Conditional Synchronized Diagnoser for Modular Discrete-Event Systems

Felipe G. Cabral, Maria Z. M. Veras and Marcos V. Moreira
COPPE-Electrical Engineering Program, Universidade Federal do Rio de Janeiro, Cidade Universiária, Ilha do Fundão, Rio de Janeiro, 21.945-970, RJ, Brazil

Keywords: Failure Diagnosis, Modular Systems, Automata, Petri Nets.

Abstract: In general, systems are formed by the composition of several modules, and may exhibit a large number of states. The growth of the global system model with the number of components leads to a high computational cost for diagnosis techniques. In order to circumvent this problem, in a recent work, a diagnosis scheme based on the observation of the nonfailure behavior model of the system components, and their synchronization due to observable events, is proposed. Although the computation of the global system model for diagnosis is avoided, the estimated observed nonfailure language in this scheme can be a larger set than the actual observed nonfailure language of the system, which leads to the notion of synchronous diagnosability. This scheme is implemented using a diagnoser, called synchronized Petri net diagnoser (SPND). In this work, we propose the addition of conditions for the observable transitions of the SPND, leading to a conditional synchronized Petri net diagnoser (CSPND). We show that the addition of such conditions can cause a decrease in the observed nonfailure language, and systems that are not synchronously diagnosable can be conditionally synchronously diagnosable, and the delay bound can be smaller than using the synchronous diagnosis scheme.

1 INTRODUCTION

Several works in the literature address the problem of failure diagnosis of discrete-event systems (DESs) (Sampath et al., 1995; Sampath et al., 1996; Qiu and Kumar, 2006; Carvalho et al., 2011; Carvalho et al., 2012; Basilio et al., 2012; Fanti et al., 2013; Cabasino et al., 2010; Cabasino et al., 2013; Carvalho et al., 2013; Zaytoon and Lafortune, 2013; Cabral et al., 2015b; Tomola et al., 2016; Santoro et al., 2017). In the seminal work (Sampath et al., 1995), a centralized diagnoser for DESs, constructed based on the plant model, is proposed. However, in general, systems are formed by the parallel composition of several subsystems, local components or modules, and the state space of the plant model grows, in the worst-case, exponentially with its number of subsystems. In order to avoid the use of the global plant model for diagnosis, several failure diagnosis schemes that take advantage of the modularity of systems have been proposed in the literature (Debouk et al., 2002; Contant et al., 2006; Zhou et al., 2008; Kan John et al., 2010). In these works, different modular diagnosability definitions are introduced and local diagnosers are proposed to detect the occurrence of failure events. The diagnosis decision of the global system is determined based solely on the observations of the failure module.

In (García et al., 2006), a different approach for modular diagnosis is proposed. Differently from (Debouk et al., 2002; Contant et al., 2006; Zhou et al., 2008; Kan John et al., 2010), the method presented in (García et al., 2006) consists of splitting the global plant model into subsystems, constructing a minimum controller for each subsystem, and then constructing a local diagnoser for each subsystem composed with its minimum controller. In (Schmidt, 2013), an incremental abstraction-based approach for the verification of modular language diagnosability of DESs is proposed, and the differences between the online diagnosis methods presented in (Debouk et al., 2002; Contant et al., 2006; Zhou et al., 2008) are reviewed.

More recently, in (Cabral et al., 2015a; Cabral and Moreira, 2017), a new approach for online failure diagnosis of modular DESs modeled as automata is proposed. Differently from (Debouk et al., 2002; Contant et al., 2006; Zhou et al., 2008; García et al., 2006; Kan John et al., 2010), a centralized synchronized Petri net diagnoser (SPND) is proposed. The SPND is formed by Petri net observers, constructed from the nonfailure behavior models of the system

Cabral, F., Veras, M. and Moreira, M.
Conditional Synchronized Diagnoser for Modular Discrete-event Systems.
DOI: 10.5220/0006435700880097
In *Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2017) - Volume 2*, pages 88-97
ISBN: Not Available

components, and provides a superset of the state estimate of the global system. The Petri net observers are naturally synchronized by the observable events executed by the plant, and if the observation of a trace is not recognized in the SPND, *i.e.*, if the observation of a trace executed by the system does not belong to the nonfailure behavior of at least one component of the system, the occurrence of the failure event is indicated by using a failure detection logic. In (Cabral et al., 2015a; Cabral and Moreira, 2017), the authors show that if two or more components have unobservable events in common, then the estimated nonfailure observed language can be a larger set then the actual observable nonfailure language of the system. This fact can increase the delay bound for synchronous diagnosis compared with the traditional diagnosis scheme or, in the worst-case, the failure event is not synchronously diagnosable.

In this work, we propose a modification in the Petri net observers that form the SPND. This modification relies on the addition of conditions to the transitions of the Petri net observers, such that if an event is observed by the diagnoser, the Petri net observers update their state estimate only if the occurrence of the event is possible in the nonfailure model of the global system, leading to the conditional synchronized Petri net diagnoser (CSPND). If an event is observed, and the transitions labeled with this event cannot occur in the nonfailure behavior model of the system, then the failure event has certainly occurred, and it is diagnosed by the CSPND. In this diagnosis scheme, the estimated observed nonfailure language can be a smaller set than the estimated observed nonfailure language obtained by using the synchronous diagnosis scheme. In addition, in the worst-case, a modular system can be conditionally synchronously diagnosable and not synchronously diagnosable. In this regard, we introduce the definition of conditional synchronous diagnosability of the language of a modular system with respect to the languages of its modules. The verification of this property can be done by using the algorithm proposed in (Cabral et al., 2015a; Cabral and Moreira, 2017). An example is used throughout the paper to illustrate our results.

This paper is organized as follows. In Section 2, we present some preliminary concepts, including the definitions of synchronous diagnosability of modular DESs and synchronized Petri net diagnoser (SPND). In Section 3, we present the conditional synchronized Petri net diagnoser (CSPND). Finally, in Section 4, the conclusions are drawn.

2 PRELIMINARIES

2.1 Notation and Definitions

Let $G = (Q, \Sigma, f, \Gamma, q_0)$ denote the automaton model of a DES, where Q is the state-space, Σ is the finite set of events, $f : Q \times \Sigma^\star \to Q$ is the transition function, where $\Sigma^\star$ is the Kleene-closure of Σ, $\Gamma : Q \to 2^\Sigma$ is the feasible event function, and q_0 is the initial state of the system. For the sake of simplicity, the feasible event function will be omitted unless stated otherwise. The language generated by G, $\mathcal{L}(G)$, is denoted in this paper by L. The accessible part of G, denoted by $Ac(G)$ is obtained as usual (Cassandras and Lafortune, 2008).

Let G_1 and G_2 be two automata. Then, $G_1 \times G_2$ and $G_1 \| G_2$ denote the product and the parallel composition of G_1 and G_2, respectively (Cassandras and Lafortune, 2008).

