2017 American Control Conference
Sheraton Seattle Hotel
May 24–26, 2017, Seattle, USA

Asynchronous Fault Diagnosis of Discrete Event Systems

Alejandro White, *Student Member, IEEE,* Ali Karimoddini, *Senior Member, IEEE*

***Abstract*—In this paper, a novel diagnoser is developed for diagnosing Discrete Event Systems (DESs) to detect, identify, and locate the occurred faults. In contrast to existing diagnosis techniques, the proposed diagnoser is capable of diagnosing faults that occur prior to and/or after the diagnoser's activation; thus allowing the diagnoser and the understudied system to be initialized asynchronously. Illustrative examples are provided to explain the construction of the proposed diagnoser.**

1. Introduction

Fault diagnosis determines the condition of a system using the symptoms (observed behavior) of a system. Despite progresses on diagnosis techniques, today's complex involved systems have increased the difficulty of applying the existing developed fault diagnosis tools. This creates a need for developing novel diagnosis techniques that take into consideration the ever increasing complexity of modern systems ranging from manufacturing and telecommunications to swarms of autonomous systems. Fault diagnosis is the act of detecting, isolating, and identifying a fault. An overview of fault diagnosis techniques is available in [1]–[4], which include fault tree analysis [5], [6], model based approaches [7], [8], expert systems [9], [10], template structures [11], [12], and bayesian networks [13].

In this paper we address fault diagnosis within Discrete Event System (DES) framework [14]–[16]. DES is capable of providing a pictorial abstract representation of a complex system (for the purpose of fault diagnosis) that allows for a more intuitive analysis of system behaviour, in contrast to differential and difference equations. A pictorial representation of a system allows the manned/automated diagnoser to quickly determine a system's behavior (faulty, or nonfaulty), with an understanding of the system's behaviour related to cause and effect, thus facilitating the analysis of a system. For example, pictorially representing a DES may allow to capture faulty and nonfaulty states as partitioned areas. Therefore, just by looking at which partition the system is in allows for an immediate conclusion of whether the system is operating in a normal or abnormal manner, while providing a visual trajectory of the system's behavior.

The authors are with the Department of Electrical and Computer Engineering, North Carolina Agricultural and Technical State University, Greensboro, NC 27411 USA.
Corresponding author: A. Karimoddini. Address: 1601 East Market Street, Department of Electrical and Computer Engineering North Carolina A&T State University Greensboro, NC, US 27411. Email: akarimod@ncat.edu *(Tel: +13362853847).*

Various capable techniques for automated fault diagnosis of systems modelled as Discrete Event Systems (DESs) have been developed over the course of research. One of the earliest form was an off-line diagnosis of a system introduced by [17]. Later in [18], an on-line diagnosis technique for DESs was developed and the concept of disposability of DESs was introduced. Diagnosability allows the system modeler to determine definitively whether all modelled system faults can or cannot be diagnosed within a finite number of post-fault occurring system observations. These presented techniques, developed for a centralized fault diagnosis, were later extended to decentralized [19], [20], modular/distributed [21], [22], learning-based approaches [23], robust and safe [24]–[26] diagnosis structures. A comprehensive review of fault diagnosis techniques for discrete event systems can be found in [27].

All of the aforementioned techniques share the quality of being cornerstones of fault diagnosis in DESs. However, to diagnose obscured fault correctly, they need to be synchronously initialized with the systems that they are monitoring. Synchronous initialization allows the systems to be in lock step with the systems that they are monitoring from the beginning of the monitored system's operation. This allows the diagnoser to form its diagnosis using an aggregation of system diagnostic observations from the system's initial powering up. Synchronous initialization provides the diagnoser a great amount of information to use for its diagnosis. This process, however, is not always an attractive option. For many situations, this can prove to be quite costly. In many other situations, it is not feasible to synchronously initialize the diagnoser and the system under diagnosis. Even if synchronously initialized, there is a chance that the diagnoser misses an observation (event) after the synchronous initialization. In this cases, the process has to start all over again.

To address this setback, we introduce a novel asynchronous diagnoser, that is capable of diagnosing a system fault without the requirement that it is synchronously initialized with the monitored system. This will lower the costs (time and money) of diagnosis in today's complex systems. Since the presented diagnoser is not synchronously initialized with the monitored system (it is activated after the system initialization), it misses out the information before the activation of the diagnoser. To address this problem, the proposed diagnoser, naturally starts with a conservative diagnostic estimation of the system's condition that include all possible situations. Observing the system's behavior, the diagnoser then narrows down its initial guess toward more accurate estimation.

