Active Prediction in Discrete-event Systems

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: This paper investigates the problem of prediction of failures that may lead to violations of critical safety requirements if they are not dealt with in a timely manner. The main contributions are as follows. First, the new notion of active predictability of discrete-event systems (DESs) is introduced and formalized to capture the feature that the occurrences of failure events can not only be predicted based on observed sequences of events, but also be prevented from occurring by properly controlling system behavior. The control actions are posed and addressed in the framework of supervisory control theory, which could be required to prevent failure events from developing into safety hazards. It is indicate through comparison that active predictability is stronger than the predictability proposed by Genc and Lafortune. Second, in order to achieve the performance of active prediction, a nondeterministic automaton called verifier is constructed and the necessary and sufficient condition for verifying the active predictability of DESs based on the verifier is presented. Third, an algorithm is developed to verify the active predictability, which can be applied to the control of robots. It is worth noting that both constructing the verifier and verifying the active predictability can be realized with polynomial complexity in the number of states and events of the system.

Keywords: discrete event systems, predictability, supervisory control.

1. INTRODUCTION

Failure detection and isolation is an important task in the automatic control of large complex systems. Failure diagnosis of DESs is to timely identify incipient failures that have occurred but may not be directly observed by the sensors, which has been widely investigated during the past decades (Sampath, 1995, 1998; Thorsley, 2007; Moreira, 2011; Liu, 2014; Chen, 2014; Yao, 2016; Zhao, 2017; Deng, 2017; Keroglou, 2018; Masopust, 2019; Viana, 2019). The main objective of failure prediction is to develop methodologies for predicting the occurrences of possible failures arising in the operation of a dynamic system. And the research problem has received considerable attention (Jeron, 2008; Genc, 2009; Briones, 2012; Takai, 2011, 2012; Chang, 2013; Chen, 2014; Grastien, 2015; Benmessahel, 2017; Yin, 2016, 2018; Liu, 2019; Zhao, 2019). Jeron, Marchand et al. developed a method to predict the occurrence of sequence patterns which contain some ordered significant observable or unobservable events (Jeron, 2008). Genc and Lafortune defined the predictability of DESs and proposed two model-based approaches to perform the online prediction and the offline verification, respectively (Genc, 2009). Briones and Madalinski refined lower and upper bound of predictability, and proposed the notion of (lb,ub)-predictability (Briones, 2013). In (Takai, 2012), the robustness of failure prediction was discussed, and a robust prognoser was introduced to predict the failures prior to their occurrence. Chang and Chen (Chang, 2013; Chen, 2014) extended the framework proposed by Genc and Lafortune (Genc, 2009) to stochastic models and introduced two new concepts of AAS-predictability and $S\_\{m\}$ -Prognosability respectively. The definition of predictability was further extended to consider the time interval and (i,j)-predictability was formulated in (Grastien, 2015). A fuzzy approach was proposed (Benmessahel, 2017) by introducing fuzzy predictability functions to characterize the predictability degree of a faulty trace as well as a faulty event in a fuzzy DES. As for distributed systems, the decentralized prognoser was defined and employed to perform the coprediction (Takai, 2011); The notion of k-reliable coprognosability was proposed (Yin, 2016) as the necessary and sufficient condition for the existence of a decentralized prognoser under the presence of unreliable local prognostic decisions. Yin and Li (Yin, 2019) proposed two novel decentralized protocols for the purpose of fault prognosis.

The copredictability of DESs was formalized (Liu, 2019) under the decentralized framework to capture the feature of copredictable DESs where the occurrences of failure events can be predicted in advance based on at least one local observation. In the paper (Zhao, 2019), the relative predictability of DESs is investigated.

This paper aims to deal with the new issue of how to prevent failure events from occurring when they are predicted to occur. This is what can be called the active prediction problem. The term active is used to distinguish the prediction from passive prediction (Jeron, 2008; Genc, 2009; Briones, 2012; Takai, 2011, 2012; Chang, 2013; Chen, 2014; Grastien, 2015; Benmessahel, 2017; Yin, 2016, 2018; Liu, 2019; Zhao, 2019) wherein the role of the prediction module is simply to observe the system behaviour and draw inference about the occurrence of potential failures. But the active prediction is one that combined observation and control and the control issues are posed and addressed in the framework of supervisory control theory. Supervisors can disable some controllable events at any time to change the behaviour of the

system to avoid failures. In order to capture the feature of the system, the notion of active predictability is introduced and formalized. And the relationship with the notion of predictability introduced by Genc and Lafortune (Genc, 2009) is analysed. It is illustrated that active predictability is stronger than the predictability. In order to achieve the necessary and sufficient condition for verifying the active predictability, a new verifier is constructed. And a verification algorithm of polynomial computational complexity is proposed.

Notice that active diagnosis and its application were investigated in (Sampath, 1998), (Thorsley, 2007) and (Chen, 2014). Inspired by these works, this paper presents an integrated approach to control and predict. But the idea of this paper is different from those references. The main differences are as follows. (Sampath, 1998), (Thorsley, 2007) and (Chen, 2014) presented some approaches of using control actions to alter the diagnosability properties of a given system, but what they did was to restrict the behaviour of a nondiagnosable system by appropriate control, to obtain a diagnosable system. This paper is not concerned with using control actions to alter the predictability property of a given system, but rather using control actions to prevent impending failure events from occurring.

Our main contributions are (1) formal definition of the notion of active predictability; (2) derivation of the necessary and sufficient condition for verifying active predictability; (3) an algorithm for checking whether a system is actively predictable.

