DOI: 10.48011/asba.v2i1.1705

Redução de Diagnosticadores de Sistemas a Eventos Discretos

Augusto P. Vasconcellos * **Gustavo S. Viana** *
Marcos V. Moreira *

* *Universidade Federal do Rio de Janeiro*
COPPE - Programa de Engenharia Elétrica
21949-900, Rio de Janeiro, R.J., Brasil, (e-mail: augustopmv; gustavo.viana; moreira.mv@poli.ufrj.br)

Abstract: The main drawback of implementing the traditional diagnoser of Discrete-Event Systems proposed in the literature is that its state set can be very large, requiring the use of a large amount of memory to store the diagnoser for large complex systems. In this paper, we propose an algorithm for the computation of a reduced diagnoser, that preserves the diagnosability of the system language and the same diagnosis delay as the original diagnoser. We show that the proposed reduction strategy can lead to a reduced diagnoser with fewer states than by using other strategies proposed in the literature.

Resumo: A principal desvantagem na implementação do diagnosticador tradicional de Sistemas a Eventos Discretos proposto na literatura é que seu conjunto de estados pode ser muito grande, requerendo o uso de uma quantidade de memória demasiadamente grande para armazenar o diagnosticador para sistemas grandes e complexos. Neste artigo, é proposto um algoritmo para computar um diagnosticador reduzido, que preserva tanto a diagnosticabilidade da linguagem do sistema quanto o mesmo atraso para o diagnóstico que o diagnosticador original. Além disso, é mostrado que a estratégia de redução proposta pode levar a um diagnosticador reduzido com menos estados do que outras estratégias propostas na literatura.

Keywords: Discrete-event systems; Fault diagnosis; Automata.
Palavras-chaves: Sistemas a eventos discretos; Diagnóstico de falhas; Autômatos.

1. INTRODUÇÃO

Para garantir uma operação confiável de sistemas de automação industriais, é necessário diagnosticar de maneira eficiente a ocorrência de falhas, *i.e.*, detectar e isolar a falha que ocorreu antes que ela danifique componentes do sistema ou cause risco aos operadores. Se o sistema puder ser abstraído como um Sistema a Evento Discreto (SED), então a falha pode ser modelada como um evento que deve ser identificado pelo sistema de diagnóstico. Neste caso, o atraso para o diagnóstico, definido como o maior número de ocorrências de eventos que o sistema pode executar após a falha ocorrer até sua detecção (Yoo and Garcia, 2003; Tomola et al., 2017; Viana et al., 2019), pode ser utilizado para determinar a eficiência do método de diagnóstico.

Em Sampath et al. (1995), um diagnosticador é construído para detectar e isolar eventos de falha não-observáveis, e a propriedade de diagnosticabilidade, que está relacionada com a capacidade de identificar a ocorrência de uma falha após um número limitado de ocorrências de eventos, é introduzida. Diagnosticadores podem ser utilizados para verificar a diagnosticabilidade do sistema e para diagnose em tempo real. A principal desvantagem de implementar diretamente o diagnosticador proposto em Sampath et al. (1995) é que seu conjunto de estados cresce no pior caso exponencialmente com o número de estados do modelo do sistema. Em Clavijo and Basilio (2017), é mostrado que, no caso médio, o diagnosticador cresce polinomialmente com o número de estados do sistema. Contudo, em sistemas complexos, o diagnosticador pode ainda ter um número elevado de estados, requerendo uma grande quantidade de memória para ser implementado. Assim, é desejável reduzir o tamanho do diagnosticador para sua implementação, preservando a diagnosticabilidade do sistema e o atraso para o diagnóstico. Para tanto, é necessário que a linguagem do diagnosticador reduzido simule a linguagem original do diagnosticador. Em geral, entretanto, a redução de estados de um autômato leva a um aumento da linguagem em comparação ao autômato original. Isso ocorre devido à geração de uma linguagem excedente por conta da introdução de novos ciclos no autômato reduzido.

Ao longo dos anos, o problema de minimização de autômatos finitos determinísticos (AFD) foi abordado para diferentes objetivos, conforme apresentado em Zad et al. (2003), Su and Wonham (2004), Hopcroft et al. (2006) e Cai et al. (2019). Em Hopcroft et al. (2006), a minimização de um AFD é baseada em encontrar e agregar os estados equivalentes do autômato, ou seja, os estados que ao serem agregados não levam a uma modificação da linguagem marcada do sistema. Em Zad et al. (2003), é apresentada uma abordagem de redução de diagnosticadores para o diagnóstico de falhas baseado em estado em tempo real.

Outros métodos de redução foram abordados na literatura, mais notadamente na teoria de controle supervisório (Minhas, 2002; Su and Wonham, 2004; Sivollela et al., 2006; da Silva Neto, 2008). Em Minhas (2002), o processo de redução é baseado em encontrar *coberturas de controle*, que são conjuntos formados por conjuntos de estados que se comportam de maneira consistente no que diz respeito às suas ações de controle, *i.e.*, para qualquer par de estados de um subconjunto de uma cobertura de controle, todos os eventos que estão habilitados por um desses estados não pode ser desabilitado pelo outro. A limitação do método proposto em Minhas (2002) é que, em alguns casos, a determinização do autômato, ou mesmo a redução do supervisor, não pode ser garantida. Para superar este problema, Su and Wonham (2004) introduziram o conceito de *congruência de controle*, que é um caso especial da cobertura de controle. Se os subconjuntos da cobertura de controle formam uma partição sobre os estados do supervisor, então a cobertura de controle é denominada congruência de controle. O determinismo do supervisor reduzido é garantido quando a congruência de controle é utilizada, porém não é possível garantir a obtenção de um supervisor mínimo, que foi demonstrado em Vaz and Wonham (1986) ser um problema NP-difícil.

Neste artigo é proposto um algoritmo que computa um diagnosticador determinístico reduzido, que preserva a diagnosticabilidade da linguagem do sistema e tem o mesmo atraso para o diagnóstico do diagnosticador original. Semelhante a Su and Wonham (2004) e Cai et al. (2019), os conjuntos de estados agregados do diagnosticador reduzido são disjuntos. Diferentemente dos outros métodos apresentados na literatura, o método proposto neste trabalho não depende de um ordenamento prévio dos estados, dado como entrada nos algoritmos de redução. O algoritmo de redução adotado procura, a cada passo, maximizar a possibilidade de redução do diagnosticador no passo seguinte. Um exemplo será utilizado ao longo do texto para ilustrar a aplicação do método, e para mostrar que a estratégia de redução proposta pode levar a um diagnosticador com menos estados que o diagnosticador obtido considerando o mesmo critério utilizado por Su and Wonham (2004) e Cai et al. (2019).

Este artigo está estruturado da seguinte forma. Na seção 2 alguns conceitos preliminares são apresentados. Na seção 3 o algoritmo para redução de diagnosticadores é descrito e são apresentados alguns exemplos da redução de diagnosticadores utilizando o algoritmo proposto. As conclusões são apresentadas na seção 4.

2. CONCEITOS PRELIMINARES

2.1 Notações e Definições

Sejam A e B dois conjuntos. Então, $|A|$ denota a cardinalidade de A, e $A \setminus B$ denota o diferença entre A e B.

Considere que $G = (X, \Sigma, f, x_0)$ denota um AFD que modela um SED, em que X é conjunto de estados, Σ é o conjunto finito de eventos, $f : X \times \Sigma^* \to X$ é a função de transição de estados definida em $X \times \Sigma^*$, em que Σ^* denota o fecho de Kleene de Σ e x_0 é o estado inicial do sistema. Além disso, considere que Σ pode ser particionado como $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$, em que Σ_o e Σ_{uo} são os conjuntos de eventos observáveis e não-observáveis, respectivamente.