978-1-5090-5992-8/$31.00 ©2017 AACC

The rest of the paper is organized as follows. Section 2, provides the necessary background and notations for DES modeling of the original system, and formulates the asynchronous fault diagnosis problem. In Section 3, an algorithm for constructing the asynchronous diagnoser is presented, accompanied by an illustrative example detailing the steps of the proposed algorithm. In Section 4, a discussion on asynchronous diagnosability. Section 5 concludes the paper.

2. Preliminaries and Background

The DES model of the monitored plant is a non-deterministic, finite state automaton signified by G, a four tuple,

$$G = (X, \Sigma, \delta, x_0) \tag{1}$$

whose elements include: the finite set of discrete system states, X, the finite set of system events, Σ, the state transition relation, δ, and the initial state of system, x_0. Pictorially, automaton G is a directed graph whose vertices (the states of the system, $x \in X$) are connected by directed edges. Each directed edge symbolizes a feasible transition between system states due to a system event, $e \in \Sigma$. The state transition relation, $\delta : X \times \Sigma \to 2^X$ (2^X is the power set of X), is a partial function which defines all transitions (directed edges) in G. The initial state x_0 is represented by an input arrow connected to a single state. (See, e.g., the automaton model shown in Fig. 1).

An event is a notable occurrence of an asynchronous discrete change in a system. The event set of system G consists of the union of two disjoint sets: $\Sigma = \Sigma_o \dot{\cup} \Sigma_u$, where Σ_o is the set of observable events, and Σ_u is the set of unobservable events. A sequence of one or more events is called a *string*. We use $e \in s$ to state that event e is an event in the composition of string s. The set of all possible finite strings formed from Σ is represented by Σ^* (the Kleene closure of Σ). The set Σ^* also includes the zero-length string ε. The length of a string t tells the number of events that compose the string, and is denote by $\|t\|$. A concatenation of strings s_1 and s_2, is shown by $s_1.s_2$. The state transition relation is recursively extended from domain $X \times \Sigma$ to domain $X \times \Sigma^*$ as follows:

- $\delta(x, \varepsilon) = x$
- $\delta(x, s.e) = \delta(\delta(x, s), e)$ $for\ any\ s \in \Sigma^*\ and\ e \in \Sigma$

All states $x_2 \in X$ are considered reachable from state $x_1 \in X$ if $\delta(x_1, s.e)$ is defined for $s \in \Sigma^*$, $e \in \Sigma$, and $x_2 \in \delta(x_1, s.e)$.

A set of strings is called a *language*. The set of strings that can be generated by G from a state $x \in X$ is denoted by $\mathcal{L}_G(x) = \{s \in \Sigma^* \mid \delta(x, s) \text{ is defined}\}$. The set of all strings generated by G from the initial state x_0, is called the language of system G, and is captured by $\mathcal{L}_G(x_0)$. For simplicity, for the remainder of the paper, we use $\mathcal{L}_G$ to denote the language of G, unless otherwise noted. A collection of all system strings that occur immediately following $s \in \mathcal{L}_G(x_0)$ is defined by the set $\mathcal{L}_{G/s} = \{t \in \Sigma^* \mid s.t \in \mathcal{L}_G(x_0)\}$. The extension closure of the language $\mathcal{L}_G(x_0)$, denoted by $ext(\mathcal{L}_G(x_0))$, is the language $ext(\mathcal{L}_G(x_0)) := \{v \in \Sigma^* \mid \exists u \in \mathcal{L}_G(x_0) : uv \in \mathcal{L}_G(x_0)\}$. We also define the unobservable reach of $x \in X$ as $UR(x) = \{y \in X \mid \exists u \in \Sigma_u^*,\ \delta(x, u) = y\}$, and the unobservable extension of $x \in X$ as $UE(s, x) = \{s.t \mid t \in \Sigma_u^*\ and\ s.t \in \mathcal{L}_G(x)\}$.

