ALGORITHMS FOR THE VERIFICATION OF SYNCHRONOUS DIAGNOSABILITY AND COMPUTATION OF THE DELAY BOUND FOR DIAGNOSIS OF MODULAR DISCRETE EVENT SYSTEMS

Felipe G. Cabral*, Jean H. A. Tomola†, Marcos V. Moreira*

*COPPE-Programa de Engenharia Elétrica, Universidade Federal do Rio de Janeiro, Cidade universitária, Ilha do Fundão, Rio de Janeiro, 21.945-970, RJ, Brasil

†Instituto Federal do Rio de Janeiro, Paracambi, Rio de janeiro, Brasil

Emails: felipecabral@poli.ufrj.br, jean.adebai@ifrj.edu.br, moreira.mv@poli.ufrj.br

Abstract— In general, systems are formed by the composition of several components, and may exhibit a large number of states. The growth of the model with the number of system components leads to high computational costs for diagnosis techniques based on the global model. Recently, a new approach to online diagnosis of modular systems that avoids the computation of the global system model is presented. The new approach has lead to the definition of synchronous diagnosability of modular systems. Although this method avoids the computational growth for online diagnosis, the verification of the synchronous diagnosability of the system is exponential in the number of system components and states. Moreover, the delay bound for synchronous diagnosis has not been computed. In this paper, we present an algorithm for the verification of synchronous diagnosability with lower computational complexity than the algorithm proposed in the literature, and we also present an algorithm for the computation of the delay bound for diagnosis of modular systems.

Keywords— Discrete-event systems, Synchronous diagnosability, Modular systems.

1 Introduction

Several works in the literature address the problem of fault diagnosis of discrete-event systems (DESs) (Sampath et al., 1995; Basilio et al., 2012; Carvalho et al., 2013). In Sampath et al. (1995), a diagnoser that can be used to verify the diagnosability of the language of the system, and can be straightforwardly implemented for online fault diagnosis is proposed. The diagnoser proposed in Sampath et al. (1995) is an observer computed using the automaton model of the system, which can be formed by the composition of several components, and may exhibit a large number of states. In fact, it is known that the global system model, obtained from the parallel composition of all system components, can grow exponentially with the number of its components.

In order to avoid the computation of the global system model, fault diagnosis schemes that take advantage of the system modular structure have been proposed in the literature (Debouk et al., 2002; Contant et al., 2006; Zhou et al., 2008). Recently, in Cabral et al. (2015a), a new approach to online fault diagnosis of modular systems has been proposed. The diagnosis method presented in Cabral et al. (2015a) is based on the computation of a Petri net diagnoser that estimates the states of the normal behavior of each component of the system, and synchronizes the occurrence of observable events in these components. Moreover, in Cabral et al. (2015a), the definition of synchronous diagnosability, and an algorithm to verify this property, are presented.

Although the method presented in Cabral et al. (2015a) avoids the computation of the global system model and observers for online fault diagnosis, the synchronous diagnosability verification requires the computation of observers of the system modules, and, therefore, the verifier can grow exponentially with the number of states of the system components.

Another issue of the diagnosis method presented in Cabral et al. (2015a) lies on the fact that if the system is synchronously diagnosable, the language observed by the Petri net diagnoser can be larger than the projection of the normal language generated by the system. This fact can lead to a larger delay bound for diagnosis by using the method proposed in Cabral et al. (2015a) than by using the traditional monolithic diagnoser (Sampath et al., 1995). The increase in the delay bound for diagnosis reduces the efficiency of the diagnostic system, and, in some cases, can make this system useless.

In this paper, we propose a polynomial time algorithm in the size of the system modules, to verify the synchronous diagnosability of the system language. In addition, we also propose an algorithm for the computation of the delay bound for synchronous diagnosis. The value of the delay bound can be used to determine if the synchronous diagnosis scheme can be efficiently applied to the system.

2 Preliminaries

2.1 Notation and definitions

Let $G = (Q, \Sigma, f, q_0, Q_m)$ 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 (Cassandras and Lafortune, 2008), q_0 is the initial state of the system, and Q_m is the set

of marked states. For the sake of simplicity, the set of marked states 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 as $Ac(G)$ and the coaccessible part of G, denoted as $CoAc(G)$, are defined 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 compositions of G_1 and G_2, respectively (Cassandras and Lafortune, 2008).

