Semi-asynchronous Fault Diagnosis of Discrete Event Systems

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

***Abstract*—This paper proposes a diagnostics tool for a Discrete-Event System (DES) under uncertain activation conditions. This diagnosis tool, the diagnoser (as it is called), detects, identifies, and locates system faults in relation to a set of states of which the system under diagnosis could possibly be located, upon the diagnoser's instance of activation. This diagnoser is designed to diagnose system faults that occur prior to and/or after the diagnoser's activation; thus removing the procedural constraint of initializing the system and diagnoser synchronously. Illustrative examples are provided to detail the proposed diagnosis procedure.**

I. Introduction

Highly complex autonomous systems are increasingly becoming dependent upon in daily societal activities and operations. This leads to an increase in liability [1]. The challenge is that even with the best practices for developing and using high quality components, faults may occur in a system unexpectedly, whose consequences might be very costly, and even deadly. Therefore, today's systems must comply with stringent requirements for system safety and reliability. As systems become more complex, diagnosing (detecting, identifying, and locating) system faults becomes more complicated. Systematic robust fault diagnosis techniques are essential for a timely and accurate diagnosis of system faults. A collective study of fault diagnosis methods can be found in [2], [3].

In this paper, a capable methodology for fault diagnosis is carried out within discrete event system (DES) framework [4], [5]. DES is an abstract, yet effective, framework to model a system by sequences of evens (actions and observations) which changes the system's states (modes of operation). Our decision to use a DES approach for fault diagnosis is supported by the fact that in many cases a fault leads to abrupt changes in the system, which can be best modelled by an event to be diagnosed by observing the behavior of the system. There are different techniques for DES fault diagnosis in the literature [6], including event-based [7] and state-based [8], appearing in different structures including decentralized [9], [10] and modular/distributed [11], [12] architectures.

These DES fault diagnosis techniques are carried out by a diagnosis tool which is called the diagnoser. Almost in all of these methods, the diagnoser has to be simultaneously initialized with the system under diagnosis. It also should synchronously execute the events in parallel with system under diagnosis to keep its past history of exhibited normal and faulty behaviors. This synchronous process is quite complicated for most industrial processes and it is preferred to use the diagnosis tool only after a fault occurs in the system. Also, the existing methods require the system under diagnosis to be restarted to make the system and the diagnoser synchronized from the initial state. This restart requirement might be practically very costly and time consuming in many industrial processes.

To address this problem, this paper proposes a semi-asynchronous DES fault diagnoser, which does not require to be synchronously initialized with the system under diagnosis. When the proposed diagnoser be activated, instead of restarting the system under diagnosis, it only needs an estimation of the active state of the system under diagnosis. For this purpose, given an estimated system state, the developed diagnoser is capable of diagnosing system faults that occur before and/or after the diagnoser's activation. A challenge to this capability, is that knowledge of the system's behavior pre-activation of the diagnoser is inaccessible. Furthermore, in the conventional setup, when a diagnoser is synchronously initialized with the system, it is often assumed that the initial state of a system is non-faulty. In the case of semi-asynchronous diagnosis, this assumption is impractical. This paper addresses these challenges by developing a systematic and analytical approach to construct a semi-asynchronous diagnoser. Our design takes advantage of the fact that many systems exhibit behaviors where the system's possible state locations can be deduced to a set of states, or that a system can be derived to a situation where its possible system state locations can be deduced to a set.

The rest of the paper is organized as follows. Section II, provides the necessary background and notations for DES modeling of the original system, and formulates the semi-asynchronous fault diagnosis problem. In Section III, an algorithm for constructing the semi-asynchronous diagnoser is presented, accompanied by an illustrative example, which details the steps of the proposed algorithm. In Section IV, the implementation of the proposed diagnoser and the way that it detects system faults are discussed. Section V concludes the paper.

II. Problem Formulation

For the purpose of diagnosing system faults within the Discrete Event Systems (DES) framework, the original system is modelled as a non-deterministic DES which can be captured

A. White and A. Karimoddini are with the Department of Electrical and Computer Engineering, North Carolina Agricultural and Technical State University, Greensboro, NC 27411, USA.

