Proceedings of the 3rd International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/6/20230801

Diagnosability verification of discrete event systems

Jingyu Cui[1,†] and Yuze Xia[2,3,†]

[1]Bell Honors School, Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu, 210023, China
[2]Computer Science Department, University of Buffalo, Buffalo, NY 14260, United States

[3]yuzexia@buffalo.edu
[†]These authors contributed equally

Abstract. Fault diagnosis is one of the key topics in the study of computer program operation and automatic control of large complex systems these days. It can be used in wide-spanning areas. As early as the last century, researchers started to study diagnosis structures and made a series of progress. Moreover, the problem of diagnosability verification of a system received much attention from many researchers. Therefore, in this paper, a discrete event system (DES) is proposed and a diagnoser is constructed as an automaton model to verify the diagnosability of a given system. A method is proposed to test if a given system is diagnosable under the discrete event system structure. The states of a system are classified into three categories, and a diagnoser structure with basic algorithms and functions is defined to verify diagnosability. The proposed diagnoser structure can better capture the behavior of the system and verify its diagnosability.

Keywords: Diagnosability verification, fault diagnosis, discrete event system, automaton

1. Introduction

In recent years, fault diagnosis techniques of discrete event systems (DESs) have been applied to wide-spanning areas such as power, factory, and aircraft systems. From a safety standpoint, modern industrial systems require high reliability and correctness of production monitoring systems. As it has been a growing concern when modern production systems became more complex, robustness for diagnosers is increasingly important.

In Sampath et al. [1], a diagnoser built on finite state automata with formalized definitions is proposed, of which formal syntax of the diagnoser system became the basic structure. Many works based on different form-driven methods were implemented to create newer and more efficient models for the diagnosis of DES. They are classified into diverse categories, for example, formalism modeling for which diagnosers were constructed in formalizing methodology, e.g., Hierarchical States Machine implementation [2-3], and models based on Petri net that is a powerful mathematical representation for distributed systems [4-5]. The model includes the classification of especially unobservable acts based on observed sequential behaviors, e.g., implementation of diagnostics of intermittent faults [6].

The diagnosis of a fault occurrence is determined by the observed sequence of events, and unexpected system events would be caused by the occurrence of a fault. Studying those behaviors has helped to characterize them more precisely to materialize faults so that more diagnosers have better accuracy and

DOI: 10.54254/2755-2721/6/20230801

completeness to a more certain problem. In Discrete Event Systems, faults have their identities. They could be classified into three types due to their distinct patterns: Permanent Faults, which will cause irreversible results in a system and make it acts faulty after it occurs; Drift-like (Incipient) faults, which show some gradual process abnormal behaviors increasing; and Intermittent faults.

Diagnosis of faults is to decide whether a fault has occurred when a system runs online. To achieve It is necessary to characterize faults and label system states "normal," "uncertain," or "faulty" to implement a diagnoser, and faulty functioning should be isolated as soon and accurately as possible. Diagnosability is also an important property in diagnosis. If any fault could be perceived within finite observable sequential events, this fault is called diagnosable. Many models can test whether a fault is *diagnosable* [1], [6]. It is an important topic to exploit the space of diagnosability because it is an essential property of a system for diagnosis. In this paper, the diagnosability verification problem in the automata framework will be discussed.

This paper can be organized as follows. In Section 2, some basic notions used in the paper are recalled. In Section 3, the diagnosability verification problem is formulated, and in Section 4, the verification method is proposed. Finally, Section 5 concludes the paper.

2. Preliminaries

In this section, we review some basic notations and definitions for approaches to fault diagnosis in discrete event systems. The system to be diagnosed is modeled as a deterministic finite state automaton (DFA).

A deterministic finite state automaton (DFA) over an alphabet A is a 4-tuple structure $G = (X, E, \delta, x_0)$ where:

--X is a finite set of states of the system;

--E is a finite(non-empty) set of events;

--δ is the transition function, where $\delta: X \times E^* \to X$;

--$x_0 \in X$ is the initial state.