The projection operation $P_s : \Sigma^\star \to \Sigma_s^\star$, where $\Sigma_s \subset \Sigma$, is defined as usual (Cassandras and Lafortune, 2008).

Let $L_A, L_B \subseteq \Sigma^\star$, and let $\Sigma_s \subset \Sigma$. Then

$$P_s(L_A \cap L_B) \subseteq P_s(L_A) \cap P_s(L_B). \qquad (1)$$

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 the set of observable and unobservable events, respectively, and let $\Sigma_f \subseteq \Sigma_{uo}$ denote the set of fault events. In this paper, for the sake of simplicity, it is assumed that there is only one fault event, *i.e.*, $\Sigma_f = \{\sigma_f\}$.

The unobservable reach of a state $q \in Q$, with respect to an observable event set Σ_o, is defined as the set of states that are reached from q by traces in $\Sigma_{uo}^\star$, *i.e*,

$$UR(q, \Sigma_o) = \{q' \in Q : [\exists s \in \Sigma_{uo}^\star][f(q, s) = q']\}.$$

The unobservable reach can be extended to a set $\Theta \subseteq Q$ as:

$$UR(\Theta, \Sigma_o) = \bigcup_{q \in \Theta} UR(q, \Sigma_o).$$

Let G_k, $k = 1, \ldots, r$, be the automaton models of the components of the system, and let $\Sigma_k \subseteq \Sigma$ be the set of events of G_k. Let $\Sigma_{o_k} = \Sigma_o \cap \Sigma_k$ denote the set of observable events of G_k. Notice that, if $\sigma \in \Sigma_{o_k} \cap \Sigma_j$, than $\sigma \in \Sigma_{o_j}$, *i.e.*, if an event is observable for a component, than it is observable for all components for which σ is defined.

A faulty 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 $Obs(G, \Sigma_o) = (Q_{obs}, \Sigma_o, f_{obs}, q_{0,obs})$ (Cassandras and Lafortune, 2008).

2.2 Diagnosability of Discrete-Event Systems

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

Definition 1 *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*

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

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

According to Definition 1, L is diagnosable with respect to P_o and Σ_f if, and only if, for all faulty traces st with arbitrarily long length after the occurrence of a fault 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 fault event after a bounded number of occurrences of events.

3 Synchronous diagnosability verifier

In Cabral et al. (2015b), a Petri net diagnoser, built from the non-faulty part of the system G_N, is proposed. Although the proposed diagnoser grows polynomially with the size of G_N, G_N has exponential growth with the number of systems components, since it can be obtained from the parallel composition of the normal behavior of the system components, *i.e.*, $G_N = G_{N_1} \| G_{N_2} \| \ldots \| G_{N_r}$, where G_{N_k} models the normal behavior of the system component G_k.

In order to circumvent this problem, in Cabral et al. (2015a) a synchronous diagnoser, based on the on-line state estimate of the non-faulty part of the system components, is proposed. In other words, considering, without loss of generality, that the normal part of the system G_N is composed of two modules G_{N_1} and G_{N_2}, the observed traces of these modules, synchronized on observable events, are used to diagnose the fault events of system G. In Cabral et al. (2015a) it is shown that the observed language of a synchronous diagnoser can be a larger set than the language observed by a diagnoser based on the monolithic system model, *i.e.*, $\mathcal{L}(Obs(G_1 \| G_2, \Sigma_o)) \subseteq \mathcal{L}(Obs(G_1, \Sigma_{o_1}) \| Obs(G_2, \Sigma_{o_2}))$. This fact leads to the definition of synchronous diagnosability of the language of a modular system. Without loss of generality, in the following definition we consider that the system G is composed of only two components G_1 and G_2, *i.e.*, $G = G_1 \| G_2$.