Corresponding author: A. Karimoddini, Tel: +13362853313, akarimod@ncat.edu.

by the four-tuple, $G = (X, \Sigma, \delta, x_0)$, where X, Σ, δ, and x_0 represent the system's state space, event set, state transition relation, and the system's initial state, respectively. X and Σ are finite sets, and $\delta : X \times \Sigma \rightarrow 2^X$ (2^X is the power set of the state space). The DES plant G encompasses the original system's normal and failed behavior.

The system's event set $\Sigma = \Sigma_o \dot{\cup} \Sigma_u$, is composed of events that are either observable ($e \in \Sigma_o$), or unobservable ($e \in \Sigma_u$). A concatenation of one or more events is called a trace or string. Σ^*, the Kleene closure of Σ, is the set of all possible finite strings s, composed of events $e \in \Sigma$ including the zero length string ε. The notation $e \in s$ implies that event e is one of the events forming string $s \in \Sigma^*$.

As observable faulty events can be trivially diagnosed, without loss of generality, we assume that faulty events in the system are unobservable events. During the system operation, different faulty event occurrences may produce the same system behavior (e.g., stuck valve, incorrect sensor readout). Therefore, similar to [7], faulty system events are partitioned into a set $\Sigma_f = \{\Sigma_{f_1}, \ldots, \Sigma_{f_m}\} \subseteq \Sigma_u$, allowing one or more faulty events to represent the same fault type Σ_{f_i}.

Modelling faulty events as unobservable events, makes fault diagnosis an arduous task of saying for certain whether a particular unseen event has or has not occurred from the observable behavior of the system. To model the observable behavior of the system, we use the natural projection operator P, which removes all unobservable events from any trace $s \in \Sigma^*$, while preserving the sequential order of all observable events in trace s. The natural projection onto the observable event set, $P : \Sigma^* \rightarrow \Sigma_o^*$, 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 $e \in \Sigma$.

A system's language is a discrete representation of the system's behaviors (normal and faulty) in the form of sequences of events. The language of a plant G, $\mathcal{L}_G(x_0)$, is the collective set of all system traces generated by G that originate at x_0. To mathematically represent the language of the system, $\mathcal{L}_G(x_0)$, the transition rule, δ, is first recursively extended to $X \times \Sigma^*$ as follows:

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

We now can define the language generated by G as $\mathcal{L}_G(x_0) := \{s \in \Sigma^* \mid \delta(x_0, s)\ is\ defined\}$. The set of all traces that can be generated by automaton G from the state $x \in X$, can be captured by $\mathcal{L}_G(x)$. This is also true for the the set of all traces that can be generated by automaton G from the states in $X_s \subseteq X$, which can be captured by $\mathcal{L}_G(X_s)$. The set of strings that can be generated from the set X_s is $Post(X_s) = \mathcal{L}_G(X_s)$, and the set of strings that are leading to X_s is shown by $Pre(X_s) = \{s \in \mathcal{L}_G(x_0) \mid \delta(x_0, s) \in X_s\}$. $\mathcal{L}_{G/s} := \{t \in \Sigma^* \mid st \in \mathcal{L}_G(x_0)\}$ is the set of all traces in $\mathcal{L}_G(x_0)$ that occur immediately following $s \in \mathcal{L}_G(x_0)$. The extension closure of the language $\mathcal{L}_G(x_0)$, denoted by $ext(\mathcal{L}_G(x_0))$, is the language $ext(\mathcal{L}_G(x_0)) := \{v \in \Sigma^* \mid \exists u \in \mathcal{L}_G(x_0) : uv \in \mathcal{L}_G(x_0)\}$. Unless specified otherwise, $\mathcal{L}_G$ will denote $\mathcal{L}_G(x_0)$.

In some cases a system state may reach other system states by a string of unobservable events. The unobservable reach is used to capture these states as $UR(x) = \{y \in X \mid \exists u \in \Sigma_u^*,\ \delta(x, u) = y\}$, which include the set of all system states (with the inclusion of x itself) reachable from state x via strings solely consisting of unobservable events. Also, $UE(s, x) = \{s.t \mid t \in \Sigma_u^*\ and\ s.t \in \mathcal{L}_G(x)\}$ specifies the set of all unobservable extensions of s concatenated with the string s and generated from the state x.