This paper is organized as follows. In section 2, the necessary background on predictability of DESs is presented. In section 3, an example is provided to illustrate the motivation of the research. In section 4, the notion of active predictability is defined. In section 5, the necessary and sufficient condition of active predictability is given and an algorithm is designed to verify active predictability. In section 6, active predictability with multiple failure types is considered. Finally, in section 7, conclusions are drawn.

2. PRELIMINARIES

A discrete-event system is modeled as a deterministic automaton

$$G = (X, \Sigma, \delta, X_m, x_0)$$

where X is the finite state space, Σ is the set of events, $\delta : X \times \Sigma \to X$ is the transition function, X_m is the set of marked states and $x_0 \in X$ is the initial state of the system. Σ^* denotes the set of all finite strings over Σ, including the empty string ϵ. The transition function δ can be extended to the domain $X \times \Sigma^*$: for any $\mathrm{x} \in X$, $\delta(\mathrm{x},\varepsilon)=\{\mathrm{x}\}$ and $\delta(\mathrm{x},\mathrm{s}\sigma)=\bigcup_{x' \in \delta(x,s)} \delta(x',\sigma)$ for $s \in \Sigma^*$ and $\sigma \in \Sigma$.

Given an event $\sigma \in \Sigma$ and a string $s \in \Sigma^*$, if σ appears at least once in s, then denote it as $\sigma \in s$ or with slight abuse of notation, denote it as $s \cap \Sigma = \{\sigma\}$. And ϕ denotes empty sets.

The language generated by G is defined as

$$L = \{s \in \Sigma^* : \delta(x_0, s) \in X\}$$

and it is often denoted by $L(G)$. $L_m(G)$ is a set of strings that can reach the marked state and $L_m(G) \subseteq L(G)$. Given a trace s originating from x_0, 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 prefix language of s. Given a string s, the length of s (number of events including repetitions) is denoted by $\|s\|$.

All events of G are partitioned as $\Sigma = \Sigma_o \bigcup \Sigma_{uo}$, where Σ_o denotes the set of observable events and Σ_{uo} denotes the set of unobservable events. Let $\Sigma_f \subseteq \Sigma_{uo}$ denote the set of failure events, define

$$L_{\sim f} = L \bigcap (\Sigma - \Sigma_f)^*$$

as the set of all traces in L containing no failure events.

Let x be a state of G (i.e. $x \in X$). The feasible event set of x is denoted by

$$\Gamma(x) = \{\sigma \in \Sigma : \delta(x, \sigma) \in X\}.$$

The accessible part (Genc, 2009) of G with respect to x is denoted by $Ac(G,x)$ and

$$Ac(G,x) = (X_{ac}, \Sigma, \delta_{ac}, x),$$

Where $X_{ac} = \{x' \in X : \exists s \in \Sigma^* \text{ s.t. } \delta(x,s) = x'\}$, and $\delta_{ac} = \delta|_{X_{ac} \times \Sigma \to X_{ac}}$.

Define

$$\Upsilon(\Sigma_f) = \{s \in L_{\sim f} : \Gamma(\delta(x_0, s)) \bigcap \Sigma_f \neq \phi\}.$$

as the set of all traces in $L_{\sim f}$ whose next feasible events contain a failure event, where ϕ is the empty set.

Let $P : \Sigma^* \to \Sigma_o^*$ denote the usual projection operator with filters the unobservable events from a trace, which is defined as $P : \varepsilon \to \varepsilon$, and for $s \in \Sigma^*$ and $\sigma \in \Sigma$,

$$P(s\sigma) = \begin{cases} P(s)\sigma & \text{if } \sigma \in \Sigma_o \\ P(s) & \text{otherwise.} \end{cases}$$

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

For a given DES, supervisory control theory deals with the design of controller that ensures that the controlled system meets certain qualitative specifications. These specifications define the legal languages of the system. A supervisor of a DES is an external agent or controller that, based on its partial view of the system, dynamically enables or disables

the controllable events of the system in order to ensure that the resulting closed-loop language lies within the legal language. The control structure of G is defined by a partition

$$\Sigma = \Sigma_c \bigcup \Sigma_{uc}$$

where Σ_c is the set of controllable events whose occurrences can be disabled, and Σ_{uc} is the set of uncontrollable events that cannot be disabled. This paper makes the following assumption:

$$\Sigma_c \subseteq \Sigma_o$$

i.e., no unobservable event can be prevented from occurring by control.

A supervisor S is a map

$$S\colon\ L(G) \to 2^{\Sigma}$$

together with a language $L_S \subseteq L_m(G)$ such that, for each string s generated by G, the control action of S over G (denoted by S/G) enables events of $S(s) \subseteq \Sigma$, with $\Sigma_{uc} \subseteq S(s)$, and marks strings $s \in L_S$.

Definition 1 (Predictability (Genc, 2009)) Given L a prefix-closed, live language over Σ, occurrences of failure event $\sigma_f (\sigma_f \in \Sigma_f)$ are predictable in L w.r.t. P if $(\exists n \in \mathbb{N})$ s.t.

$$(\forall s \in \Upsilon(\Sigma_f))(\exists t \in \overline{s})[\mathbb{Z}(t) = 1],$$

where $\mathbb{N}$ is the set of natural numbers and the condition $\mathbb{Z}(t)$ is defined as follows:

$$\mathbb{Z}(t) = \begin{cases} 1 & \text{if } (\forall \mathrm{v} \in L/(P^{-1}(P(t)) \bigcap L_{\sim f})) \\ & (\| \mathrm{v}\| \geq \mathrm{n} \Rightarrow (\sigma_f \in \mathrm{v})) \\ 0 & \text{otherwise.} \end{cases}$$

Intuitively, occurrences of failure events in the language L are predictable if for any trace s whose next feasible events contain a failure event, it is possible to infer about the occurrence of the failure event based on the observable record of the trace. Predictability of L can be verified through constructing corresponding verifier (Genc, 2009).