Definition 2 *Let $G_N = G_{N_1} \| G_{N_2}$, 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 $k = 1, 2$. Then, L is said to be synchronously diagnosable with respect to L_{N_1}, L_{N_2},*

$P_1 : \Sigma^\star \to \Sigma_1^\star$, $P_2 : \Sigma^\star \to \Sigma_2^\star$, $P_o : \Sigma^\star \to \Sigma_o^\star$, *and* Σ_f *if*

$$(\exists n \in \mathbb{N})(\forall s \in L_F)(\forall st \in L_F, \|t\| \geq n) \Rightarrow$$
$$(P_o(st) \notin P_o(P_1^{-1}(L_{N_1})) \cap P_o(P_2^{-1}(L_{N_2}))). \quad \square$$

It is important to remark that Definition 2 of synchronous diagnosability of a language L is equivalent to the standard definition of diagnosability (Definition 1) of a language $L_a = L_F \cup L_{N_a}$, where L_{N_a} is such that $P_o(L_{N_a}) = P_o(P_1^{-1}(L_{N_1})) \cap P_o(P_2^{-1}(L_{N_2}))$.

The algorithm presented in Cabral et al. (2015a) to verify synchronous diagnosability of DESs is exponential in the number of system components, and also in the number of states of the components. In this paper, we propose a modification of the verification algorithm presented in Cabral et al. (2015a), based on the algorithm proposed in Moreira et al. (2011), in order to prevent the exponential computational complexity with the number of states of the system components. This modification lies in the introduction of a renaming function of unobservable events, and prevent the use of observers.

Algorithm 1 G_V^M

Input: *Automata G_1, G_2 and G.*
Output: *Verifier automaton G_V^M.*

1: *Compute automaton G_F that models the faulty behavior of the system by using the method presented in Moreira et al. (2011).*

2: *Compute automaton G_N that models the normal behavior of G by using the method presented in Moreira et al. (2011).*

3: *Compute automata G_{N_k}, for $k = 1, 2$ as follows:*

  3.1: *Remove from G_k the transitions that are not associated with a transition in G_N, resulting in automata G'_{N_k}, $k = 1, 2$.*

  3.2: *Compute automata $G_{N_k} = Ac(G'_{N_k})$. Notice that the set of events of G_{N_k}, Σ_{N_k}, can be partitioned as $\Sigma_{N_k} = \Sigma_{uo,N_k} \dot\cup \Sigma_{o_k}$, where $\Sigma_{uo,N_k} = \Sigma_k \setminus (\Sigma_{o_k} \cup \Sigma_f)$.*

4: *Compute automaton G_N^R as follows:*

  4.1: *Define function $R_k : \Sigma_{N_k} \to \Sigma_{N_k}^R$, as:*

$$R_k(\sigma) = \begin{cases} \sigma, & \text{if } \sigma \in \Sigma_{o_k} \\ \sigma_{R_k}, & \text{if } \sigma \in \Sigma_{uo,N_k} \end{cases}. \quad (2)$$

  4.2: *Construct automata $G_{N_k}^R = (Q_{N_k}, \Sigma_{N_k}^R, f_{N_k}^R, q_{0,N_k})$, for $k = 1, 2$, with $f_{N_k}^R(q_{N_k}, R_k(\sigma)) = f_{N_k}(q_{N_k}, \sigma)$, $\forall \sigma \in \Sigma_{N_k}$.*

  4.3: *Compute $G_N^R = G_{N_1}^R \| G_{N_2}^R$.*

5: *Compute the verifier automaton $G_V^M = (Q_V, \Sigma_V, f_V, q_{0,V}) = G_F \| G_N^R$. Notice that a state of G_V^M is given by $q_V = (q_F, q_N^R)$, where q_F and q_N^R are states of G_F and G_N^R, respectively, and $q_F = (q, q_l)$, where $q \in Q$ and $q_l \in \{N, F\}$.*

6: *Verify the existence of a cyclic path $cl = (q_V^i, \sigma_i, q_V^{i+1}, \ldots, q_V^l, \sigma_l, q_V^i)$, where $l \geq i > 0$, in G_V^M such that:*

$$\exists j \in \{i, i+1, \ldots, l\} \text{ s.t. for some } q_V^j,$$
$$(q_l^j = F) \wedge (\sigma_j \in \Sigma).$$

If the answer is yes, then L is not synchronously diagnosable with respect to L_{N_1}, L_{N_2}, $P_1 : \Sigma^\star \to \Sigma_1^\star$, $P_2 : \Sigma^\star \to \Sigma_2^\star$, $P_o : \Sigma^\star \to \Sigma_o^\star$, and Σ_f. Otherwise, L is synchronously diagnosable.