All system traces $s \in \mathcal{L}_G$ are categorized as follows:

- "*F_i-faulty*": if there exists an event $f \in \Sigma_{f_i}$ such that $f \in s$,
- "*non F_i-faulty*": if for all $f \in \Sigma_{f_i}$, $f \notin s$,
- "*normal*": if for all $i = 1, ..., m$, $f \notin s$.

Extending the natural projection operator to a language as $P(\mathcal{L}_1) = \{P(s) \mid s \in \mathcal{L}_1\}$, makes it possible to capture the observable system behaviors. The inverse projection of a string $w \in \Sigma_o^*$ into $\mathcal{L}_1 \subseteq \Sigma^*$ is $P_{\mathcal{L}_1}^{-1}(w) = \{s \in \mathcal{L}_1 \mid P(s) = w\}$, and the inverse projection of a language $\mathcal{L}_2$ into $\mathcal{L}_1$ is $P_{\mathcal{L}_1}^{-1}(\mathcal{L}_2) = \bigcup_{w \in \mathcal{L}_2} P_{\mathcal{L}_1}^{-1}(w)$.

To ensure that detecting faults in a DES system G from its observable behaviors is feasible, two assumptions are placed on G. It is assumed that $\mathcal{L}_G$ is live, i.e., $\forall x \in X, \exists\ \sigma \in \Sigma$ such that $\delta(x, \sigma)$ is defined. This ensures that after the occurrence of a fault, there is ample time provided to monitor the system's behavior, and diagnose the fault occurrence. In addition, it is assumed that the length of unobservable strings in $\mathcal{L}_G$ are bounded by n_o, otherwise the plant G may become trapped in a cycle of unobservable events, in turn making the diagnosis impossible.

Fault diagnosis is the art of distinctively characterizing the system's behavior in order to detect, identify, and locate fault occurrences solely based upon external observations of the system. For the case of semi-asynchronous diagnosis, this problem can be formally formulated as follows:

Problem 1: Given X_s (the initial estimation of states of a DES plant G), for all $s \in \mathcal{L}_G$ with $\delta(x_0, s) \in X_s$ and for any its (sufficiently long) successive string $t \in \mathcal{L}_{G/s}$, where t occurs after 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 system state $x \in X$ that has subsequently reached by $s.t$.

III. Constructing the Diagnoser

To solve Problem 1, we propose a new technique to develop a diagnoser represented by a deterministic finite-state DES in the form of the four-tuple $G_d = (Q_d, \Sigma_d, \delta_d, q_0)$, where $Q_d \subseteq 2^{X \times L}$ is the state space, $\Sigma_d = \Sigma_o$ is the event set, δ_d is the state transition rule, and q_0 is the diagnoser's initial state. Diagnoser G_d uses the original system's observed behavior as input, and provides an estimate of the system's state and condition (faulty or non-faulty) as output. Each state of the diagnoser $q_d \in Q_d$, provides diagnostic information in the form of $q_d = \{(x_1, \ell_1), \ldots, (x_k, \ell_k)\}$, where $x_j \in X$ and $\ell_j \in L$, $j = 1, \ldots, k$. With this format, each ordered pair $(x_j, \ell_j) \in q_d$ consists of an estimation of the system state,

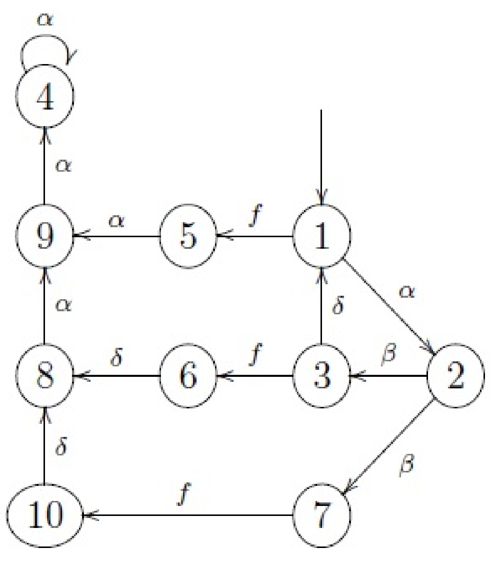

Fig. 1: The DES plant G_1

x_j, adjoined by the corresponding estimated system condition, ℓ_j. These ordered pairs help to detect, isolate, and identify a fault's occurrence. The set $L = \{N\} \cup 2^F$ represents all possible condition labels for each system state $x \in X$. The set $F = \{F_1, F_2, \ldots, F_m\}$ represents all possible fault labels, where the label F_i represents the label for the occurrence of a fault(s) that belongs to fault type Σ_{f_i}, $i = 1, ..., m$, and label $\{N\}$ represents a normal system operation. System fault $f \in \Sigma_{f_i}$ has been diagnosed if for all $(x_j, \ell_j) \in q_d$, $F_i \in \ell_j$.