An unobservable event is unrecorded, thus cannot be used for system diagnosis. A tool used to capture observable system events is the natural projection, P, which removes the unobservable events from a string, while preserving the order of the string's observable events. Formally, the natural projection $P : \Sigma^* \to \Sigma_o^*$ can be defined as follows:

- $P(\varepsilon) = \varepsilon$,
- $P(e) = e$, if $e \in \Sigma_o$,
- $P(e) = \varepsilon$, if $e \notin \Sigma_o$,
- $P(s.e) = P(s)P(e)$, for $s \in \Sigma^*$ and $e \in \Sigma$.

To extract the observable behavior of the system, the natural projection should be extended to $\mathcal{L}_G$ as $P(\mathcal{L}_G) = \{P(s) \mid s \in \mathcal{L}_G\}$. The inverse projection of a string $w \in \Sigma_o^*$ into $\mathcal{L}_G \subseteq \Sigma^*$ is $P^{-1}_{\mathcal{L}_G}(w) = \{s \in \mathcal{L}_G \mid P(s) = w\}$.

Plant faults are modelled by fault events $f \in \Sigma_f$, where Σ_f is the set of all faulty events of the system. In this paper, faulty events are assumed to be unobservable, $\Sigma_f \subseteq \Sigma_u$, as diagnosing an observable event is trivial. The fault event set Σ_f can be partitioned into m different types of faults Σ_{f_1}, Σ_{f_2}, ..., Σ_{f_m}. This is done to allow one or more faults events to represent the same fault type Σ_{f_i}. During the plant operation, different faulty event occurrences may produce the same system behavior (e.g., stuck valve and incorrect valvae sensor readout).

To facilitate the diagnosis of fault occurrences in the system, the following assumptions are made: $\mathcal{L}_G$ is live ($\forall x \in X, \exists \sigma \in \Sigma$ such that $\delta(x, \sigma)$ is defined), and all strings of unobservable events are of finite length ($\forall s \in \Sigma_u^*$, $\exists n_o \in \mathbb{N}$ such that $\|s\| \leq n_o$). We assume that the system is live to provide the diagnoser enough time to gather information for fault diagnosis. Without a bound on traces of unobservable events, the diagnoser will arbitrarily traverse a cycle of unobservable events. This situation results in no recorded data for fault diagnosis; thus the bound ensures that the diagnoser will receive observable events for information gathering within a finite interval of system events.

Fault diagnosis is the investigative analysis of a system's observed behavior to provide the detection (verification of fault occurrence), identification (determination of the fault's nature), and isolation (finding the location of system state reached upon detection of fault) of a fault's occurrences. We formulate this problem as:

Problem 1. In a discrete event system G with the generated string s, for any successive string $t \in \mathcal{L}_{G/s}$, where t occurs upon diagnoser activation, from the observation $P(t)$, determine if $\exists f \in \Sigma_f$ such that $f \in s.t$. If yes, identify the type of fault, Σ_{f_i}, where $f \in \Sigma_{f_i}$, and locate the fault by finding the system state $x \in X$ subsequently reached upon fault occurrence.

3. Constructing the Diagnoser

In this section, to address Problem 1 we present a novel algorithm for construction of a diagnoser that can be activated asynchronously with respect to the monitored system. The proposed asynchronous diagnoser is a deterministic, finite-state DES, represented by the four tuple:

$$G_d = (Q_d, \Sigma_d, \delta_d, q_0) \tag{2}$$

whose elements include, $Q_d \subseteq 2^{X \times L}$ (the finite set of discrete diagnoser states), $\Sigma_d = \Sigma_o$ (the finite set of diagnoser events), δ_d (the diagnoser transition relation), and q_0 (the diagnoser's initial state). The estimated condition (faulty, normal) of the monitored plant is illustrated using various diagnostic labels from the set $L = \{N\} \cup 2^F$, where $F = \{F_1, F_2, \ldots, F_m\}$, F_i represents the labels for the occurrence of events of fault type Σ_{f_i}, $i = 1, ..., m$, and the diagnostic label $\{N\}$ represents a normal plant operation. Each diagnoser state $q = \{(x_1, \ell_1), \ldots, (x_k, \ell_k)\} \in Q_d$ is a set of ordered pairs (x_j, ℓ_j) characterizing the diagnoser's estimations, $x_j \in X$, $j = 1, ..., k$, and the diagnosed condition represented by a diagnostic label $\ell_j \in L$. Upon reaching a state q in Q_d, with $(x_j, \ell_j) \in q$, if $F_i \in \ell_j$, it implies that $\ell_j \neq \{N\}$, a fault event of type F_i may have been occurred. Next we will discuss the procedure for finding the states of the diagnoser, and the mechanics of the diagnoser state transition relation.

