$ such that $\tilde{P}_\Sigma(\tilde{t}') = t' \wedge \tilde{P}_\Delta(\tilde{t}') = l$ and $\tilde{P}_\Sigma(\tilde{t}) = \tilde{t} \wedge \tilde{P}_\Delta(\tilde{t}) = l$. As a result, $\tilde{P}_\Delta(\tilde{t}') = \tilde{P}_\Delta(\tilde{t})$.

2) According to Lemma 1, for $\forall \tilde{t}', \tilde{t} \in \tilde{L}$, there exist $t', t \in L$ such that $\tilde{P}_\Sigma(\tilde{t}') = t'$ and $\tilde{P}_\Sigma(\tilde{t}) = t$. Let $\tilde{P}_\Delta(\tilde{t}') = \tilde{P}_\Delta(\tilde{t}) = l$. From Lemma 2, it can be inferred that $\tilde{P}_\Delta(\tilde{t}') \in M(t')$ and $\tilde{P}_\Delta(\tilde{t}) \in M(t)$. That is, $l \in M(t')$ and $l \in M(t)$. As a result, $M(t') \cap M(t) \ne \phi$ holds.

Lemma 4: Assume that L_f and $\tilde{L}_f$ are the corresponding sets of fault-ending strings in L and $\tilde{L}$. Hence, the findings listed below are true.

1) $\forall s \in L_f \Rightarrow \forall \tilde{s} \in \tilde{P}_\Sigma^{-1}(s), \tilde{s} \in \tilde{L}_f$

1) $\forall \tilde{s} \in \tilde{L}_f \Rightarrow s = \tilde{P}_\Sigma(\tilde{s}), s \in L_f$.

Proof:

1) For $\forall s \in L_f$, there exists $\tilde{s} \in \tilde{L}$ such that $\tilde{P}_\Sigma(\tilde{s}) = s$ by Lemma 1. As $\tilde{P}_\Sigma(\tilde{s}) = s$, it is obvious that $\tilde{s} \in \tilde{L}_f$ in accordance with the definition of $\tilde{L}_f$.

2) For any $\tilde{s} \in \tilde{L}_f$, assume that $\tilde{s} = \tilde{t}(\sigma_f, \varepsilon)$, where $\tilde{t} \in \tilde{\Sigma}^*$. Following that, $\tilde{P}_\Sigma(\tilde{s}) = \tilde{P}_\Sigma(\tilde{t}(\sigma_f, \varepsilon)) = \tilde{P}_\Sigma(\tilde{t})\sigma_f$. In other words, $s = \tilde{P}_\Sigma(\tilde{s}) = \tilde{P}_\Sigma(\tilde{t})\sigma_f$. As a result, $s \in L_f$ is true.

Theorem 1: Let L and $\tilde{L}$ be languages generated by G and $\tilde{G}$, respectively. L is actively prognosable w.r.t. M, Σ_f, and Σ_c, if and only if $\tilde{L}$ is actively prognosable w.r.t. $\tilde{P}_\Delta$, $\tilde{\Sigma}_f$, and $\tilde{\Sigma}_c$.

Proof: The proof is divided into two parts.

($\Rightarrow$) Since L is actively prognosable w.r.t. M, Σ_f, and Σ_c, for each $s \in L_f$, there exists $tu \in \hat{s}$ with $\|u\| \ge 1$ and $\sigma_c \in u$ such that for any $t'u' \in L_{\sim f}$ with $M(t') \cap M(t) \ne \phi$, $\|u\| \ge 1$, and $\sigma_c \in u'$, there is a $\omega \in L/t'u'$ and $\sigma_f \in \omega$. According to

lemma 4, there is a $\tilde{s} = \tilde{P}_{\Sigma}^{-1}(s)$ and $\tilde{s} \in \tilde{L}_f$ for every $s \in L_f$. For $t,u,t'u' \in \Sigma^*$, according to Lemma 1, there are matching strings $\tilde{t},\tilde{u},\tilde{t}', \tilde{u}' \in \tilde{\Sigma}^*$, such that $\tilde{P}_{\Sigma}(\tilde{t}) = t$, $\tilde{P}_{\Sigma}(\tilde{u}) = u$, $\tilde{P}_{\Sigma}(\tilde{t}') = t'$, and $\tilde{P}_{\Sigma}(\tilde{u}') = u'$. Since $M(t') \cap M(t) \neq \phi$, Lemma 3 states that there exist $\tilde{t},\tilde{t}' \in \tilde{L}$ such that $\tilde{P}_{\Delta}(\tilde{t}') = \tilde{P}_{\Delta}(\tilde{t})$. Thus, for the $\tilde{s} \in \tilde{L}_f$, there exists $\tilde{t}\tilde{u} \in \hat{s}$ with $\|\tilde{u}\| \geq 1$ and $(\sigma_c, e) \in \tilde{u}$ such that for all $\tilde{t}'\tilde{u}' \in \tilde{L}_{\sim f}$ with $\tilde{P}_{\Delta}(\tilde{t}') = \tilde{P}_{\Delta}(\tilde{t})$, $\|\tilde{u}'\| \geq 1$ and $(\sigma_c, e) \in \tilde{u}'$, there is a $\tilde{\omega} \in \tilde{L}/\tilde{t}\tilde{u}$ and $(\sigma_f, \varepsilon) \in \tilde{\omega}$. Hence, $\tilde{L}$ is actively prognosable w.r.t. $\tilde{P}_{\Delta}$, $\tilde{\Sigma}_f$, and $\tilde{\Sigma}_c$.

($\Leftarrow$) Similarly, if $\tilde{L}$ is actively prognosable w.r.t. $\tilde{P}_{\Delta}$, $\tilde{\Sigma}_f$, and $\tilde{\Sigma}_c$, then for each $\tilde{s} \in \tilde{L}_f$, there exists $\tilde{t}\tilde{u} \in \hat{s}$ with $\|\tilde{u}\| \geq 1$ and $(\sigma_c, e) \in \tilde{u}$ such that for all $\tilde{t}'\tilde{u}' \in \tilde{L}_{\sim f}$ with $\tilde{P}_{\Delta}(\tilde{t}') = \tilde{P}_{\Delta}(\tilde{t})$, $\|\tilde{u}'\| \geq 1$, and $(\sigma_c, e) \in \tilde{u}'$, there is a $\tilde{\omega} \in \tilde{L}/\tilde{t}'\tilde{u}'$ and $(\sigma_f, \varepsilon) \in \tilde{\omega}$. Lemma 4 states that there is a $s = \tilde{P}_{\Sigma}(\tilde{s})$ and $s \in L_f$ for each $\tilde{s} \in \tilde{L}_f$. According to Lemma 1, for $\tilde{t},\tilde{u},\tilde{t}',\tilde{u}' \in \tilde{\Sigma}^*$, there are matching strings $t,u,t'u' \in \Sigma^*$ such that $t = \tilde{P}_{\Sigma}(\tilde{t})$, $u = \tilde{P}_{\Sigma}(\tilde{u})$, $t' = \tilde{P}_{\Sigma}(\tilde{t}')$, and $u' = \tilde{P}_{\Sigma}(\tilde{u}')$. Given that $t = \tilde{P}_{\Sigma}(\tilde{t})$ and $t' = \tilde{P}_{\Sigma}(\tilde{t}')$, from Theorem 2, it can be inferred that $\tilde{P}_{\Delta}(\tilde{t}) \in M(t)$ and $\tilde{P}_{\Delta}(\tilde{t}') \in M(t')$. As it is known that $\tilde{P}_{\Delta}(\tilde{t}') = \tilde{P}_{\Delta}(\tilde{t})$. Hence, $M(t') \cap M(t) \neq \phi$. It follows that for the $s \in L_f$, there exists $tu \in \hat{s}$ with $\|u\| \geq 1$ and $\sigma_c \in u$ such that for all $t'u' \in L_{\sim f}$ with $M(t') \cap M(t) \neq \phi$, $\|u'\| \geq 1$, and $\sigma_c \in u'$, there is a $\omega \in L/t'u'$ and $\sigma_f \in \omega$. As a result, L is actively prognosable w.r.t. M, Σ_f, and Σ_c.