System condition labels $\ell \in L$ are created and updated by the *Label Appending Function*, $\nabla : L \times \Sigma^* \to L$, where $\ell' = \nabla(\ell, t) :=$

$$\begin{cases} \cup\{F_i \in F\}, \; if \; (F_i \in \ell) \; or \; (\exists f \in \Sigma_{fi} \; and \; f \in t) \\ \{N\} \qquad\qquad\qquad otherwise \end{cases} \tag{1}$$

Upon activation, the semi-asynchronous diagnoser begins with the assumption that the system under diagnosis is either in one of the states $x \in X_s$, or their unobservable reach states $UR(X_s)$. Therefore, the composition of the diagnoser's initial state is $q_0 =:$

$$\{(\delta(x_0, s), \nabla(\{N\}, s)) \mid \delta(x_0, s) \in UR(X_s), \; s \in \mathcal{L}_G\} \tag{2}$$

Starting with this uncertain estimation, the diagnoser proceeds to use its observations of the system's behavior to narrow down its estimate of the system's state and condition (faulty, non-faulty). For this purpose, upon observing $e \in \Sigma_o$ from the original system, the diagnoser updates its estimation of the system's state and condition, using the diagnoser's state transition rule as follows:

$$\delta_d(q, e) = \bigcup_{\substack{(x,\ell)\in q \\ t\in UE(e,x)}} \{(\delta(x, t), \nabla(\ell, t))\} \tag{3}$$

The following algorithm provides a step-by-step procedure for constructing the semi-asynchronous diagnoser.

Algorithm 1 Constructing a Semi-Asynchronous Diagnoser

```
    Initialization:
    $q_s := \{(x_0, N)\}$;
    Step 1: Constructing $q_0$
    $q_s := q_s \bigcup \{(x, \ell) \mid x \in \delta(x_0, u), \; u \in \Sigma_u^*, \; \ell = \nabla(\{N\}, u)\}$;
    repeat
        for $(x, \ell) \in q_s$ and $e \in \Sigma_o$ do
            if $\delta(x, e)$ is defined, $\exists t \in UE(e, x)$ , and $(\delta(x, t), \nabla(\ell, t)) \notin q_s$ then
                $q_s = q_s \cup \{(\delta(x, t), \nabla(\ell, t)) \mid t \in UE(e, x)\}$;
            end if
        end for
    until There is no new pair $(x, l)$ in $q_s$.
    $q_0 = \{(x, \ell) \mid (x, \ell) \in q_s, \; x \in UR(X_s)\}$
    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 $e \in \Sigma_o$.
```

Remark 1: Although the developed semi-asynchronous diagnoser starts with an uncertain estimation of the system under diagnosis, as it moves forward, the diagnoser narrows down its estimate of the original system's state and condition as it acquires more information from its observations.

The following example, illustrates the implementation of the algorithm.

Example 1: Consider the plant G_1, shown in Fig. 1, with $\Sigma = \{\alpha, \beta, \delta, f\}$, $\Sigma_o = \{\alpha, \beta, \delta\}$, $\Sigma_u = \Sigma_f = \{f\}$. Also, consider that upon activation of the diagnoser, the plant G_1 is in one of the states $X_s = \{2, 9\}$.