The projection operation $P_s^l : \Sigma_l^\star \to \Sigma_s^\star$, where $\Sigma_s \subset \Sigma_l$ is defined as $P_s^l(\varepsilon) = \varepsilon$, $P_s^l(\sigma) = \sigma$, if $\sigma \in \Sigma_s$ or $P_s^l(\sigma) = \varepsilon$, if $\sigma \in \Sigma_l \setminus \Sigma_s$, where $\setminus$ denotes set difference, and $P_s^l(s\sigma) = P_s^l(s)P_s^l(\sigma)$, for all $s \in \Sigma_l^\star$, and $\sigma \in \Sigma_l$. The projection can also be applied to language L, by applying the projection to all traces $s \in L$. The inverse projection $P_s^{l^{-1}} : \Sigma_s^\star \to 2^{\Sigma_l^\star}$ when applied to a trace $s \in \Sigma_s^\star$ generates all traces of $\Sigma_l^\star$ whose projection is equal to s. The inverse projection can also be applied to languages.

Let us now suppose that the event set of G is partitioned as $\Sigma = \Sigma_o \dot\cup \Sigma_{uo}$, where Σ_o and Σ_{uo} denote, respectively, the set of observable and unobservable events, and let $\Sigma_f \subseteq \Sigma_{uo}$ denote the set of failure events. In this paper, we assume, without loss of generality, that there is only one failure event, *i.e.*, $\Sigma_f = \{\sigma_f\}$.

Definition 1. *(Failure and normal traces) A failure trace is a sequence of events s such that σ_f is one of the events that form s. A normal trace, on the other hand, does not contain the event σ_f.* □

The normal language $L_N \subset L$ denotes the set of all normal traces of L, and the subautomaton of G that generates L_N is denoted by G_N. Thus, the set of all traces generated by the system that contain σ_f is $L_F = L \setminus L_N$.

Let $P_o : \Sigma^\star \to \Sigma_o^\star$ be a projection. Then, it is always possible to obtain a deterministic automaton whose generated language is equal to $P_o(L)$. This automaton is the observer of G, denoted by $Obs(G, \Sigma_o) = (Q_{obs}, \Sigma_o, f_{obs}, \Gamma_{obs}, q_{0,obs})$ (Cassandras and Lafortune, 2008).

A Petri net is another formalism usually used to model a DES (Cassandras and Lafortune, 2008; Davi

and Alla, 2005). Let $\mathcal{N} = (P, T, Pre, Post, x_0)$ denote a Petri net where P is the set of places, T is the set of transitions, $Pre : (P \times T) \to \mathbb{N}$ is the function of arcs that connect places to transitions, $Post : (T \times P) \to \mathbb{N}$ is the function of arcs that connect transitions to places, and $x_0 : P \to \mathbb{N}$ is the initial marking of the system.

The set of places is denoted here by $P = \{p_1, p_2, \ldots, p_n\}$ and the set of transitions by $T = \{t_1, t_2, \ldots, t_m\}$. Thus, $|P| = n$ and $|T| = m$, where $|.|$ denotes cardinality. The set of input places (resp., transitions) of a transition $t_j \in T$ (resp., place $p_i \in P$) is denoted by $I(t_j)$ (resp., $I(p_i)$), and is formed by the places $p_i \in P$ (resp., transitions $t_j \in T$) such that $Pre(p_i, t_j) > 0$ (resp., $Post(t_j, p_i) > 0$).

The number of tokens assigned to a place p_i is represented by $x(p_i)$, where $x : P \to \mathbb{N}$. Thus, the marking of a Petri net is given by the vector $\underline{x} = [x(p_1)\ x(p_2)\ \ldots\ x(p_n)]^T$ formed with the number of tokens of each place p_i, for $i = 1, \ldots, n$. A place $p_i \in P$ is said to be safe if $x(p_i) \leq 1$ for all reachable markings of the Petri net.

A transition t_j is said to be enabled when $x(p_i) \geq Pre(p_i, t_j)$, $\forall p_i \in I(t_j)$. If a transition t_j is enabled for a marking $\underline{x}$, then t_j can fire reaching a new marking $\bar{\underline{x}}$. The evolution of the markings is given by:

$$\bar{x}(p_i) = x(p_i) - Pre(p_i, t_j) + Post(t_j, p_i), i = 1, \ldots, n. \quad (1)$$

A binary Petri net can be defined as a Petri net with a different evolution rule for the place markings reached after the firing of a transition t_j given by (Alayan and Newcomb, 1987):

$$\bar{x}(p_i) = \begin{cases} 0, \text{if } x(p_i) - Pre(p_i, t_j) + Post(t_j, p_i) = 0 \\ 1, \text{if } x(p_i) - Pre(p_i, t_j) + Post(t_j, p_i) > 0 \end{cases}, \quad (2)$$

for $i = 1, \ldots, n$. Notice that in a binary Petri net all places are forced to be safe.

In order to model DESs, events are associated with transitions in the Petri net, leading to the so-called labeled Petri net. A labeled Petri net is the seven-tuple $\mathcal{N}_l = (P, T, Pre, Post, x_0, \Sigma, l)$, where $(P, T, Pre, Post, x_0)$ is a Petri net, Σ is the set of events used to label transitions, and $l : T \to 2^{\Sigma}$ is the transition labeling function that associates a subset of Σ to a transition in T. An enabled transition t_j in a labeled Petri net fires when one of the events associated to t_j occurs.

2.2 Diagnosability of Discrete-Event Systems

The following definition of language diagnosability can be stated (Sampath et al., 1995).

Definition 2. *Let L and $L_N \subset L$ be the live and prefix-closed languages generated by G and G_N, respectively. Then, L is said to be diagnosable with respect to projection $P_o : \Sigma^\star \to \Sigma_o^\star$ and Σ_f if*

$$(\exists z \in \mathbb{N})(\forall s \in L \setminus L_N)(\forall st \in L \setminus L_N, \|t\| \geq z) \Rightarrow (P_o(st) \notin P_o(L_N)),$$

where $\|.\|$ denotes the length of a trace.

According to Definition 2, L is diagnosable with respect to P_o and Σ_f if, for all failure traces st with arbitrarily long length after the occurrence of a failure event, there does not exist a normal trace $s_N \in L_N$, such that $P_o(st) = P_o(s_N)$. Therefore, if L is diagnosable, then it is always possible to identify the occurrence of a failure event after a bounded number of observations of events.

A polynomial-time algorithm to verify language diagnosability is presented in (Moreira et al., 2011).

2.3 Synchronous Diagnosability of Modular Discrete-Event Systems

In (Cabral et al., 2015a; Cabral and Moreira, 2017), the definition of synchronous diagnosability of a modular DES is presented. In order to do so, it is assumed that the system is composed of r modules G_k, $k = 1, \ldots, r$, *i.e.*, the plant $G = \|_{k=1}^{r} G_k$. It is also assumed that the event set of each module G_k can be partitioned as $\Sigma_k = \Sigma_{k,o} \dot{\cup} \Sigma_{k,uo}$, where $\Sigma_{k,o}$ and $\Sigma_{k,uo}$ denote, respectively, the sets of observable and unobservable events of G_k. Moreover, each component has its nonfailure behavior modeled by automaton G_{N_k}, such that the nonfailure behavior of the plant is given by $G_N = \|_{k=1}^{r} G_{N_k}$. The main idea in (Cabral et al., 2015a; Cabral and Moreira, 2017) is to implement observers for each normal part of the modules of the system, which are naturally synchronized with the observable events executed by the plant, and then, using a failure detection logic, identify the occurrence of a failure event. The following definition can be stated.