Theorem 1 states that the active prognosability of L is equivalent to the property of $\tilde{L}$. As a consequence, the verification $L's$ active prognosability can be transformed into the verification of $\tilde{L}'s$ corresponding attributes.

The following proposition is easily obtained using Theorem 1.

Proposition 1: $\tilde{L}$ is actively prognosable w.r.t. $\tilde{P}_{\Delta}$, $\tilde{\Sigma}_f$, and $\tilde{\Sigma}_c$ iff the occurrence of the event (σ_f, ε) in each string $\tilde{s} \in \tilde{L}_f$ is prognosable and preventable w.r.t. $\tilde{P}_{\Delta}$ and $\tilde{\Sigma}_c$.

Proof: Theorem 1, Definition 2, and Definition 3 all lead us to this conclusion with ease.

4.2 Automata $\tilde{G}_p$ and $\tilde{G}_f$

It is necessary to define some key concepts at the outset of this section in order to construct automata $\tilde{G}_p$ and $\tilde{G}_f$ later.

Definition 4: Assume $\tilde{G} = (X_g, \tilde{\Sigma}, \tilde{f}_g, \tilde{\Gamma}_g, x_0, X_m)$ and $\tilde{\Sigma}_f = \{(\sigma_f, \varepsilon)\}$ is the fault event set. State $x \in X_g$ is considered a fault state if $\exists \tilde{s} \in \tilde{L}_f$, such that $\tilde{f}_g(x_0, \tilde{s}) = x$; state $x' \in X_g$ is considered a pre-fault state if $(\sigma_f, \varepsilon) \in \Gamma_g(x')$. F represents the set of fault states, while $\prec F$ represents the set of pre-fault states. State $x' \in (X_g - F)$ is referred to as the acc-fault state if there exist $\tilde{s} \in \Sigma^*$ and $y \in F$ such that $\tilde{f}_g(x', \tilde{s}) = y$; the set of acc-fault states is denoted by $\rightarrow F$.

Definition 5: In $\tilde{G}$, a path $(x_1, \tilde{\sigma}_1, x_2, ..., \tilde{\sigma}_{n-1}, x_n)$ forms a fault path if $x_1 = x_0$ and there exists a state $x_i \in F (i < 1 \leq n)$.

Definition 6: Let $(x_0, \tilde{\sigma}_1, x_1, ..., \tilde{\sigma}_n, x_n)$ be a fault path. Assume $\tilde{s}$ and $\tilde{t}$ are strings on the path. If there is a state $x_i \in F (i < 1 \leq n)$ such that $\tilde{f}_g(x_0, \tilde{s}) = x_i$ and for any $\tilde{t} \in \hat{\tilde{s}}$, $\tilde{f}_g(x_0, \tilde{t}) \notin F$, then state x_i is referred to as first-entered fault state on the path.

The definition of prognosability makes it clear that if the longest non-faulty prefix in a string that ends in a fault event satisfies the unpredictable constraints, then the occurrence of the fault event must be unpredictable. In this sense, the verification approach's main idea is to track two types of strings: the permanent normal string and the longest non-faulty prefix string, which always keep the same observation during system execution. The system behavior is divided into two parts for this purpose: normal and faulty.

For describing the normal behavior of $\tilde{G}$, the automaton $\tilde{G}_n = (X_n, \tilde{\Sigma}, \tilde{f}_n, \tilde{\Gamma}_n, x_{0n})$ will be computed. Create an automaton $\tilde{A}_n$ first, which consists of a single state N with a self-loop attached by all events of $\tilde{G}$ except fault events. Using $\tilde{G}$ and $\tilde{A}_n$, build automaton $\tilde{G}_n = \tilde{G} \times \tilde{A}_n$.

On the basis of $\tilde{G}_n$, a new automaton $\tilde{G}_p = (X_p, \tilde{\Sigma}, \tilde{f}_p, \tilde{\Gamma}_p, x_{0n})$ is established to describe the permanent normal behavior of $\tilde{G}$. If all states of $\tilde{G}_n$ in cycles are considered to be the marked states, then $\tilde{G}_p = CoAc(\tilde{G} \times \tilde{A}_n)$.

The automaton $\tilde{G}_f = (X_f, \tilde{\Sigma}, \tilde{f}_f, \tilde{\Gamma}_f, x_{0n})$ can be used to express the fault language of the system. To obtain $\tilde{G}_f$, it is necessary to first construct a label automaton $\tilde{A}_l$, which has three state types (N, C, and F), with transitions that can only be triggered by controllable or fault events. Compute $\tilde{G} \| \tilde{A}_l$. Let the first-entered fault state on each fault path be the marked state, then $\tilde{G}_f = CoAc(\tilde{G} \| \tilde{A}_l)$. Add a self-cycle containing all events to all dead states.

4.3 Testing automaton $V_{\tilde{G}}$

Build a test automaton $V_{\tilde{G}} = (X_V, \Sigma_V, f_V, x_{V,0})$ using $\tilde{G}_p$ and $\tilde{G}_f$, where $x_{V,0} = (x_{0n}, x_{0n})$, $X_V = X_f \times X_p$, and $\Sigma_V = (\tilde{\Sigma} \bigcup \{(\varepsilon,\varepsilon)\}) \times (\tilde{\Sigma} \bigcup \{(\varepsilon,\varepsilon)\})$. $f_V : X_V \times \Sigma_V \to X_V$ is defined as follows.