In order to prove the correctness of Algorithm 1 we first present the following lemmas.

Lemma 1 *Let G_{N_k} be the automaton that models the normal behavior of G_k, and L_{N_k} be the language generated by G_{N_k}, for $k = 1, 2$. Let $G_{N_k}^R$ be the automaton obtained from G_{N_k} by applying Step 4 of Algorithm 1, and $L_{N_k}^R$ be the language generated by $G_{N_k}^R$, for $k = 1, 2$. Then, $P_o^R[P_{N_1}^{R^{-1}}(L_{N_1}^R) \cap P_{N_2}^{R^{-1}}(L_{N_2}^R)] = P_o^R[P_{N_1}^{R^{-1}}(L_{N_1}^R)] \cap P_o^R[P_{N_2}^{R^{-1}}(L_{N_2}^R)]$, where $P_o^R : \Sigma^{R^\star} \to \Sigma_o^\star$, $P_{N_k}^R : \Sigma^{R^\star} \to \Sigma_{N_k}^\star$, for $k = 1, 2$, and $\Sigma^R = \Sigma_{N_1}^R \cup \Sigma_{N_2}^R$.*

Proof: In order to show that $P_o^R[P_{N_1}^{R^{-1}}(L_{N_1}^R) \cap P_{N_2}^{R^{-1}}(L_{N_2}^R)] = P_o^R[P_{N_1}^{R^{-1}}(L_{N_1}^R)] \cap P_o^R[P_{N_2}^{R^{-1}}(L_{N_2}^R)]$, we must only proof that $P_o^R[P_{N_1}^{R^{-1}}(L_{N_1}^R) \cap P_{N_2}^{R^{-1}}(L_{N_2}^R)] \supseteq P_o^R[P_{N_1}^{R^{-1}}(L_{N_1}^R)] \cap P_o^R[P_{N_2}^{R^{-1}}(L_{N_2}^R)]$, since $P_o^R[P_{N_1}^{R^{-1}}(L_{N_1}^R) \cap P_{N_2}^{R^{-1}}(L_{N_2}^R)] \subseteq P_o^R[P_{N_1}^{R^{-1}}(L_{N_1}^R)] \cap P_o^R[P_{N_2}^{R^{-1}}(L_{N_2}^R)]$ is true, according to equation (1).

Let $s = \sigma_{o_1}\sigma_{o_2}\ldots\sigma_{o_n}$ be a sequence of events s.t. $s \in P_o^R[P_{N_1}^{R^{-1}}(L_{N_1}^R)] \cap P_o^R[P_{N_2}^{R^{-1}}(L_{N_2}^R)]$. Thus, $s \in P_o^R[P_{N_1}^{R^{-1}}(L_{N_1}^R)]$ and $s \in P_o^R[P_{N_2}^{R^{-1}}(L_{N_2}^R)]$. Therefore, there exists at least one trace $y_1 \in P_{N_1}^{R^{-1}}(L_{N_1}^R)$, and one trace $y_2 \in P_{N_2}^{R^{-1}}(L_{N_2}^R)$, such that $P_o(y_1) = P_o(y_2) = s$. Since $P_o(y_1) = P_o(y_2) = s$, y_1 and y_2 can be written as $y_1 = v_1\sigma_{o_1}v_2\sigma_{o_2}v_3\ldots v_n\sigma_{o_n}v_{n+1}$, and $y_2 = \omega_1\sigma_{o_1}\omega_2\sigma_{o_2}\omega_3\ldots\omega_n\sigma_{o_n}\omega_{n+1}$, where $v_i, \omega_i \in (\Sigma^R \setminus \Sigma_o)^\star$. Moreover, since $y_1 \in P_{N_1}^{R^{-1}}(L_{N_1}^R)$, and $y_2 \in P_{N_2}^{R^{-1}}(L_{N_2}^R)$, v_i and ω_i can be written as $v_i = t_1\sigma_{1,1}^R t_2\sigma_{1,2}^R\ldots t_\eta$, and $\omega_i = z_1\sigma_{2,1}^R z_2\sigma_{2,2}^R\ldots z_r$, where $\sigma_{1,j}^R \in \Sigma_{N_1}^R$, $t_i \in (\Sigma_{N_2}^R \setminus \Sigma_{N_1}^R)^\star$, $\sigma_{2,j}^R \in \Sigma_{N_2}^R$, and $z_i \in (\Sigma_{N_1}^R \setminus \Sigma_{N_2}^R)^\star$. Therefore, there exists at least one trace $y \in P_{N_1}^{R^{-1}}(L_{N_1}^R) \cap P_{N_2}^{R^{-1}}(L_{N_2}^R)$ with the same observation as y_1 and y_2. Since $P_o^R(y) = s$, then $s \in P_o^R[P_{N_1}^{R^{-1}}(L_{N_1}^R) \cap P_{N_2}^{R^{-1}}(L_{N_2}^R)]$, which concludes the proof. $\square$