To ease the implementation of Algorithm 1, first, we incrementally construct a state reachability table (SRT) to record all system diagnostic information. To illustrate the process we will explain the construction of SRT for the plant G_1 given in Example 1. First consider $q_s = \{(x_0, N)\} = \{(1, N)\}$. Then, we should find the states and their labels that are reachable from x_0 by unobservable strings, $q_s := q_s \bigcup \{(x, \ell) \mid x \in \delta(x_0, u), \; u \in \Sigma_u^*, \; \ell = \nabla(\{N\}, u)\} = \{(1, \{N\}), (5, \{F\})\}$. These pairs will form the first column of the table. Then, to complete the rest of the table, for all $(x, \ell) \in q_s$ 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_s	α	β	δ
$1N$	$2N$	-	-
$5F$	$9F$	-	-

If a new pair (x, ℓ) appears in the table which does not exist in column q_s, it has to be added to q_s. Again, for this new member of q_s, 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 and sorted table will be as follows:

q_s	α	β	δ
$1N$	$2N$	-	-
$2N$	-	$3N$, $7N$, $6F$, $10F$	-
$3N$	-	-	1N,5F
$4F$	$4F$	-	-
$5F$	$9F$	-	-
$6F$	-	-	$8F$
$7N$	-	-	-
$8F$	$9F$	-	-
$9F$	$4F$	-	-
$10F$	-	-	$8F$

In the SRT, the first column entry of each row contains a state $x \in X$ and its corresponding possible diagnostic label $\ell \in L$. All remaining column entries of each row independently corresponds to transitions due to $e_o \in \Sigma_o$, and provides diagnostic information for all reachable states. These column entries contain $(x', \ell') \in \{(\delta(x,t), \nabla(\ell,t)) \mid t \in UE(e_o, x)\}$.

Now, from the first column of this SRT we can find the state q_0 as, $q_0 = \{(x,\ell) \mid (x,\ell) \in q_s,\ x \in UR(X_s)\} = \{2N, 9F\}$. Next, following Step 2 in the algorithm 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 $q_1 = \delta(q_0, \alpha) = \bigcup_{\substack{(x,\ell)\in q_0 \\ t \in UE(\alpha,x)}} \{(\delta(x,t), \nabla(\ell,t))\} = \{(4,F)\}$. Continuing this process for G_1, the completed diagnoser is shown in Fig. 2.

IV. Diagnosing the Faults

Using the procedure proposed in the previous section, it is possible to construct a semi-asynchronous diagnoser for any DES G with a given set X_s. Running the diagnoser, we can obtain useful diagnostic information about the system including the possible current states, and the possibly occurred faults in the original system. In particular, if the diagnoser is at an F_i-certain state, this implies that the fault type F_i has determinatively occurred. If the diagnoser is at a *non F_i-faulty* state, then for sure a no fault of type F_i has occurred, and if the diagnoser is at a normal state, then we can conclude that no fault has occurred in the system. An important question is that will the constructed diagnoser, semi-asynchronously diagnose all possible occurrences of $f \in \Sigma_f$ in G. For example, in system G_1 discussed in Example 1, it can be verified that all faults can be detected after a finite number of transitions. When diagnoser G_{d1} is activated, it is uncertain of the system state and assumes G_1 to be $x \in X_s = \{2, 9\}$. Let's assume the actual state of system G_1 is $x = 2$, and that the following feasible trace occurs, $2 \xrightarrow{\beta} 3 \xrightarrow{f} 6 \xrightarrow{\delta} 8 \xrightarrow{\alpha} 9 \xrightarrow{\alpha} 4$, producing the corresponding diagnoser state transitions $q_0 \xrightarrow{\beta} q_1 \xrightarrow{\delta} q_2 \xrightarrow{\alpha} q_0 \xrightarrow{\alpha} q_3$. Diagnoser state q_3 is F-certain. Therefore it can be determinatively concluded that a system fault type F has occurred. For all feasible system traces that may originate from any $x \in X = \{2, 9\}$, if a fault type Σ_f occurs, diagnoser G_{d1} will reach an F-certain state within a finite number of observed system events. However, this may not be true for all systems. See the following example.