For any $x_v = (x_1 l_1, x_2 l_2) \in X_V$ and $\sigma_V = ((\sigma_1, e_1)(\sigma_2, e_2)) \in \Sigma_V$,

1) if $(\sigma_1, e_1) \in \tilde{\Gamma}_f(x_1 l_1), (\sigma_2, e_2) \in \tilde{\Gamma}_p(x_2 l_2)$, and $e_1 = e_2$,

$f_V(x_v, \sigma_V) = (\tilde{f}_f(x_1 l_1, (\sigma_1, e_1)), \tilde{f}_p(x_2 l_2, (\sigma_2, e_2)))$;

2) if $(\sigma_1, e_1) \in \tilde{\Gamma}_f(x_1 l_1)$ and $e_1 = \varepsilon \wedge e_2 \neq \varepsilon$,

$f_V(x_v, ((\sigma_1, e_1)(\varepsilon, \varepsilon))) = (\tilde{f}_f(x_1 l_1, (\sigma_1, e_1)), x_2 l_2)$;

3) if $(\sigma_2, e_2) \in \tilde{\Gamma}_p(x_2 l_2)$ and $e_2 = \varepsilon \wedge e_1 \neq \varepsilon$,

$f_V(x_v, ((\varepsilon, \varepsilon)(\sigma_2, e_2))) = (x_1 l_1, \tilde{f}_p(x_2 l_2, (\sigma_2, e_2)))$.

In order to provide Theorem 2 later, the following definitions and lemmas must be introduced.

Definition7: Assume $V_{\tilde{G}}$ is the verifier of $\tilde{G}$, $\{(x_1^1 l_1^1, x_2^1 l_2^1), (x_1^2 l_1^2, x_2^2 l_2^2), ..., (x_1^n l_1^n, x_2^n l_2^n)\} \subseteq X_V$, and $\{((\sigma_1^1, e_1^1)(\sigma_2^1, e_2^1)), ((\sigma_1^2, e_1^2)(\sigma_2^2, e_2^2)), ..., ((\sigma_1^n, e_1^n)(\sigma_2^n, e_2^n))\} \subseteq \Sigma_V$. A NF-uncertain cycle is formed by a path denoted by $(x_1^1 l_1^1, x_2^1 l_2^1) \xrightarrow{((\sigma_1^1, e_1^1)(\sigma_2^1, e_2^1))} (x_1^2 l_1^2, x_2^2 l_2^2) \xrightarrow{((\sigma_1^2, e_1^2)(\sigma_2^2, e_2^2))} ... \xrightarrow{((\sigma_1^i, e_1^i)(\sigma_2^i, e_2^i))} (x_1^{i+1} l_1^{i+1}, x_2^{i+1} l_2^{i+1}), ..., \xrightarrow{((\sigma_1^n, e_1^n)(\sigma_2^n, e_2^n))} (x_1^1 l_1^1, x_2^1 l_2^1)$, where $l_1^i = F$ and $l_2^i = N$ for all $i = 1, 2, ...n$.

Lemma 5: Assume that $\tilde{L}_f$ is the language generated by $\tilde{G}_f$, $\tilde{L}_{\sim f}^*$ is the language generated by $\tilde{G}_p$, and $V_{\tilde{G}}$ is the verifier created by combining $\tilde{G}_f$ and $\tilde{G}_p$. There exist $\tilde{s}_f \in \tilde{L}_f$ and $\tilde{s}_n \in \tilde{L}_{\sim f}^*$ such that $\tilde{f}_f(x_{0n}, \tilde{s}_f) = x_1^1 l_1^1$, $\tilde{f}_p(x_{0n}, \tilde{s}_n) = x_2^1 l_2^1$, and $\tilde{P}_\Delta(\tilde{s}_f) = \tilde{P}_\Delta(\tilde{s}_n)$, if there is a NF-uncertain cycle $(x_1^1 l_1^1, x_2^1 l_2^1) \xrightarrow{((\sigma_1^1, e_1^1)(\sigma_2^1, e_2^1))} (x_1^2 l_1^2, x_2^2 l_2^2) \xrightarrow{((\sigma_1^2, e_1^2)(\sigma_2^2, e_2^2))} ....., \xrightarrow{((\sigma_1^i, e_1^i)(\sigma_2^i, e_2^i))} (x_1^{i+1} l_1^{i+1}, x_2^{i+1} l_2^{i+1}), ..., \xrightarrow{((\sigma_1^n, e_1^n)(\sigma_2^n, e_2^n))} (x_1^1 l_1^1, x_2^1 l_2^1)$ with $l_1^i = F$ and $l_2^i = N$ for all $i = 1, 2, ...n$ in the $V_{\tilde{G}}$.

Proof: Consider the state $(x_1^1 l_1^1, x_2^1 l_2^1)$. Since $l_1^1 = F$, there exists a string $\tilde{s}(\sigma_f, \varepsilon) \in \tilde{L}_f$ such that $\tilde{f}_f(x_{0n}, \tilde{s}) = x_1^1 l_1^1$. Because $l_2^1 = N$, a string $\tilde{s}' \in \tilde{L}_{\sim f}^*$ exists such that $\tilde{f}_p(x_{0n}, \tilde{s}') = x_2^1 l_2^1$. According to the versifier's construction process, it is obvious that $\tilde{P}_\Delta(\tilde{s}(\sigma_f, \varepsilon)) = \tilde{P}_\Delta(\tilde{s}')$. If $\tilde{s}_f = \tilde{s}(\sigma_f, \varepsilon)$ and $\tilde{s}_n = \tilde{s}'$, the conclusion is confirmed.

Definition 8: Let $\tilde{t}, \tilde{u}, \tilde{r} \in \tilde{\Sigma}^*$, $\tilde{s} = \tilde{t}\tilde{u}(\sigma_f, \varepsilon) \in \tilde{L}_f$, and $\tilde{t} = \tilde{r}(\sigma, e)$, where $e \neq \varepsilon$. If the occurrence of event (σ_f, ε) in $\tilde{s}$ is prognosable w.r.t. $\tilde{P}_\Delta$ and $\tilde{t}$, but not w.r.t. $\tilde{P}_\Delta$ and $\tilde{r}$, then $\tilde{t}$ is called the minimum prognosable prefix of $\tilde{s}$; $\tilde{r}$ is called the maximum unpredictable prefix of $\tilde{s}$. The state reached by $\tilde{t}$ is expressed as $x_{\tilde{t}\uparrow}(x \in X_g)$; the set composed of all $x_{\tilde{t}\uparrow}$ is denoted by $\mathrm{X}_{\tilde{t}\uparrow}$; the state reached by $\tilde{r}$ is expressed as $x_{\tilde{r}\uparrow}(x \in X_g)$; the set composed of all $x_{\tilde{r}\uparrow}$ is denoted by $\mathrm{X}_{\tilde{r}\uparrow}$.

Lemma 6: Let $\tilde{s} \in \tilde{L}_f$ and the occurrence of event (σ_f, ε) of $\tilde{s}$ be prognosable w.r.t. $\tilde{P}_\Delta$. Assume that $\tilde{r}$ is the maximum unpredictable prefix of $\tilde{s}$. If there exists an event (σ, e) with $e \neq \varepsilon$ such that $\tilde{t} = \tilde{r}(\sigma, e) \in \hat{\tilde{s}}$, then $\tilde{t}$ is the minimum prognosable prefix of $\tilde{s}$.