When the diagnoser detects an observable event $e \in \Sigma_o$, it carries out a process that updates its estimations of the plant's state and condition. The diagnoser's updated estimation of the plant's condition, is determined by the *Label Appending Function*, ∇. Given $t \in \Sigma^*$, the update of the label ℓ to a new label ℓ' is carried out by $\nabla : L \times \Sigma^* \to L$, where $\ell' = \nabla(\ell, t) :=$

$$\begin{cases} \cup\{F_i \in F\}, \ if \ (F_i \in \ell) \ or \ (\exists f \in \Sigma_{fi} \ and \ f \in t) \\ \{N\} \qquad\qquad\qquad otherwise \end{cases} \tag{3}$$

Note that from Equation 3, once a fault event $f \in \Sigma_{f_i}$ occurs in a system trace in G, its corresponding fault label $F_i \in F$ is propagated for all future system event occurrences. Since the diagnoser is asynchronously activated at any time instance of the plant's operation, initially the diagnoser is unknowing of the system's state and condition. This is captured by the diagnoser's initial state:

$$q_0 := \{(x, \ell) = (\delta(x_0, t), \nabla(\{N\}, t)) | \forall t \in \mathcal{L}_G(x_0)\} \tag{4}$$

Equation 4 is stating that the diagnoser's initial estimate of the system consists of all possible system conditions for every reachable system state. Starting from this wide estimation, the asynchronous diagnoser narrows down its estimate of the plant's state and condition as it sequentially observes events $e \in \Sigma_o$. This is accomplished by the diagnoser's state transition relation $\delta_d : Q_d \times \Sigma_o \to Q_d$:

$$\delta_d(q, e) = \bigcup_{\substack{(x,\ell) \in q \\ t \in UE(e,x)}} \{(\delta(x, t), \nabla(\ell, t))\} \tag{5}$$

Starting from q_0, the diagnoser's state transition relation operator given in Eq. 5 builds each accessible diagnoser state based upon the recorded $e \in \Sigma_o$. Algorithm 1 summarizes this diagnoser construction process.

Algorithm 1 Constructing an Asynchronous Diagnoser

Initialization:
$q_0 := \{(x_0, N)\}$;
Step 1: Constructing q_0
$q_0 := q_0 \bigcup \{(x, \ell) | x \in \delta(x_0, u), \ u \in \Sigma_u^*, \ \ell = \nabla(\{N\}, u)\}$;
repeat
  for $(x, \ell) \in q_0$ and $e \in \Sigma_o$ **do**
    if $\delta(x, e)$ is defined, $\exists t \in UE(e, x)$, and $(\delta(x, t), \nabla(\ell, t)) \notin q_0$ **then**
      $q_0 = q_0 \cup \{(\delta(x, t), \nabla(\ell, t)) | t \in UE(e, x)\}$;
    end if
  end for
until There is no new pair (x, l) in q_o.
Step 2: Constructing Q_d
$Q_d := q_0$;
repeat
  for $q \in Q_d$ and $e \in \Sigma_o$ **do**
    if $\delta_d(q, e)$ is defined and $\delta_d(q, e) \notin Q_d$ **then**
      Add $\delta_d(q, e)$ to Q_d;
    end if
  end for
until There is no new state $\delta_d(q, e)$ for all $e \in \Sigma_o$.

Algorithm 1 is divided into two steps. In Step 1, the diagnoser's initial state q_0 is constructed. The algorithm is initialized with $q_0 = \{(x_0, N)\}$ assuming that the plant starts operating normally. The remaining statements in Step 1, recursively build q_0 to cover all possible traces in $\mathcal{L}_\mathcal{G}$. In Step 2, the remaining diagnoser states $q \in Q_d$ are recursively constructed from all possible sequences of observable events emitted by the plant. The following example illustrates the implementation of the algorithm.

Example 1. Consider a DES plant G, shown in Fig. 1, with $\Sigma = \{\alpha, \beta, \tau, f_1, f_2\}$, $\Sigma_o = \{\alpha, \beta\, \tau\}$, $\Sigma_u = \{f_1, f_2\}$, and $\Sigma_f = \{f_1, f_2\}$, $\Sigma_{f_1} = \{f_1\}$, and $\Sigma_{f_2} = \{f_2\}$.