Seja $\Gamma_G : X \to 2^\Sigma$ a função de eventos viáveis de G, tal que $\Gamma_G(x) = \{\sigma \in \Sigma : f(x, \sigma)!\}$, em que ! denota *é definido*. Uma vez que f é uma função parcial, então a linguagem gerada por G, denotada por $\mathcal{L}(G) = L$, é definida como $L = \{s \in \Sigma^* : f(x_0, s)!\}$. Dessa forma, o fecho de prefixo de L, denotado por $\overline{L}$, é definido como $\overline{L} = \{s \in \Sigma^* : (\exists t \in \Sigma^*)[st \in L]\}$. Se $L = \overline{L}$, então L é dita ser prefixo-fechada. A linguagem $L \subseteq \Sigma^*$ é dita ser viva se, para todas as sequências $s \in L$, existe um evento $\sigma \in \Sigma$, tal que $s\sigma \in L$ (Cassandras and Lafortune, 2008). O comprimento de uma sequência $s \in L$ é denotado por $|s|$. Um caminho em G é dado por $(x_1, \sigma_1, x_2, \ldots, \sigma_{n-1}, x_n)$, em que $x_i \in X$, $\sigma_i \in \Sigma$, e $x_{i+1} = f(x_i, \sigma_i)$, $i = 1, 2, \ldots, n-1$, e o caminho é dito ser cíclico se $x_1 = x_n$.

Sejam Σ_s e Σ_l conjuntos de eventos tais que $\Sigma_s \subset \Sigma_l$. A projeção $P : \Sigma_l^* \to \Sigma_s^*$ é definida como: (i) $P(\varepsilon) = \varepsilon$; (ii) $P(\sigma) = \sigma$, se $\sigma \in \Sigma_s$; (iii) $P(\sigma) = \varepsilon$, se $\sigma \in \Sigma_l \setminus \Sigma_s$; (iv) $P(s\sigma) = P(s)P(\sigma)$, para $s \in \Sigma_l^*$, $\sigma \in \Sigma_l$, em que ε denota a sequência vazia. Além disso, a projeção inversa $P^{-1} : \Sigma_s^* \to 2^{\Sigma_l^*}$ pode ser definida como $P^{-1}(t) := \{s \in \Sigma_l^* : P(s) = t\}$.

Sejam G_1 e G_2 dois autômatos. Então, a composição paralela entre G_1 e G_2 é denotada por $G_1 \| G_2$ (Cassandras and Lafortune, 2008). A operação de tomar a parte acessível de um autômato G é denotada por $Ac(G)$ (Cassandras and Lafortune, 2008).

O alcance não-observável de um estado x é definido como $UR(x) = \{y \in X : (\exists t \in \Sigma_{uo}^*)\,[(f(x,t) = y)]\}$. Seja $A \subseteq X$, então o alcance não-observável de A é dado por $UR(A) = \bigcup_{x \in A} UR(x)$. O observador de G pode ser definido como $Obs(G) = (X_{obs}, \Sigma_o, f_{obs}, x_{0,obs})$, em que $X_{obs} \subseteq 2^X$, $x_{0,obs} = UR(x_0)$, e para todo $x_{obs} \in X_{obs}$, $\Gamma_{G_{obs}}(x_{obs}) = \bigcup_{x \in x_{obs}} \Gamma_G(x) \cap \Sigma_o$, $f_{obs}(x_{obs}, \sigma) = \bigcup_{(x \in x_{obs}) \wedge (f(x,\sigma)!)} UR\left(f(x, \sigma)\right)$, se $\sigma \in \Gamma_{G_{obs}}(x_{obs})$, ou não definido, caso contrário.

2.2 Diagnosticabilidade de Sistemas a Eventos Discretos

Suponha que $\Sigma_f \subseteq \Sigma_{uo}$ é o conjunto formado pelos eventos de falha que devem ser diagnosticados. Suponha também, sem perda de generalidade, que $\Sigma_f = \{\sigma_f\}$. O caso de múltiplas falhas pode ser abordado analisando-se cada tipo de falha separadamente (Yoo and Lafortune, 2002). Em Sampath et al. (1995), a propriedade da linguagem do sistema relacionada com a capacidade de diagnosticar a ocorrência do evento de falha σ_f, denominada *diagnosticabilidade*, é apresentada. Para definir formalmente essa propriedade, em Sampath et al. (1995) as seguintes hipóteses são consideradas: (i) a linguagem L é viva; e (ii) o autômato G que representa a planta não contém caminhos cíclicos formados apenas por eventos não-observáveis. Neste artigo, as mesmas hipóteses apresentadas em Sampath et al. (1995) são consideradas.

Seja $L_N \subset L$ a linguagem prefixo-fechada formada por todas as sequências de L que não contém o evento de falha σ_f, e seja G_N o subautômato de G que gera a linguagem L_N. Assim, a seguinte definição de diagnosticabilidade de L pode ser apresentada (Sampath et al., 1995).

Definição *1.* Seja L a linguagem gerada por G, e suponha que L seja viva. Seja $L_N \subset L$ a linguagem livre de falha de L. Seja $P_o : \Sigma^* \to \Sigma_o^*$ uma operação de projeção. Então, L é dita ser diagnosticável com relação à P_o e Σ_f, se

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

$$\Rightarrow \left(\forall \omega \in P_o^{-1}\left(P_o(st)\right) \cap L, \omega \in L \setminus L_N\right).$$

□

De acordo com a Definição 1, L não é diagnosticável com relação à P_o e Σ_f se, e somente se, existir uma sequência de falha st, de comprimento arbitrariamente longo após a falha, com a mesma observação que uma sequência livre de falha ω em L_N, *i.e.*, $P_o(st) = P_o(\omega)$. Se L for diagnosticável com relação à P_o e Σ_f, então é possível definir o atraso para o diagnóstico, denotado por z^*, como o menor valor de z que satisfaz a condição de diagnosticabilidade da Definição 1.

Em Sampath et al. (1995), o autômato diagnosticador é proposto para realizar diagnóstico em tempo real e verificar a diagnosticabilidade do sistema. Para tal, primeiro é definido o autômato rotulador $A_l = (X_l, \Sigma_f, f_l, x_{0,l})$, em que $X_l = \{N, Y\}, x_{0,l} = \{N\}, f_l(N, \sigma_f) = Y$, e $f_l(Y, \sigma_f) = Y$. O rótulo Y indica a ocorrência do evento de falha σ_f, e o rótulo N significa que a falha não ocorreu. Assim, o diagnosticador é definido como $G_d = Obs(G_l) = (X_d, \Sigma_o, f_d, x_{0_d})$, em que $G_l = G \| A_l$.

Note que os estados de G_d são da seguinte forma $x_d = \{(x_1, l_1), \ldots, (x_n, l_n)\}$, em que $x_i \in X$ e $l_i \in \{Y, N\}$, para $i = 1, \ldots, n$. Se o rótulo $l_i = Y$ para $i = 1, \ldots, n$, então x_d é dito ser um estado *positivo*. Por outro lado, se $l_i = N$ para $i = 1, \ldots, n$, x_d é dito ser um estado *negativo*. Finalmente, se existe $l_i = Y$ e $l_j = N$, $i \neq j$, x_d é dito ser um estado *incerto*.

O conjunto de estados do autômato diagnosticador pode ser particionado como $X_d = X_Y \dot{\cup} X_N \dot{\cup} X_{NY}$, em que X_Y, X_N e X_{NY} são, respectivamente, os conjuntos formados por todos os estados positivos, negativos e incertos de G_d.