Proof: Contradiction can be used to prove the conclusion. Assume that $\tilde{t}$ is not the minimum prognosable prefix of $\tilde{s}$. There are two possibilities: first, $\tilde{t}$ is the unpredictable prefix of $\tilde{s}$; second, $\tilde{t}$ is the prognosable prefix of $\tilde{s}$ but not the minimum prognosable prefix. If $\tilde{t}$ is the unpredictable prefix of $\tilde{s}$, then $\tilde{r}$ is not the maximum unpredictable prefix of $\tilde{s}$. This contradicts the known conditions. If $\tilde{t}$ is the prognosable prefix of $\tilde{s}$ but not its minimum prognosable prefix, then $\tilde{r}$ is the prognosable prefix of $\tilde{s}$. The initial conditions are in conflict with this. As a result, $\tilde{t}$ is the minimum prognosable prefix of $\tilde{s}$.

This lemma demonstrates that it is simple to determine a fault string's minimum prognosable prefix given its maximum unpredictable prefix.

Lemma 7: Let $\tilde{s} = (\sigma_1, e_1)...(\sigma_i, e_i)...(\sigma_j, e_j)...(\sigma_f, \varepsilon) \in \tilde{L}_f$, $(x_0 l_0, (\sigma_1, e_1)...(\sigma_i, e_i), x_i l_i, ..., (\sigma_j, e_j), x_j l_j, ...(\sigma_f, \varepsilon), x_k, l_k)$ be the fault path generated by $\tilde{G}_f$ and the occurrence of event (σ_f, ε) in the string $\tilde{s}$ is prognosable. If the following conditions are met:

$\mathrm{x_i} \in \mathrm{X}_{\tilde{t}\uparrow}$, $l_j = C$, and $l_k = F$, where $0 < i < j < k$, then the occurrence of event (σ_f, ε) in the string $\tilde{s}$ is preventable.

Proof: If $\tilde{t} = (\sigma_1, e_1)(\sigma_2, e_2)...(\sigma_i, e_i)$, $\tilde{u} = (\sigma_{i+1}, e_{i+1})...(\sigma_j, e_j)$, and $\tilde{\omega} = (\sigma_{j+1}, e_{j+1})...(\sigma_f, \varepsilon)$, then $\tilde{s} = \tilde{t}\tilde{u}\tilde{w}$ and $\tilde{f}_f(x_0, \tilde{t}) = x_i l_i$. Since $\mathrm{x_i} \in \mathrm{X}_{\tilde{t}\uparrow}$, $\tilde{t}$ is the minimum prognosable prefix of $\tilde{s}$, according to Definition 8. That is, the occurrence of event (σ_f, ε) in $\tilde{s}$ is prognosable w.r.t. $\tilde{P}_\Delta$ and $\tilde{t}$. $\tilde{f}_f(x_i l_i, \tilde{u}) = x_j l_j$ and $l_j = C$ in the path. As a consequence, $\|\tilde{u}\| \geq 1$ and there exists $(\sigma_c, e) \in \tilde{\Sigma}_c$ such that $(\sigma_c, e) \in \tilde{u}$. Furthermore, $\tilde{f}_f(x_j l_j, \omega) = x_k l_k$ with $l_k = F$ and $j < k$, resulting in $\tilde{\omega} \in \tilde{L}/t'u'$ and $(\sigma_f, \varepsilon) \in \tilde{\omega}$. This indicates that there is a controllable event

following the prognosable prefix but prior to the fault event. Therefore, the occurrence of event (σ_f, ε) in the string $\tilde{s}$ is preventable.

The lemma provides a method for determining whether a fault in a predictable fault string can be avoided.

After that, a theorem is provided to verify the active prognosability of $L(\tilde{G})$.

Theorem 2: Given $\tilde{G}_p$ and $\tilde{G}_f$, $V_{\tilde{G}}$ is the verifier constructed by them. $\tilde{L}$ generated by $\tilde{G}$ is actively prognosable w.r.t. $\tilde{P}_\Delta$, $\tilde{\Sigma}_f$, and $\tilde{\Sigma}_c$ iff the following conditions hold:

1) There is no NF-uncertain cycle in the $V_{\tilde{G}}$;

2) For each fault string in $\tilde{G}_f$, the occurrence of event (σ_f, ε) is preventable.

Proof: This proof procedure is split into two parts as well.

$(\Rightarrow)$: Assume that $\tilde{L}$ generated by $\tilde{G}$ is actively prognosable in terms of $\tilde{P}_\Delta$, $\tilde{\Sigma}_f$, and $\tilde{\Sigma}_c$. It is self-evident that the occurrence of event (σ_f, ε) in each fault string of $\tilde{G}_f$ is prognosable and preventable. For any string $\tilde{s} \in \tilde{L}_f$, there must be a state in the $\tilde{G}_f$, assuming that it is expressed as $x_1 l_1$, such that $\tilde{f}_f(x_{0n}, \tilde{s}) = x_1 l_1$ and $l_1 = F$. Because the occurrence of event (σ_f, ε) in $\tilde{s}$ is prognosable, by Definition 2, there exists no $s' \in \tilde{L}_{\sim f}^{*}$ such that $\tilde{P}_\Delta(s') = \tilde{P}_\Delta(s)$. Therefore, according to lemma 5, there is no NF-uncertain cycle in the $V_{\tilde{G}}$.

$(\Leftarrow)$: Assume that there is an NF-uncertain cycle $(x_1^1 l_1^1, x_2^1 l_2^1) \xrightarrow{((\sigma_1^1, e_1^1)(\sigma_2^1, e_2^1))} (x_1^2 l_1^2, x_2^2 l_2^2) \xrightarrow{((\sigma_1^2, e_1^2)(\sigma_2^2, e_2^2))} \ldots \xrightarrow{((\sigma_1^i, e_1^i)(\sigma_2^i, e_2^i))}$ $(x_1^{i+1} l_1^{i+1}, x_2^{i+1} l_2^{i+1}), \ldots, \xrightarrow{((\sigma_1^n, e_1^n)(\sigma_2^n, e_2^n))} (x_1^1 l_1^1, x_2^1 l_2^1)$ in the $V_{\tilde{G}}$ with $l_1^i = F$ and $l_2^i = N$ for all $i = 1, 2, \ldots n$. Lemma 5 states that there exist $\tilde{s}_f \in \tilde{L}_f$ and $\tilde{s}_n \in \tilde{L}_{\sim f}^{*}$ with $\tilde{P}_\Delta(\tilde{s}_f) = \tilde{P}_\Delta(\tilde{s}_n)$ such that $\tilde{f}_f(x_{0n}, \tilde{s}_f) = x_1^1 l_1^1$ and $\tilde{f}_p(x_{0n}, \tilde{s}_n) = x_2^1 l_2^1$. Furthermore, because for all $i = 1, 2, \ldots n$, $l_2^i = N$, it can be deduced that for any $\tilde{\omega} \in \tilde{L} / \tilde{s}_n$, $\sigma_f \notin \tilde{\omega}$ must be true. As a result, $\tilde{L}$ generated by $\tilde{G}$ is not prognosable w.r.t. $\tilde{P}_\Delta$, $\tilde{\Sigma}_f$, and $\tilde{\Sigma}_c$. This clearly shows that if $\tilde{L}$ is actively predictable, the assumption is impossible. Further to that, suppose that there exists $\tilde{s} \in \tilde{L}_f$ and the occurrence of event (σ_f, ε) in $\tilde{s}$ is not preventable, then it is determined by Proposition 1 that $\tilde{L}$ generated by $\tilde{G}$ is not actively prognosable w.r.t. $\tilde{P}_\Delta$, $\tilde{\Sigma}_f$, and $\tilde{\Sigma}_c$. This goes against the stated conditions.Thus, if conditions 1) and 2) are both met, $\tilde{L}$ generated by $\tilde{G}$ is actively prognosable w.r.t. $\tilde{P}_\Delta$, $\tilde{\Sigma}_f$, and $\tilde{\Sigma}_c$.