Definition 3. *Let L and $L_N \subset L$ be the languages generated by G and G_N, respectively, and let $L_F = L \setminus L_N$. Consider that the system G is composed of r modules, such that $G_N = \|_{k=1}^{r} G_{N_k}$, where G_{N_k} is the automaton that models the normal behavior of G_k, and let L_{N_k} denote the language generated by G_{N_k}, for $k = 1, \ldots, r$. Then, L is said to be synchronously diagnosable with respect to L_{N_k}, $P_k : \Sigma^\star \to \Sigma_k^\star$, for $k = 1, \ldots, r$, $P_o : \Sigma^\star \to \Sigma_o^\star$, and Σ_f if*

$$(\exists z \in \mathbb{N})(\forall s \in L_F)(\forall st \in L_F, \|t\| \geq z) \Rightarrow (P_o(st) \notin \cap_{k=1}^{r} P_o(P_k^{-1}(L_{N_k}))).$$

Notice that Definition 3 of synchronous diagnosability of a language L is equivalent to the standard definition of diagnosability (Definition 2) of a language $L_a = L_F \cup L_{N_a}$, where L_{N_a} is such that $P_o(L_{N_a}) = \cap_{k=1}^{r} P_o(P_k^{-1}(L_{N_k}))$.

It is important to remark that since $P_o(L_{N_a}) \supseteq P_o(L_N)$, then diagnosability is a necessary condition for synchronous diagnosability, but it is not sufficient, *i.e.*, a system can be diagnosable but not synchronously diagnosable. Moreover, since $P_o(L_{N_a}) \supseteq P_o(L_N)$, the delay bound for synchronous diagnosis can be greater that the delay bound for diagnosis. In (Cabral et al., 2015a; Cabral and Moreira, 2017) it is also shown that if there do not exist unobservable events in common between the components, *i.e.*, if $\Sigma_{i,uo} \cap \Sigma_{j,uo} = \emptyset$ for all $i, j \in \{1, \ldots, r\}$, and $i \neq j$, then $P_o(L_N) = P_o(L_{N_a})$, and diagnosability becomes a necessary and sufficient condition for synchronous diagnosability.

2.4 Synchronous Diagnosability Verification

In (Cabral et al., 2015a; Cabral and Moreira, 2017), a method for the verification of synchronous diagnosability of modular discrete event systems is proposed. The method is based on the comparison between automaton G_N^R, whose observable language is equal to $P_o(L_{N_a})$, and G_F, that models the failure behavior of the system G. Automaton G_N^R is constructed in two steps: (i) compute automata $G_{N_k}^R$ from automata G_{N_k} by renaming its unobservable events using function $R_k : \Sigma_{N_k} \to \Sigma_{N_k}^R$, defined as:

$$R_k(\sigma) = \begin{cases} \sigma, & \text{if } \sigma \in \Sigma_{k,o} \\ \sigma_{R_k}, & \text{if } \sigma \in \Sigma_{k,uo} \end{cases}, \tag{3}$$

and; (ii) compute $G_N^R = \|_{k=1}^{r} G_{N_k}^R$.

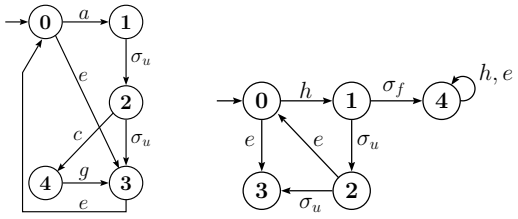
In the synchronous diagnosis scheme, the synchronization of unobservable events of the system modules is lost, which leads to the possible growth of the estimated normal language by using this scheme. In order to model the observation of this augmented language, the unobservable events of the normal behavior automaton models of the system components G_{N_k} are renamed using the renaming function (3), which leads to automata $G_{N_k}^R$. Thus, since the unobservable events of $G_{N_k}^R$ are private events, for $k \in \{1, \ldots, r\}$, the observed language of automaton $G_N^R = \|_{k=1}^{r} G_{N_k}^R$ models the observation of the augmented normal language for synchronous diagnosis, *i.e.*, $P_o(L_{N_a}) = P_o^R(\mathcal{L}(G_N^R))$, where $P_o^R : \Sigma_N^R \to \Sigma_o$, with $\Sigma_N^R = \cup_{k=1}^{r} \Sigma_{N_k}^R$.

According to Definition 3, in order to verify if the language L of a modular system is synchronously diagnosable, it is necessary to verify if the projection $P_o : \Sigma^\star \to \Sigma_o^\star$ of any failure trace st, with arbitrarily long length after the occurrence of the failure event σ_f, belongs to $P_o(L_{N_a})$. If the answer is yes, than L is not synchronously diagnosable with respect to L_{N_k}, $P_k : \Sigma^\star \to \Sigma_k^\star$, for $k = 1, \ldots, r$, $P_o : \Sigma^\star \to \Sigma_o^\star$, and Σ_f. Thus, the synchronous diagnosability verification is carried out by comparing automaton G_N^R with the failure behavior automaton G_F. Automaton G_F is obtained from G following the algorithm proposed in (Moreira et al., 2011). The event set of G_F is Σ, and its states are labeled with N or F, such that if a state of G_F has the label F, then this state is reachable after the occurrence of the failure event σ_f.

Since the unobservable events of $G_N^R = \|_{k=1}^{r} G_{N_k}^R$ are private events with respect to G_F, and since $P_o(L_{N_a}) = P_o^R(\mathcal{L}(G_N^R))$, the verification of synchronous diagnosability can be done by searching for cyclic paths in $G_V = G_N^R \| G_F$ formed by states labeled with F and with at least one event from Σ. The language L is synchronously diagnosable if and only if there does not exist a cyclic path with these characteristics in G_V. In the sequel, we present an example that illustrates the synchronous diagnosability verification.

Example 1. *Consider the system $G = G_1 \| G_2$, where G_1 and G_2 are depicted in Figure 1. The set of events of G_1 and G_2 are $\Sigma_1 = \{a, c, e, g, \sigma_u\}$ and $\Sigma_2 = \{e, h, \sigma_u, \sigma_f\}$, where $\Sigma_{1,o} = \{a, c, e, g\}$, $\Sigma_{2,o} = \{e, h\}$, $\Sigma_{1,uo} = \{\sigma_u\}$, $\Sigma_{2,uo} = \{\sigma_u, \sigma_f\}$, and σ_f is the failure event. In Figures 2 and 3, we present automata G_N and G_F, respectively, obtained by following the method presented in (Moreira et al., 2011). Automaton G is equal to automaton G_F, except for the labels N and F. In order to verify the synchronous diagnosability, it is necessary to obtain the automaton models of the normal behavior of the components of the system G_{N_1} and G_{N_2}, which can be seen in Figure 4. In the sequel, automata $G_{N_1}^R$ and $G_{N_2}^R$, depicted in Figure 5, are computed by applying the renaming function (3) to automata G_{N_1} and G_{N_2}, respectively. Automaton $G_N^R = G_{N_1}^R \| G_{N_2}^R$, whose observed generated language is $P_o^R(\mathcal{L}(G_N^R)) = P_o(L_{N_a})$, is shown in Figure 6.*

Notice that the gray states of G_N^R do not belong to G_N and, thus, all observable transitions related to such states can contribute to the growth of the estimated normal language obtained by using the synchronous diagnosis scheme. Finally, in order to verify the synchronous diagnosability of the system G, it is necessary to compute the verifier automaton $G_V = G_N^R \| G_F$ and search for cyclic paths formed by states with the label F and at least one event $\sigma \in \Sigma$. Since in G_V there is a cyclic path that violates the synchronous



(a) G_1 (b) G_2

Figure 1: Automata G_1 and G_2 of Example 1.

