DIAGNOSE DE FALHAS EM SISTEMAS A EVENTOS DISCRETOS MODELADOS POR AUTÔMATOS FINITOS

João Carlos Basilio*
basilio@dee.ufrj.br

Lilian Kawakami Carvalho*
lilian@coep.ufrj.br

Marcos Vicente Moreira*
moreira@dee.ufrj.br

*Universidade Federal do Rio de Janeiro
COPPE - Programa de Engenharia Elétrica
Cidade Universitária, Ilha do Fundão, Centro de Tecnologia
21949-900, Rio de Janeiro, R.J., Brasil

ABSTRACT

Fault diagnosis of discrete event systems modeled as automata
This tutorial presents the background necessary to the study and research on fault diagnosis of discrete-event systems modeled as automata. Both centralized diagnosability and co-diagnosability with coordination are considered. Besides presenting necessary and sufficient conditions for the verification of diagnosability and co-diagnosability, the paper also presents tests using diagnosers and verifiers. Recent results on centralized diagnosis under partial observation are also addressed.

KEYWORDS: Fault diagnosis, Discrete Event Systems, Automata.

RESUMO

Este tutorial apresenta os fundamentos necessários para o estudo e a pesquisa em diagnose de falhas de sistemas a eventos discretos modelados por autômatos. Tanto a diagnose centralizada quanto a descentralizada com coordenação (codiagnose) são consideradas. Além de apresentar as condições necessárias e suficientes para a verificação da diagnosticabilidade e codiagnosticabilidade, este artigo também apresentada testes utilizando diagnosticadores e verificadores. Resultados recentes envolvendo a diagnose centralizada em sistemas sob observação parcial são também considerados.

PALAVRAS-CHAVE: Diagnose de falhas, Sistemas a Eventos Discretos, Autômatos.

1 INTRODUÇÃO

A diagnose de falhas em sistemas a eventos discretos (SEDs) tem despertado grande interesse nos últimos anos. As metodologias desenvolvidas para a diagnose de falhas de SEDs podem ser aplicadas não só a sistemas em que o modelo por eventos discretos é o mais apropriado (redes de comunicação e sistemas de computação e de manufatura), como também a diversos sistemas dinâmicos de variáveis contínuas (SDVC), uma vez que esses sistemas podem também ser modelados como SEDs dependendo do grau de abstração.

Dois paradigmas norteiam a diagnose de falhas em SEDs:

1. As falhas a serem diagnosticadas são eventos não observáveis, isto é, eventos cujas ocorrências não podem ser registradas por sensores;

2. A ocorrência de falhas altera o comportamento do sistema, porém não necessariamente leva o sistema a uma parada; por exemplo, em sistemas de manufatura, a ocorrência de uma

Artigo submetido em 09/10/2009 (Id.: 01046)
Revisado em 30/11/2009, 13/05/2010, 20/05/2010
Aceito sob recomendação da Editora Associada Profa. Emilia Villani

falha não diagnosticada pode levar a uma degradação dos indicadores de eficácia global dos equipamentos (disponibilidade, eficiência e qualidade).

De uma maneira informal, diz-se que um evento de falha pode ser diagnosticado se a sua ocorrência puder ser detectada após a ocorrência de um número finito de eventos observáveis. Para esse fim, são construídos sistemas para a diagnose de falhas, cujo objetivo é inferir e informar a ocorrência de falhas tendo como base somente os eventos que tenham sido observados, isto é, registrados pelos sensores. O projeto desses sistemas requer, em primeiro lugar, a construção de um modelo a eventos discretos do sistema que capture tanto o comportamento normal quanto o comportamento do sistema levando-se em consideração a ocorrência da falha. A segunda parte do projeto é calcada em um arcabouço teórico desenvolvido nas duas últimas décadas e que será revisto neste tutorial e consiste no desenvolvimento de um conjunto de regras (protocolo) a serem seguidas para a identificação e a diagnose de falhas.

Neste tutorial, somente SEDs modelados por autômatos serão considerados (Cassandras e Lafortune, 2008; Hopccroft et al., 2007). O objetivo principal é apresentar os fundamentos necessários para o estudo e a pesquisa em diagnose de falhas considerando tanto a diagnose centralizada quanto a descentralizada com coordenação (codiagnose). Nesse contexto, será apresentada uma breve revisão bibliográfica, cujo objetivo é apenas dar uma idéia geral dos principais tópicos de pesquisa já considerados.