4.4 An algorithm

The following algorithm is offered to verify the active prognosability based on the aforementioned theorem.

Algorithm 1: Verifying active prognosability of $L(\tilde{G})$

Input: $\tilde{G}$, $\tilde{\Sigma}_f$, $\tilde{\Sigma}_c$

Output: Yes- $L(\tilde{G})$ is actively prognosable;

No- $L(\tilde{G})$ is not actively prognosable.

1. Construct the transformed automaton $\tilde{G}$.
2. Construct the automaton $\tilde{G}_f$.
3. Build the automaton $\tilde{G}_p$.
4. Create the testing automaton $V_{\tilde{G}}$.
5. Check whether conditions in Theorem 2 are satisfied; If satisfied, then output YES; otherwise output NO.

Remark 2: The computational complexity of the first three steps of the algorithm is obviously $O(|X_g||\tilde{\Sigma}|)$. The complexity of computing verifier automaton $V_{\tilde{G}}$ in the fourth step is $O(|X_g|^2 |\tilde{\Sigma}|^2)$. The verification of the existence of an NF-uncertain cycle in step 5 is based on the $V_{\tilde{G}}$, so the complexity will not exceed that of the verifier construction; the computational complexity for verifying the second condition in Theorem 2 is the same as that of step 2. As a result, the total computational complexity of the algorithm is $O(|X_g|^2 |\tilde{\Sigma}|^2)$, implying that executing the algorithm takes polynomial time in terms of the number of states and events.

The example below shows how to use the algorithm to validate a system's active prognosability.

Example 2: Consider the automaton $\tilde{G}_2$, a variant of the G_1 shown in Fig.2. The only difference between G_1 and G_2 is that for the event α starting from state 2, its observation label becomes e_2. Let us now verify the active prognosability of $\tilde{L}(G_2)$.

First, as shown in Fig.3, the transformed automaton $\tilde{G}_2$ is built. The automata $\tilde{A}_n$ and $\tilde{A}_l$ are then built, as shown in Fig. 4 and Fig.5 respectively.

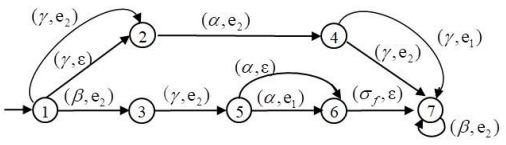

Fig. 3. Transformed automaton $\tilde{G}_2$ of G_2.

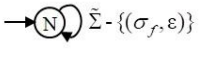

Fig. 4. Automaton $\tilde{A}_n$.

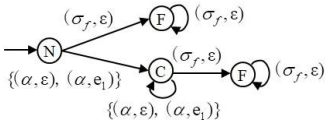

Fig. 5. Automaton $\tilde{A}_f$.

Establish automata $\tilde{G}_{2f}$ and $\tilde{G}_{2p}$ based on $\tilde{G}_2$, $\tilde{A}_n$ and $\tilde{A}_f$, as shown in Figs. 6 and 7, respectively.

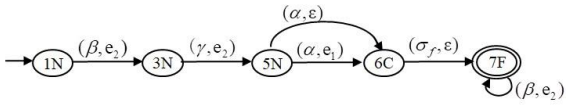

Fig. 6. $\tilde{G}_{2f}$ of Example 2.

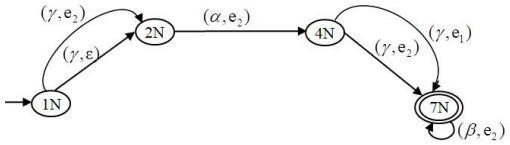

Fig. 7. $\tilde{G}_{2p}$ of Example 2.

Build test automaton $V_{\tilde{G}_2}$ in accordance with $\tilde{G}_{2f}$ and $\tilde{G}_{2p}$ as shown in Fig.8.

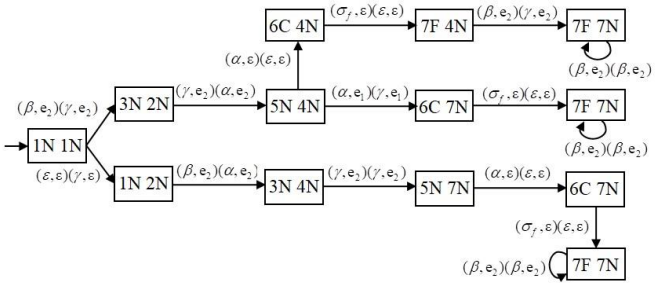

Fig. 8. Test automaton $V_{\tilde{G}_2}$ of Example 2.

It is clear that cycle $7F7N \xrightarrow{(\beta,e_2)(\beta,e_2)} 7F7N$ in $V_{\tilde{G}_2}$ is NF-uncertain. Theorem 2 states that $L(\tilde{G}_2)$ is not actively prognosable w.r.t. $\tilde{P}_\Delta$, $\tilde{\Sigma}_f$, and $\tilde{\Sigma}_c$. This indicates that $L(G_2)$ is not actively prognosable w.r.t. M, $\tilde{\Sigma}_f$, and $\tilde{\Sigma}_c$.

Example 3: Once more, think about the plant G_1 in Fig. 2. To verify the active prognosability of the system, Algorithm 1 will be used.

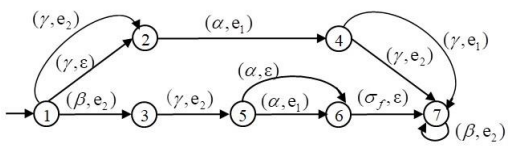

Fig. 9. Transformed automaton $\tilde{G}_1$ of G_1.

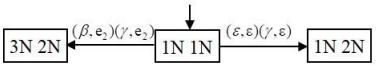

Fig. 10. Test automaton $V_{\tilde{G}_1}$ of Example 3.