Example 2: Consider the automaton G_2 in Fig. 3 with $\Sigma = \{\alpha, \beta, \delta, f_1, f_2\}$, $\Sigma_o = \{\alpha, \beta, \delta\}$, $\Sigma_u = \{f_1, f_2\}$, $\Sigma_f = \{f_1, f_2\}$, $\Sigma_{f_1} = \{f_1\}$, and $\Sigma_{f_2} = \{f_2\}$. The initial set of estimated states is $X_s = \{2, 12, 21\}$. The semi-asynchronous diagnoser for this automaton is shown in Fig. 4. Using this diagnoser as reference, it can be verified that diagnoser G_{d2} does not detect all faults in G_2. For example, consider that the system G_2 is in state 2 when the diagnoser is activated. The failure f_2 occurs through the following sequence in the original plant: $2 \xrightarrow{\alpha} 3 \xrightarrow{\beta} 4 \xrightarrow{f_2} 30 \xrightarrow{\alpha} 31 \xrightarrow{\delta} 29 \xrightarrow{\beta} 24 \xrightarrow{\alpha} 25 \xrightarrow{\delta} 29 \xrightarrow{\beta} 24 \xrightarrow{\alpha} 25 \cdots$, cycling the states 24,25, and 29 infinitely. This corresponds to the sequence $q_0 \xrightarrow{\alpha} q_1 \xrightarrow{\beta} q_2 \xrightarrow{\alpha} q_3 \xrightarrow{\delta} q_4 \xrightarrow{\beta} q_2 \xrightarrow{\alpha} q_3 \xrightarrow{\delta} q_4 \xrightarrow{\beta} q_2 \xrightarrow{\alpha} q_3 \cdots$, which never reaches an F_2-certain state.

These examples motivates us to formally define a new concept of Semi-asynchronous diagnosablity as follows:

Definition 1: (F_i-Semi-Asynchronous Diagnosability) The plant G with the live language, $\mathcal{L}_G$, is said to be F_i-semi-asynchronously diagnosable with respect to the failure type Σ_{f_i}, the natural projection P, and the initial set of estimated states, X_s, if and only if

- for all $s_2 \in Post(X_s)$ and $f \in \Sigma_{Fi}$ with $f \in s_2$, there exists an upper-bound $n_i \in \mathbb{N}$ such that for any s_1, s_3 with $s_1.s_2 \in \mathcal{L}_G$, $s_3 \in \mathcal{L}_{G/s_1.s_2}$, $\|s_3\| \geq n_i$, the following condition holds:

$$\begin{aligned}\{\forall u, v \in \Sigma^* \ \ with \ \ u.v \in \mathcal{L}_G, u \in Pre(X_s) \ and \\ v \in P^{-1}_{ext(\mathcal{L}_G)}(P(s_2.s_3))\} \Rightarrow f \in uv\end{aligned} \tag{4}$$

- for all $s_1 \in Pre(X_s)$ and $f \in \Sigma_{Fi}$ with $f \in s_1$, there exists an upper-bound $n_i \in \mathbb{N}$ such that for any $s_2 \in \mathcal{L}_{G/s_1}$ with $\|s_2\| \geq n_i$, the following condition holds:

$$\begin{aligned}\{\forall u, v \in \Sigma^* \ with \ u.v \in \mathcal{L}_G, u \in Pre(X_s) \ and \\ v \in P^{-1}_{ext(\mathcal{L}_G)}(P(s_2))\} \Rightarrow f \in uv\end{aligned} \tag{5}$$

where the first case refers to the situations that failures occur post-diagnoser activation, while the second case captures the situations that failures occur pre-diagnoser activation.

Using this definition, we can formally verify that the system G_2 in Example 2, is not F_2-semi-asynchronously diagnosable. Assume that the actual state of system G_2 is at state 2 at the time of diagnoser activation and the system G_2 runs the following sequence $2 \xrightarrow{\alpha} 3 \xrightarrow{\beta} 4 \xrightarrow{f_2} 30 \xrightarrow{\alpha} 31 \xrightarrow{\delta} 29 \xrightarrow{\beta} 24 \xrightarrow{\alpha} 25 \xrightarrow{\delta} 29 \xrightarrow{\beta} 24 \xrightarrow{\alpha} 25 \cdots$, cycling the states 24, 25, and 29 infinitely. Let $s_1 = \alpha$, $s_2 = \alpha\beta f_2 \alpha\delta$, where $f_2 \in$