Com o objetivo de verificar a diagnosticabilidade de L utilizando o diagnosticador, é necessário fazer a busca por ciclos indeterminados em G_d (Sampath et al., 1995). A seguir, são apresentadas as definições de ciclos e ciclos indeterminados.

Definição *2.* Um ciclo em G é o conjunto de estados formado por um caminho cíclico $(x_k, \sigma_1, x_{k+1}, \sigma_2, ..., \sigma_l, x_{k+l})$, em que $x_{k+l} = x_k$. □

Definição *3.* Um ciclo indeterminado em G_d é o conjunto $\left\{x_{d_1}, x_{d_2}, ..., x_{d_p}\right\} \subseteq X_d$ formado por estados incertos, satisfazendo as seguintes condições:

(i) $\left\{x_{d_1}, x_{d_2}, ..., x_{d_p}\right\}$ forma um ciclo em G_d.

(ii) $\exists (x_l^{k_l}, Y), (\tilde{x}_l^{r_l}, N) \in x_{d_l}$, com $x_l^{k_l}$ não necessariamente distinto de $\tilde{x}_l^{r_l}$, $l = 1, 2, ..., p$, $k_l = 1, 2, ..., m_l$, e $r_l = 1, 2, ..., \tilde{m}_l$, tal que os estados $\{x_l^{k_l}\}$, $l = 1, 2, ..., p$, $k_l = 1, 2, ..., m_l$, e $\{\tilde{x}_l^{r_l}\}$, $l = 1, 2, ..., p$, $k_l = 1, 2, ..., m_l$, podem ser rearranjados para formar ciclos em G. □

De acordo com a Definição 3, uma condição necessária e suficiente para a diagnosticabilidade de L, baseada na construção do autômato diagnosticador G_d, pode ser enunciada a seguir.

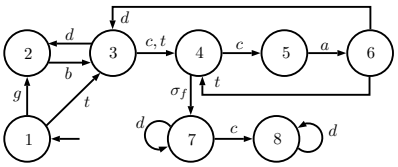

Figura 1. Autômato G.

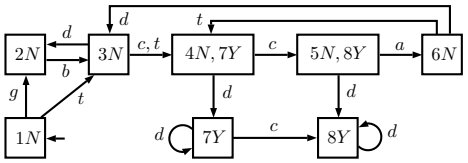

Figura 2. Autômato diagnosticador G_d.

Teorema *1.* A linguagem L é diagnosticável com relação à P_o e $\Sigma_f = \{\sigma_f\}$ se, e somente se, G_d não possui ciclos indeterminados.

Prova. Ver Sampath et al. (1995). □

O exemplo 1 ilustra a construção de G_d e a verificação da diagnosticabilidade da linguagem L.

Exemplo *1.* Considere o sistema modelado pelo autômato G, ilustrado na Figura 1, em que $\Sigma = \{a, b, c, d, t, \sigma_f\}$, $\Sigma_o = \{a, b, c, d, t\}$ e $\Sigma_{uo} = \Sigma_f = \{\sigma_f\}$, e suponha que se deseja verificar se a linguagem L é diagnosticável com relação à P_o e σ_f. Para tanto, o autômato diagnosticador G_d, apresentado na Figura 2, é obtido. Uma vez que G_d não possui ciclos indeterminados, então L é diagnosticável com relação à P_o e σ_f. □

3. ALGORITMO PARA REDUÇÃO DE DIAGNOSTICADORES

Considere que a linguagem L é diagnosticável com relação à P_o e Σ_f. Nesse caso, o diagnosticador proposto em Sampath et al. (1995) pode ser construído. Uma vez que a construção do diagnosticador G_d é baseado em um observador, então G_d pode ter um número elevado de estados (Clavijo and Basilio, 2017; Sampath et al., 1995). Nesta seção, um algoritmo para redução do diagnosticador G_d, de tal maneira que a diagnosticabilidade de L e o atraso para o diagnóstico sejam ambos preservados, é proposto. O método utilizado é baseado em um algoritmo guloso que, a cada passo, escolhe uma agregação de estados do diagnosticador baseada em uma medida que estima a agregabilidade de estados no próximo passo do algoritmo. Quanto maior o valor dessa medida, maior a possibilidade de novas agregações de estados, o que implica em uma maior redução do diagnosticador.

O primeiro passo para reduzir o diagnosticador consiste em agregar todos os estados positivos de G_d em um único estado, denotado por $\{F\}$, gerando um novo autômato diagnosticador $G'_d = (X'_d, \Sigma_o, f'_d, x'_{0_d})$, em que $X'_d = \{\{x_d\} : x_d \in X_N \cup X_{NY} \cup \{F\}\}$, $f'_d(\{x_d\}, \sigma) = \{f_d(x_d, \sigma)\}$, se $f_d(x_d, \sigma) \in X_N \cup X_{NY}$, $f'_d(\{x_d\}, \sigma) = \{F\}$, se $f_d(x_d, \sigma) \in X_Y$, e $f'_d(\{F\}, \sigma) = \{F\}$ para todo $\sigma \in \Sigma_o$, e $x'_{0_d} = \{x_{0_d}\}$. A agregação dos estados positivos de G_d é possível uma vez que o conhecimento do estado do sistema após a detecção da falha é desnecessário para o diagnóstico e para o cálculo do atraso.

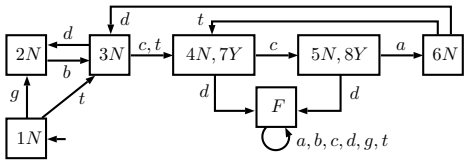

Figura 3. Autômato diagnosticador G'_d.

O exemplo 2 ilustra a construção do autômato diagnosticador G'_d, obtido a partir da redução de G_d.

Exemplo *2.* Considere o mesmo autômato G apresentado no Exemplo 1, e seu diagnosticador G_d mostrado na Figura 2. Os estados positivos de G_d são $\{7Y\}$ e $\{8Y\}$. Assim, é possível agregar $\{7Y\}$ e $\{8Y\}$ em um único estado, denotado por $\{F\}$, gerando o autômato diagnosticador G'_d, ilustrado na Figura 3. Note que um autolaço é introduzido em $\{F\}$ rotulado com todos os eventos observáveis. □

No primeiro passo para a redução de diagnosticadores apenas estados positivos são agregados. Entretanto, em alguns casos, é possível reduzir G'_d, obtendo um novo autômato diagnosticador $G^r_d = (X^r_d, \Sigma_o, f^r_d, x^r_{0_d})$, agregando também estados negativos e incertos de G'_d. No Algoritmo 1, o autômato G''_d, obtido após a agregação de dois estados x e y de um autômato $\hat{G}_d$ em um único estado $x \cup y$, é calculado. Vale ressaltar que todos os diagnosticadores reduzidos neste trabalho são obtidos através da agregação de estados do diagnosticador G'_d. Assim, uma vez que cada estado de G'_d é um conjunto unitário formado por cada estado de G_d ou $\{F\}$, então um estado de um diagnosticador reduzido tem a seguinte forma $\{x_{d_1}, x_{d_2}, \ldots, x_{d_\eta}\}$, em que $\{x_{d_i}\} \in X'_d$. No Algoritmo 1, $\hat{G}_d$ é um autômato reduzido obtido a partir de G'_d ou o próprio G'_d. Note que, no Algoritmo 1, $\hat{G}_d$ e G''_d são considerados autômatos não-determinísticos, *i.e.*, $\hat{f}_d : \hat{X}_d \times \Sigma_o \to 2^{\hat{X}_d}$ e $f''_d : X''_d \times \Sigma_o \to 2^{X''_d}$.