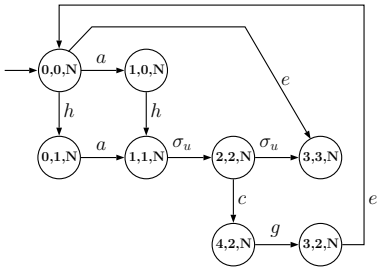


Figure 2: Automaton G_N of Example 1.

diagnosability condition, L is not synchronously diagnosable. It is important to notice that, according to Definition 2, L is diagnosable.

2.5 Synchronized Petri Net Diagnoser

In order to implement the synchronous diagnosis scheme, in (Cabral et al., 2015a; Cabral and Moreira, 2017), the authors propose a synchronized Petri net diagnoser (SPND). The SPND is a centralized diagnoser, consisting of r Petri net state observers that provide the state estimate of the normal behavior of the system components G_{N_k}, for $k = 1,\ldots,r$, and a failure detection logic. If an event that is not feasible in at least one of the current state estimate of a given nonfailure model component, than the failure event is

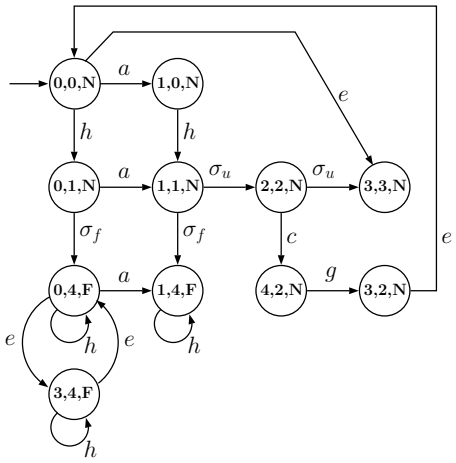
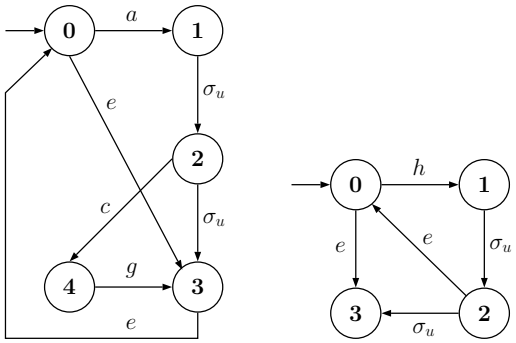
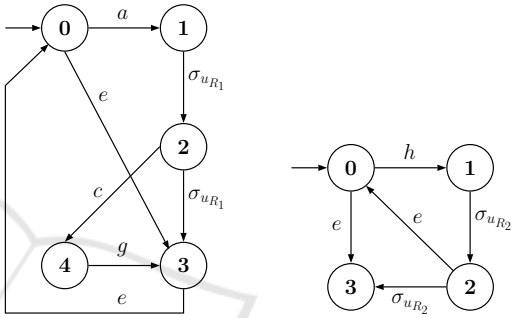


Figure 3: Automaton G_F of Example 1.



(a) G_{N_1} (b) G_{N_2}

Figure 4: Automata G_{N_1} and G_{N_2} of Example 1.



(a) $G^R_{N_1}$ (b) $G^R_{N_2}$

Figure 5: Automata $G^R_{N_1}$ and $G^R_{N_2}$ of Example 1.

diagnosed.

The synchronized Petri net diagnoser $\mathcal{N}_D = (P_D, T_D, Pre_D, Post_D, x_{0,D}, \Sigma_o, l_D)$ is a labeled binary Petri net formed by Petri net state observers $\mathcal{N}_{SO_k} = (P_{SO_k}, T_{SO_k}, Pre_{SO_k}, Post_{SO_k}, x_{0,SO_k}, \Sigma_{k,o}, l_{SO_k})$, for $k = 1,\ldots,r$, where its set of transitions is defined as $T_{SO_k} = T_{k,o} \dot{\cup} T'_{k,o}$, where $T_{k,o}$ is the set of observable transitions of $\mathcal{N}_{SO_k}$, such that each transition $t^i_{k,o} \in T_{k,o}$ corresponds to an observable transition of G_{N_k}, and $T'_{k,o}$ is the set of complementary transitions, whose

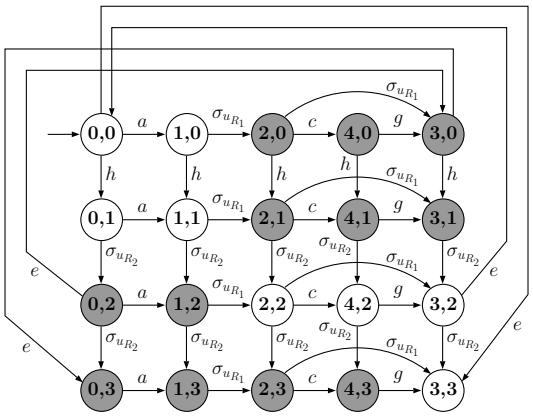


Figure 6: Automaton G^R_N of Example 1.

function is to remove tokens from the places that do not belong to the estate estimate of G_{N_k} after the observation of an event. Consider a state $q_j \in Q$ of G_{N_k}, the complementary transition $t'^{j}_{k,o} \in T'_{k,o}$ is labeled with all observable events that do not belong to the feasible event set of q_j, *i.e.*, $t'^{j}_{k,o}$ is labeled with all events of $\Sigma_{k,o} \setminus \Gamma(q_j)$. Therefore, if an event that is not in the feasible event set of a state that belongs to the current state estimate of G_{N_k} is observed, then this state does not belong to the state estimate after the observation of this event. In order to correctly implement this behavior, the complementary transition of the place associated with this state of G_{N_k} will fire and the token of its input place is removed.

After the Petri net state observers $\mathcal{N}_{SO_k}$, for $k = 1,\ldots,r$ have been computed, the next step to obtain $\mathcal{N}_D$ is to build the Petri nets $\mathcal{N}_{D_k}$ by adding a transition t_{f_k} to $\mathcal{N}_{SO_k}$, labeled with the always occurring event. All places of $\mathcal{N}_{D_k}$ are connected to t_{f_k} by inhibitor arcs, such that if all places of $\mathcal{N}_{D_k}$ lose all their tokens, transition t_{f_k} is enabled and fires since it is labeled with the always occurring event. Finally, the synchronized Petri net diagnoser $\mathcal{N}_D$ is obtained by grouping all $\mathcal{N}_{D_k}$ into one Petri net, and adding a place p_F to represent the diagnosis of the failure event. The place p_F is an output place of all transitions t_{f_k} such that if one of the Petri nets $\mathcal{N}_{D_k}$ loses all its tokens, transition t_{f_k} fires and a token is assigned to place p_F, indicating the occurrence of the failure event. In the sequel, we present an example of the SPND for the system $G = G_1 \| G_2$, where G_1 and G_2 are presented in Figure 1.