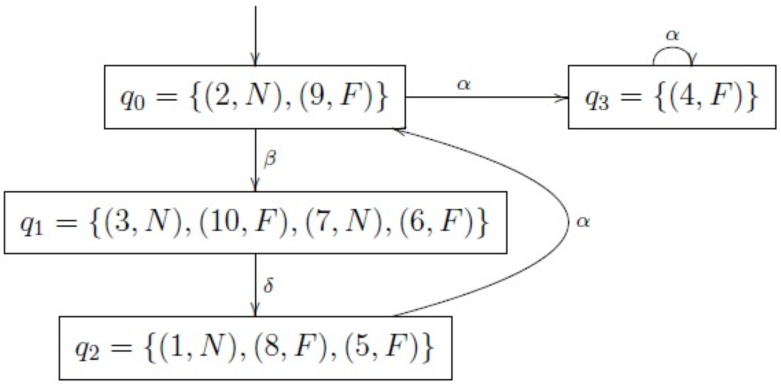

Fig. 2: The constructed diagnoser for DES plant G_1 with $X_s = \{2, 9\}$

s_2, $s_1.s_2 = \alpha\alpha\beta f_2\alpha\delta \in \mathcal{L}_G$. Also consider $s_3 = (\beta\alpha\delta)^* \in \mathcal{L}_{G/s_1.s_2}$, which is an arbitrarily large string. Now, for another run in the system $1 \xrightarrow{\alpha} 2 \xrightarrow{\alpha} 3 \xrightarrow{\beta} 4 \xrightarrow{\alpha} 5 \xrightarrow{\delta} 6 \xrightarrow{\beta} 4 \xrightarrow{\alpha} 5 \xrightarrow{\delta} 6 \xrightarrow{\beta} 4 \xrightarrow{\alpha} 5 \cdots$, let $u = \alpha$, $v = \alpha\beta(\alpha\delta\beta)^*$, where $u.v \in \mathcal{L}_G$, $u \in Pre(X_s)$, $v \in P^{-1}_{ext(\mathcal{L}_G)}(P(s_2.s_3))$. However, $f \notin uv$, which violates the semi-asynchronous diagnosability, given in (4). Therefore system G_2 is not semi-asynchronously diagnosable with respect to the give X_s.

Remark 2: Whether or not F_i-semi-asynchronously diagnosable, we can always construct the diagnoser G_d for the plant G. If the plant G is F_i-semi-asynchronously diagnosable, then the constructed diagnoser can determine if a fault $f \in \Sigma_{f_i}$ has occurred or not in a finite number of transitions. Therefore, it is preferred to have the plant G F_i-semi-asynchronously diagnosable. However, if the plant G is not F_i-semi-asynchronously diagnosable, still the diagnoser G_d can be constructed and can provide its best estimation of the failures' status in G, though in some cases there might be an ambiguity in the occurrence of failures of type Σ_{f_i}, and the diagnoser cannot resolve the ambiguity even after observing a large number of transitions in G.

V. Conclusion

In this paper, a systematic and analytical approach was developed to construct a diagnoser capable of diagnosing system fault occurrences. A unique feature of this diagnoser is that it may be activated asynchronously during system operation with respect to a set of estimated system state locations. The constructed diagnoser is able to diagnose fault occurrences in the system without access to information on system operation prior to the system reaching estimated states. Future study directions of this work may include examining the scalability of the proposed approach to extend the proposed framework to a decentralized and distributed diagnosis architectures, as well as developing systematic methods to verify the semi-asynchronous diagnosability of a given system.

Acknowledgment

The authors would like to acknowledge the support from Air Force Research Laboratory and OSD for sponsoring this research under agreement number FA8750-15-2-0116. 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, OSD or the U.S. Government.

References

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

[2] Z. Gao, C. Cecati, and S. X. Ding, "A survey of fault diagnosis and fault-tolerant techniques—part i: Fault diagnosis with model-based and signal-based approaches," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 6, pp. 3757–3767, 2015.

[3] I. Hwang, S. Kim, Y. Kim, and C. E. Seah, "A survey of fault detection, isolation, and reconfiguration methods," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 3, pp. 636–653, 2010.

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

[5] 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.

[6] 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.

[7] 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.