O problema da diagnose de falhas foi trazido para o contexto de SEDs por Lin (1994), que introduziu o conceito da capacidade de se diagnosticar a ocorrência de uma falha em um sistema. Logo a seguir, Sampath et al. (1995) apresentaram condições necessárias e suficientes para a diagnose de falhas de SEDs e propuseram a construção de um autômato diagnosticador que permite tanto inferir sobre a capacidade de diagnosticar as falhas presentes no sistema quanto ser usado para realizar a diagnose de falhas em tempo real. Em um trabalho correlacionado, Sampath et al. (1996) consideraram o problema do desenvolvimento de modelos a eventos discretos para a diagnose de falhas.

Para tornar possível a diagnose de uma falha em SEDs cujos modelos não satisfazem as condições para diagnosticabilidade apresentadas em Sampath et al. (1995), as seguintes soluções podem ser adotadas:

1. Introdução de mais sensores no sistema. Essa abordagem tem a desvantagem de introduzir outros sensores além daqueles realmente necessários para a operação normal do sistema. É, em geral, rejeitada por razões econômicas.

2. Introdução dos chamados sensores virtuais (Sampath, 2001). Sensores virtuais são usados para aumentar a quantidade de informações fornecidas pelos sensores reais do sistema. Essas informações são derivadas por meios analíticos.

3. Uso de ações de controle para alterar a propriedade de diagnosticabilidade de um sistema (Sampath et al., 1998), restringindo-se o comportamento de um sistema não-diagnosticável através de ações de controle apropriadas para torná-lo diagnosticável. Essa abordagem, diferentemente das soluções 1 e 2 acima, que tratam a diagnose de falhas como passiva, combina observação e controle, sendo esse último problema formulado e resolvido utilizando a teoria de controle supervisório (Ramadge e Wonham, 1989).

Inspirado nos resultados de Lin e Wonham (1990) para controle supervisório descentralizado, Debouk et al. (2000) propuseram uma arquitetura descentralizada com coordenação, denominada codiagnose, que consiste de módulos locais capazes de observar a ocorrência de parte dos eventos observáveis do sistema. Esses módulos locais se comunicam com um coordenador, que é responsável pela diagnose das falhas que venham a ocorrer no sistema. A noção de diagnosticabilidade introduzida por Sampath et al. (1995) foi estendida em Debouk et al. (2000) levando ao conceito de diagnose descentralizada. Em um trabalho posterior, Contant et al. (2006) introduziram o conceito de diagnosticabilidade modular em sistemas que podem ser modelados pela composição paralela de autômatos, em que cada autômato representa um componente local (ou subsistema, ou módulo) do sistema global. Foi mostrado que se o sistema for modularmente diagnosticável, isto é, se cada subsistema for diagnosticável, então a diagnose de falha do sistema global será obtida utilizando-se somente os diagnosticadores locais (*i.e.*, os diagnosticadores projetados para cada um dos subsistemas). Mais recentemente, Basilio e Lafortune (2009) apresentaram o conceito de codiagnose robusta, segundo a qual, uma arquitetura descentralizada diagnosticável será robusta se e somente se continuar diagnosticável mesmo com a perda de comunicação entre um ou mais módulos e o coordenador. Condições necessárias e suficientes para codiagnose robusta foram apresentadas em Basilio e Lafortune (2009).

O problema da diagnose de falhas em SEDs estocásticos foi primeiramente considerado por Lunze e Schroder (2001), tendo sido resolvido a partir da formulação de um problema de observação de estados de autômatos estocásticos. Um autômato estocástico é um autômato ao qual é adicionada uma estrutura probabilística para estimar a probabilidade de ocorrência de eventos específicos. Seguindo a mesma linha de Sampath et al. (1995), Thorsley e Teneketzis (2005) apresentaram duas noções de diagnosticabilidade que incorporam a estrutura estocástica do autômato e determinam condições necessárias e suficientes para diagnosticabilidade. A diferença principal entre os trabalhos de Sampath et al. (1995)

e Thorsley e Teneketzis (2005) é que no primeiro o modelo do SED não pode distinguir entre sequências ou estados que têm elevada probabilidade de ocorrer e aqueles que têm reduzidas chances de ocorrer, enquanto que no último, tais comportamentos improváveis são descartados. Posteriormente, Liu et al. (2008) generalizaram os resultados de Thorsley e Teneketzis (2005) para diagnose descentralizada em SEDs estocásticos utilizando vários diagnosticadores locais baseados no modelo completo do sistema estocástico. De acordo com Liu et al. (2008), um SED estocástico será codiagnosticável se, após a ocorrência de uma falha, existir pelo menos um módulo local tal que a probabilidade desse módulo não diagnosticá-la seja suficientemente pequena.