Algorithm 1: MERGE($\hat{G}_d$,x,y)

Input: $\hat{G}_d = (\hat{X}_d, \Sigma_o, \hat{f}_d, \hat{x}_{0_d})$, $x, y \in \hat{X}_d$
Output: $G''_d = (X''_d, \Sigma_o, f''_d, x''_{0_d})$

1 $X''_d = (\hat{X}_d \setminus \{x, y\}) \cup \{x \cup y\}$
2 **if** $\hat{x}_{0_d} \in \{x, y\}$ **then**
3 $\quad x''_{0_d} \leftarrow x \cup y$
4 **else**
5 $\quad x''_{0_d} \leftarrow \hat{x}_{0_d}$
6 Seja $xy = x \cup y$. Defina a função de transição f''_d, para todo $\sigma \in \Sigma_o$, como:
(i) $f''_d(xy, \sigma) = \hat{f}_d(x, \sigma) \cup \hat{f}_d(y, \sigma)$, se $\left(\hat{f}_d(x, \sigma) \cup \hat{f}_d(y, \sigma)\right) \cap \{x, y\} \neq \emptyset$
(ii) $f''_d(xy, \sigma) = \{xy\} \cup (\hat{f}_d(x, \sigma) \setminus \{x, y\}) \cup (\hat{f}_d(y, \sigma) \setminus \{x, y\})$, se $\hat{f}_d(x, \sigma) \cap \{x, y\} \neq \emptyset$ ou $\hat{f}_d(y, \sigma) \cap \{x, y\} \neq \emptyset$
(iii) $f''_d(z, \sigma) = \hat{f}_d(z, \sigma)$, para todo $z \in \hat{X}_d \setminus \{x, y\}$, se $\hat{f}_d(z, \sigma) \cap \{x, y\} = \emptyset$
(iv) $f''_d(z, \sigma) = \{xy\} \cup (\hat{f}_d(z, \sigma) \setminus \{x, y\})$, para todo $z \in \hat{X}_d \setminus \{x, y\}$, se $\hat{f}_d(z, \sigma) \cap \{x, y\} \neq \emptyset$

Teorema *2.* $L(\hat{G}_d) \subseteq L(G''_d)$.

Prova. A prova é obtida diretamente da operação de agregação de estados e será omitida devido à falta de espaço. □

De acordo com o Teorema 2, a linguagem gerada por G''_d, obtido pela agregação de estados de $\hat{G}_d$, contém a linguagem gerada por $\hat{G}_d$. Além disso, como é suposto que o próprio $\hat{G}_d$ é obtido após a agregação de estados de G'_d, então o seguinte resultado pode ser enunciado.

Corolário *1.* $L(G'_d) \subseteq L(G''_d)$.

Esse crescimento da linguagem não deve alterar a diagnosticabilidade do evento de falha σ_f, e nem alterar o atraso para o diagnóstico. Para tanto, a propriedade a seguir deve ser verdadeira.

Propriedade 1:

(*i*) Para todo $s \in L \backslash L_N$, tal que $P_o(s) \notin P_o(L_N)$, tem-se que $f''_d(x''_{0_d}, P_o(s)) = \{\{F\}\}$.
(*ii*) Para todo $\omega \in L_N$, $f''_d(x''_{0_d}, P_o(\omega)) \cap \{\{F\}\} = \emptyset$. □

A condição (*i*) da Propriedade 1 garante que todas as sequências de falha $s \in L \backslash L_N$, que podem ser diagnosticadas por G_d, também podem ser diagnosticadas utilizando G''_d e com o mesmo atraso. Além disso, a condição (*ii*) da Propriedade 1 evita a geração de falsos positivos.

Para obter G''_d, apenas estados negativos e incertos de G'_d são agregados. Assim, é necessário definir quais estados de G'_d podem ser agregados para garantir a Propriedade 1.

Definição *4.* Estados $x, y \in X'_d \setminus \{\{F\}\}$ são ditos contraditórios se existe $\sigma \in \Sigma_o$ tal que $f'_d(x, \sigma) = \{F\}$ e $f'_d(y, \sigma) \subset X_N \cup X_{NY}$, ou vice-versa. Caso contrário, os estados x e y são ditos não-contraditórios. □

O próximo teorema apresenta uma condição necessária para garantir que G''_d satisfaz a Propriedade 1.

Teorema *3.* Se a Propriedade 1 é válida, então estados contraditórios de G'_d não são agregados na obtenção de G''_d.

Prova. Considere, sem perda de generalidade, dois estados contraditórios $x, y \in X'_d \setminus \{\{F\}\}$, *i.e.*, existe $\sigma \in \Sigma_o$ tal que $f'_d(x, \sigma) = \{F\}$ e $f'_d(y, \sigma) \subset X_N \cup X_{NY}$. Seja $\tilde{s} \in L$ a sequência de maior comprimento tal que $f'_d(x'_{0_d}, P_o(\tilde{s})) = x$. Assim, de acordo com o cálculo de G'_d, tem-se que $s = \tilde{s}\sigma \in L \setminus L_N$ e $P_o(s) \notin P_o(L_N)$. Se os estados x e y forem agregados, formando um único estado $xy = x \cup y$, então, de acordo com a linha 6 do Algoritmo 1, $f''_d(x''_{0_d}, P_o(\tilde{s})) = \{xy\}$, e $f''_d(x''_{0_d}, P_o(\tilde{s})\sigma) = \{f'_d(x, \sigma)\} \cup \{f'_d(y, \sigma)\}$, se $f'_d(y, \sigma) \notin \{x, y\}$, ou $f''_d(x''_{0_d}, P_o(\tilde{s})\sigma) = \{f'_d(x, \sigma)\} \cup \{xy\}$, se $f'_d(y, \sigma) \in \{x, y\}$. Assim, $f'_d(y, \sigma) \subset X_N \cup X_{NY}$, e G''_d é incerto sobre a ocorrência da falha após a observação de $P_o(\tilde{s})\sigma$, violando a Propriedade 1. □

Note que não agregar estados contraditórios é apenas uma condição necessária para satisfazer a Propriedade 1, *i.e.*, é possível violar a Propriedade 1 agregando-se apenas estados não-contraditórios. Por exemplo, quando existir uma sequência de eventos $s \in \Sigma^*_o$ de comprimento maior do que um tal que $f'_d(x, s) = \{F\}$ e $f'_d(y, s) \subset X_N \cup X_{NY}$ ou vice-versa, e x e y forem agregados, então a Propriedade 1 não será válida. Isso leva à seguinte definição.

Definição *5.* Estados $x, y \in X'_d \setminus \{\{F\}\}$ são ditos possivelmente agregáveis se não existir uma sequência de eventos $s \in \Sigma_o^*$ tal que $f'_d(x, s) = \{F\}$ e $f'_d(y, s) \subset X_N \cup X_{NY}$, ou vice-versa. □

Teorema *4.* Dois estados $x, y \in X'_d \setminus \{\{F\}\}$ podem ser agregados para formar G''_d satisfazendo a Propriedade 1, apenas se x, y forem possivelmente agregáveis.

Prova. A prova é obtida de maneira direta e será omitida devido à falta de espaço. □

Note que a condição apresentada no Teorema 4 é apenas necessária, uma vez que na Definição 5 não é levada em consideração a linguagem em excesso que pode ser gerada após a agregação de x e y.