The finite language generated by G is denoted as $L(G)$. $L(G) = \{s \in E^*: \delta(x_0, s)!\}$ represents the set of a given system containing all possible sequences of executions or events that occurred from the state x_0 . In this definition, "!" means "is defined". We hypothesize that system G is live. This means that $\forall x \in X, \exists a: \delta(x, a)!$, i.e., G cannot get to a state where no event occurs. In this paper, the length of a sequence s is indicated by the symbol |s|.

For a partially-observed deterministic finite state automaton, it is assumed some occurrences of the events set E could not be observed directly. An approach to solve is to partition it into binary sets: the observable events set E_o and unobservable events set E_{uo}, i.e., $E = E_o \cup E_{uo}$. The unobservable event set E_{uo} continues to be further partitioned into normal unobservable events set E_n and fault events set E_f, i.e., $E_{uo} = E_n \cup E_f$. Fault events are classified into different types, i.e., $E_f = \{f_1, f_2, f_3, \ldots, f_n\}$ while $f_1, f_2, f_3, \ldots, f_n$ refer to types of faults. For sake of simplicity, it is regarded that $E_f = \{f\}$.

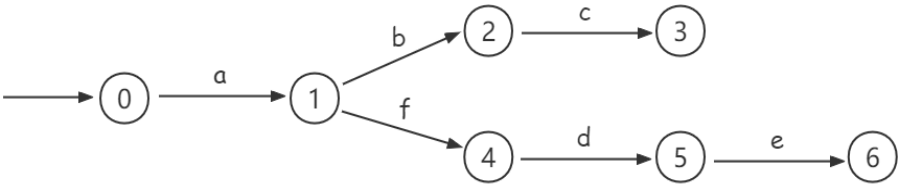

Figure 1. Automaton G.

For example, a partially-observed automaton $G = (X, E, \delta, x_0)$ is shown in Figure 1. Then we have:

--$Q = \{0,1,2,3,4,5,6\}$;

DOI: 10.54254/2755-2721/6/20230801

--$E = E_o \cup E_{uo}$, $E_o = \{a, b, c, d, e\}$, $E_{uo} = \{f\}$;

--$\delta : \delta(0, a) = \{1,4\}$, $\delta(1, b) = \{2\}$, $\delta(2, c) = \{3\}$, $\delta(1, d) = \delta(4, d) = \{5\}$, $\delta(5, e) = \{6\}$;

--$x_0 = 0$.

Once a fault has occurred, we assume that the system is subject to faults with occurrences being diagnosed within a finite number of steps. To determine whether the fault has occurred, we focus on an $s \in L(G)$ from which we can observe a word w contains fault f .

A projection function $P: E^* \to E_o^*$ used to capture the partially unobservable events in a sequence of events is defined as:

$$P(\varepsilon) = \varepsilon;$$

$$P(a) = a \quad if \quad a \in E_o;$$

$$P(a) = \varepsilon \quad if \quad a \in E_{uo}.$$

This projection function P can erase the events set E_{uo}, transferring an $a \in E^*$ produced by the deterministic finite automaton to an observed $w \in {E_o}^*$, i.e., $P(a) = w \in {E_o}^*$. Accordingly, the inverse projection function is defined as $P^{-1} = \{a \in E^* | P(a) = w\}$, meaning the set of a that can produce w.

To classify the states of automaton due to the result of our observation, the diagnosis function φ is introduced:

$$\varphi: E_0^* \to \{N, F, U\}.$$

Case 1: $\varphi(w) = \{N\}$, if for all $s \in P^{-1}(w)$ it holds that $f \notin s$.

In this case, there is no fault observed from s which is related to the observed word w, meaning no fault has occurred.

Case 2: $\varphi(w) = \{F\}$, if for all $s \in P^{-1}(w)$ it holds that $f \in s$.

In this case, there is a fault observed from s which is related to the observed word w, meaning fault f has occurred.

Case 3: $\varphi(w) = \{U\}$, if there are $s, s' \in P^{-1}(w)$ such that $f \in s$and $f \notin s'$.

In this case, there are both s and s' related to the observed word while one observes fault $f$and another does not. This means fault f might occur or not, therefore it is uncertain.