Outra forma de descrever a incerteza de SEDs é através do autômato fuzzy (Lin e Ying, 2002; Belohlavek, 2002; Li et al., 2006). Para tanto, a definição de autômato de estados finitos é reformulada para permitir a incorporação dos conceitos de lógica fuzzy e conjuntos fuzzy. A diagnosticabilidade de SEDs foi generalizada para o caso de SED fuzzy por Kilic (2008) que propôs o conceito de grau de diagnosticabilidade fuzzy. De acordo com Kilic (2008), se o grau de diagnosticabilidade do sistema for igual a 1, então as ocorrências de todas as falhas do sistema poderão ser diagnosticadas. Caso o grau de diagnosticabilidade seja entre zero e 1, não é possível precisar o tipo da falha que tenha ocorrido. Se o grau de diagnosticabilidade for igual a zero, a linguagem não é diagnosticável.

A inclusão da informação de tempo em SEDs levou aos chamados autômatos temporizados. Nos modelos temporizados, as trajetórias não são especificadas somente em termos de sequências de estados ou eventos, mas devem incluir alguma informação do tempo de ocorrência. Alur e Dill (1994) propuseram o chamado autômato temporizado com guarda que emprega uma forma generalizada para o mecanismo de temporização no qual um conjunto de *clocks* com dinâmicas dirigidas pelo tempo são incorporados aos autômatos e cujas transições possuem pré-condições estabelecidas em termos dos valores dos relógios, denominadas guardas. Tripakis (2002) estendeu os resultados de Sampath et al. (1995) para SEDs modelados pelos autômatos temporizados com guarda de Alur e Dill (1994), sendo a diagnose de falhas baseada não somente nas sequências de eventos observáveis, mas também nos intervalos de tempo decorridos entre dois eventos sucessivos. Outras abordagens para o problema da diagnose de falhas em SEDs temporizados foram apresentadas por Chen e Provan (1997), Zad et al. (1999) e Zad et al. (2005) que consideraram a diagnose de falhas em modelos a tempo-discreto. Nestes trabalhos, o tempo decorrido entre eventos é modelado tendo como base um evento observável especial denominado "*clock tick*", sendo o problema da diagnose de falha resolvido utilizando-se técnicas de modelos não-temporizados. Além dessas abordagens, é importante mencionar o trabalho de Holloway e Chand (1996), que propôs uma nova técnica para diagnose de falha distribuída denominada monitoração de padrões, que utiliza conjuntos de temporizações e relações sequenciais para determinar quando estão previstas as ocorrências dos eventos e para precisar se um evento ocorreu ou não.

O conceito de diagnosticabilidade segura foi introduzido por Paoli e Lafortune (2005) segundo o qual, um sistema é dito ter a propriedade da diagnosticabilidade segura se ele for diagnosticável e a detecção de uma falha for realizada antes da execução de um dado conjunto de sequências proibidas após a ocorrência da falha. Recentemente, Qiu et al. (2009) estendeu a propriedade de diagnosticabilidade segura de Paoli e Lafortune (2005) para o caso descentralizado, denominando-a codiagnosticabilidade segura. Nesse caso, quando o sistema executar uma sequência que contenha o evento de falha, deve existir pelo menos um diagnosticador local que possa detectá-la com atraso limitado e antes que viole uma dada especificação de segurança.

Um outro problema frequente em todas as áreas da engenharia é a falha intermitente. Esse tipo de falha pode ocorrer devido a ligações elétricas ruins, componentes que emperram temporariamente, superaquecimento de circuitos integrados, ruído de medição em sensores, entre outros. As metodologias para diagnose de falhas mencionadas nos parágrafos anteriores não são apropriadas para o tratamento de falhas intermitentes pois supõem que, uma vez que a falha tenha ocorrido, o sistema não é capaz de se recuperar da falha; daí a terminologia falhas permanentes. Em um trabalho preliminar, Jiang et al. (2003) consideraram um problema correlato, *i.e.*, a diagnose de falhas repetidas em SEDs, estendendo os resultados de Jiang et al. (2001) e apresentando algumas definições de diagnosticabilidade de falhas repetidas. O problema da diagnose de falhas intermitentes foi, de fato, considerado pela primeira vez por Contant et al. (2004), que propuseram uma extensão do diagnosticador de Sampath et al. (1995) para incorporar as falhas intermitentes. As principais diferenças entre o diagnosticador proposto por Sampath et al. (1995) e o estendido são a introdução de eventos "*reset*" associados às falhas e de novos rótulos associados aos estados para indicar as seguintes situações: *(i)* se não houve ocorrência de falha; *(ii)* se a falha ocorreu e não houve recuperação da falha; *(iii)* se a falha ocorreu e houve a recuperação da falha. Além disso, condições necessárias e suficientes para a diagnose dessas falhas são apresentadas considerando novos tipos de ciclos indeterminados.