No Algoritmo 2, um método para calcular todos os pares de estados do autômato G'_d que certamente não podem ser agregados de acordo com a Definição 5 é apresentado. Os pares de estados que certamente não podem ser agregados são armazenados no conjunto M_{not}. Note, de

Algorithm 2: NOT_MERGEABLE(G'_d)

Input: $G'_d = (X'_d, \Sigma_o, f'_d, x'_{0_d})$
Output: Conjunto M_{not} formado por todos os pares de estados que não são agregáveis

1 $C := \{(x, y) \in X'_d \setminus \{\{F\}\} \times X'_d \setminus \{\{F\}\} : x, y$ são contraditórios$\}$
2 $M_{not} \leftarrow C$
3 Sinalize todos os pares em M_{not}
4 Forme uma fila *first-in, first-out* Q com todos os pares de estados de C em qualquer ordem
5 **while** $Q \neq \emptyset$ **do**
6 $\quad (x, y) \leftarrow head[Q]$
7 $\quad$ **for** $(w, z) \in X'_d \setminus \{\{F\}\} \times X'_d \setminus \{\{F\}\}$ *não sinalizado tal que existe* $\sigma \in \Sigma_o$ *satisfazendo* $f'_d(w, \sigma) = x$ *e* $f'_d(z, \sigma) = y$ **do**
8 $\quad\quad M_{not} \leftarrow M_{not} \cup \{(w, z)\}$
9 $\quad\quad$ Sinalize (w, z)
10 $\quad\quad Enqueue(Q, (w, z))$
11 $\quad Dequeue(Q)$

acordo com a definições 4 e 5, que estados contraditórios não são agregáveis. Assim, na linha 2 do Algoritmo 2, C é adicionado ao M_{not}. Além disso, note que se um par de estados (w, z) leva, após a ocorrência de um evento $\sigma_o \in \Sigma_o$, a um par de estados não-agregáveis, então (w, z) também é um par não-agregável. Dessa forma, nas linhas 7 e 8, esses estados são adicionados ao M_{not}. Uma fila *first-in first-out* Q é utilizada no Algoritmo 2, em que $head[Q]$, $Enqueue(Q, (w, z))$, e $Dequeue(Q)$, denotam, respectivamente, os procedimentos para obter o primeiro elemento de Q, adicionar (w, z) ao final de Q, e remover o primeiro elemento de Q.

Como cada par de estados é sinalizado apenas uma vez no Algoritmo 2, então cada par é adicionado e retirado da fila apenas uma vez. Assim, a complexidade do Algoritmo 2 é $O(|X'_d|^2)$, que está associada ao número de pares de estados de G'_d.

Exemplo *3.* Considere o diagnosticador G'_d ilustrado na Figura 3, em que $\Sigma = \{a, b, c, d, t, \sigma_f\}$, $\Sigma_o = \{a, b, c, d, t\}$ e $\Sigma_{uo} = \Sigma_f = \{\sigma_f\}$, e suponha que é desejado calcular todos os pares de estados de G'_d que não podem ser agregados de acordo com a Definição 5. Assim, é necessário obter M_{not} utilizando o Algoritmo 2. Na linha 1 do Algoritmo 2, o conjunto C, formado por todos os pares de estados contraditórios de G'_d, é obtido:

$$C = \{(\{3N\}, \{4N, 7Y\}), (\{3N\}, \{5N, 8Y\}), (\{4N, 7Y\}, \{6N\}), (\{5N, 8Y\}, \{6N\})\}.$$

Na linha 2, C é adicionado ao conjunto M_{not}, e, na linha 4, a fila Q é formada com todos os pares de estados de C. Após isso, o procedimento principal do laço while da linha 5 começa. Se não for possível alcançar um par de estados (x, y) em M_{not} a partir de um par de estados (w, z) após a ocorrência de um evento $\sigma \in \Sigma_o$, então (x, y) é removido de Q. Caso contrário, o par (w, z) é adicionado a M_{not}. Considere que o primeiro elemento na fila Q é $(\{3N\}, \{4N, 7Y\})$. Pela Figura 3, pode-se notar que o par $(\{3N\}, \{4N, 7Y\})$ é alcançado pelo par $(\{1N\}, \{6N\})$ através do evento t. Assim, $(\{1N\}, \{6N\})$ é adicionado a M_{not}, mas $(\{3N\}, \{4N, 7Y\})$ não é removido de Q, uma vez que esse estado ainda é alcançado por outro par de estados. O algoritmo continua e é visto que o par $(\{3N\}, \{4N, 7Y\})$ é alcançado pelo par $(\{1N\}, \{3N\})$ através do evento t. Dessa forma, $(\{1N\}, \{3N\})$ é adicionado a M_{not}. Como não existe mais nenhum outro par de estados que alcança o estado $(\{3N\}, \{4N, 7Y\})$, então o mesmo é removido de Q. O algoritmo continua até Q estar vazia. Procedendo dessa maneira obtém-se o conjunto $M_{not} = \{(\{1N\}, \{3N\}), (\{1N\}, \{6N\}), (\{3N\}, \{4N, 7Y\}), (\{3N\}, \{5N, 8Y\}), (\{4N, 7Y\}, \{6N\}), (\{5N, 8Y\}, \{6N\})\}$.

□

No Algoritmo 3, é apresentado um método para encontrar um diagnosticador determinístico G''_d obtido após a agregação de dois estados $x, y \in \hat{X}_d$ de um autômato $\hat{G}_d$. Para tal, o Algoritmo 1 é executado até a função de transição $\tilde{f}_d$ se tornar determinística, *i.e.*, apenas um estado pode ser alcançado após a ocorrência de um evento ativo. Assim, G''_d obtido pelo Algoritmo 3 será determinístico. Outra importante característica de G''_d é que, assim como em Su and Wonham (2004) e Cai et al. (2019), seu espaço de estados X''_d é formado por uma partição de conjuntos disjuntos.

Algorithm 3: DET_MERGE($\hat{G}_d, x, y$)

Input: $\hat{G}_d = (\hat{X}_d, \Sigma_o, \hat{f}_d, \hat{x}_{0_d})$, e $x, y \in \hat{X}_d$
Output: $G''_d = (X''_d, \Sigma_o, f''_d, x''_{0_d})$

1 $\tilde{G}_d \leftarrow$ MERGE($\hat{G}_d, x, y$)
2 **while** $\exists w \in \tilde{X}_d$ *tal que* $|\tilde{f}_d(w, \sigma)| > 1$, *para* $\sigma \in \Sigma_o$ **do**
3 $\quad \tilde{G}_d \leftarrow$ MERGE($\tilde{G}_d, \tilde{x}, \tilde{y}$), em que $\tilde{x}, \tilde{y} \in \tilde{f}_d(w, \sigma)$
4 $\hat{G}_d \leftarrow \tilde{G}_d$

Note que, para garantir o determinismo do autômato, após agregar dois estados x, y, é necessário agregar todos os pares de estados que são alcançados a partir de x e y após a ocorrência da mesma sequência observável. Assim, é possível que a agregação de dois estados possivelmente agregáveis x, y de acordo com a Definição 5 leve à agregação de dois estados não-agregáveis, o que mostra que x, y

não podem ser agregados de fato. Além disso, é possível que agregar dois estados (x, y), obrigue a agregação de outro par de estados que contenha x ou y. Considere, sem perda de generalidade, que o estado alcançado por (x, y) após a ocorrência de um evento observável, seja (x, z). Nesse caso, para garantir o determinismo, é necessário agregar os três estados x, y, e z em um único estado. Para tal, todos os pares de estados formados por x, y, e z devem ser agregáveis.

Uma vez que a condição do Teorema 4 para garantir a Propriedade 1 é apenas necessária, torna-se importante enunciar uma condição necessária e suficiente para verificar a Propriedade 1. Considere que $MS(x, y)$ denota o conjunto formado por todos os conjuntos de estados de G'_d que devem ser agregados para garantir o determinismo do diagnosticador reduzido, após a agregação dos estados $x, y \in X'_d \setminus \{\{F\}\}$. Então, a seguinte condição necessária e suficiente para garantir que os estados $x, y \in X'_d$ são agregáveis pode ser enunciada.