Since $V_{\tilde{G}_1}$ lacks an NF-uncertain cycle, $L(\tilde{G}_1)$ is predictable. Let $\tilde{s} = (\beta, e_2)(\gamma, e_2)(\alpha, e_1)(\sigma_f, \varepsilon) \in \tilde{L}_f$ or $\tilde{s} = (\beta, e_2)(\gamma, e_2)(\alpha, \varepsilon)(\sigma_f, \varepsilon) \in \tilde{L}_f$. In addition, it is clear that (β, e_2) is the maximum unpredictable prefix of $\tilde{s}$ from $V_{\tilde{G}_1}$. Let $\tilde{t} = (\beta, e_2)(\gamma, e_2) \in \hat{\tilde{s}}$ According to Lemma 6, $\tilde{t}$ is the minimum prognosable prefix of $\tilde{s}$. Additionally, as depicted in Fig. 6, there are three states 5N, 6C, and 7F, respectively, such that $5 \in X_{\tilde{t}\uparrow}$, $\tilde{f}_f(5N, (\alpha, e_1)) = 6C$ (or $\tilde{f}_f(5N, (\alpha, \varepsilon)) = 6C$), and $\tilde{f}_f(6C, (\sigma_f, \varepsilon)) = 7F$. Thus, the occurrence of event (σ_f, ε) in the string $\tilde{s}$ is preventable by Lemma 7. Theorem 2 shows that $L(\tilde{G}_1)$ is actively prognosable w.r.t. $\tilde{P}_\Delta$, $\tilde{\Sigma}_f$, and $\tilde{\Sigma}_c$, i.e., $L(\tilde{G}_1)$ is actively prognosable w.r.t. M, Σ_f, and Σ_c.

5. ACTIVE PROGNOSIS WITH SUPERVISORY CONTROL

The previous part discussed the verification of active prognosability. If a system is actively prognosable, the fault behavior of the system can be predicted in advance and can be forcibly terminated through supervisory control. Thus, the issue of supervisory control based on failure prognosis information will be addressed in the section that follows.

Language L might contain strings that breach safety; they might be some of $G's$ undesirable behaviors that need to be avoided. This indicates that what people actually need or are interested in is the sub-language, also sometimes called specification, that represents the "legal" or "admissible" behavior of the system. Due to this, a supervisor, denoted by S, should be added to the system and S/G denotes the closed-loop system controlled by the supervisor S.

Definition 9: Given a closed-loop system S/G and the specification language K, for any $s(s \in L)$ generated so far by G and $\forall t \in M(s)$,

$$S(t) = (\Sigma_{uc} \cap [\bigcup_{s' \in M^{-1}(t)} \Gamma_g(x_0, s')] \cup \{\sigma \in \Sigma_c : (\exists s'\sigma \in \bar{K})[M(s') = t]\}. \quad (13)$$

Definition 10: Given a DES G with the non-deterministic observations M and a supervisor S, the language generated by S/G is defined recursively as follows:

$$\varepsilon \in L(S/G),$$

$$s \in L(S/G) \wedge s\sigma \in L(G) \wedge (\exists t \in M(s))\sigma \in S(t) \Leftrightarrow s\sigma \in L(S/G). \quad (14)$$

Previous research has demonstrated that $L(G_n)$ characterizes the normal behavior of the system. Therefore, the "admissible" behavior of the system is represented by the legal sub-language of L. Assume $K = L(G_n)$. The next question is: is it possible to find a supervisor S such that $L(S/G) = K$.

As previously stated, the nondeterministic supervisory control problem is solvable given G, M, and K if and only if K is controllable with respect to L and Σ_{uc} and O-observable with respect to L and M (Zhou, 2019).

Let's first consider whether K is controllable.

Controllability theorem (Cassandras, 2008): For a given system G with non-deterministic observations M, the specification language K is controllable with respect to L and Σ_{uc} if

$$\bar{K}\Sigma_{uc} \bigcap L(G) \subseteq \tilde{K}. \tag{15}$$

The theorem states that a supervisor cannot disable uncontrollable events, i.e.,

$$(\forall s \in L(G))(\forall t \in M(x_0, s))$$

$$(\Sigma_{uc} \bigcap [\bigcup_{s' \in M^{-1}(t)} \Gamma_g(x_0, s')]) \subseteq S(t). \tag{16}$$

According to the theorem, it is easy to draw the following conclusion.

Proposition2: If $K = L(G_n)$ and $\sigma_f \in \Sigma_{uc}$, then the specification language K is not controllable with respect to L and Σ_{uc}.

Proof: The conclusion will be proven through contradiction. Take note that K is prefix-closed. Assume that K is controllable with respect to L and Σ_{uc} and S is a supervisor attached to it. Since $K \subseteq L(G)$, $K \neq \phi$, and $K = \bar{K}$, the controllability theorem states that $\bar{K}\Sigma_{uc} \bigcap L(G) \subseteq \tilde{K}$. This means that there is $t \in M(s)$ such that $(\Sigma_{uc} \bigcap \bigcup_{s' \in M^{-1}(t)} \Gamma_g(x_0, s')]) \subseteq S(t)$ for each string $s \in K$ with $f_g(x_0, s) \in \prec F$. It is well known that $s \in M^{-1}(t)$, $\sigma_f \in \Sigma_{uc}$, and $\sigma_f \in \Gamma_g(x_0, s)$. As a result, $\sigma_f \in S(t)$, which implies that σ_f is an allowed event under the action of the supervisor S. Because $s \in K$, $\sigma_f \in S(t)$, and $s\sigma_f \in L(G)$, by Definition 10, it is known that $s\sigma_f \in K$. However, $K = L(G_n)$, so $s\sigma_f \notin K$. This is a logical contradiction. Thus, K is uncontrollable with respect to L and Σ_{uc}.

Example 4: Reconsider the plant G_1 in Fig.2. G_{1n} is the automaton that generates the system's normal behavior, as demonstrated in Fig.11. Assume $K = L(G_n)$ and $\sigma_f \in \Sigma_{uc}$. If take $t = e_2e_2e_1$, then $M^{-1}(t) = \{\beta\gamma\alpha\}$. Suppose $s' = \beta\gamma\alpha$, then $\Sigma_{uc} \bigcap \Gamma_g(x_0, s') = \sigma_f$. Controllability Theorem states that $\sigma_f \in S(t)$. Therefore, $s'\sigma_f = \beta\gamma\alpha\sigma_f \in K$. But the truth is $s'\sigma_f \notin K$. As a result, K is not controllable with respect to L and Σ_{uc}.

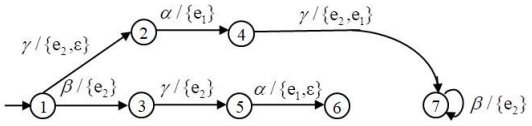

Fig. 11. G_{1n} of Example 4.

Because the given language K is not controllable, it is necessary to find the largest sub-language of K that is controllable.

Given DE G with uncontrollable event set Σ_{uc} and admissible language $K(K = \bar{K} \subseteq L(G))$, find a supervisor S such that $L(S/G) \subseteq K$ and $L(S/G)$ is ``the largest it can be", that is, for any other supervisor S', there is $L(S'/G) \subseteq L(S/G)$. This is a fundamental issue in supervisory control. Solving the problem entails selecting a supervisor S such that $L(S/G) = K^{\uparrow C}$. According to Cassandras and Lafortune in the work (Cassandras, 2008), $K^{\uparrow C}$ is the supreme prefix-closed sub-language of K that is controllable with respect to L and Σ_{uc}.

$K^{\uparrow C}$ can be calculated using the following theorem.