Example 2. *Consider again the modular system $G = G_1 \| G_2$, where G_1 and G_2 are shown in Figure 1. Although, as pointed out in Example 1, L is not synchronously diagnosable, let us construct the SPND for this example. Following the method presented in (Cabral et al., 2015a; Cabral and Moreira, 2017), the SPND depicted in Figure 7 is obtained. Notice that if the system generates the failure trace $h\sigma_f eh(eh)^\star$, the failure event σ_f is not diagnosed since none of the Petri nets $\mathcal{N}_{D_1}$ or $\mathcal{N}_{D_2}$ loses all their tokens.*

When the system generates the failure trace $h\sigma_f eh(eh)^\star$, as a consequence of the occurrence of event h, observable transition $t_{2,1}$ of Petri net $\mathcal{N}_{D_2}$ fires, removing a token from place $0N_2$ and adding a token to places $1N_2$, $2N_2$, and $3N_2$. Then, when event e is observed, transition $t_{1,2}$ of $\mathcal{N}_{D_1}$ and transition $t_{2,4}$ of $\mathcal{N}_{D_2}$ fire, removing a token from places $0N_1$ and $2N_2$, and adding a token to places $3N_1$ and $0N_2$. However, transition $((0,2,N),e,(3,0,N))$ does not exist in automaton G_N, as shown in Figure 2, and therefore, the simultaneous firing of transitions $t_{1,2}$ and $t_{2,4}$ should be avoided. Indeed, it can be seen in Figure 2 that event e is feasible only in states $(0,0,N)$ or $(3,2,N)$ of G_N, i.e., if the system is in state 0 in automaton G_1 and state 0 in automaton G_2, or in state 3 in automaton G_1 and state 2 in automaton G_2. Thus, if we add a condition to the firing of transition $t_{1,2}$, associated with the marking of place $0N_2$ of $\mathcal{N}_{D_2}$, and a condition to the firing of $t_{2,4}$ associated with the marking of place $3N_1$ of $\mathcal{N}_{D_1}$, the simultaneous firing of $t_{1,2}$ and $t_{2,4}$ would be avoided.

In the following section, we propose a modification of the SPND in order to decrease the estimated normal observed language for synchronous diagnosis.

3 CONDITIONAL SYNCHRONIZED PETRI NET DIAGNOSER

In this paper, we propose a modification in the SPND, in order to allow an observable transition to fire in a state observer Petri net only if this transition also exists in the normal automaton of the system G_N, leading to the conditional synchronized Petri net diagnoser (CSPND) $\mathcal{N}_{D,c}$. In order to do so, we add conditions to the observable transitions of the Petri net state observers $\mathcal{N}_{SO_k}$, for $k = 1,\ldots,r$, based on G_N. These conditions are boolean expressions associated with places of the Petri net state observers $\mathcal{N}_{SO_j}$, for $j = 1,\ldots,r$, and $j \neq k$.

As illustrated in Example 2, the addition of conditions to the observable transitions of $\mathcal{N}_{SO_k}$ based on the normal automaton model G_N can contribute to the diagnosis of the failure event. This leads to conditional Petri net state observers $\mathcal{N}^c_{SO_k}$, where each transition is labeled with observable events and conditions that depend on the marking of places of Petri nets $\mathcal{N}^c_{SO_j}$, for $j = 1,\ldots,r$, $j \neq k$. These conditions are selected based on the possible observable transitions of G_N. Thus, Petri net $\mathcal{N}^c_{SO_k}$ is an eight-tuple $\mathcal{N}^c_{SO_k} = (P_{SO_k}, T^c_{SO_k}, Pre^c_{SO_k}, Post^c_{SO_k}, x_{0,SO_k}, \Sigma_{k,o}, C_{SO_k}, l^c_{SO_k})$, where $l^c_{SO_k} : T^c_{SO_k} \to 2^{\Sigma_{k,o}} \times C_{SO_k}$ is a labeling function that associates to each transition in $T^c_{SO_k}$ a set of events from $2^{\Sigma_{k,o}}$ and a condition C from C_{SO_k}, associated with the places of Petri nets $\mathcal{N}^c_{SO_j}$, for $j = 1,\ldots,r$, $j \neq k$.

In the sequel, we present Algorithm 1 for the computation of the conditional synchronized Petri net diagnoser $\mathcal{N}_{D,c}$.

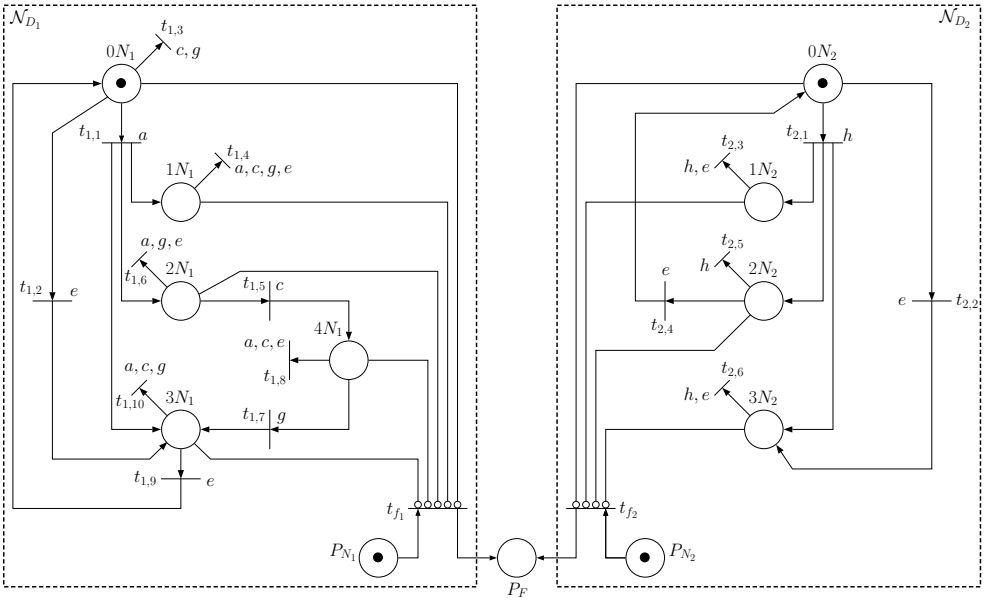


Figure 7: Synchronized Petri net diagnoser of Example 2.