Teorema *5.* Estados $x, y \in X'_d$ são agregáveis se, e somente se, $F \notin M$ para todo $M \in MS(x, y)$.

Prova. ($\Rightarrow$) Seja G''_d um diagnosticador determinístico obtido pela agregação dos estados x, y de G'_d de acordo com o Algoritmo 3. Pela construção de G''_d, tem-se que se $s \in L(G'_d)$ é tal que $f'_d(x'_{0_d}, s) = x'_d$, então $x'_d \subseteq x''_d$, em que $x''_d = f''_d(x''_{0_d}, s)$. Assim, se $F \notin M$, para todo $M \in MS(x, y)$, então todas as sequências de falha que levam G'_d à $\{F\}$, também levam G''_d ao estado $\{F\}$, *i.e.*, todas as sequências de falha que podem ser diagnosticadas utilizando G'_d também podem ser diagnosticadas utilizando G''_d. Seguindo o mesmo raciocínio, pode-se notar que todas as sequências livres de falha não levam G''_d ao estado $\{F\}$, o que mostra que G''_d não gera falso positivo para a detecção da falha.

($\Leftarrow$) Suponha agora que $F \in M$, em que $M \in MS(x, y)$. Então, existe um estado $x_d \in X_N \cup X_{NY}$, tal que $x_d \in M$. Pela construção de G''_d, há sequências diferentes $s_1, s_2 \in L(G'_d)$, tais que $f'_d(x'_{0_d}, s_1) = \{x_d\}$ e $f'_d(x'_{0_d}, s_2) = \{F\}$. Assim, se um estado M de G''_d for alcançado, não é possível afirmar qual sequência s_1 ou s_2 ocorreu e, consequentemente, se houve a ocorrência do evento de falha. Assim, o diagnóstico do evento de falha, se possível, será atrasado se a sequência s_2 for observada. □

No Algoritmo 4, um método para encontrar todos os conjuntos $MS(x, y)$, formados apenas pelos conjuntos de estados agregáveis de um autômato determinístico $\hat{G}_d$, obtido após a agregação de estados $x, y \in \hat{X}_d$, é apresentado. Os conjuntos $MS(x, y)$ são armazenados em uma lista ML. Assim, se x, y forem não-agregáveis, então $MS(x, y)$ não é adicionado à lista ML. Como no Algoritmo 4 a condição necessária apresentada no Teorema 4 é utilizada, sem a obtenção do autômato G''_d de maneira completa obtida utilizando o Algoritmo 3, o custo computacional para calcular $MS(x, y)$ pode ser reduzido.

No Algoritmo 4, nas linhas 8 e 9, os estados (w, z) alcançados pelo estado (x, y) são computados, e a agregação de w, z é testada nas linhas 10 a 12 utilizando a condição necessária fornecida pelo Teorema 4. Note que esta condição pode ser facilmente verificada e, se não for verdadeira, então x, y não será agregável e o algoritmo poderá continuar verificando a agregação de outros pares de estados. Se (w, z) for possivelmente agregável, na linha 14 é verificado se existe em $MS(x, y)$ um conjunto de estados com pelo menos um elemento igual a w ou z. Se esse conjunto não existir, então, na linha 15, (w, z) é adicionado a $MS(x, y)$. Por outro lado, se há um conjunto $M \in MS(x, y)$ tal que w ou z pertençam a M, então, nas linhas 17 a 26, é verificado se é possível a substituição de M pelo novo conjunto de estados agregados em $MS(x, y)$. Para tal, todos os pares de estados formados com um elemento de $\{w, z\}$ e outro de M devem ser possivelmente agregáveis. Finalmente, nas linhas 29 e 30, a condição necessária e suficiente do Teorema 5 é verificada e, se (x, y) for agregável, então o par $[(x, y), MS(x, y)]$ é adicionado à lista ML.

Algorithm 4: COMPUTE_MS($\hat{G}_d, x, y$)

Input: $\hat{G}_d = (\hat{X}_d, \Sigma_o, \hat{f}_d, \hat{x}_{0_d})$
Output: Lista ML em que cada elemento é da forma $[(x, y), MS(x, y)]$

```
1  M̂_not ← NOT_MERGEABLE(Ĝ_d)
2  ML ← NULL
3  for (x, y) ∈ (X̂_d × X̂_d) \ M̂_not do
4  |  MS(x, y) ← {{x, y}}
5  |  Forme uma fila first-in, first-out Q com o par
   |    (x, y)
6  |  while Q ≠ ∅ do
7  |  |  (x, y) ← head[Q]
8  |  |  for σ ∈ Γ_{Ĝ_d}(x) ∩ Γ_{Ĝ_d}(y) do
9  |  |  |  (w, z) ← (f̂_d(x, σ), f̂_d(y, σ))
10 |  |  |  if (w, z) ∈ M̂_not then
11 |  |  |  |  MS(x, y) ← ∅
12 |  |  |  |  vá para a linha 3
13 |  |  |  else
14 |  |  |  |  if ∄M ∈ MS(x, y) tal que {w, z} ∩ M ≠ ∅
   |  |  |  |    then
15 |  |  |  |  |  MS(x, y) ← MS(x, y) ∪ {{w, z}}
16 |  |  |  |  else
17 |  |  |  |  |  M̂ ← ∅
18 |  |  |  |  |  for M ∈ MS(x, y) tal que
   |  |  |  |  |    {w, z} ∩ M ≠ ∅ do
19 |  |  |  |  |  |  U = {(u, v) : u ∈ {w, z} e v ∈ M}
20 |  |  |  |  |  |  if U ∩ M̂_not ≠ ∅ then
21 |  |  |  |  |  |  |  MS(x, y) ← ∅
22 |  |  |  |  |  |  |  vá para a linha 3
23 |  |  |  |  |  |  else
24 |  |  |  |  |  |  |  M̂ ← M̂ ∪ M
25 |  |  |  |  |  |  |  MS(x, y) ← MS(x, y) \ M
26 |  |  |  |  |  |  MS(x, y) ← MS(x, y) ∪ {M̂ ∪ {w, z}}
27 |  |  |  |  Enqueue(Q, (w, z))
28 |  |  |  Dequeue(Q)
29 |  |  if ∄M ∈ MS(x, y) tal que F ∈ M then
30 |  |  |  Adicione [(x, y), MS(x, y)] à lista ML
```

Como no Algoritmo 4, pares de estados são verificados para cada par de estados $(x, y) \in \hat{X}_d \times \hat{X}_d$, a complexidade computacional do Algoritmo 4 é $O(|\hat{X}_d|^4)$.