A label function $L: \{N, F\} \times E^* \to \{N, F\}$ will yield a "normal" label only when: s ∈ E* and $f \notin s \Rightarrow L(N, s) = N$. Otherwise, it would yield a "fault" label.

3. Problem formulation

Fault diagnosis of discrete event systems (DESs) gained massive concern during the last several decades. In practice, the discrete event systems' diagnosis problem is mainly to identify which faults (unobservable events) can be explained for a given sequence of observed events on the system-based based model. The diagnosis tasks are mainly to detect a fault when it occurred and isolate the occurrence of a fault after its observation.

The two main topics discussed in DES-based diagnosis are online diagnosis and offline diagnosability analysis. Online diagnosis involves inferring the occurrence of a scheduled fault from observed system behavior. Diagnosability refers to the ability to provide evidence and judgment. Thus, the goal of diagnosability analysis of a given system is to precisely determine whether any fault can be observed and identified within a finite delay after it occurred.

In this paper, we assume the system remains in an irreversible fault state once a fault has occurred. Each automaton G considered here meets the two hypotheses below:

(1) The language $L(G)$ is live, i.e., $\forall x \in X, \exists a: \delta(x, a)!$, which means G cannot reach a state that has no out paths; and (2) G is a finite state automaton, which does not exist in any cyclic path that involves unobservable events only.

DOI: 10.54254/2755-2721/6/20230801

Since diagnosis is the process of inferring the causes of a given system's behavior by a given set of observations, there raises one question: is it possible for a diagnosis to infer accurate and sufficient run-time information on the behavior of the occurrences in the observed system?

To capture whether these occurrences could be detected within a finite number of steps, the notion of diagnosability is to be defined. We first define $\Psi(G)$ as the set of words of L that ends with a fault f. Now given a system $G = (X, E, \delta, x_0)$, G is said to be diagnosable if the following condition holds:

$$\big(\forall s \in \Psi(G)\big)(\exists n \in IN)\ \big(\forall st \in L(G)\big)[|t| \geq n \Rightarrow Diag]$$

in which the diagnosability condition Diag is

$$\forall w \in P^{-1}\big(P(st)\big) \Rightarrow f \in w.$$

In this definition, s is any set of words ending with a fault from E_f, and t is any sufficiently finite long continuation of s. The diagnosability condition Diag means any word that generates the same observable events as the word st does would have to contain a fault from E_f. This ensures us to detect whether fault f has occurred after a finite number of steps n after s, as we would not observe a word with no fault, and thus we say it is diagnosable.

Diagnosability is a significant attribute in diagnosing fault activities. Informally, it refers to the capability to accurately project fault behavior by partial-observed executions within a finite delay after a fault may have occurred.

Diagnosability is closely related to people's lives and is used in a lot of areas, e.g. in communication systems [7] for the users could monitor and troubleshoot networks to ensure security and stability when communicating; in web services [8], businesses processes could have failed so fault diagnosis should necessarily be introduced and recover from their effects; in the aeronautical field [9], one important application is to diagnosis, troubleshoot, and localize besides guaranteeing the operational performance of aircraft; in traffic management systems [10] or automated highway systems [11], an observer is significantly needed to identify those unexpected situations and take measures to avoid traffic accidents. Overall, diagnosability plays a vital role in people's daily life.

To analyze diagnosability, diagnosability verification with a systematic approach of the discrete event system is necessarily proposed. In this article, we propose a method for verifying the diagnosability of a system that is based on a specialized deterministic finite automaton, the diagnoser, derived from the original model. We assume it consists of a system-specific observer associated with a token function, and each state in that observer is marked with a token, suggesting whether the state is reached by a faulty execution, i.e., whether any spectacular fault has occurred during the execution.

4. Diagnosability verification

In this section, we first define the diagnoser $G_d = \big(X_d, E_o, \delta_d, x_{d_0}\big)$ of a system $G = (X, E, \delta, x_0)$.