To implement Algorithm 1 for constructing an asynchronous diagnsoer for the plant G, we first incrementally construct a state reachability table (SRT) which provides all system diagnostic information in a compact form. We initialize the SRT with $q_0 = \{(x_0, N)\} = \{(1, N)\}$. Next, we determine the states that are reachable from x_0 by unobservable strings, and adjoin to them all resulting condition labels. These ordered pairs are then added to q_0 creating $q_0 := q_0 \bigcup \{(x, \ell) \mid x \in \delta(x_0, u), \ u \in \Sigma_u^*, \ \ell = \nabla(\{N\}, u)\} = \{(1, \{N\}), (4, \{F_1\}), (7, \{F_2\})\}$. These pairs will form the first column of the table. To

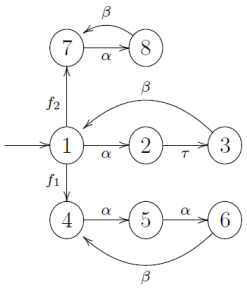

Figure 1: The DES model of the plant G in Example 1.

complete the rest of the table, for all $(x, \ell) \in q_0$ and $e_0 \in \Sigma_o$, we will find the states that are reachable by $UE(e_o, x)$, i.e. $\bigcup_{t \in UE(e_o,x)} \{(\delta(x,t), \nabla(\ell,t))\}$, as shown in the following table:

q_0	α	β	τ
$1N$	$2N$	-	-
$4F_1$	$5F_1$	-	-
$7F_2$	$8F_2$	-	-

If a new pair (x, ℓ) appears in the table which does not exist in column q_0, it has to be added to q_0. Again, for this new member of q_0, for each column $e_0 \in \Sigma_o$, we should find the states that are reachable by $UE(e_o, x)$, i.e. $\bigcup_{t \in UE(e_o,x)} \{(\delta(x,t), \nabla(\ell,t))\}$. This has to be continued until no new state can be found. For Example 1, the completed SRT table is shown in Table 1.

In the SRT, the first column forms q_0 which contains all states $x \in X$ and their corresponding possible diagnostic label $\ell \in L$. For any element of the first column, (x, ℓ), the other columns of the same row independently corresponds to transitions due to $e_o \in \Sigma_o$ as $(x', \ell') \in \{(\delta(x,t), \nabla(\ell,t)) \mid t \in UE(e_o, x)\}$. Upon completion of the SRT, the first column contains all elements of q_0 as, $q_0 = \{1N, 2N, 3N, 4F_1, 5F_1, 6F_1, 7F_2, 8F_2\}$.

Next, following Step 2 of Algorithm 1 and using this SRT table, all remaining states of the diagnoser will be constructed. For example, due to the event α, the state q_0 will be transited to a new state q_1 which can be found as $q_1 = \delta(q_0, \alpha) = \bigcup_{\substack{(x,\ell) \in q_0 \\ t \in UE(\alpha,x)}} \{(\delta(x,t), \nabla(\ell,t))\} = \{(2, N), (5, F_1), (6, F_1), (8, F_2)\}$. Continuing this process for G, the completed diagnoser is shown in Fig. 2.

4. Asynchronous Fault Diagnosis

Algorithm 1 in Section 3 constructs an asynchronous diagnoser for any given DES system. The constructed diagnoser will provide fault diagnosis by detecting, isolating, and identifying a fault's occurrence without requiring the restarting of the system. A fault of type F_i is detected upon the diagnoser reaches an F_i-certain state. Imagine that f_1 has occurred and the plant G is in state 4. Then, we turn on the diagnoser, and the diagnoser starts from q_0. When the event $\alpha \in \Sigma_o$ happens in the plant G, the diagnoser switches to $q_5 = \{(2, N), (5, F_1), (8, F_2)\}$, which is an F_1-uncertain and F_2-uncertain state. But if we wait for another transition, again the event $\alpha \in \Sigma_o$ happens in the plant G, resulting the diagnoser transition to $q_4 = \{(6, F_1)\}$ which is an F_1-certain state, implying that the failure of type F_1 has occurred during the system's operation. As this can be checked for all fault occurrences in G_1, upon the activation of the diagnoser in Figure 2, it will indeed diagnose the occurrence of all fault events f_1 and f_2, by reaching a corresponding F_i-certain state in a finite number of system observations. Such a discrete event system that is diagnosable for all fault occurrences and under all activation conditions is called asynchronously diagnosable. Asynchronous diagnosability can be formally defined as follows:

TABLE 1: The State Reachability Table (SRT) for the plant G in Example 1.

q_0	α	β	τ
$1N$	$2N$	-	-
$2N$	-	-	$3N$
$3N$	-	$1N$, $4F_1$, $7F_2$	-
$4F_1$	$5F_1$	-	-
$5F_1$	$6F_1$	-	-
$6F_1$	-	$4F_1$	-
$7F_2$	$8F_2$	-	-
$8F_2$	-	$7F_2$	-

Definition 1. (F_i-Asynchronous Diagnosability) The plant G with the live language $\mathcal{L}_G$, is said to be F_i-Asynchronously diagnosable with respect to the failure type F_i and the natural projection P, if for all $s \in \mathcal{L}_G$, $f \in \Sigma_{f_i}$, $f \in s$, there exist an upper bound $n_i \in \mathbb{N}$, such that for any string $t \in \mathcal{L}_{G/s}$ with $\|t\| \geq n_i$, the following condition holds:

$$\{\forall uv \in \mathcal{L}_G, v \in P^{-1}_{ext(\mathcal{L}_G)}(P(t))\} \Rightarrow f \in uv \quad (6)$$

Example 2. Consider the automaton G_2 in Fig. 3 with $\Sigma = \{\alpha, \beta, \delta, \tau, f_1, f_2\}$, $\Sigma_o = \{\alpha, \beta, \delta, \tau\}$, $\Sigma_u = \{f_1, f_2\}$, $\Sigma_f = \{f_1, f_2\}$, $\Sigma_{f_1} = \{f_1\}$, and $\Sigma_{f_2} = \{f_2\}$. Similar to Example 1, we can construct the asynchronous diagnoser G_{d_2} as shown in Fig. 4. It can be verified that the diagnoser G_{d2} does not detect all faults in G_2. For instance, let's assume that the diagnoser G_{d_2} is activated while the system G_2 is located at system state 10. At this point, in the system G_2, the fault $f_2 \in \Sigma_{f_2}$ has occurred prior to the diagnoser's activation, however this information is not available to the diagnoser G_{d_2}. From system state 10, the trace

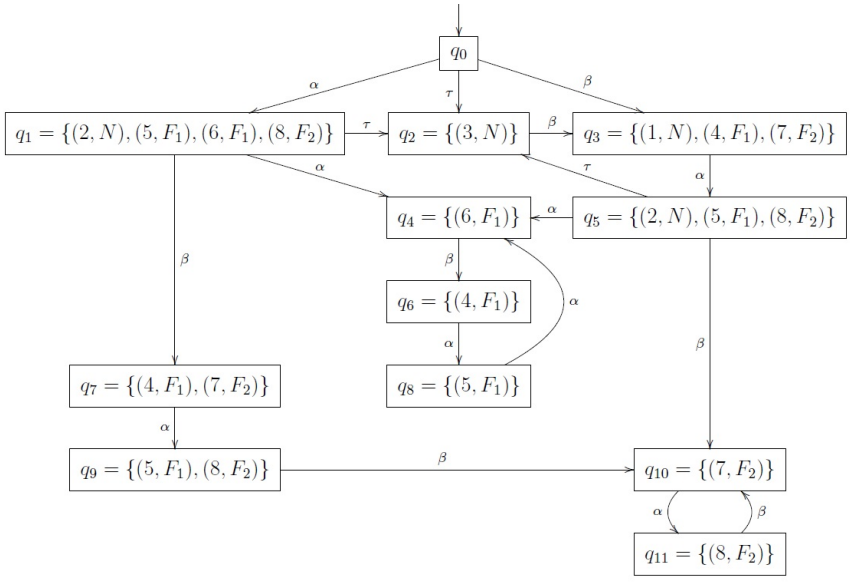

Figure 2: The constructed diagnoser for the plant G given in Fig. 1

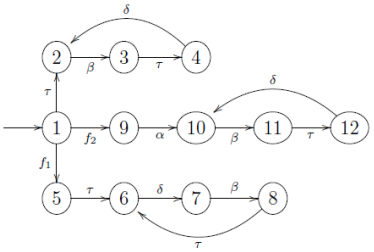

Figure 3: The DES model of the plant G_2 in Example 2.