3. PROBLEM MOTIVATION

Example 1: Consider the plant G described in Fig.1, in which α, β, γ and τ are observable events, σ_{uo} is an unobservable event, $\sigma_f \in \Sigma_f$ is a failure event and state 1 is the initial state.

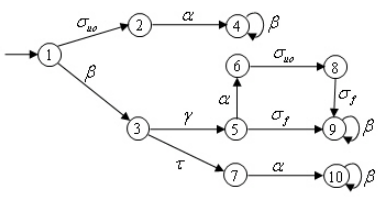

Fig. 1. Automaton G.

As Fig.1 shows, $\Upsilon(\Sigma_f) = \{\beta\gamma, \beta\gamma\alpha\sigma_{uo}\}$. Let $s = \beta\gamma$ or $s = \beta\gamma\alpha\sigma_{uo}$ and $t = \overline{s}$. Take $t = \beta\gamma$, $P^{-1}(P(t)) \bigcap L_{\sim f} = \{\beta\gamma\}$ and $L/P^{-1}(P(t)) \bigcap L_{\sim f} = \{\sigma_f\beta^n, \quad \alpha\sigma_{uo}\sigma_f\beta^n\}$. For $\forall v \in L/P^{-1}(P(t)) \bigcap L_{\sim f}$, $\|v\| \geq \mathrm{n} \Rightarrow \quad (\sigma_f \in v)$. This means that for any string $s \in \Sigma_f$, there is a $t \in \overline{s}$ which satisfies the condition $\mathbb{Z}(t) = 1$ in **Definition 1**. Therefore, all the occurrences of σ_f can be predicted, i.e. L is predictable w.r.t. P and σ_f.

Now consider the case: if the system starts from the initial state 1 and event β has occurred, then event γ or τ may occur. As long as event γ occurs, then failure event σ_f is bound to occur in the future. In order to avoid the occurrence of the failure event σ_f, a supervisor can step in and take some preventive actions, such as preventing event γ from occurring, of course, if event γ is controllable. If event γ is prevented from occurring, the next event that will occur is τ. After event τ, no failure events will occur. By this way, failure event σ_f that may cause serious security hazards can not only be predicted, but also be avoided. Such predictability can be called active predictability.

4. FORMALIZATION OF ACTIVE PREDICTABILITY

Definition 2 (Active Predictability) A prefix-closed language L is said to be actively predictable w.r.t. P, Σ_f and Σ_c if the following conditions hold:

C1. Predictability condition:

$$(\exists n \in \mathbb{N})(\forall s \in \Upsilon(\Sigma_f))(\exists t \in \overline{s}) \Rightarrow \mathbb{Z},$$

where $\mathbb{Z}$ is:

$$(\forall v \in L/(P^{-1}(P(t)) \bigcap L_{\sim f})) \wedge (\| v \| \geq n) \Rightarrow (\sigma_f \in v).$$

C2. Active predictability condition:

for $\forall s \in \Upsilon(\Sigma_f)$, if exists $t \in \overline{s}$ such that $\mathbb{Z}$ holds, and $t \bigcap \Sigma_c \neq \phi$.

Roughly speaking, the occurrences of failure events in the language L are actively predictable if for any trace whose next feasible events contain a failure event, it is possible to infer about the occurrence of the failure event based on the observable part of the trace and the prefix of the trace contains some controllable events that can be posed the plant to prevent the failure from occurring.

By comparing **definition 1** and **definition 2**, it is known that active predictability is stronger than the predictability (Genc, 2009).

Let event γ of G described in Fig.1 be controllable (i.e. $\gamma \in \Sigma_c$, then for $s = \beta\gamma$ or $s = \beta\gamma\alpha\sigma_{uo}$ ($s \in \Upsilon(\Sigma_f)$), there exists $t = \beta\gamma(t \in \overline{s})$ such that the predictability condition **C1**

holds and $t \bigcap \Sigma_c = \{\gamma\} \neq \phi$ (i.e. activity predictability condition **C2** holds). According to the **definition 2**, it is known that L generated by G is actively predictable w.r.t. P, σ_f and Σ_c. So supervisory controller of the plant can prevent the predictable failure event σ_f from occurring by disabling event γ.

5. VERIFICATION OF ACTIVE PREDICTABILITY OF DESS

In this section, the necessary and sufficiency condition of active predictability is derived and an algorithm based on a verifier is developed to verify the property of DESs.

5.1 The Necessary and Sufficiency Conditions

Definition 3 (Marked states) Let $G = (X, \Sigma, \delta, X_m, x_0)$ and $\Sigma_f = \{\sigma_f\}$ be the failure event set of G. State $x \in X$ is referred to as marked state if $\sigma_f \in \Gamma(x)$; And the set of marked states is X_m.

Definition 4 (Simplified model) Let G_N be the non-failure automaton of G. Remove the " N " labels from the states of G_N and mark all marked states of G in G_N by double-circle. Then G_N is referred to as the simplified model of G, denoted by $G_S = (\mathrm{X}_S, \Sigma, \delta, \mathrm{x}_0)$.

Definition 5 (1-class and 2-class marked states) Let G_S be the simplified model of G, state x be a marked state of G_S (i.e. $x \in \mathrm{X}_m$) and the accessible part of state x be $Ac(G,x) = (X_{ac}, \Sigma, \delta_{ac}, x)$. (a) State x is referred to as 2-class marked state if $X_{ac} \neq \{x\}$ and $X_{ac} \bigcap \mathrm{X}_m = \{x\}$; Otherwise, $x \in X_m$ is referred to as 1-class marked state. (b) The set of 2-class marked states is denoted by X_m^2; the set of 1-class marked states is denoted by X_m^1. And $X_m^1 \bigcup X_m^2 = X_m$.