O diagnosticador proposto por Sampath et al. (1995), embora intuitivo e com aplicabilidade na diagnose em tempo real de SEDs, pode ter a sua utilização na verificação da diagnosticabilidade de SEDs comprometida tendo em vista que o espaço de estados do diagnosticador tem complexidade exponencial

em relação à cardinalidade do espaço de estados do autômato cuja linguagem gerada se deseja diagnosticar. Para contornar esse problema, Jiang et al. (2001) e Yoo e Lafortune (2002) propuseram um novo método para verificar a diagnosticabilidade de SEDs baseado na construção de autômatos não determinísticos denominados verificadores, cujo número de estados cresce de forma polinomial. Mais recentemente, Qiu e Kumar (2006) e Wang et al. (2007) estenderam esses verificadores para a codiagnose, levando aos chamados verificadores descentralizados.

Um outro formalismo para a modelagem de SEDs são as redes de Petri (Peterson, 1981; Murata, 1989; David e Alla, 2005). Recentemente, o problema da diagnose de falhas em sistemas modelados por redes de Petri tem recebido grande atenção (Ushio et al., 1998; Chung et al., 2003; Benveniste et al., 2003; Ramirez-Trevino et al., 2004; Giua e Seatzu, 2005; Genc e Lafortune, 2007; Lefebvre e Delherm, 2007; Manyari-Rivera et al., 2007; Ru e Hadjicostis, 2009; Basile et al., 2009; Dotoli et al., 2009). Contudo, um dos principais problemas ao se considerar modelos em redes de Petri no contexto de diagnose de falhas é que, conforme mostrado por Gaubert e Giua (1999), uma rede de Petri não determinística não pode ser convertida em uma determinística equivalente. Por essa razão, os diagnosticadores obtidos para SEDs modelados por redes de Petri são ainda autômatos. Mais recentemente, Cabasino et al. (2009) apresenta condições necessárias e suficientes para a diagnosticabilidade de redes de Petri rotuladas ilimitadas e propõe um teste para verificar a sua diagnosticabilidade baseado na análise do grafo de cobertura de uma rede de Petri obtida a partir da rede de Petri do sistema inicial.

Este artigo está estruturado da seguinte forma. A seção 2 apresenta uma breve revisão da teoria de SEDs. Na seção 3 é formulado o problema da diagnose de falhas e na seção 4 são apresentadas as condições (necessárias e suficientes) para a diagnose centralizada, e em seguida, é considerado o problema de se diagnosticar uma falha utilizando como conjunto de eventos um subconjunto do conjunto de eventos observáveis original. Ainda na seção 4, é abordado o problema da diagnose descentralizada com coordenação. Na seção 5 consideram-se as condições necessárias e suficientes para a diagnose centralizada e descentralizada utilizando verificadores. Comentários finais e sugestões de tópicos para pesquisa futura são apresentados na seção 6.

2 SISTEMAS A EVENTOS DISCRETOS

Sistemas a eventos discretos (SEDs) são sistemas dinâmicos de estados discretos cuja transição de estados se dá através da ocorrência, em geral assíncrona, de eventos. O fato do estado do sistema ser discreto implica que ele pode assumir valores simbólicos, como por exemplo $\{$ligado, desligado$\}$, $\{$verde, amarelo, vermelho$\}$, ou valores discretos tais como valores numéricos pertencentes aos conjuntos $\mathbb{N}$ ou $\mathbb{Z}$, ou ser formado por um subconjunto enumerável de elementos de $\mathbb{R}$. Eventos podem estar associados a ações específicas (por exemplo, alguém aperta um botão, um avião levanta vôo etc) ou ser o resultado de diversas condições que são satisfeitas (uma peça atinge um determinado ponto de uma linha de produção, o líquido dentro de um tanque atinge uma determinada altura etc). Embora seja possível modelar qualquer sistema físico como um SED de acordo com o grau de abstração considerado, determinados sistemas são naturalmente discretos e com evolução determinada pela ocorrência de eventos.