$10 \xrightarrow{\beta} 11 \xrightarrow{\tau} 12 \xrightarrow{\delta} 10 \xrightarrow{\beta} 11 \xrightarrow{\tau} 12 \xrightarrow{\delta} 10 \cdots$ cycles the states 10, 11, and 12 infinitely. This corresponds to the sequence $q_0 \xrightarrow{\beta} q_2 \xrightarrow{\tau} q_6 \xrightarrow{\delta} q_3 \xrightarrow{\beta} q_2 \xrightarrow{\tau} q_6 \xrightarrow{\delta} \cdots$, which never reaches an F_2-certain state.

This can be also verified by checking Definition 1 that the plant G_2 is not F_2-Asynchronously diagnosable. For the above example, state 10 is reachable from x_0 by $s = f_2\alpha$. From there, the string $t = (\beta\tau\delta)^*$ can cycle the states 10, 11, and 12 as $10 \xrightarrow{\beta} 11 \xrightarrow{\tau} 12 \xrightarrow{\delta} 10 \xrightarrow{\beta} 11 \xrightarrow{\tau} 12 \xrightarrow{\delta} 10 \cdots$. So, the string $s.t = f_2\alpha(\beta\tau\delta)^*$ contains the fault f_2. Now, in plant G_2 consider the trace $u = f_1\tau$ and $v = (\beta\tau\delta)^*$, where $u.v \in L_{G_2}$. As it can be checked $v \in P^{-1}_{ext(\mathcal{L}_{G_2})}(P(t))\}$, however, $f_2 \notin uv$. Therefore, based on Definition 1, system G_2 in Example 2 is not F_2-asynchronously diagnosable.

Remark 1. Regardless of diagnosability of a system, the diagnoser G_d can always be constructed. If the the system G is F_i-asynchronously diagnosable, then diagnoser G_d will diagnose all existing plant faults. However in the case that the system is not F_i-asynchronously diagnosable, the diagnoser will always provide its best estimate of the plant's state and condition based upon its observations of the plant.

5. Conclusion

We introduced and presented a novel asynchronous diagnoser that can be activated asynchronously any time during system operation to diagnose different types of faults. Therefore, unlike other existing diagnosis techniques, our developed diagnoser does not need the past history of the system before the activation of the diagnoser and hence, it does not require the monitored system to be restarted, in turn saves time and cost of diagnosis. Several examples were provided to illustrate the proposed algorithm. Online implementation of the proposed algorithm were investigated. Future works include extending the proposed approach to distributed and decentralized structures.

Acknowledgment

This research is supported by Air Force Research Laboratory and Office of the Secretary of Defense under agreement number FA8750-15-2-0116 as well as US ARMY Research Office under agreement number W911NF-16-1-0489. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory, ARMY Research Office, OSD, or the U.S. Government.

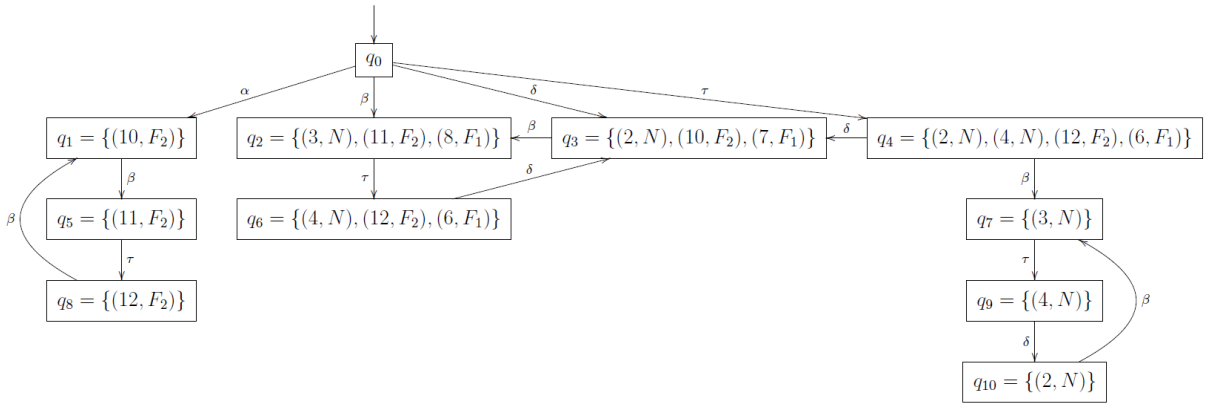

Figure 4: The constructed diagnoser G_{d_2} for the plant G_2 given in Fig. 3

References

[1] R. M. Murray, K. J. Astrom, S. P. Boyd, R. W. Brockett, and G. Stein, "Future directions in control in an information-rich world," *IEEE Control Systems Magazine*, vol. 23, no. 2, pp. 20–33, 2003.