Exemplo *4.* Considere o autômato G ilustrado na Figura 1, e seu diagnosticador G'_d, ilustrado na Figura 3, cujo conjunto de estados contraditórios M_{not} foi obtido no Exemplo 3. Para obter a lista ML que contém os estados agregáveis, é necessário computar $MS(x,y)$ para cada par de estados $(x,y) \in (X'_d \times X'_d) \setminus M_{not}$, de acordo com o Algoritmo 4. Na linha 4, o primeiro par (x,y) é adicionado a $MS(x,y)$ e na linha 5 a fila Q é formada começando pelo par (x,y). Na linha 6, o laço while começa. Considere que o primeiro estado em Q é $(\{1N\},\{2N\})$. Note que o par $(\{1N\},\{2N\})$ não alcança nenhum par de estados de G'_d através da execução do mesmo evento. Como $(\{1N\},\{2N\})$ é agregável, então o conjunto $(\{1N\},\{2N\})$ é adicionado a $MS(\{1N\},\{2N\})$. Suponha que $(\{3N\},\{6N\})$ seja o próximo par de estados adicionado a Q na linha 5. Então, o par $(\{2N\},\{3N\})$ que é alcançado a partir de $(\{3N\},\{6N\})$ através do evento d é computado na linha 9, e a agregabilidade de $(\{2N\},\{3N\})$ é testada na linha 10. Como $(\{2N\},\{3N\}) \notin M_{not}$, então é verificado na linha 14 se $\{\{2N\},\{3N\}\} \cap \{\{3N\},\{6N\}\} = \emptyset$. Como $\{\{2N\},\{3N\}\} \cap \{\{3N\},\{6N\}\} \neq \emptyset$, então na linha 19 o conjunto $U = \{(\{2N\},\{6N\})\}$ é formado, e na linha 20 é verificado se $(\{2N\},\{6N\})$ é não-agregável. Como $(\{2N\},\{6N\})$ é agregável, o conjunto $\{\{2N\},\{3N\},\{6N\}\}$ é adicionado a $MS(\{3N\},\{6N\})$. O Algoritmo 4 continua até que todos os pares agregáveis x,y, cuja agregação leva a um diagnosticador determinístico, sejam adicionados à lista ML. $MS(x,y)$ para cada par de estados agregáveis é dado por: $MS(\{1N\},\{2N\}) = \{\{1N\},\{2N\}\}$; $MS(\{1N\},\{4N,7Y\}) = \{\{1N\},\{4N,7Y\}\}$; $MS(\{1N\},\{5N,8Y\}) = \{\{1N\},\{5N,8Y\}\}$; $MS(\{2N\},\{3N\}) = \{\{2N\},\{3N\}\}$; $MS(\{2N\},\{4N,7Y\}) = \{\{2N\},\{4N,7Y\}\}$; $MS(\{2N\},\{5N,8Y\}) = \{\{2N\},\{5N,8Y\}\}$; $MS(\{2N\},\{6N\}) = \{\{2N,6N\}\}$; $MS(\{3N\},\{6N\}) = \{\{2N,3N,6N\}\}$; $MS(\{4N,7Y\},\{5N,8Y\}) = \{\{4N,7Y\},\{5N,8Y\}\}$. □

Após agregar um par de estados $x,y \in \hat{X}_d$ de um autômato $\hat{G}_d$, todas as agregações de estados definidas em $MS(x,y)$ devem ser realizadas. Então, um novo autômato $G''_d = (X''_d, \Sigma_o, f''_d, x''_{0_d})$ é computado. Para escolher qual par de estados (x,y) deve ser agregado para obter G''_d, o número de pares de estados possivelmente agregáveis de G''_d, N_m, é definido da seguinte forma:

$$N_m = \frac{|X''_d|(|X''_d|-1)}{2} - |M''_{not}|,$$

em que $\frac{|X''_d|(|X''_d|-1)}{2}$ é o número total de pares de estados de G''_d, e M''_{not} é o conjunto de pares de estados de G''_d que não são agregáveis, obtido através do Algoritmo 2.

Se N_m for grande, então há várias possibilidades de agregação de pares de estados de G''_d, o que implica que a chance de agregar mais estados no próximo passo do processo de redução será maior do que se N_m for um número pequeno. Utilizando este raciocínio escolhemos agregar, a cada passo do processo de redução, o par de estados do diagnosticador $\hat{G}_d$ cuja agregação leva ao diagnosticador G''_d com o maior valor de N_m. Este procedimento está descrito no Algoritmo 5. Seguindo os passos do Algoritmo 5, o diagnosticador reduzido G^r_d é computado a partir de G'_d. Nas linhas 2 e 3, M_{not} e ML são computados para o autômato $\hat{G}_d$. Então, enquanto houver pares agregáveis em ML, na linha 7 o autômato G''_d é obtido para cada par $[(x,y), MS(x,y)]$ de

Algorithm 5: REDUCED_DIAGNOSER(G'_d)

Input: G'_d
Output: G^r_d

1 $\hat{G}_d \leftarrow G'_d$
2 $ML \leftarrow$COMPUTE_MS($\hat{G}_d, M_{not}$)
3 **while** $ML \neq NULL$ **do**
4 $\quad LN \leftarrow$ NULL
5 $\quad$ **for** *cada elemento* $[(x,y), MS(x,y)]$ *em* ML **do**
6 $\quad\quad G''_d \leftarrow$ DET_MERGE($\hat{G}_d, x, y$)
7 $\quad\quad M''_{not} \leftarrow$ NOT_MERGEABLE(G''_d)
8 $\quad\quad$ Calcule $N_m = \frac{|X''_d|(|X''_d|-1)}{2} - |M''_{not}|$
9 $\quad\quad$ Adicione $[N_m, G''_d]$ à lista LN
10 $\quad$ Encontre o número máximo N_{max} de N_m na lista LN
11 $\quad$ Escolha um autômato $G''_{d_{max}}$ associado a N_{max} em LN
12 $\quad \hat{G}_d \leftarrow G''_{d_{max}}$
13 $\quad ML \leftarrow$COMPUTE_MS($\hat{G}_d, M_{not}$)
14 $G^r_d \leftarrow \hat{G}_d$

ML e, na linha 9, N_m é computado para cada G''_d. Na linha 10, o número máximo N_{max} de N_m é calculado. Note que mais de um autômato G''_d pode ter o mesmo valor de $N_m = N_{max}$. Assim, na linha 11, um autômato $G''_{d_{max}}$ associado a N_{max} é aleatoriamente escolhido. Em seguida, na linha 13, ML é computado para G''_d. Se ML for NULL, *i.e.*, se não houver pares de estados agregáveis, o algoritmo para e retorna $G^r_d = G''_d$. Caso contrário, o algoritmo continua e um novo processo de redução ocorrerá.

No pior caso, o laço while do Algoritmo 5 agrega pares de estados até restar apenas um estado, *i.e.*, ocorrerão $|X'_d| - 1$ agregações. Além disso, a cada iteração do laço for do Algoritmo 5, pares de estados são verificados para cada par de estados $(x,y) \in X'_d \times X'_d$, e ML é computada. Consequentemente, a complexidade do Algoritmo 5 é $O(|X'_d|^5)$. O processo de redução do diagnosticador está ilustrado no exemplo a seguir.

Exemplo *5.* Considere novamente o autômato G ilustrado na Figura 1, e o diagnosticador G'_d ilustrado na Figura 3. Desejamos obter um diagnosticador reduzido, G^r_d, tal que o evento de falha possa ser diagnosticado com o mesmo atraso se utilizarmos G'_d. Para tanto, será utilizado o Algoritmo 5. Na linha 2, o conjunto M_{not}, apresentado no Exemplo 3, é computado. Na linha 3, $MS(x,y)$, obtido no Exemplo 4, é computado para cada par de estados agregáveis de G'_d. Na linha 9, N_m é calculado para cada $MS(x,y)$ obtido no passo anterior, e a lista LN é obtida. Note que G'_d possui seis estados, logo $|X'_d| = 6$. Os valores de N_m para cada par de estados (x,y) de ML são: (i) 3, para $(\{1N\},\{2N\})$; (ii) 6, para $(\{1N\},\{4N,7Y\})$; (iii) 6, para $((\{1N\},\{5N,8Y\})$; (iv) 4, para $(\{2N\},\{3N\})$; (v) 3, para $(\{2N\},\{4N,7Y\})$; (vi) 3, para $(\{2N\},\{5N,8Y\})$; (vii) 4, para $(\{2N\},\{6N\})$; $(viii)$ 4, para $(\{3N\},\{6N\})$; e (ix) 6, para $(\{4N,7Y\},\{5N,8Y\})$. Assim, $N_{max} = 6$. Então, de acordo com o Algoritmo 5, qualquer par de estados tal que $N_m = 6$ pode ser escolhido para ser agregado. Por exemplo, suponha que os estados $\{1N\}$ e

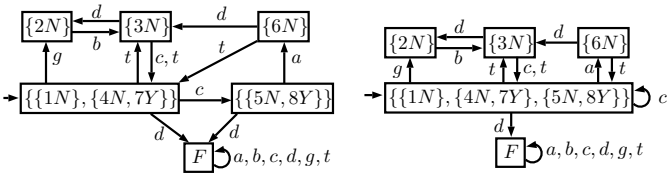

Figura 4. Agregação dos estados $\{1N\}$ e $\{4N, 7Y\}$ (a); e $\{\{1N\}, \{4N, 7Y\}\}$ com $\{5N, 8Y\}$ (b), no Exemplo 5.