Algorithm 1. *Conditional synchronized Petri net diagnoser $\mathcal{N}_{D,c}$.*

Input: *Petri net state observers $\mathcal{N}_{SO_k} = (P_{SO_k}, T_{SO_k}, Pre_{SO_k}, Post_{SO_k}, x_{0,SO_k}, \Sigma_{k,o}, l_{SO_k})$, for $k = 1, \ldots, r$, and automaton G_N.*

Output: *Conditional synchronized Petri net diagnoser $\mathcal{N}_{D,c}$.*

1: Compute the conditional state observer Petri nets $\mathcal{N}^c_{SO_k} = (P_{SO_k}, T^c_{SO_k}, Pre^c_{SO_k}, Post^c_{SO_k}, x_{0,SO_k}, \Sigma_{k,o}, C_{SO_k}, l^c_{SO_k})$, as follows:

1.1: Let $T^{c'}_{SO_k} = \emptyset$. Create a new transition t^c_k for each transition $\tilde{q}_{N_k} = f_{N_k}(q_{N_k}, \sigma)$ defined in G_{N_k}, where $\tilde{q}_{N_k}, q_{N_k} \in Q_{N_k}$, and $\sigma \in \Sigma_{k,o}$. For each transition t^c_k, define $Pre^c_{SO_k}(p_k, t^c_k) = 1$, if p_k corresponds to state q_{N_k}, and $Pre^c_{SO_k}(p_k, t^c_k) = 0$, otherwise, and do $T^{c'}_{SO_k} = T^{c'}_{SO_k} \cup \{t^c_k\}$.

1.2: Define $T^c_{SO_k} = T_{SO_k} \cup T^{c'}_{SO_k}$.

1.3: Define $Pre^c_{SO_k} : P_{SO_k} \times T^c_{SO_k} \to \mathbb{N}$ and $Post^c_{SO_k} : T^c_{SO_k} \times P_{SO_k} \to \mathbb{N}$ such that $Pre^c_{SO_k}(p_k, t_k) = Pre_{SO_k}(p_k, t_k)$, and $Post^c_{SO_k}(t_k, p_k) = Post_{SO_k}(t_k, p_k)$ for all $p_k \in P_{SO_k}$ and $t_k \in T_{SO_k}$, and $Post^c_{SO_k}(t^c_k, p_k) = Post^c_{SO_k}(t^c_k, p_k) = 0$, for all $t^c_k \in T^{c'}_{SO_k}$ and $p_k \in P_{SO_k}$.

1.4: Define $l^c_{SO_k} : T^c_{SO_k} \to 2^{\Sigma_{k,o}} \times C_{SO_k}$ as:

$$l^c_{SO_k}(t_{k,i}) = \begin{cases} (l_{SO_k}(t_{k,i}), C_{k,i}), \text{ if } t_{k,i} \in T_{k,o} \cup T^{c'}_{SO_k} \\ (l_{SO_k}(t_{k,i}), 1), \quad \text{otherwise,} \end{cases} \tag{4}$$

with

$$C_{k,i} = \begin{cases} [\bigwedge^r_{j=1, j\neq k}(\bigvee_\ell p_{j,\ell})], \text{ if } t_{k,i} \in T_{k,o} \\ \overline{[\bigwedge^r_{j=1, j\neq k}(\bigvee_\ell p_{j,\ell})]}, \text{ if } t_{k,i} \in T^{c'}_{SO_k} \end{cases} \tag{5}$$

for all places $p_{j,\ell} \in P_{SO_j}$ such that $I(t_{k,i})$ and $p_{j,\ell}$ correspond to states in Q_{N_k} and Q_{N_j} that are the k-th and j-th coordinates of a state $q_N \in Q_N$, respectively, where $f_N(q_N, \sigma)$ is defined for $\sigma \in l_{SO_k}(t_{k,i})$.

1.5: Define the initial marking of $\mathcal{N}^c_{SO_k}$ as $x^c_{0,SO_k} = x_{0,SO_k}$, for $k = 1, \ldots, r$.

2: Compute the Petri net $\mathcal{N}^c_{D_k} = (P^c_{D_k}, T^c_{D_k}, Pre^c_{D_k}, Post^c_{D_k}, In^c_{D_k}, x^c_{0,D_k}, \Sigma_{k,o}, C_{SO_k}, l^c_{SO_k})$, where $In^c_{D_k} : P^c_{D_k} \times T^c_{D_k} \to \{0, 1\}$ denotes the function of inhibitor arcs, as follows:

2.1: Add to $\mathcal{N}^c_{SO_k}$ a transition t_{f_k} labeled with the always occurring event λ. Define $T^c_{D_k} = T_{SO_k} \cup \{t_{f_k}\}$.

2.2: Add to $\mathcal{N}^c_{SO_k}$ a place p_{N_k}, and define $Pre^c_{D_k}(p_{N_k}, t_{f_k}) = 1$. Set $x^c_{0,D_k}(p_{N_k}) = 1$, and define $P^c_{D_k} = P_{SO_k} \cup \{p_{N_k}\}$.

2.3: Define $In^c_{D_k}(p^c_{D_k}, t_{f_k}) = 1$ and $In_{D_k}(p^c_{D_k}, t^c_{SO_k}) = 0$, $\forall p^c_{D_k} \in P^c_{D_k}$ and $\forall t^c_{SO_k} \in T^c_{SO_k}$.

3: Compute the conditional synchronized Petri net diagnoser $\mathcal{N}_{D,c} = (P^c_D, T^c_D, Pre^c_D, Post^c_D,$

$In^c_D, x^c_{0,D}, \Sigma_o, C^c_D, l^c_D$), *as follows:*

3.1: Form a unique Petri net by grouping all Petri nets $\mathcal{N}^c_{D_k}$, *for* $k = 1, \ldots, r$.

3.2: Add a place p_F *and define* $Post^c_D(t_{f_k}, p_F) = 1$, *for* $k = 1, \ldots, r$. *Set* $x^c_{0,D}(p_F) = 0$.

In the following, we present an example of the CSPND $\mathcal{N}_{D,c}$ for the modular system G of Example 1.

Example 3. *Consider the modular system* $G = G_1 \| G_2$, *where* G_1 *and* G_2 *are depicted in Figure 1. Following the steps of Algorithm 1, the conditional synchronized Petri net diagnoser* $\mathcal{N}_{D,c}$, *shown in Figure 8, is constructed. Notice that, if the system generates the failure trace* $h\sigma_f eh(eh)^\star$, *the failure event* σ_f *is diagnosed by the CSPND* $\mathcal{N}_{D,c}$ *after the first observation of event e, since both Petri nets* $\mathcal{N}_{D,c_1}$ *and* $\mathcal{N}_{D,c_2}$ *lose all tokens.*

It is important to notice that the conditions added to $\mathcal{N}_D$ prevent observable transitions that cannot occur in G_N to be considered as belonging to the estimated normal observed behavior of the system. The practical consequence of this fact is a decrease in the observed augmented normal language for synchronous diagnosis $P_o(L_{N_a})$, leading to an observed conditional augmented normal language $P_o(L_{N_{a,c}})$, where $P_o(L_{N_{a,c}}) \subseteq P_o(L_{N_a})$. Moreover, since the observed language of automaton G^R_N is equivalent to $P_o(L_{N_a})$, in order to model the language $P_o(L_{N_{a,c}})$, we have to erase the observable transitions of G^R_N according to G_N, leading to the conditional normal behavior model automaton $G^R_{N_c}$. This can be done by following the steps of the algorithm presented in the sequel.