[2] J. Carreno, G. Galdorisi, S. Koepenick, and R. Volner, "Autonomous systems: Challenges and opportunities," DTIC Document, Tech. Rep., 2010.

[3] A. Pouliezos and G. S. Stavrakakis, *Real time fault monitoring of industrial processes*. Springer Science & Business Media, 2013, vol. 12.

[4] P. M. Frank, "Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy: A survey and some new results," *Automatica*, vol. 26, no. 3, pp. 459–474, 1990.

[5] W.-S. Lee, D. L. Grosh, F. A. Tillman, and C. H. Lie, "Fault tree analysis, methods, and applications - a review," *IEEE Transactions on Reliability*, vol. 34, no. 3, pp. 194–203, 1985.

[6] J. D. Andrews and S. J. Dunnett, "Event-tree analysis using binary decision diagrams," *IEEE Transactions on Reliability*, vol. 49, no. 2, pp. 230–238, 2000.

[7] J. Chen and R. J. Patton, *Robust model-based fault diagnosis for dynamic systems*. Springer Science & Business Media, 2012, vol. 3.

[8] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri, "A review of process fault detection and diagnosis: Part i: Quantitative model-based methods," *Computers & chemical engineering*, vol. 27, no. 3, pp. 293–311, 2003.

[9] X. Zhang, T. Parisini, and M. M. Polycarpou, "Adaptive fault-tolerant control of nonlinear uncertain systems: an information-based diagnostic approach," *IEEE Transactions on Automatic Control*, vol. 49, no. 8, pp. 1259–1274, 2004.

[10] Y. Zheng, H. Fang, and H. O. Wang, "Takagi-sugeno fuzzy-model-based fault detection for networked control systems with markov delays," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 36, no. 4, pp. 924–929, 2006.

[11] D. N. Pandalai and L. E. Holloway, "Template languages for fault monitoring of timed discrete event processes," *IEEE Transactions on Automatic Control*, vol. 45, no. 5, pp. 868–882, 2000.

[12] S. R. Das and L. E. Holloway, "Characterizing a confidence space for discrete event timings for fault monitoring using discrete sensing and actuation signals," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 30, no. 1, pp. 52–66, 2000.

[13] U. Lerner, R. Parr, D. Koller, G. Biswas *et al.*, "Bayesian fault detection and diagnosis in dynamic systems," in *AAAI/IAAI*, 2000, pp. 531–537.

[14] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer Science & Business Media, 2009.

[15] P. J. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81–98, 1989.

[16] A. White and A. Karimoddini, "Semi-asynchronous fault diagnosis of discrete event systems," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2016, pp. 003 961–003 966.

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

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

[19] Y. Wang, T.-S. Yoo, and S. Lafortune, "Diagnosis of discrete event systems using decentralized architectures," *Discrete Event Dynamic Systems*, vol. 17, no. 2, pp. 233–263, 2007.

[20] W. Qiu and R. Kumar, "Decentralized failure diagnosis of discrete event systems," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 36, no. 2, pp. 384–395, 2006.

[21] O. Contant, S. Lafortune, and D. Teneketzis, "Diagnosability of discrete event systems with modular structure," *Discrete Event Dynamic Systems*, vol. 16, no. 1, pp. 9–37, 2006.

[22] R. Su and W. M. Wonham, "Global and local consistencies in distributed fault diagnosis for discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 50, no. 12, pp. 1923–1935, 2005.

[23] M. M. Karimi, A. Karimoddini, A. P. White, and I. W. Bates, "Event-based fault diagnosis for an unknown plant," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec 2016, pp. 7216–7221.

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

[25] L. K. Carvalho, J. C. Basilio, and M. V. Moreira, "Robust diagnosis of discrete event systems against intermittent loss of observations," *Automatica*, vol. 48, no. 9, pp. 2068–2078, 2012.

[26] S. T. S. Lima, J. C. Basilio, S. Lafortune, and M. V. Moreira, "Robust diagnosis of discrete-event systems subject to permanent sensor failures." in *WODES*, 2010, pp. 90–97.

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