Theorem 1: Let L be the language of $G = (X, \Sigma, \delta, X_m, x_0)$ and G_S be the simplified model of G. L is not actively predictable w.r.t. P, Σ_f and Σ_c if $X_m^2 \neq \phi$.

Proof: Since $X_m^2 \neq \phi$, there is a state x of G_S and $x \in X_m^2$. This means that the reachable states from state x contain only a few unmarked states except for itself. Therefore, suppose that there is a string $s \in L$ such that $\delta(x_0, s) = x$, then there exist $st_1 \in L$ and $\sigma_f \in t_1$, $st_2 \in L$ and $\|t_2\| \geq \mathrm{n}$ ($\mathrm{n} \in \mathbb{N}$) but $\sigma_f \notin t_2$. The fact indicates that the occurrence of σ_f cannot be predicted according to the observation of $\overline{s}$. According to **Definition 1**, it is known that L is not predictable w.r.t. P and σ_f. So by **Definition 2**, the conclusion that L is also not actively predictable can be drawn.

Let there be no 2-class marked states in G_S. The Verifier of G_S is a finite automaton

$$V_G = (\mathrm{X}_V, \Sigma, \delta_V, \mathrm{x}_{V,0})$$

where X_V is the state space, δ_V is the state transition function, and $\mathrm{x}_{V,0}$ is the initial state. Verifier state $\mathrm{x}_V \in \mathrm{X}_V$ is of the form $\mathrm{x}_V = [x_1 l_1; x_2 l_2]$, where $\mathrm{x}_i \in \mathrm{X}$, $l_i \in \{N, F, CN, CF\}$ and $i \in \{1, 2\}$. The state transition function $\delta_V(x_V, \sigma)$ is defined as follows.

For $\sigma \in \Sigma$,

1) if $\sigma \notin \Sigma_c$ and $\sigma \in \Sigma_o$,

a. if $\delta(x_0, \sigma) \notin X_m^1$, then

$$\delta_V([x_0 N, x_0 N], \sigma) = [\delta(x_0, \sigma)N, \delta(x_0, \sigma)N];$$

b. if $\delta(x_0, \sigma) \in X_m^1$, then

$$\delta_V([x_0 N, x_0 N], \sigma) = [\delta(x_0, \sigma)F, \delta(x_0, \sigma)F];$$

2) if $\sigma \notin \Sigma_c$ and $\sigma \in \Sigma_{uo}$,

a. if $\delta(x_0, \sigma) \notin X_m^1$, then

$$\delta_V([x_0 N, x_0 N], \sigma) = \begin{cases} [\delta(x_0, \sigma)N, x_0 N] \\ [x_0 N, \delta(x_0, \sigma)N] \\ [\delta(x_0, \sigma)N, \delta(x_0, \sigma)N]; \end{cases}$$

b. if $\delta(x_0, \sigma) \in X_m^1$, then

$$\delta_V([x_0 N, x_0 N], \sigma) = \begin{cases} [\delta(x_0, \sigma)F, x_0 N] \\ [x_0 N, \delta(x_0, \sigma)F] \\ [\delta(x_0, \sigma)F, \delta(x_0, \sigma)F]; \end{cases}$$

3) if $\sigma \in \Sigma_c$,

a. if $\delta(x_0, \sigma) \notin X_m^1$, then

$$\delta_V([x_0 N, x_0 N], \sigma) = [\delta(x_0, \sigma)CN, \delta(x_0, \sigma)CN];$$

b. if $\delta(x_0, \sigma) \in X_m^1$, then

$$\delta_V([x_0 N, x_0 N], \sigma) = [\delta(x_0, \sigma)CF, \delta(x_0, \sigma)CF].$$

For $s = t\sigma \in L(G)$ and $\delta_V([x_0 N, x_0 N], t) = [x_1 l_1, x_2 l_2]$, let $l_1' = l_1 \bigcap "C" \bigcup "F"$ and $l_2' = l_2 \bigcap "C" \bigcup "F"$ (Here, with slight abuse of notation, label merge operation is denoted by symbols of sets).

1) if $\sigma \notin \Sigma_c$ and $\sigma \in \Sigma_o$,

a. if $\delta(x_0, s) \notin X_m^1$, then

$$\delta_V([x_1 l_1, x_2 l_2], \sigma) = [\delta(x_1, \sigma)l_1, \delta(x_2, \sigma)l_2];$$

b. if $\delta(x_0, \sigma) \in X_m^1$, then

$$\delta_V([x_1 l_1, x_2 l_2], \sigma) = [\delta(x_1, \sigma)l_1', \delta(x_2, \sigma)l_2'];$$

2) if $\sigma \notin \Sigma_c$ and $\sigma \in \Sigma_{uo}$,

a. if $\delta(x_0,s) \notin X_m^1$, then

$$\delta_V([x_1l_1,x_2l_2],\sigma)=\begin{cases}[\delta(x_1,\sigma)l_1,x_2l_2]\\ [x_1l_1,\delta(x_2,\sigma)l_2]\\ [\delta(x_1,\sigma)l_1,\delta(x_2,\sigma)l_2];\end{cases}$$

b. if $\delta(x_0,s) \in X_m^1$, then

$$\delta_V([x_1l_1,x_2l_2],\sigma)=\begin{cases}[\delta(x_1,\sigma)l_1^{'},x_2l_2]\\ [x_1l_1,\delta(x_2,\sigma)l_2^{'}]\\ [\delta(x_1,\sigma)l_1^{'},\delta(x_2,\sigma)l_2^{'}];\end{cases}$$