Remark 1. $U(e, x)$ is the function that generates all possible unobserved events $\sigma \in E_{uo}^*$ which could happen after an observed event $e \in E_o$ with a known start state x. It is defined as follows:

$$U(e, x) = \{\sigma \in \mathrm{E}_{\mathrm{uo}}^* \mid \delta(\mathrm{x}, \mathrm{e}\sigma)\ is\ defined\}.$$

Remark 2. The behavior of labeling states is the same as the conjunction operator in Boolean Algebra: a state will be marked as "normal" if and only if all its preceding transitions didn't produce any fault label F. In other words, a state goes through a trace of events all labeled as normal. Thus, we can apply logical conjunction $\wedge$ over labels:

$$\mathrm{N} \wedge N = N;$$

$$\mathrm{N} \wedge F = F;$$

$$F \wedge N = F;$$

$$\mathrm{F} \wedge F = F.$$

DOI: 10.54254/2755-2721/6/20230801

For example:

$$N \wedge F \wedge \mathrm{N} = (N \wedge F) \wedge \mathrm{N} = \mathrm{F} \wedge \mathrm{N} = \mathrm{F}.$$

The set of states of a diagnoser X_d is defined as:

$$X_d \subseteq 2^{X \times \{N,F\}}.$$

This means each state x is denoted with a label N(normal) or F(fault) in this form $x' = (x, l)$ and $l \in \{N, F\}$. The process that generates X_d from the original G equation is similar to constructing the equivalence DFA from NFA in automata theory (see Figure 1, 2), the formal definition is defined as follows:

$$X_d = \{\{x_0, N\}\} \cup \{\{(x_d, l_d) \mid (x, l) \in X_d, e \in E_o, x_d = \delta(x, e), \sigma \in U(e, x), l_d = l \wedge L(e\sigma)\}\}.$$

The event of a diagnoser is defined as E_o according to the definition of diagnosis since diagnosers only make decisions based on observable events, thus, we define the events of G_d simply as the set of observable events E_o. Another element, x_{d_0}, refers to the initial state of a diagnoser, and it is a two-tuple (x_0, N) in which x_0 with a "normal" flag N. δ_d is the set of transition functions of a diagnoser, which we could formally define as:

$$\delta_d(x_d, e) = \{(x', \mathrm{L}(\mathrm{l}, \mathrm{e}\sigma)) \mid (x, l) \in x_d, x' \in \delta(x, e\sigma) \text{ when } \sigma \in U(e, x)\}.$$

X_d and δ_d are generated by its formal definition above (see *Lemma 1,2*); also, given a system G, a level order traversal algorithm to construct a diagnoser $G_d = (X_d, E_o, \delta_d, x_{d_0})$ is provided as follows:

Algorithm 1: Construction of diagnoser G_d.

Given a system $G = (X, E, \delta, x_0)$
$x_{d_0} = (\mathrm{x}_0, N)$
$X_d = \{\{x_{d_0}\}\}, X'_d = \{\{x_{d_0}\}\}$

while x in X'_d:
    for each $e \in E_o$:
        $x' = \delta_d(x, e)$
        $X_d = X_d \cup \{x'\}$, $X'_d = X'_d \cup \{x'\}$
        $\delta_d(x, e) = x'$ (Set the input-output mapping for transition functions)
    end
    $X'_d = X'_d \setminus \{X\}$
end

Let us consider the following example. Given a system G in Figure 2, where $E_{uo} = \{f, g\}, E_f = \{f\}$. We can derive the corresponding G_d according to G, which is shown in Figure 3.

DOI: 10.54254/2755-2721/6/20230801

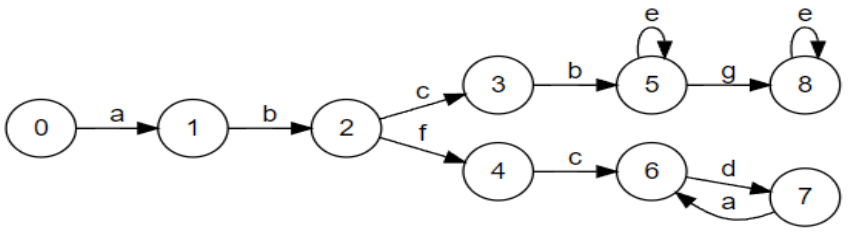

Figure 2. DFA of a system G.