Lemma 2 *Let L_{N_k} be the language generated by G_{N_k}. Then, $P_o^R[P_{N_1}^{R^{-1}}(L_{N_1}^R)] \cap P_o^R[P_{N_2}^{R^{-1}}(L_{N_2}^R)] = P_o(P_1^{-1}(L_{N_1})) \cap P_o(P_2^{-1}(L_{N_2}))$.*

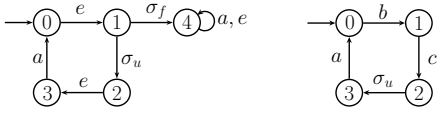
Proof: The proof is straightforward and will be omitted due to lack of space. □

Theorem 3 *Let L_{N_k} denote the language generated by G_{N_k}, for $k = 1, 2$. Then, L is not synchronously diagnosable with respect to L_{N_1}, L_{N_2}, $P_1 : \Sigma^\star \to \Sigma_1^\star$, $P_2 : \Sigma^\star \to \Sigma_2^\star$, $P_o : \Sigma^\star \to \Sigma_o^\star$, and Σ_f if, and only if, there exists a cyclic path $cl = (q_V^i, \sigma_i, q_V^{i+1}, \ldots, q_V^l, \sigma_l, q_V^i)$ in G_V^M, where $l \geq i > 0$, such that:*

$$\exists j \in \{i, i+1, \ldots, l\} \text{ s.t. for some } q_V^j,$$
$$(q_l^j = F) \wedge (\sigma_j \in \Sigma). \quad (3)$$

Proof: According to Definition 2, in order to verify the synchronous diagnosability of the language of the system L, it is necessary to check if there exists a faulty trace st such that $P_o(st) \in P_o(P_1^{-1}(L_{N_1})) \cap P_o(P_2^{-1}(L_{N_2}))$. Lemma 1 shows that $P_o^R[P_{N_1}^{R^{-1}}(L_{N_1}^R) \cap P_{N_2}^{R^{-1}}(L_{N_2}^R)] = P_o^R[P_{N_1}^{R^{-1}}(L_{N_1}^R)] \cap P_o^R[P_{N_2}^{R^{-1}}(L_{N_2}^R)]$, and according to Lemma 2, $P_o^R[P_{N_1}^{R^{-1}}(L_{N_1}^R)] \cap P_o^R[P_{N_2}^{R^{-1}}(L_{N_2}^R)] = P_o(P_1^{-1}(L_{N_1})) \cap P_o(P_2^{-1}(L_{N_2}))$. Therefore, $P_o(P_1^{-1}(L_{N_1})) \cap P_o(P_2^{-1}(L_{N_2})) = P_o^R[P_{N_1}^{R^{-1}}(L_{N_1}^R) \cap P_{N_2}^{R^{-1}}(L_{N_2}^R)]$. Thus, in order to check the synchronous diagnosability of L, it can be verified if there exists a faulty trace st such that $P_o(st) \in P_o^R[P_{N_1}^{R^{-1}}(L_{N_1}^R) \cap P_{N_2}^{R^{-1}}(L_{N_2}^R)]$. Since the unobservable events of G_N^R are renamed, and hence, are private events of G_N^R, it can be seen that the verifier automaton G_V^M proposed here is equal to the verifier automaton G_V obtained by applying the method proposed in Moreira et al. (2011) to a system whose faulty automaton generates L_F and whose normal behavior automaton generates L_{N_a}. Moreover, the same necessary and sufficient condition (3) would be obtained by using the verification method proposed in Moreira et al. (2011), which concludes the proof. □