Theorem 3 (Cassandras, 2008): If $K = \bar{K}$, then

$$K^{\uparrow C} = K - [(L - K) \setminus \Sigma_{uc}^*]\Sigma^*. \tag{17}$$

The basic idea of the calculation is to restrict the behavior of the system G in order to keep it safe, but do not restrict them more than necessary. From the language viewpoint, all strings in K that possess a prefix that violates the controllability condition must be deleted from K to make the language controllable. The algorithm for calculating $K^{\uparrow C}$ can be found in reference (Cassandras, 2008).

Example 5: Continue to consider the prefix-closure language $K = L(G_n)$. Assume that $\Sigma_c = \{\alpha\}$ and $\Sigma_{uc} = \{\beta, \gamma, \sigma_f\}$. Example 4 demonstrated that K is not controllable with respect to L and Σ_{uc}. Let's use the algorithm to calculate $K^{\uparrow C}$ in the sequence.

Delete state 6 and its related links from H to get a new automaton H', as shown in Fig.12. $K^{\uparrow C} = L(H')$ and it is controllable with respect to L and Σ_{uc}.

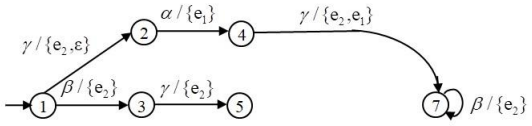

Fig. 12. Automaton H' of Example 5.

Next, its observability will be discussed.

Definition 11 (Zhou,2019):For DES G with non-deterministic observation M, the nonempty specification language K,

generated by the sub-automaton H' of G, is O-observable with respect to L and M if for $\forall s \in L$ and $\sigma \in \Sigma$,

$$s\sigma \in L \wedge s\sigma \in K \Rightarrow (\exists t \in M(s))$$

$$\forall s' \in M^{-1}(t) s' \in K \wedge s'\sigma \in L \Rightarrow s'\sigma \in K. \quad (18)$$

The following example exemplifies the definition.

Example 6: Still consider the automaton G_1 described by Fig.2. As shown in example 5, for $K = L(G_n)$, $K^{\uparrow C}$ is controllable with respect to L and Σ_{uc}. Now demonstrate that $K^{\uparrow C}$ is O-observable with regard to L and M.

Let $s = \gamma$ and $\sigma = \alpha$. It is evident that $s\sigma = \gamma\alpha \in L(G) \wedge s\sigma \in K^{\uparrow C}$. $M(s) = M(\gamma) = \{e_2, \varepsilon\}$. Let $s' = \beta$. If take $t = e_2$, then $M^{-1}(t) = \{s, s'\}$. For $s' \in M^{-1}(t), s' \in K^{\uparrow C}$ but $s'\sigma = \beta\alpha \notin L(G)$. If take $t = \varepsilon$, then $M^{-1}(t) = \{s\}$. $s' \in M^{-1}(t)$ indicates that $s' = s$. In this instance, $s'\sigma \in L(G)$ and $s'\sigma \in K^{\uparrow C}$. As a consequence, O-observability is satisfied.

Since $K^{\uparrow C}$ is controllable and O-observable, let's create a supervisor for it.

The unobservable reach of each state $x \in X_g$ is denoted by $UR(x)$ and is defined as

$$UR(x) = \{y \in X_g : (\exists s \in \Sigma^*)[\varepsilon \in M(x,s) \wedge (f_g(x,s) = y)]\}.$$

The definition is extended to sets of states $B \subseteq X_g$ by

$$UR(B) = \bigcup_{x \in B} UR(x). \quad (19)$$

An observer of H' denoted by H'_{obs} is constructed. $H'_{obs} = (X_{obs}, \Delta, f_{obs}, x_{0,obs})$, where $x_{0,obs} := UR(x_0)$. For any $B \in X_{obs}$ and $e \in \Delta - \{\varepsilon\}$.

$$f_{obs}(B,e) := UR(\{x \in Xg : (\exists x' \in B)(\exists \sigma \in \Sigma \wedge M(x',\sigma) = e)$$

$$[x = fg(x',\sigma)]\}). \quad (20)$$

Theorem 4: Given a DES G with non-deterministic observations M and the specification language $K^{\uparrow C}$, if the nondeterministic supervisory control problem is solvable, the following supervisor is a solution.

$$S_p(t) = \bigcup_{x \in x_{current,obs}} \Gamma_{h'}(x), \quad (21)$$

where t is the current string of observable labels and $x_{current,obs}$ is the state of H'_{obs} following t's observation.

Based on the Theorem, calculate the active events for each state belonging to x_{obs} and add self-loops for the events with a unique tag ε. As a result, the desired supervisor is implemented as an automaton.

Example 7: Consider the plant G_1 shown in Fig.2 once more. As seen by example 6, $K^{\uparrow C} = L(H')$ is O-observable with respect to L, Σ_{uc}, and M. According to Theorem 4, the observer shown in Fig.13 can be used to obtain the standard realization of a supervisor, as depicted in Fig.14. Since there are no active events with the unique tag ε for each state in the H', there is no need to add self-loops to the supervisor.

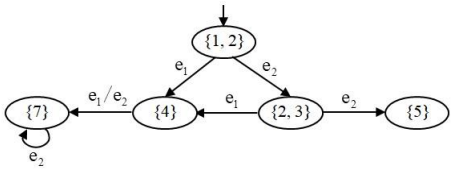

Fig. 13. Observer H'_{obs}.

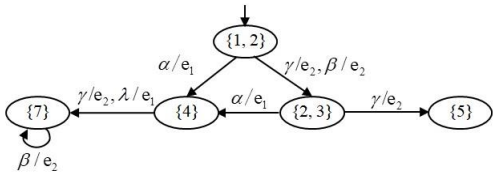

Fig. 14. Supervisor S.

The following control actions are encoded by the supervisor:

Enable α, β, γ at state 1 and 2;

Enable α, γ at state 2 and 3;

Enable γ at state 4;

Disable α at state 5;

Enable β at state 7.

For $s = \beta\gamma\alpha\sigma_f \in L_f$, Example 3 has shown us that $t = \beta\gamma$ is the minimum prognosable prefix of s. $M(t) = M(\beta\gamma) = \{e_1 e_2\}$, so when $e_1 e_2$ is observed during system operation, it can be inferred that a fault will occur. Moreover, observer H'_{obs} leads us to conclude that the system's current state is 5. In state 5, the supervisor's control action should be to disable the event α in order to prevent the occurrence of fault event σ_f and ensure the system's normal operation.

It is worth noting that the supervisor in the preceding example is built on the foundation of an observer. This differs from the reference (Zhou, 2019), which first creates an augmented automaton to evaluate the system's observability before synthesizing a supervisor.

6. CONCLUSIONS

New findings about the prognosis of faults in DESs with non-deterministic observations were reported in this work. An offline approach for testing the new notion of active prognosability is described along with its definition. According to the analysis, the algorithm is polynomial in the number of events and system states. Furthermore, a supervisor is developed, whose purpose is to maintain the

safety of the system's behavior. It should be mentioned that the study findings presented in this work can be applied to multiple fault types as well as just single fault type. Because in systems with many fault event kinds, other fault event types can be considered as non-fault unobservable events. Based on this assumption, supervisors and test automata are created separately for every kind of fault. Future research will continue to examine the pattern prognosability of stochastic or fuzzy DESs under non-deterministic observations.