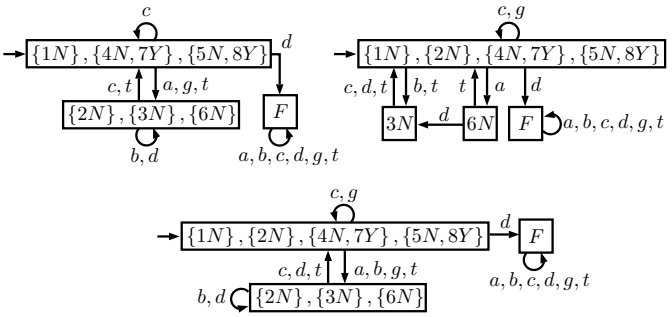

Figura 5. Autômato G_d^r do Exemplo 5 (a); diagnosticadores reduzidos aplicando os métodos propostos em Su and Wonham (2004) (b) e Minhas (2002) (c).

$\{4N, 7Y\}$ sejam agregados. Então, o diagnosticador G_d'', ilustrado na Figura 4(a), é obtido.

Como ainda há estados agregáveis, o algoritmo continua computando ML e, para cada $[(x, y), MS(x, y)]$ de ML, é procurado o diagnosticador reduzido determinístico que leva ao maior valor de N_m. Os valores de N_m para cada par de estados (x, y) de ML são: (i) 1, para $(\{\{1N\}, \{4N, 7Y\}\}, \{2N\})$; (ii) 4, para $(\{\{1N\}, \{4N, 7Y\}\}, \{5N, 8Y\})$; (iii) 2, para $(\{2N\}, \{3N\})$; (iv) 1, para $(\{2N\}, \{5N, 8Y\})$; (v) 2, para $(\{2N\}, \{6N\})$; e (vi) 2, para $(\{3N\}, \{6N\})$. Como $N_{max} = 4$, então, o par de estados $\{\{1N\}, \{4N, 7Y\}\}, \{\{5N, 8Y\}\}$ é agregado, e o diagnosticador G_d'', ilustrado na Figura 4(b), é obtido. Como G_d'' ainda possui estados agregáveis, o Algoritmo 5 continua. Assim, os valores de N_m são calculados: (i) 0, para $(\{\{1N\}, \{4N, 7Y\}, \{5N, 8Y\}\}, \{2N\})$; (ii) 1, para $(\{2N\}, \{3N\})$; (iii) 1, para $(\{2N\}, \{6N\})$; e (iv) 1, para $(\{3N\}, \{6N\})$. Como $N_{max} = 1$, então qualquer par de estados associado a esse valor de N_m pode ser agregado. Suponha que os estados $\{3N\}, \{6N\}$ sejam agregados. Então, como $MS(\{3N\}, \{6N\}) = \{\{2N\}, \{3N\}\{6N\}\}$, os três estados $\{2N\}, \{3N\}, \{6N\}$ são agregados. Note que não é mais possível reduzir G_d'', já que $ML = NULL$. Portanto, o diagnosticador reduzido G_d^r é obtido, conforme ilustrado na Figura 5(a). É importante ressaltar que o diagnosticador reduzido obtido aplicando-se o método proposto em Su and Wonham (2004) possui mais estados que G_d^r (Figura 5(b)). Já aplicando o método de Minhas (2002), um autômato com o mesmo número de estados de G_d^r é obtido, porém esse autômato é não-determinístico (Figura 5(c)). □

4. CONCLUSÕES

Neste artigo, um método para computar um diagnosticador reduzido determinístico, mantendo tanto a diagnosticabilidade quanto o atraso para o diagnóstico do diagnosticador original, é proposto. A estratégia de redução proposta não depende do ordenamento prévio dos estados, podendo levar a um diagnosticador determinístico com um número menor de estados do que quando são utilizadas outras estratégias propostas na literatura.

REFERÊNCIAS

Cai, K., Giua, A., and Seatzu, C. (2019). On consistent reduction in discrete-event systems. In *Proc. 15th IEEE Int. Conf. on Automation Science and Engineering*, 474–479. Vancouver, Canada.

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

Clavijo, L.B. and Basilio, J.C. (2017). Empirical studies in the size of diagnosers and verifiers for diagnosability analysis. *Discrete Event Dynamic Systems*, 27(4), 701–739.

da Silva Neto, F.A.C. (2008). *Redução de Supervisores Utilizando Marcação por Eventos e Métodos de Otimização*. Dissertação de mestrado, Departamento de Automação e Sistemas, Universidade Federal de Santa Catarina, Florianópolis, Brasil.

Hopcroft, J.E., Motwani, R., and Ullman, J.D. (2006). *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Boston, MA: USA, 3 edition.

Minhas, R.S. (2002). *Complexity Reduction in Discrete Event Systems*. Ph.D. thesis, ECE Department, University of Toronto, Canada.

Sampath, M., Sengupta, R., and Lafortune, S. (1995). Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 9(40), 1555–1575.

Sivollela, L.F., da Cunha, A.E.C., and Ades, R. (2006). Redução de supervisores como ferramenta para a implementação de supervisores em controladores discretos. In *Congresso Brasileiro de Automática*. Salvador, Brasil.

Su, R. and Wonham, W.M. (2004). Supervisor reduction for discrete-event systems. *Journal of Discrete Event Dynamic Systems*, 14(1), 31–53.

Tomola, J.H.A., Cabral, F.G., Carvalho, L.K., and Moreira, M.V. (2017). Robust disjunctive-codiagnosability of discrete-event systems against permanent loss of observations. *IEEE Transactions on Automatic Control*, 62(11), 5808–5815.

Vaz, A.F. and Wonham, W.M. (1986). On supervisor reduction in discrete-event systems. *International J. Control*, 44(2), 475–491.

Viana, G.S., Moreira, M.V., and Basilio, J.C. (2019). Codiagnosability analysis of discrete-event systems modeled by weighted automata. *IEEE Transactions on Automatic Control*, 64(10), 4361–4368.

Yoo, T.S. and Garcia, H.E. (2003). Computation of fault detection delay in discrete-event systems. In *Proceedings of the 14th International Workshop on Principles of Diagnosis, DX'03*, 207–212. Washington, USA.

Yoo, T. and Lafortune, S. (2002). Polynomial-time verification of diagnosability of partially observed discrete-event systems. *IEEE Transactions on Automatic Control*, 47(9), 1491–1495.

Zad, S.H., Kwong, R.H., and Wonham, W.M. (2003). Fault diagnosis in discrete-event systems: Framework and model reduction. *IEEE Transactions on Automatic Control*, 48(7), 1199–1212.