Algorithm 2. *Conditional normal behavior model*

Input: *Automata* G_N *and* G^R_N.
Output: *Automaton* $G^R_{N_c}$.

1: *Flag the transitions* $f^R_N(q^R_N, \sigma) = q^{R\prime}_N$, *such that* $[(q^R_N \notin Q_N) \vee (q^{R\prime}_N \notin Q_N)] \wedge (\sigma \in \Sigma_o)$ *of* G^R_N.

2: *Compute* $G^{R\prime}_N$ *by eliminating the flagged transitions from* G^R_N.

3: *Compute* $G^R_{N_c} = Ac(G^{R\prime}_N)$.

In the following, we present a theorem that ensures that the removal of observable transitions from G^R_N by Algorithm 2 in order to compute $G^R_{N_c}$ has the same effect as the conditions added to $\mathcal{N}_D$ in order to obtain $\mathcal{N}_{D,c}$.

Theorem 1. *Consider automaton* $G^R_{N_c}$ *obtained by following the steps of Algorithm 2. The observed language of* $G^R_{N_c}$, $P^R_o(\mathcal{L}(G^R_{N_c})) = P_o(L_{N_{a,c}})$, *corresponds to the conditional augmented normal language.*

Proof. In order to prove Theorem 1, we must show that the conditions added to the SPND $\mathcal{N}_D$ have the same effect as erasing the observable transitions of G^R_N to compute automaton $G^R_{N_c}$. Notice that the conditions added to an observable transition in a Petri net state observer $\mathcal{N}_{SO_k}$ only allow this transition to fire if a set of places of the other Petri nets have tokens assigned. This set of places correspond to a set of states of the normal behavior models of the components of the system that form a state in G_N, where this observable event is active. Therefore, this transition can only fire in the CSPND $\mathcal{N}_{D,c}$ if there exists a correspondent observable transition in G_N. ■

Example 4. *Consider automata* G_N *and* G^R_N *depicted in Figures 2 and 6, respectively. Following the steps of Algorithm 2, automaton* $G^R_{N_c}$, *shown in Figure 9, is computed. Notice that there are no observable transitions in* $G^R_{N_c}$ *that do not belong to* G_N. *It is important to remark that the augmented normal trace* $\omega_{a,1} = h\sigma_{R_2} e(h\sigma_{R_2} e)^\star$, *that belongs to* G^R_N, *whose observation in* Σ_o *is* $P^R_o(\omega_{a,1}) = he(he)^\star$ *was eliminated and it is not possible to occur in* $G^R_{N_c}$. *The trace* $\omega_{a,1}$ *has the same observation in* Σ_o *that the failure trace* $st = h\sigma_f e(he)^n$, *which makes the system G not synchronously diagnosable. However, after eliminating the observable transitions of* G^R_N *that do not belong to* G_N, *the normal augmented trace* $\omega_{a,1}$ *is not possible to occur in* $G^R_{N_c}$, *and the failure trace st becomes conditionally synchronously diagnosable.*

It is important to notice that, even with the elimination of the observable transitions from G^R_N *that do not belong to* G_N, *the observable normal language for conditional synchronous diagnosis can still be a larger set than the observable normal language of the system, i.e.,* $P^R_o(G^R_{N_c}) \supseteq P_o(L_N)$. *In order to see this fact, consider the normal augmented trace* $\omega_{a,2} = ha\sigma_{u_{R2}}\sigma_{u_{R1}}\sigma_{u_{R1}} e(ha\sigma_{u_{R2}}\sigma_{u_{R1}}\sigma_{u_{R1}} e)^\star$, *whose observation in* Σ_o *is* $P^R_o(\omega_{a,2}) = hae(hae)^\star$. *Notice that* $P^R_o(\omega_{a,2})$ *does not belong to the observable normal language of the system* $P_o(L_N)$, $P^R_o(\omega_{a,2}) = hae(hae)^\star \notin P_o(L_N)$.

It is important to remark that the observed normal language for the conditional synchronous diagnosis $P_o(L_{N_{a,c}})$ is a superset of the observed normal language of the composed system $P_o(L_N)$. Therefore, even if a modular system is diagnosable, this system is not necessarily conditionally synchronously diagnosable. This leads to the following definition of conditional synchronous diagnosability.

Definition 4. *Let* L *and* $L_N \subset L$ *denote the languages generated by* G *and* G_N, *respectively, and let* $L_F = L \setminus L_N$. *Consider that the system is composed of r modules, such that* $G_N = \|_{k=1}^r G_{N_k}$, *where* G_{N_k} *is the automaton that models the normal behavior of* G_k,

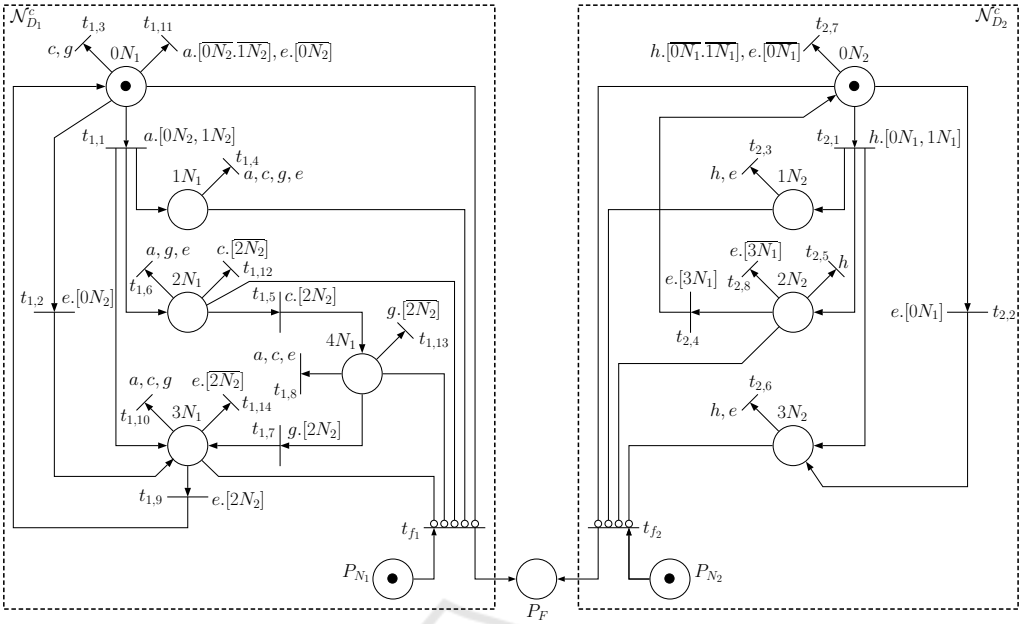


Figure 8: Conditional synchronized Petri net diagnoser $\mathcal{N}_{D,c}$ of Example 3.