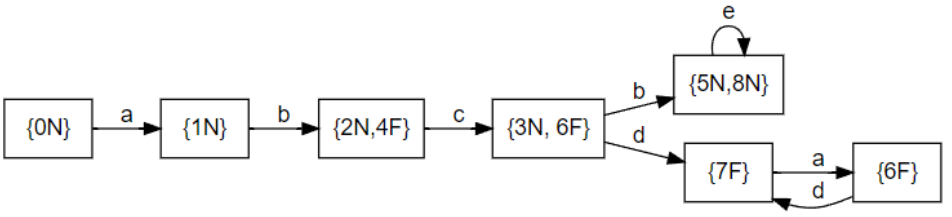

Figure 3. DFA of diagnoser of G.

For each state of diagnoser, the diagnostic types are:

1. Normal: $\forall x_{d_i} = (\mathrm{x}, \mathrm{l}) \in x_d,\ x_d \in X_d,\ l = N.$
2. Faulty: $\forall x_{d_i} = (\mathrm{x}, \mathrm{l}) \in x_d,\ x_d \in X_d,\ l = F.$
3. Uncertain: $\exists x_{d_i} = (\mathrm{x_i}, \mathrm{l_i}), x_{d_j} = (\mathrm{x_j}, \mathrm{l_j}) \in x_d, x_d \in X_d, l_i = N$ and $l_j = F.$

So far, the model of diagnoser for a plant has been defined. By using the diagnoser constructed, one can verify the diagnosability of a plant model. If the diagnoser of a plant does not contain any cycle containing uncertain states, then the plant is diagnosable.

5. Conclusion

Due to the great demand for systems capable of detecting faults in many areas, this paper introduces a diagnoser structure that can be used to verify the diagnosability of a system. A system is diagnosable if a fault can be determined after observing a finite number of events. Firstly, an automaton model with formal definitions is introduced to model the system. Based on the model, an algorithm is proposed to compute a diagnoser that is used to verify diagnosability. In future work, the complexity of verifying the diagnosability can be reduced.

References

[1] Sampath M, Sengupta R, Lafortune S, Sinnamohideen K and Teneketzis D 1995 Diagnosability of Discrete Event Systems *IEEE Trans. Automat. Contr.* **40** 1555-75

[2] Su R and Wonham W 2006 Hierarchical Fault Diagnosis for Discrete-Event Systems under Global Consistency *Discrete Event Dynamic Systems* **16** 39-70

[3] Paoli A and Lafortune S 2005 Safe diagnosability for fault-tolerant supervision of discrete-event systems *Automatica* **41** 1335-47

[4] Prock J 1991 A new technique for fault detection using Petri nets *Automatica* **27** 239-45

[5] Ma Z, Yin X and Li Z 2021 Marking diagnosability verification in labeled Petri nets *Automatica* **131** 109713

DOI: 10.54254/2755-2721/6/20230801

[6] Contant O, Lafortune S and Teneketzis D 2004 Diagnosis of Intermittent Faults *Discrete Event Dynamic Systems: Theory and Applications* **14** 171-202
[7] Moreira V M, Jesus C T and Basilio C J 2011 Polynomial time verification of decentralized diagnosability of discrete event systems *IEEE Trans. Automat. Contr.* **56** 1679-84
[8] Yan Y, Cordier M, Pencolé Y and Grastien A 2005 Monitoring Web Service Networks in a Model-based Approach *Proceedings - Third European Conference on Web Services, ECOWS2005* pp 192-203
[9] Ghelam S, Simeu-abazi Z, Derain J, Feuillebois C, Vallet S and Glade M 2006 Integration of health monitoring in the avionics maintenance system *IFAC Proceedings* **39** 1449-1454
[10] Tomlin C, Pappas G J and Sastry S 1995 Conflict resolution for air traffic management: A study in multi-agent hybrid systems *IEEE Trans. Automat. Contr.* **42** 509-21
[11] Varaiya P 1993 Smart cars on smart roads: problems of control *IEEE Trans. Automat. Contr.* **38** 195-207