ACKNOWLEDGEMENTS

This work was supported by "the National Natural Science Foundation of China" (grant number 61673122) and "Guangdong Basic and Applied Basic Research Foundation" (grant number 2023A1515012783) of China.

REFERENCES

Benmessahel, B., Touahria, M., and Nouioua, F. (2017). Predictability of fuzzy discrete event systems. *Discrete Event Dynamic Systems: Theory and Applications*, 27(4), 641-673.

Cao, W., Liu, F., and Zhao, R. (2022). Decentralized failure prognosis of stochastic discrete-event systems and a test algorithm. *IEEE Trans. on Automation Science and Engineering*, 19(4): 2944-2954.

Carvalho, L. K., Basilio, J. C., and Moreira, M. V. (2012). Robust diagnosis of discrete event systems against intermittent loss of observations. *Automatica*, 48(9), 2068-2078.

Carvalho, L. K., Moreira, M. V., and Basilio, J. C. (2021). Comparative analysis of related notions of robust diagnosability of discrete-event systems. *Annual Reviews in Control*, 51, 23-36.

Cassandras, C. G. and Lafortune, S. (2008). Introduction to discrete event systems. *Springer Berlin.*

Chang, M., Dong, W., Ji, Y., and Tong, L. (2013). On fault predictability in stochastic discrete event systems. *Asian Journal of Control*, 15(5), 1458-1467.

Genc, S. and Lafortune, S. (2009). Predictability of event occurrences in partially-observed discrete-event systems. *Automatica*, 45, 301-311.

Liao, H., Liu, F., and Zhao, R. (2022). Reliable co-prognosability of decentralized stochastic discrete-event systems and a polynomial-time verification. *IEEE Trans. on Cybernetics, 52(7), 6207-6216.*

Liu, F. and Qiu, D. (2008). Safe diagnosability of stochastic discrete event systems. *IEEE Trans. on Automatic Control*, 53(5), 1291-1296.

Liu, F. (2015). Safe diagnosability of fuzzy discrete-event systems and a polynomial-time verification. *IEEE Trans. on Fuzzy Systems*, 23(5), 1534-1544.

Liu, F. (2018). Reliable predictability of failure events for decentralized discrete-event systems. in *Proc. 37th Chin. Control Conf. (CCC)*, Wuhan, China, 2048-2053.

Liu, F., Yang, P., Zhao, R., and Dziong, Z. (2022). Verification of safe diagnosability of stochastic discrete-event systems. *International Journal of Control*, 95(2), 372-379.

Moreira, M. V., Jesus, T. C., and Basilio, J. C. (2011). Polynomial time verification of decentralized diagnosability of discrete event systems. *IEEE Trans. on Automatic Control*, 56(7), 1679-1684.

Keroglou, C. and Hadjicostis, C. N. (2018). Distributed fault diagnosis in discrete event systems via set intersection refinements. *IEEE Trans. on Automatic Control*, 63(10), 3601-3607.

Oliveira, V. de S. L., Cabral, F. G., and Moreira, M. V. (2022). K-loss robust codiagnosability of Discrete-Event Systems. *Automatica*, 140, 1-6.

Paoli, A. and Lafortune, S. (2005). Safe diagnosability for fault-tolerant supervision of discrete-event systems. *Automatica*, 41(8), 1335-1347.

Paoli, A., Sartini, M., and Lafortune, S. (2011). Active fault tolerant control of discrete event systems using online diagnostics. *Automatica* , 47, 639-649.

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

Takai, S. and Kumar, R. (2012). Distributed failure prognosis of discrete event systems with bounded-delay communications. *IEEE Trans. on Automatic Control*, 57(5), 1259-1265.

Tomola, J. H. A., Cabral, F. G., Carvalho, L. K., and Moreira, M. V. (2017). Robust disjunctive-codiagnosability of discrete-event systems against permanent loss of observations. *IEEE Trans. on Automatic Control*, 62(11), 5808-5815.

Ushio, T. and Takai, S. (2016). Nonblocking supervisory control of discrete event systems modeled by mealy automata with nondeterministic output functions. *IEEE Trans. on Automatic Control*, 61(3),799-804.

Viana, G. S. and Basilio, J. C. (2019). Codiagnosability of discrete event systems revisited: a new necessary and sufficient condition and its applications. *Automatica*, 101, 354-364.

Watanabe, Ana T. Y., Leal, A. B., and Cury, J. E. R.(2017). Safe controllability using online prognosis. *IFAC (International Federation of Automatic Control) Papers Online* 50-1, 12359-12365.

Watanabe, AnaT. Y., Leal, A. B. and Cury, J. E. R. (2021). Combining online diagnosis and prognosis for safe controllability. *IEEE Trans. on Automatic Control*, DOI 10.1109/TAC, 3124-185.

Wang, F. and Shu, S. (2016). Robust networked control of discrete event systems. *IEEE Trans. on Automation Science and Engineering*. 13(4), 1528-1539.

Xiao, C. and Liu, F. (2022). Robust fault prognosis of discrete-event systems against loss of observations. *IEEE Trans. on Automation Science and Engineering*, 19(2), 1083-1094.

Xu, S. and Kumar, R. (2009). Discrete event control under nondeterministic partial observation. *I5th Annual IEEE Conference on Automation Science and Engineering*, Bangalore, India, August 22-25, 127-132.

Yin, X. and Li, Z. (2016a). Reliable decentralized fault prognosis of discrete-event systems. *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, 46(11), 1598-1603.

Yin, X. and Li, Z. (2016b).`Decentralized fault prognosis of discrete event systems with guaranteed performance bound. *IEEE Trans. on Automatic Control*, 69, 375-379.

Yin, X. and Li, Z. (2019). Decentralized fault prognosis of discrete-event systems using state-estimate-based protocols. *IEEE Transactions on Cybernetics*, 49(4), 1302-1313.

Yin, X., Chen, J., Li, Z., and Li, S. (2019). Robust fault diagnosis of stochastic discrete event systems. *IEEE Trans. on Automatic Control*, 64(10), 4237-4244.

Yin, X., Chen, J., Li, Z., and Li, S. (2019). Robust fault diagnosis of stochastic discrete-event systems. *IEEE Trans. Automatic Control*. 64(10), 4237-4244.

Zhao, R. and Liu, F. (2021). Active prediction in discrete-event systems. *Control Engineering and Applied Informatics*. 23(1), 13-21.

Zhou, L., Shu, S., and Lin, F. (2019). Supervisory control of discrete event systems under non-deterministic observations. *18th European Control Conference* (ECC), Napoli, Italy, June 25-28.

Zhou, L., Shu, S., and Fang, H. (2020). Verifying o-observability for discrete event systems under non-deterministic observations. *IEEE 16th International Conference on Control and Automation* (ICCA)(Virtual) Sapporo, Hokkaido, Japan, Oct 9-11.