Assim como na modelagem de sistemas dinâmicos de variáveis contínuas (SDVC), um modelo para um SED deve ser capaz de reproduzir, dentro de limites de tolerância pré-estabelecidos, o comportamento do sistema. Enquanto nos SDVCs as trajetórias dos estados são descritas em função do tempo, nos SEDs elas são função de uma sequência de eventos. Todas as sequências de eventos possíveis de serem geradas por um SED caracterizam a linguagem desse SED, sendo esta definida sobre o conjunto de eventos (alfabeto) do sistema. Assim, ao se considerar a evolução dos estados de um SED, a maior preocupação é com a sequência de estados visitados e com os eventos que causaram as correspondentes transições de estado, isto é, o modelo de um SED é composto basicamente de dois elementos, estados e eventos, conforme será ilustrado no exemplo a seguir.

Exemplo 1 Considere uma célula de manufatura formada por duas máquinas (M_1 e M_2) e um robô que transporta as peças de M_1 para M_2. A máquina M_1 recebe peças brutas e quando as peças estão prontas são recolhidas pelo robô. Caso o robô esteja ocupado, a máquina M_1 retém a peça até que o robô esteja completamente livre. Caso uma outra peça chegue enquanto a máquina M_1 estiver processando/retendo alguma peça, a máquina M_1 rejeita a peça recebida. Quando o robô recebe uma peça de M_1, inicia o transporte desta até a máquina M_2. No momento em que chegar a M_2, o robô somente entregará a peça à máquina M_2 se esta estiver livre; caso contrário reterá a peça até M_2 ficar disponível. Após entregar a peça a M_2, o robô retorna à máquina M_1. A máquina M_2 recebe a peça do robô e a processa.

A tabela 1 descreve os estados e os eventos das máquinas M_1 e M_2 e do robô. Note que os eventos e_1 (entrega de peça ao robô) e a_2 (entrega/chegada de peça em M_2) pertencem a dois subsistemas: máquina M_1 e robô, e robô e máquina M_2, respectivamente. É importante notar que, para que o evento e_1 ocorra, a máquina M_1 deverá estar no estado H_1 e o robô no estado I; para que o evento a_2 ocorra, o robô deverá estar no estado H e a máquina M_2 deverá estar no estado I_2. Para os demais estados do sistema, isto é, aqueles que estão presentes em somente um dos subsistemas, a

Tabela 1: Os estados e os eventos das máquinas M_1, M_2 e do robô.

Elemento	Estados	Eventos
Máquina M_1	M_1 disponível:I_1	Chegada de peça a M_1: a_1
	M_1 processando: P_1	Fim de processamento: t_1
	M_1 retendo peça pronta: H_1	Entrega de peça ao robô: e_1
	$X_1 = \{I_1, P_1, H_1\}$	$E_1 = \{a_1, t_1, e_1\}$
Robô	Robô disponível: I,	Entrega de peça ao robô: e_1
	Transportando M_1-M_2: T_{12}	Chegada a M_2: c_2
	Esperando em M_2: H	Entrega/chegada de peça a M_2: a_2
	Retornando para M_1: R	Chegada a M_1: r_1
	$X_r = \{I, T_{12}, H, R\}$	$E_r = \{e_1, c_2, a_2, r_1\}$
Máquina M_2	M_2 disponível: I_2	Entrega/chegada de peça em M_2: a_2
	M_2 processando: P_2	Fim de processamento: t_2
	$X_2 = \{I_2, P_2\}$	$E_2 = \{a_2, t_2\}$

ocorrência não dependerá do estado em que os demais subsistemas estiverem, sendo determinada somente pelo estado atual do subsistema; por exemplo, a ocorrência do evento t_1 (fim de processamento da peça em M_1) dependerá apenas da máquina M_1 estar no estado P_1, independentemente de quais estados estiverem o robô e a máquina M_2. □

2.1 Linguagem

Uma linguagem definida sobre um conjunto de eventos Σ é um conjunto de sequências (também referidas como cadeias) de comprimentos finitos formadas com os eventos de Σ. Por exemplo, seja $\Sigma = \{a, b, c, g\}$ um conjunto de eventos. Os seguintes conjuntos são exemplos de linguagens definidas sobre Σ: $L_1 = \{aa, bb, cg\}$ e $L_2 = \{$todas as possíveis sequências de eventos de Σ terminadas com o evento a$\}$. Note que, enquanto L_1 tem apenas três elementos, L_2 é infinita (porém enumerável).