3) if $\sigma \in \Sigma_c$,

a. if $\delta(x_0,s) \notin X_m^1$, then

$$\delta_V([x_1l_1,x_2l_2],\sigma)=[\delta(x_1,\sigma)CN,\delta(x_2,\sigma)CN];$$

b. if $\delta(x_0,s) \in X_m^1$, then

$$\delta_V([x_1l_1,x_2l_2],\sigma)=[\delta(x_1,\sigma)CF,\delta(x_2,\sigma)CF].$$

A verifier state $x_V=[x_1l_1,x_2l_2] \in \mathrm{X}_V$ is: **CF-certain** if $l_i="CF"$; **normal** if $l_i="N"$ or $l_i="CN"$; and **CF-uncertain** if $"N" \in l_1$ and $"F" \in l_2$ (or $"F" \in l_1$ and $"N" \in l_2$) or $l_i="F"$, where $i \in \{1,2\}$.

Theorem 2: Let G_S be the simplified model of G, there be no 2-class marked states in G_S, and V_G be the verifier of G_S. L generated by G is not actively predictable w.r.t. P, Σ_f and Σ_c iff there exists at least a CF-uncertain state in V_G.

Proof: **Necessity**: Assume that L is not actively predictable w.r.t. P, Σ_f and Σ_c. By contradiction it will be shown that there exists at least a CF-uncertain state in V_G. If there is no any CF-uncertain state in V_G, then the states in V_G are either normal or CF-certain, which means that as long as there are failure events in the system, then failure events must be predictable and the system must have executed controllable events before occurrences of the failure events. According to **definition 2**, L is actively predictable w.r.t. P, Σ_f and Σ_c. Clearly, this is a contradiction to the intended hypothesis. Consequently, if L is not actively predictable w.r.t. P, Σ_f and Σ_c, there exists at least a CF-uncertain state in V_G.

Sufficiency: Assume that there is a CF-uncertain state $x=[x_il_i,x_jl_j]$ in V_G. Case 1: suppose $"N" \in l_i$ and $"F" \in l_j$, then there exist strings $s_1,s_2 \in L$ and $P(s_1)=P(s_2)$ such that $\delta(x_0,s_1)=x_i(x_i \notin X_m^1)$ and $\delta(x_0,s_2)=x_j(x_j \in X_m^1)$. This means there is a failure event and its occurrence cannot be predicted according to the observation of s_1 or s_2. So by **Definition 1**, it is known that L is not predictable w.r.t. P, Σ_f and Σ_c. Thus, L is not actively predictable w.r.t. P, Σ_f and Σ_c. Case 2: suppose $l_i="F"$ and $l_j="F"$, then there exists a string $s \in L$ such that $\delta(x_0,s)=x_i(x_i \in X_m^1)$, $\delta(x_0,s)=x_j(x_j \in X_m^1)$, and $s \bigcap \Sigma_c=\phi$. According to **definition 2**, L is not actively predictable w.r.t. P, Σ_f and Σ_c.

5.2 A verification Algorithm

Algorithm 1 (Checking the active predictability of DESs):

Step 1: Construct non-failure automaton G_N.

$G_N=G \times A_N$ models the normal behaviour of G, where A_N is composed of a single state N with a self-loop labelled with all events in $\Sigma \setminus \Sigma_f$.

Step 2: Construct simplified model G_S.

According to G_N, construct simplified model of G_S.

Step 3: Check whether there are a 2-class marked states in G_S.

If there are, it is known that the occurrences of failure events in language L cannot be actively predictable by **Theorem 1**, i.e., L is not actively predictable w.r.t. P, Σ_f and Σ_c; Otherwise, perform the next step.

Step 4: Based on G_S, construct verifier automaton V_G.

Step 5: Check if there are CF-uncertain states in the V_G.

If the answer is yes, then L is not actively predictable w.r.t. P, Σ_f and Σ_c according to **Theorem 2**; Otherwise, L is actively predictable.

Now, discuss the computational complexity of the algorithm. Let $|X|$ be the number of states of X; $|\Sigma|, |\Sigma_f|$ be the number of events of $|\Sigma|$ or $|\Sigma_f|$.

For the first step, since the number of states of G is $|X|$, and the number of feasible transitions from every state is $|\Sigma|\text{-}|\Sigma_f|$, the construction of G_N takes $O(|X|(|\Sigma|-|\Sigma_f|))$ time. The time complexity of step 2 and step 3 is $O(|X_N|(|\Sigma|-|\Sigma_f|))$. The number of states of V_G is in the worst case equal to $2\times|\mathrm{X}_N|^2$. Moreover, the maximum number of transitions of states in the V_G is equal to $|\Sigma|+(|\Sigma|\text{-}|\Sigma_f|)$. The time complexity of step 4 and step 5 is $2|X_N|^2\times(|\Sigma|+(|\Sigma|\text{-}|\Sigma_f|))$. So the overall computational complexity of the approach is polynomial.

5.3 Illustrative Examples

Example 2: Consider the plant G shown in Fig.2, where $\Sigma_o = \{\alpha, \beta, \gamma, \tau\}$, $\sigma_f \in \Sigma_f$ is a failure event and state 1 is the initial state. Suppose γ is a controllable event (i.e. $\gamma \in \Sigma_c$). In the following, the active predictability of G will be verified by Algorithm 1.

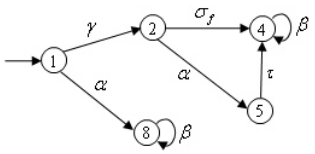

Fig. 2. Plant G of Example 2.