[8] S. H. Zad, R. H. Kwong, and W. M. Wonham, "Fault diagnosis in discrete-event systems: framework and model reduction," *IEEE Transactions on Automatic Control*, vol. 48, no. 7, pp. 1199–1212, 2003.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

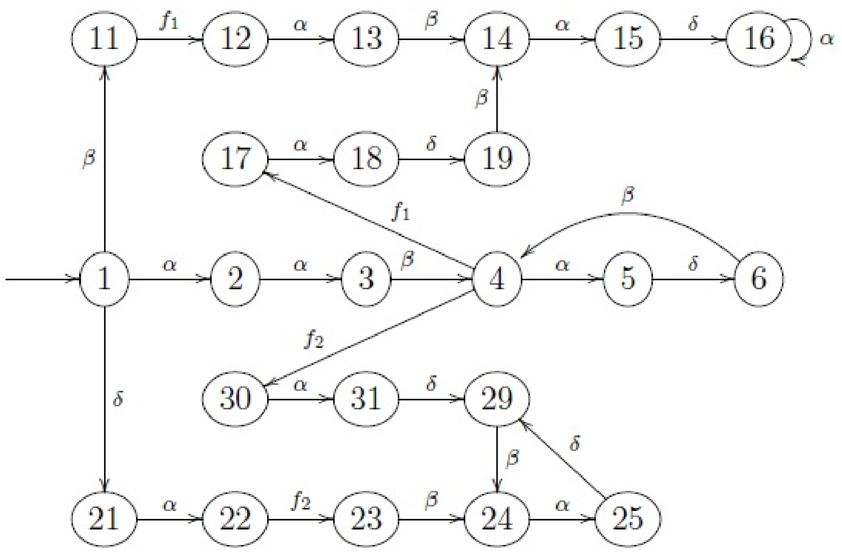

Fig. 3: The DES plant G_2.

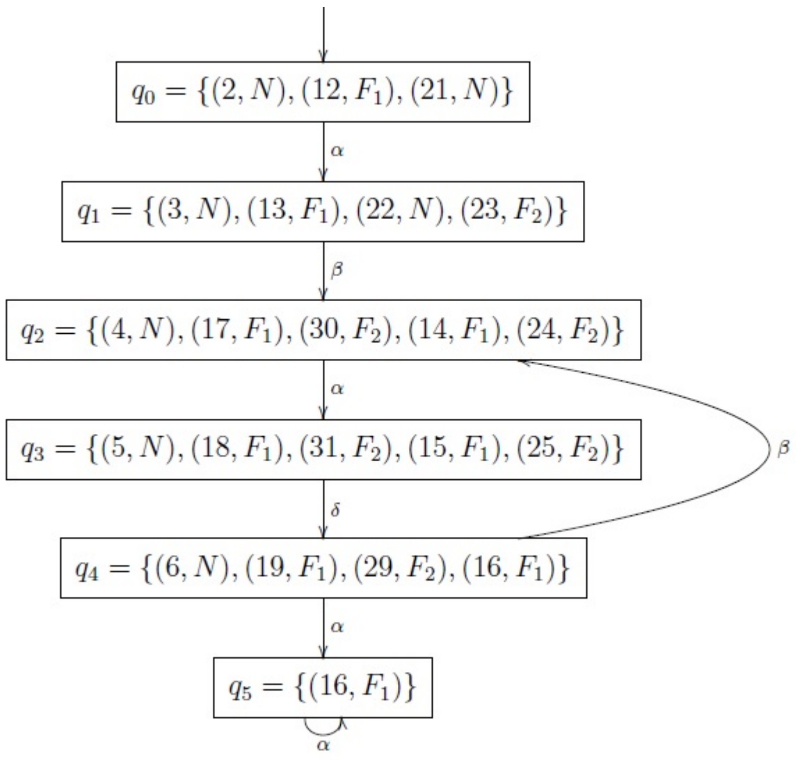
$q_0 = \{(2, N), (12, F_1), (21, N)\}$
$q_1 = \{(3, N), (13, F_1), (22, N), (23, F_2)\}$
$q_2 = \{(4, N), (17, F_1), (30, F_2), (14, F_1), (24, F_2)\}$
$q_3 = \{(5, N), (18, F_1), (31, F_2), (15, F_1), (25, F_2)\}$
$q_4 = \{(6, N), (19, F_1), (29, F_2), (16, F_1)\}$
$q_5 = \{(16, F_1)\}$
α
β
δ

Fig. 4: The constructed diagnoser for DES plant G_2 with $X_s = \{2, 12, 21\}$