Note que uma linguagem é um subconjunto do conjunto de todas as possíveis sequências de comprimentos finitos formadas com os elementos de Σ. A esse conjunto dá-se o nome de fecho de Kleene de Σ, que é denotado por Σ^*. Formalmente, o fecho de Kleene é definido como:

$$\Sigma^* = \{\epsilon\} \cup \Sigma \cup \Sigma\Sigma \cup \Sigma\Sigma\Sigma \cup \ldots,$$

em que ϵ denota a sequência vazia e $\Sigma_a\Sigma_b$ denota a operação de concatenação entre os conjuntos Σ_a e Σ_b, definida da seguinte forma:

$$\Sigma_a\Sigma_b = \{\sigma = \sigma_a\sigma_b : \sigma_a \in \Sigma_a \text{ e } \sigma_b \in \Sigma_b\}.$$

É importante salientar que, como linguagens são conjuntos, as operações usuais de união, interseção, complemento e diferença sejam também válidas para linguagens. Além dessas operações usuais, três importantes operações podem ser definidas para linguagens: a concatenação, o fecho de prefixo e a projeção.

A concatenação entre duas linguagens L_1 e L_2 é uma linguagem $L = L_1L_2$ formada concatenando-se todas as sequências de L_1 com todas as sequências de L_2, isto é,

$$L = L_1L_2 = \{s = s_1s_2 : s_1 \in L_1 \text{ e } s_2 \in L_2\}.$$

O fecho de prefixo de uma linguagem L (denotado por $\bar{L}$) é o conjunto formado por todos os prefixos dos elementos de L. Uma sequência u será um prefixo de s se existir uma sequência v tal que $s = uv$. Assim, se, por exemplo, $L = \{a, ab, ba\}$, então $\bar{L} = \{\epsilon, a, ab, b, ba\}$. Uma linguagem L tal que $L = \bar{L}$ é dita ser prefixo-fechada.

A projeção P_o é definida como (Ramadge e Wonham, 1989):

$$\begin{array}{rcccl} P_o & : & \Sigma^* & \rightarrow & \Sigma_o^*, \text{ sendo } \Sigma_o \subset \Sigma \\ & & s & \mapsto & P_o(s), \end{array} \quad (1)$$

com as seguintes propriedades:

$$\begin{array}{rcl} P_o(\epsilon) & = & \epsilon, \\ P_o(\sigma) & = & \begin{cases} \sigma, \text{ se } \sigma \in \Sigma_o, \\ \epsilon, \text{ se } \sigma \in \Sigma \backslash \Sigma_o, \end{cases} \\ P_o(s\sigma) & = & P_o(s)P_o(\sigma), s \in \Sigma^*, \sigma \in \Sigma. \end{array} \quad (2)$$

O operador projeção pode ser estendido para linguagens de forma natural aplicando a projeção (2) a todas as sequências dessa linguagem. Assim, se $L \subset \Sigma^*$ então

$$P_o(L) = \{t \in \Sigma_o^* : (\exists s \in L)[P_o(s) = t]\}. \quad (3)$$

De acordo com a definição acima, a projeção consiste em apagar das sequências de L os eventos que não pertencem a Σ_o. Do ponto de vista prático, essa operação representa a linguagem observada de um sistema, isto é, as sequências formada pelos eventos cujas ocorrências são, de alguma forma,

do conhecimento do observador. Isso pode gerar ambiguidade, isto é, duas sequências distintas da linguagem podem ter a mesma projeção, o que pode levar a dificuldades tanto no controle quanto na diagnose de falhas de SEDs. A projeção inversa P_o^{-1} é definida da seguinte forma:

$$\begin{array}{rcccl} P_o^{-1} & : & \Sigma_o^* & \rightarrow & 2^{\Sigma^*} \\ & & s & \mapsto & P_o^{-1}(s)=\{t \in \Sigma^* : P_o(t) = s\}. \end{array} \quad (4)$$