The first step of the algorithm is to obtain single state automaton A_N and to compute the automaton G_N, which are shown in Fig.3 and Fig.4, respectively.

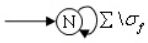

Fig. 3. Automaton A_N.

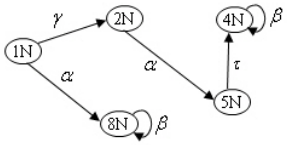

Fig. 4. G_N of Example .

Then, G_S is constructed as that in Fig.5. As Fig.5 shows, state 2 is a 2-class marked state. So according to **Theorem 1**, it is known that L of G is not actively predictable w.r.t. P, Σ_f and Σ_c.

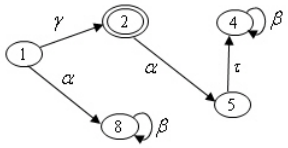

Fig. 5. G_S of Example 2.

Example 3: Consider again the plant G shown in Fig.1 and let γ be a controllable event (i.e. $\gamma \in \Sigma_c$). In the following, the result that L is actively predictable w.t.r. P, Σ_f and Σ_c will be verified by Algorithm 1.

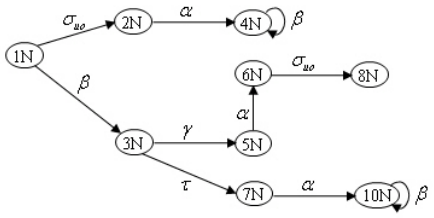

Fig. 6. G_N of Example 3.

Firstly, G_N and G_S are constructed as that in Fig.6 and Fig.7, respectively.

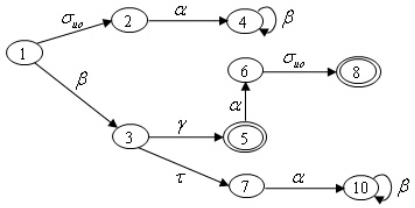

Fig. 7. G_S of Example 3.

In the G_S, state 5 and state 8 are both 1-class marked states and there are no 2-class marked states.

Construct verifier V_G by the G_S, depicted in Fig.8.

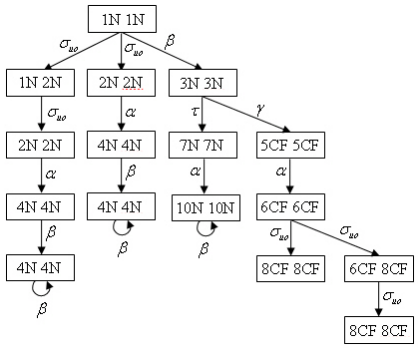

Fig. 8. Verifier V_G of Example 3.

Obviously, there is no CF-uncertain state in the verifier V_G. By **Theorem 2**, it is known that the language L generated by G is actively predictable w.r.t. P, Σ_f and Σ_c.

Example 4: Consider the system G shown in Fig.9, where $\alpha, \beta, \gamma, \tau$ are observable events, σ_{uo} is an unobservable event, σ_f is a failure event and γ is a controllable event (i.e. $\gamma \in \Sigma_c$). Next to verify the active predictability of the system.

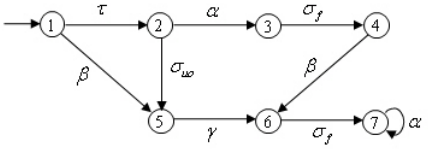

Fig. 9. G of Example 4.

G_N and G_S are firstly constructed as that in Fig.10 and Fig.11, respectively. As Fig.11 shows, state 3 and 6 are 1-class marked states, also there is no 2-class marked state in the G_S.

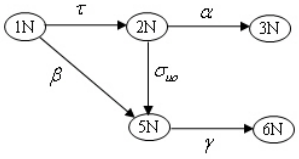

Fig. 10. G_N of Example 4.

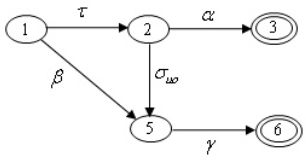

Fig. 11. G_S of Example 4.

Verifier V_G is constructed as that in Fig.12. As Fig.12 shows, there is a CF-uncertain state "6F6F" in V_G. So according to **Theorem 2**, it is known that the language L generated by G is not actively predictable w.r.t. P, Σ_f and Σ_c.

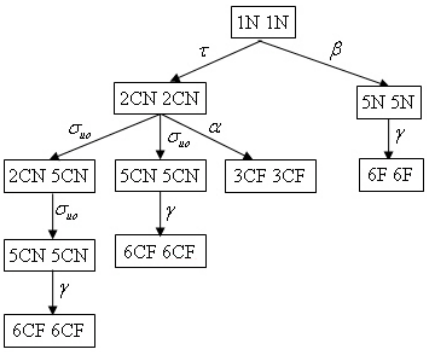

Fig. 12. V_G of Example 4.

Example 5: Consider the automatic cargo transport system shown in Fig.13, which describes the situation that a robot carries a batch of goods from warehouse A to warehouse B. In the figure, the red dashed line indicates the given route direction and black squares and dots represent obstacles and states, respectively. At first, the robot goes straight following the direction of the given route, but if obstacles or borders are predicted during the process, the robot will adjust the route appropriately.