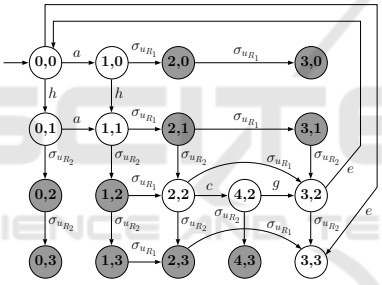


Figure 9: Automaton $G^R_{N_c}$ of Example 4.

and let L_{N_k} denote the language generated by G_{N_k}, for $k = 1,\ldots,r$. Then, L is said to be conditionally synchronously diagnosable with respect to $L_{N_{a,c}}$, $P_o : \Sigma^\star \to \Sigma_o^\star$, and Σ_f if

$$(\exists n \in \mathbb{N})(\forall s \in L_F)(\forall st \in L_F, \|t\| \geq n) \Rightarrow$$
$$(P_o(st) \notin P_o(L_{N_{a,c}})).$$

Notice that, according to Definition 4, in order to verify if a system is conditionally synchronously diagnosable, it is necessary to verify if there is an arbitrarily long length failure trace with the same observation as a normal trace that belongs to $P_o(L_{N_{a,c}})$. Since, as shown in Theorem 1, $P^R_o(\mathcal{L}(G^R_{N_c})) = P_o(L_{N_{a,c}})$, and all unobservable events of $G^R_{N_c}$ are renamed, in order to verify the conditional synchronous diagnosability of a system, the algorithm proposed in (Cabral and Moreira, 2017) for verifying synchronous diagnosability can be used. In order to do so, instead of using $G_V = G^R_N \| G_F$, it is necessary to build $G_{V,c} = G^R_{N_c} \| G_F$ and search for cyclic paths formed with states labeled with F and events there are not renamed. If there exists a cyclic path in $G_{V,c}$ with these characteristics, then the system is not conditionally synchronously diagnosable. It can be seen that for the running example of this paper $G_{V,c}$ does not have cyclic paths whose states are labeled with F and at least one event belongs to Σ. Thus, L is conditionally synchronously diagnosable.

Remark 1. *It is important to remark that since $P_o(L_{N_a}) \supseteq P_o(L_{N_a,c})$, even if a system is synchronously diagnosable, the delay bound for conditional synchronous diagnosis can be smaller than for synchronous diagnosis. In (Cabral and Moreira, 2017), a method for the computation of the delay bound for synchronous diagnosis that uses the verifier automaton G_V is proposed. The same method can be used for the computation of the delay bound for conditional synchronous diagnosis by using the verifier automaton $G_{V,c}$ instead of G_V.*

4 CONCLUSIONS

In this paper, a conditional synchronized Petri net diagnoser is proposed. In order to do so, we propose the addition of conditions to the observable transitions of the synchronized Petri net diagnoser (SPND) presented in (Cabral et al., 2015a; Cabral and Moreira, 2017). We show that the conditional synchronous diagnosis can have a smaller delay bound than the synchronous diagnosis approach. Moreover, systems that

are not synchronously diagnosable can be conditionally synchronously diagnosable.

ACKNOWLEDGEMENTS

This paper was partially supported by the Brazilian Research Council (CNPq) under grant 309084/2014-8.

REFERENCES

Alayan, H. and Newcomb, R. W. (1987). Binary Petri-net relationships. *IEEE Transactions on Circuits and Systems*, CAS-34:565–568.

Basilio, J. C., Lima, S. T. S., Lafortune, S., and Moreira, M. V. (2012). Computation of minimal event bases that ensure diagnosability. *Discrete Event Dynamic Systems: Theory And Applications*, 22:249–292.

Cabasino, M. P., Giua, A., Paoli, A., and Seatzu, C. (2013). Decentralized Diagnosis of Discrete Event Systems using labeled Petri nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(6):1477–1485.

Cabasino, M. P., Giua, A., and Seatzu, C. (2010). Fault detection for discrete event systems using Petri nets with unobservable transitions. *Automatica*, 46():1531–1539.

Cabral, F. G. and Moreira, M. V. (2017). Online failure diagnosis of modular discrete-event systems. *Automatic Control, IEEE Transactions on*. Submitted for publication.

Cabral, F. G., Moreira, M. V., and Diene, O. (2015a). Online fault diagnosis of modular discrete-event systems. In *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*, pages 4450–4455. IEEE.

Cabral, F. G., Moreira, M. V., Diene, O., and Basilio, J. C. (2015b). A Petri net diagnoser for discrete event systems modeled by finite state automata. *IEEE Transactions on Automatic Control*, pages 59–71.

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

Carvalho, L. K., Moreira, M. V., and Basilio, J. C. (2011). Generalized robust diagnosability of discrete event systems. In *18th IFAC World Congress*, pages 8737–8742, Milano, Italy.

Carvalho, L. K., Moreira, M. V., Basilio, J. C., and Lafortune, S. (2013). Robust diagnosis of discrete-event systems against permanent loss of observations. *Automatica*, 49(1):223–231.

Cassandras, C. and Lafortune, S. (2008). *Introduction to Discrete Event System*. Springer-Verlag New York, Inc., Secaucus, NJ.

Contant, O., Lafortune, S., and Teneketzis, D. (2006). Diagnosability of discrete event systems with modular structure. *Discrete Event Dynamic Systems: Theory And Applications*, 16(1):9–37.

Davi, R. and Alla, H. (2005). *Discrete, Continuous and Hybrid Petri Nets*. Springer.

Debouk, R., Malik, R., and Brandin, B. (2002). A modular architecture for diagnosis of discrete event systems. In *41st IEEE Conference on Decision and Control*, pages 417–422, Las Vegas, Nevada USA.

Fanti, M. P., Mangini, A. M., and Ukovich, W. (2013). Fault detection by labeled petri nets in centralized and distributed approaches. *Automation Science and Engineering, IEEE Transactions on*, 10(2):392–404.

García, E., Correcher, A., Morant, F., Quiles, E., and Blasco-Giménez, R. (2006). Centralized modular diagnosis and the phenomenon of coupling. *Discrete Event Dynamic Systems*, 16(3):311–326.

Kan John, P., Grastien, A., and Pencolé, Y. (2010). Synthesis of a distributed and accurate diagnoser. In *21st International Workshop on Principles of Diagnosis (DX-10)*, pages 209–216.

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

Qiu, W. and Kumar, R. (2006). Decentralized failure diagnosis of discrete event systems. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, 36(2):384–395.

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., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. (1996). Failure diagnosis using discrete-event models. *IEEE Trans. on Control Systems Technology*, 4(2):105–124.

Santoro, L. P. M., Moreira, M. V., and Basilio, J. C. (2017). Computation of minimal diagnosis bases of discrete-event systems using verifiers. *Automatica*, 77:93–102.

Schmidt, K. W. (2013). Verification of modular diagnosability with local specifications for discrete-event systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(5):1130–1140.

Tomola, J. H. A., Cabral, F. G., Carvalho, L. K., and Moreira, M. V. (2016). Robust disjunctive-codiagnosability of discrete-event systems against permanent loss of observations. *IEEE Transactions on Automatic Control*. DOI: 10.1109/TAC.2016.2638042.

Zaytoon, J. and Lafortune, S. (2013). Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 37(2):308–320.

Zhou, C., Kumar, R., and Sreenivas, R. S. (2008). Decentralized modular diagnosis of concurrent discrete event systems. In *9th Workshop on Discrete Event Systems*, pages 388–393, Göteborg, Sweden.