A projeção inversa de uma linguagem M restrita à linguagem L é definida como:

$$P_{o_L}^{-1}(M) = \{s \in L : (\exists y \in M)[P(s) = y]\}. \quad (5)$$

Para ilustrar o conceito de projeção, considere os seguintes conjuntos de eventos: $\Sigma_o = \{a, b\}$ e $\Sigma_{uo} = \{c\}$. Seja $P_o : (\Sigma_o \cup \Sigma_{uo})^* \rightarrow \Sigma_o^*$. Então $P_o(\{cbac\}) = \{ba\}$ e $P_o^{-1}(\{ba\}) = \{c\}^*\{b\}\{c\}^*\{a\}\{c\}^*$. Pode-se, portanto, verificar que $P_o[P_o^{-1}(L)] = L$, porém $P_o^{-1}[P_o(L)] \supseteq L$.

2.2 Autômatos

Uma das maneiras de se modelar SEDs é através de autômatos (também chamados máquinas de estados finitos ou geradores). Formalmente, um autômato (determinístico) é uma sêxtupla

$$G = (X, \Sigma, f, \Gamma, x_0, X_m), \quad (6)$$

em que X denota o espaço de estados, Σ o conjunto de eventos, $f : X \times \Sigma \rightarrow X$ a função de transição de estados (possivelmente parcial[1]), Γ a função dos eventos ativos, x_0 o estado inicial do sistema, e $X_m \subseteq X$ o conjunto dos estados marcados.

Autômatos são representados graficamente através de diagramas de transição de estados. Nesses diagramas os estados são representados por circunferências e são conectados entre si por arcos identificados (rotulados) com símbolos, que representam os eventos que determinam as transições entre os estados ligados pelo arco. Os estados marcados são identificados por duas circunferências concêntricas e estão, em geral, relacionados ao cumprimento de uma tarefa a ser realizada pelo sistema modelado pelo autômato. O estado inicial é indicado por uma seta apontada a ele, não oriunda de qualquer outro estado.

Exemplo 2 A figura 1 mostra um diagrama de transição de um autômato $G = (X, \Sigma, f, \Gamma, x_0, X_m)$ em que $X = \{0, 1, 2, 3\}$, $\Sigma = \{a, b\}$, $x_0 = 0$ e $X_m = \{2, 3\}$. A evolução dinâmica do autômato representado na figura 1 se dá da seguinte forma. Quando ligado, o sistema se encontra no

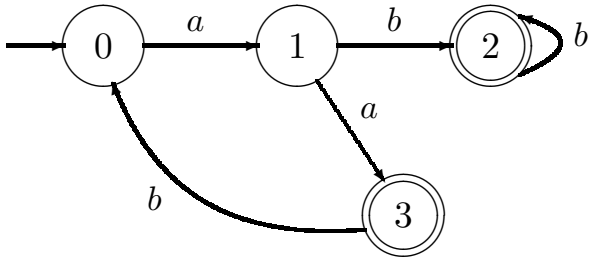


Figura 1: Autômato simples.

estado $x_0 = 0$. Nessa situação, somente o evento a pode ocorrer e, portanto, $\Gamma(0) = \{a\}$. A ocorrência do evento a muda o estado do autômato de 0 para 1; formalmente tem-se que $f(0, a) = 1$. No estado $x = 1$, há duas possibilidades de ocorrência de eventos: a ou b, o que é caracterizado pela função dos eventos ativos, isto é, $\Gamma(1) = \{a, b\}$. Se o evento b ocorrer, o estado do sistema mudará para $x = 2$ e se a ocorrer, ter-se-á a transição para o estado $x = 3$. Note que existe uma transição definida por um autolaço no estado $x = 2$, significando que o autômato permanecerá no estado $x = 2$, mesmo com a ocorrência do evento b. □

Um autômato é um dispositivo capaz de representar uma linguagem de acordo com regras bem definidas. São dois os tipos de linguagens que podem ser associadas ao comportamento de um autômato: a linguagem gerada e a linguagem marcada. A linguagem gerada (denotada por L) representa todos os caminhos que podem ser seguidos no diagrama de transição de estados, começando pelo estado inicial. A linguagem marcada (denotada por L_m) é um subconjunto da linguagem gerada e consiste de todos os caminhos que terminam em um estado marcado no diagrama de transição de estados. Para se definir L e L_m formalmente, deve-se inicialmente estender o domínio de f de $X \times \Sigma$ para $X \times \Sigma^*$ da seguinte forma recursiva:

$$\begin{array}{rcl} f(x, \epsilon) & := & x, \\ f(x, se) & := & f[f(x, s), e] \text{ para } s \in \Sigma^* \text{ e } e \in \Sigma. \end{array}$$

Feito isso, pode-se definir L e L_m da seguinte forma:

$$L = \{s \in \Sigma^* : (\exists x \in X)[f(x_0, s) = x]\}$$

e

$$L_m = \{s \in L : f(x_0, s) \in X_m\}.$$

Observe que a linguagem L é prefixo-fechada.