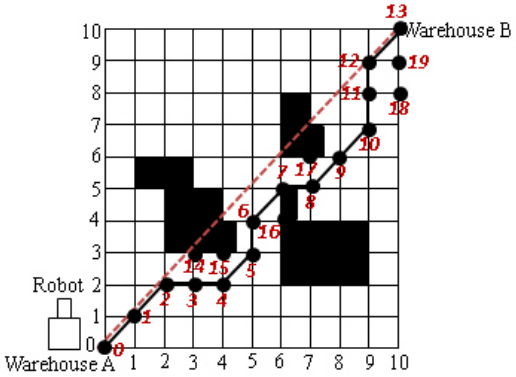

Fig. 13. Automatic cargo transport system.

The model of the system is shown in Fig.14, where events are three execution instructions:

α : walk following the direction of the given route;

β : turn right and goes straight;

γ : turn left and goes straight;

Here, assume that the direction of the given route can be adjusted appropriately, i.e. α is a controllable event.

If the robot executes wrong instructions, it is possible to encounter obstacles or borders while walking., i.e., the failure event σ_f occurred.

Initially, the robot is in the warehouse A (i.e., state 0. When the robot receives instruction α, then it will carry the goods and go straight along the direction of the given route until arrives state 1. From state 1, continue to move forward to state 2 with instruction α. In state 2, if the robot executes instruction α, it will reach state 14. In state 14, it will encounter obstacles as it moves on, that is, it will reach a failure state (i.e. state 20). In state 2, if the robot can predict that it will encounter obstacles when it continues to go straight, it will turn right and go straight to state 3 with instruction β. In state 3, if the robot continues to execute instruction α, it will reach state 15. Similarly, if the robot go straight from state 15, it may run into obstacles, that is, it will reach the failure state (i.e. state 20). However, if it can predict that there are obstacles ahead, the robot will turn right and go straight to state 4 according to instruction β. Then, go from state 4 to state 5 with instruction α. In state 5, if the robot continues to execute instruction α, it will reach state 16. In state 16, it will encounter obstacles and cannot go straight, that is, it will reach the failure state (i.e. state 20). But in state 5, if it predict that there are obstacles ahead, then the robot will turn left and go straight to state 6 according to instruction γ. After that, the system execution process is similar to the previous one.

In the following, the active predictability of the system will be verified by Algorithm 1.

Firstly, compute automaton $G_N = G \times A_N$, which is described by Fig.15.

Next is to build a simplified model G_S of G, shown in Fig.16.

There are no 2-class marked states in G_S, So the verifier automaton V_G is computed by G_S, shown in Fig.17.

As Fig.17 shows, there is no any CF-uncertain state in the verifier V_G, so the L generated by G is actively predictable w.r.t. P, Σ_f and Σ_c.

Since the L is actively predictable, the supervisory controller can control system behaviour when the system is actually running (as shown in Table 1). For example, if the system executes the $\alpha\alpha$ event, continuing to execute the α event will inevitably cause failure event σ_f to occur. To avoid the occurrence of the failure event, the supervisory controller can disable the event α. Then the next event that will be executed is the event β. In this way, the occurrence of the failure event is avoided.

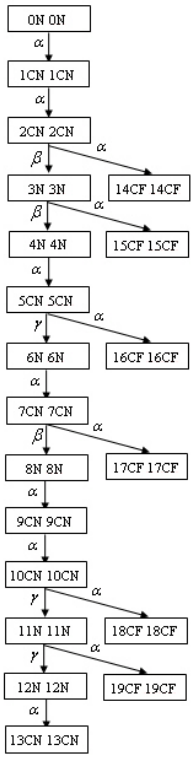

Fig. 17. V_G of Example 5.

Table 1. The behavior of supervisory controller.

Strings that may cause failures	Strings after control
$\alpha\alpha(\alpha\sigma_f)$	$\alpha\alpha\beta$
$\alpha\alpha\beta(\alpha\sigma_f)$	$\alpha\alpha\beta\beta$
$\alpha\alpha\beta\beta\alpha(\alpha\sigma_f)$	$\alpha\alpha\beta\beta\alpha\gamma$
$\alpha\alpha\beta\beta\alpha\gamma\alpha(\alpha\sigma_f)$	$\alpha\alpha\beta\beta\alpha\gamma\alpha\beta$
$\alpha\alpha\beta\beta\alpha\gamma\alpha\beta\alpha\alpha(\alpha\sigma_f)$	$\alpha\alpha\beta\beta\alpha\gamma\alpha\beta\alpha\alpha\gamma$
$\alpha\alpha\beta\beta\alpha\gamma\alpha\beta\alpha\alpha\gamma(\alpha\sigma_f)$	$\alpha\alpha\beta\beta\alpha\gamma\alpha\beta\alpha\alpha\gamma\gamma$

6. ACTIVE PREDICTABILITY OF DESs WITH MULTIPLE FAULT TYPES

Let $\Sigma_f = \Sigma_{f_1} \cup \Sigma_{f_2} \cup ... \cup \Sigma_{f_r}$ be a partition of the set of failure events, where r denotes the number of fault types, and let $\prod_f$ denote this partition. The active predictability of L with respect to $\prod_f$ is equivalent to the active predictability of L with respect to each fault type separately, as ensured by the following result.

Theorem 3: The language L of system G is actively predictable w.r.t. P, $\prod_f$ and Σ_c, if and only if L is actively predictable w.r.t. P, Σ_{f_i} and Σ_c for each $i = 1, 2, ..., r$.

Poof: it is easily obtained by **Definition 2**.

7. CONCLUSIONS

This paper concerned the problem of active failure prediction. Starting from the standard definition of predictability (Genc, 2009) of DESs, which dealt with the problem of predicting the occurrences of failure events using the available system observations, the new notion of active predictability was introduced and the verification of active predictability in the framework DESs was investigated. A verifier was constructed, and based on it the necessary and sufficient condition of active predictability was presented. Moreover, a polynomial-complexity algorithm for verifying active predictability was developed. In future work, the active predictability of fuzzy DESs or stochastic DESs will be considered.