It is important to remark that Algorithm 1 for the construction of G_V^M is polynomial in the number of states of G_{N_1} and G_{N_2}, and it is exponential in the number of system components. The synchronous diagnosability verification can be carried out off-line, and, once the synchronous diagnosability is verified, the synchronous diagnoser proposed in Cabral et al. (2015a), that has polynomial growth in the number of states of G_{N_1} and G_{N_2} and in the number of system components, can be used for on-line diagnosis. In the following example, we illustrate the use of Algorithm 1 for the verification of synchronous diagnosability.



(a) G_1. (b) G_2.

Figure 1: Components G_1 and G_2 of Example 1.

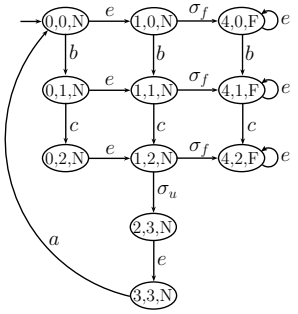


Figure 2: Automaton G_F of Example 1.

Example 1 *Let G_1 and G_2, whose transition diagrams are shown in Figures 1(a) and 1(b), respectively, be the components of a system $G = G_1 \| G_2$. Let $\Sigma = \{a, b, c, e, \sigma_u, \sigma_f\}$, $\Sigma_o = \{a, b, c, e\}$, $\Sigma_{uo} = \{\sigma_u, \sigma_f\}$, $\Sigma_f = \{\sigma_f\}$, $\Sigma_1 = \{a, e, \sigma_u, \sigma_f\}$ and $\Sigma_2 = \{a, b, c, \sigma_u\}$. In the first step of Algorithm 1 automaton G_F, shown in Figure 2, is computed. In steps 2 and 3, automata G_N, G_{N_1} and G_{N_2} are computed, and, in step 4, automaton G_N^R, depicted in Figure 3, is obtained. Notice that the gray states of G_N^R, and their corresponding transitions labeled with observable events, are not in G_N. Finally, in step 5 of Algorithm 1, the synchronous verifier automaton G_V^M, presented in Figure 4, is computed. Notice that there are no cycles in G_V^M that satisfies condition (3) of Theorem 3. Thus, L is synchronously diagnosable with respect to L_{N_1}, L_{N_2}, $P_1 : \Sigma^\star \to \Sigma_1^\star$, $P_2 : \Sigma^\star \to \Sigma_2^\star$, $P_o : \Sigma^\star \to \Sigma_o^\star$, and Σ_f.*

4 Computation of the delay bound for synchronous diagnosis

In this section, we propose an algorithm for the computation of the delay bound n^* for synchronous diagnosis. This is an important issue, since the synchronous diagnoser may consider normal traces that cannot be executed by the plant, and that can have the same projection of a faulty trace, *i.e.*, the synchronous diagnoser may aug-

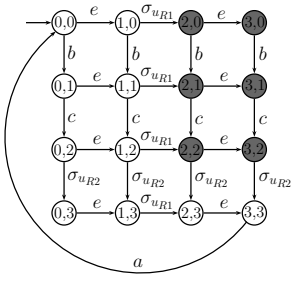


Figure 3: Automaton G_N^R of Example 1.

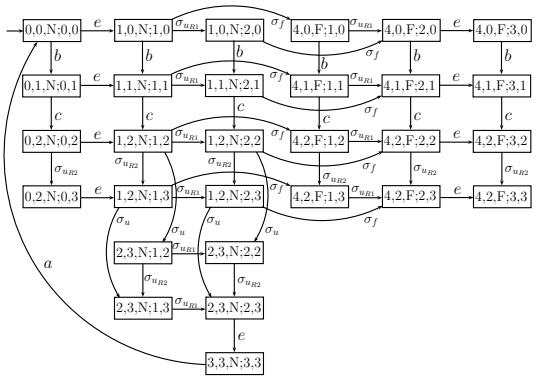


Figure 4: Automaton G_V^M of Example 1.

ment the delay for diagnosis of the fault events of the system. If the delay introduced by the synchronous diagnoser is exceedingly large, then, even when the language of the system is synchronous diagnosable, there may be no practical application to this method. Thus, the computation of the delay bound is essential for the implementation of the synchronous diagnoser.