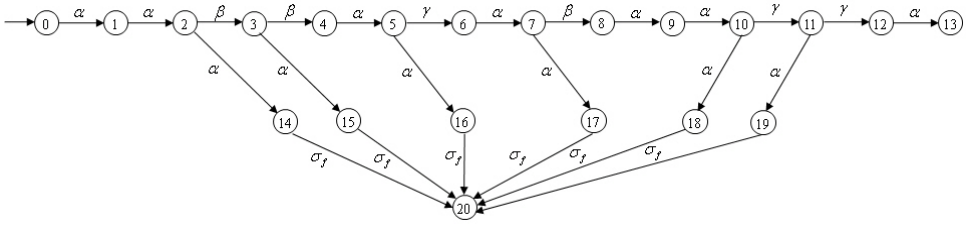

Fig. 14. The model G of the automatic cargo transport system.

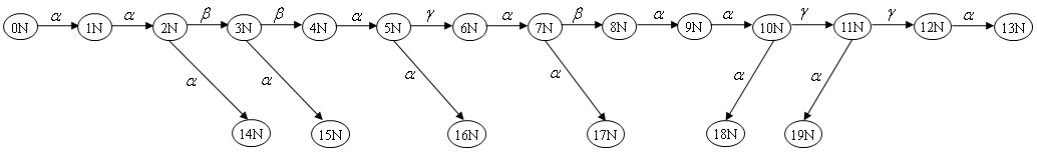

Fig. 15. G_N of Example 5.

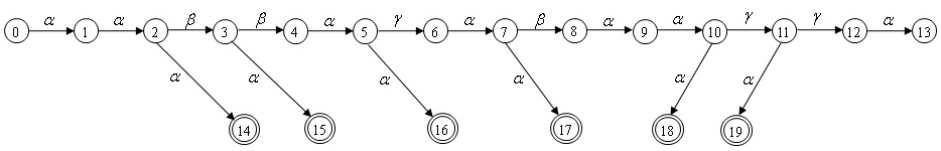

Fig. 16. G_S of Example 5.

ACKNOWLEDGEMENTS

This work was supported by "the National Natural Science Foundation of China" (grand number 61673122) and "the Natural Science Foundation of Guangdong Province" (grand number 2019A1515010548) of China.

REFERENCES

Briones, L. B. and Madalinski, A. (2012). Bounded predictability for faulty discrete event systems. *in 2011 30th International Conference of the Chilean Computer Science Society (SCCC 2011)*, 142–146.

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

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.

Chen, J. and Kumar, R. (2014). Failure prognosability of stochastic discrete event systems. *in Proc. 2014 Amer. Control Conf*, June, 2041–2046.

Chen, Z., Lin, F., Wang, C., Wang, L. and Xu, M. (2014). Active diagnosability of discrete event systems and its application to battery fault diagnosis. *IEEE Trans. on Control System Technology*, 22(5), 1892–1898.

Deng, W. and Qiu, D. (2017). State-based decentralized diagnosis of bi-fuzzy discrete event systems. *IEEE Transactions on Fuzzy Systems*, 25(4), 854–867.

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

Grastien, A. (2015). Interval predictability in discrete event systems. *arXiv:1508.00683v1[cs.SY],* Aug., 1–14.

Jeron, T., Marchand, H., Genc, S. and Lafortune, S. (2008). Predictability of sequence patterns in discrete event systems. *in Proc. 17th IFAC world conference*, July 6-11, 537–543.

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

Liu, F. (2014). Polynomial-time verification of diagnosability of fuzzy discrete event systems. *Science China Information Sciences*, 57(6), 1–10.

Liu, F. (2019). Predictability of failure event occurrences in decentralized discrete-event systems and polynomial-time verification. *IEEE Trans. on Automation science and engineering*, 16(1), 498–504.

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.

Masopust, T. and Yin, X. (2019). Complexity of detectability, opacity and A-diagnosability for modular discrete event systems. *Automatica*, 101, 290–295.

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.

Sampath, M., Lafortune, S., and Teneketzis, D. (1998). Active diagnosis of discrete-event systems. *IEEE Trans. on Automatic Control*, 43(7), 908–929.

Thorsley, D. and Teneketzis, D. (2007). Active acquisition of information for diagnosis and supervisory control of discrete-event systems. *Journal of Discrete Event Dynamic Systems*, 17, 531–583.

Takai, S. (2012). Robust failure prognosis of partially observed discrete event systems. *in Proc. 2012 Amer. Control Conf., Montreal, Canada, June*, 27-29.

Takai, S. and Kumar, R. (2011). Inference-based decentralized prognosis in discrete event systems. *IEEE Trans. on Automatic Control*, 56(1), 165-171.

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.

Yao, L., Feng, L. (2016). Fault diagnosis and fault tolerant control for non-Gaussian time-delayed singular stochastic distribution systems. *International Journal of Control, Automation and Systems*, 14(2),435–442.

Yin, X. and Li, Z. (2016). 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. (2019). Decentralized fault prognosis of discrete-event systems using state-estimate-based protocols. *IEEE Transactions on Cybernetics*, 49(4), 1302-1313.

Zhao, R., Liu, F. and Liu, Z. (2017). Relative diagnosability of discrete-event systems and its opacity-based test algorithm. *International Journal of Control, Automation and Systems*, 15(4), 1693–1700.

Zhao, R., Liu, F. and Tan, J. (2019). Relative predictability of failure event occurrences and its opacity-based test algorithm. *International Journal of Control*, 92(7), 1600–1608.