With a view to computing the delay bound n^*, it is necessary to compute first the maximum number of events d that the system can execute after the occurrence of the fault event σ_f, for which there exist a faulty trace $st \in L_F$ and a normal trace $\omega \in L_{N_a}$ with the same observation:

$$d = \max\{\|t\| : (s \in L_F)(st \in L_F), \\ (P_o(st) \in P_o(P_1^{-1}(L_{N_1})) \cap P_o(P_2^{-1}(L_{N_2})))\}.$$

In order to compute d, it is necessary to compute the length of the longest path in a directed acyclic graph (DAG) (Dasgupta et al., 2008), as presented in Algorithm 2.

Algorithm 2 *Length of the longest path in a directed acyclic graph G.*

Input: *Directed acyclic graph $G = (V, E)$.*
Output: *Length of the longest path in G, l.*

1: $(v_1, v_2, \ldots, v_\eta) =$ *Topological Sort*(G), *where* $v_j \in V$, *for* $j \in I_\eta$, *and* $\eta = |V|$.
2: *For* $j = 1, \ldots, \eta$:
$l(v_j) = 1 + \ max\{l(v_i) : (v_i, v_j) \in E\}$.
3: $l = \ max_{j \in I_\eta} l(v_j) - 1$.

It is important to remark that in the first step of Algorithm 2, a topological sort is carried out. The topological sort algorithm returns the linked list of the vertices of a DAG G, such that if G has an edge (u, v), then u appears before v in the ordering (Dasgupta et al., 2008).

Notice that Algorithm 2 can be carried out only if the graph is acyclic. Thus, the elimination of cyclic paths in G_V^M is necessary in order to use Algorithm 2 to compute d. This can be done by following the steps of Algorithm 3 presented hereafter.

Algorithm 3 *Elimination of cyclic paths of G_V^M.*

Input: $G_V^M = (Q_V, \Sigma_V, f_V, q_{0,V})$, *where* $\Sigma_V = \Sigma_{N_1}^R \cup \ldots \cup \Sigma_{N_r}^R \cup \Sigma_F$.
Output: $G_{nd}^M = (Q_{nd}, \Sigma_F, f_{nd}, q_{0,nd})$.

1: $q_{0,nd} = UR(q_{0,V}, \Sigma_F)$. $Q_{nd} = \{q_{0,nd}\}$.
2: *For each state* q_V *of* G_V^M *reached by an event* $\sigma \in \Sigma_F$*:*
2.1: *Compute* $q_{nd} = UR(q_V, \Sigma_F)$.
2.2: $Q_{nd} = Q_{nd} \cup \{q_{nd}\}$.
3: *For each* $q_{nd} \in Q_{nd}$ *and* $\sigma \in \Sigma_F$ *define the nondeterministic function* $f_{nd}(q_{nd}, \sigma)$, *as follows.*
3.1: *Compute*
$$Q_V^e = \{q_V^e \in Q_V : (\exists q_V \in q_{nd})[f_V(q_V, \sigma) = q_V^e]\}.$$
3.2: *Compute*
$$Q_{nd}^e = \{UR(q_V^e, \Sigma_F) : q_V^e \in Q_V^e\}.$$
3.3: *Define* $f_{nd}(q_{nd}, \sigma) = Q_{nd}^e$.

In Algorithm 4, we present the steps to compute the maximum number of events that can be executed by the system after the occurrence of the fault event σ_f, d.

Algorithm 4 *Computation of d.*

Input: G_{nd}^M.
Output: d.

1: *Mark all states of* G_{nd}^M *whose components have the label* F.
2: *Compute automaton* $\overline{G}_{nd}$ *by eliminating from* G_{nd}^M *all states that are not marked and their related transitions.*
3: *Compute the length* d *of the longest path in* $\overline{G}_{nd}$ *following the steps of Algorithm 2.*

In Algorithm 4, automaton $\overline{G}_{nd}$ is obtained from G_{nd}^M by eliminating all states of G_{nd}^M, and their related transitions, that are not reached after the occurrence of the faulty event σ_f. Thus, all sequences in $\overline{G}_{nd}$ are traces $t \in \Sigma^\star$ executed after a faulty sequence $s \in L \setminus L_N$. Therefore, d represents the length of the longest trace t for which there exist normal traces $\omega \in L_{N_a}$, where $P_o(\omega) = P_o(st)$, which violates definition 2.

Finally, algorithm 5 summarizes the procedure for computing de delay bound for synchronous diagnosis n^*.

Algorithm 5 *Computation of the delay bound for synchronous diagnosis n^*.*

Input: G_V^M.
Output: n^*.

1: *Compute* G_{nd}^M *from* G_V^M *by following Algorithm 3.*
2: *Compute d as presented in Algorithm 4.*
3: *Compute the delay bound for synchronous diagnosis* $n^* = d + 1$.

Example 2 *Consider the plant $G = G_1 \| G_2$, whose components G_1 and G_2 are shown in Figure 1. As shown in Example 1, L is synchronously diagnosable with respect to L_{N_1}, L_{N_2}, $P_1 : \Sigma^\star \to \Sigma_1^\star$, $P_2 : \Sigma^\star \to \Sigma_2^\star$, $P_o : \Sigma^\star \to \Sigma_o^\star$, and Σ_f. Thus, the delay bound for synchronous diagnosis n^* can be computed following the steps af Algorithm 5, which results in $n^* = 4$. The longest trace after the occurrence of the fault event σ_f is obtained from the synchronous verifier G_V^M shown in Figure 4, where it can be seen that a deadlock state is reached after traces $t_1 = ebc$, $t_2 = bce$ or $t_3 = bec$ have been executed after σ_f. Notice that the delay bound for the monolithic diagnosis is also $n^* = 4$, which shows that, even with the possibility of the growth of the observed normal language for synchronous diagnosis, the fault event can be diagnosed with the same delay as using the monolithic approach.*

The computation of the delay bound for synchronous diagnosis has polynomial complexity in the size of the plant model G. To confirm this result, it is necessary to compute the complexity of each step of Algorithm 5. In the first step, automaton G_{nd}^M is computed by eliminating all cyclic paths of G_V^M, as presented in Algorithm 3. In the worst case, the number of states of G_{nd}^M is equal to the number of states of G_V^M, and the number of transitions of G_{nd}^M is, in the worst case, equal to $|Q_V^M|^2 \times |\Sigma_F|$, due to the fact that G_{nd}^M is a nondeterministic automaton. Since the number of states of G_V^M is bounded by $(\prod_{k=1}^{r} |Q_k|) \times 2 \times |Q|$, then the computation complexity of G_{nd}^M is $O\left(((\prod_{k=1}^{r} |Q_k|) \times |Q|)^2 \times |\Sigma|\right)$.

In step 2 of Algorithm 5 d is computed following the steps of Algorithm 4. This is done by constructing automaton $\overline{G}_{nd}$, which is formed by eliminating the states of G_{nd}^M that do not have the label F, which can be executed in linear time. Next, in step 3 of Algorithm 4, d is calculated using Algorithm 2. The computation of the length of the longest path of a DAG can also be done in linear time with the size of the DAG (Dasgupta et al., 2008). Therefore, the complexity of Algorithm 5 is $O\left(((\prod_{k=1}^{r} |Q_k|) \times |Q|)^2 \times |\Sigma|\right)$.

5 Conclusions

In this paper, we propose a modification of the synchronous diagnosability verification algorithm presented in Cabral et al. (2015a). This modification lies in the introduction of a renaming event function of unobservable events, preventing the use of observers, which have exponential growth. We also propose an algorithm for the computation of the delay bound for the synchronous diagnosis method, which can be used to evaluate if the diagnosis delay added by this method allows the implementation of the diagnoser.

Acknowledgement

This work was supported by Petrobras, CNPq, and FAPERJ.

References

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.

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

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

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.

Dasgupta, S., Papadimitriou, C. and Vazirani, U. (2008). *Algorithms*, McGraw-Hill.

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

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* pp. 1679–1684.

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.

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