Para o autômato da figura 1, tem-se que $L = \{\epsilon\} \cup \{a\}\overline{[\{aba\}^*]}\{b\}^*$ e $L_m = \{a\}\{aba\}^*\{bb^*, a\}$.

Suponha que $G_1 = (X_1, \Sigma_1, f_1, \Gamma_1, x_{0_1}, X_{m_1})$ e $G_2 = (X_2, \Sigma_2, f_2, \Gamma_2, x_{0_2}, X_{m_2})$ sejam dois autômatos distintos e

[1] A função de transição f foi historicamente definida como total, isto é, definida para todo o domínio $X \times \Sigma$. Em SEDs, é usual considerá-la parcialmente definida sobre o seu domínio. O leitor interessado pode ter maiores detalhes em (Cassandras e Lafortune, 2008, pp. 60).

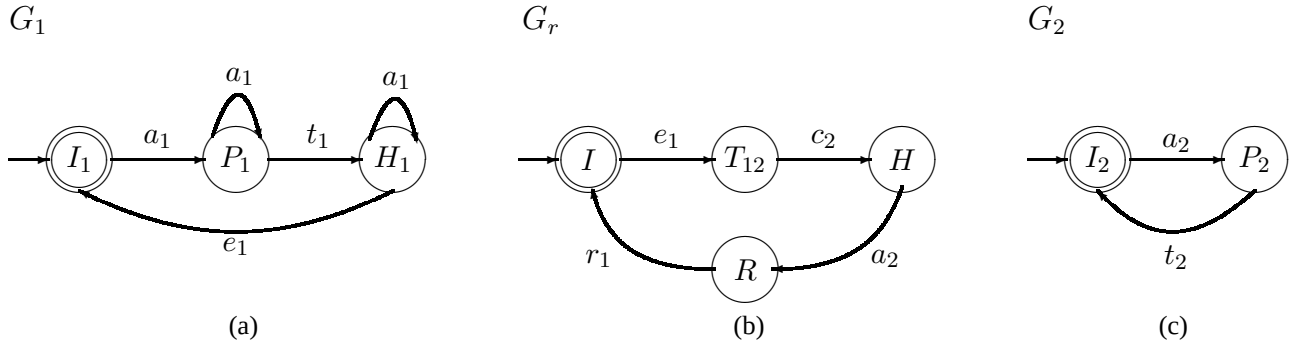


Figura 2: Máquina M_1 (a); Robô (b); Máquina M_2 (c).

que se deseje obter um autômato que modele o comportamento síncrono de G_1 e G_2, isto é: *(i)* um evento comum a G_1 e G_2 somente poderá ocorrer quando ambos, G_1 e G_2, estiverem em estados cujos conjuntos dos eventos ativos tenham esse evento como elemento; *(ii)* eventos privados, isto é, pertencentes a $\Sigma_1 \setminus \Sigma_2$ ou a $\Sigma_2 \setminus \Sigma_1$ podem ser executados sempre que possível. Tal autômato existe e pode ser obtido através da chamada composição paralela de G_1 e G_2, denotada por $G_1 \| G_2$, e definida da seguinte forma:

$$\begin{aligned} G_1\|G_2 &= \text{Ac}(X_1 \times X_2, \Sigma_1 \cup \Sigma_2, f_{1\|2}, \Gamma_{1\|2}, (x_{0_1}, x_{0_2}), \\ &\quad X_{m_1} \times X_{m_2}), \end{aligned}$$

sendo que $\times$ denota o produto cartesiano e Ac denota a parte acessível de $G_1 \| G_2$, a qual é formada pelos estados que podem ser alcançados a partir do estado inicial por uma sequência em $(\Sigma_1 \cup \Sigma_2)^*$. A função de transição de estados de $G_1 \| G_2$ é definida como:

$$f_{1\|2}((x_1,x_2),\sigma) = \begin{cases} (f_1(x_1,\sigma), f_2(x_2,\sigma)), \text{ se } \sigma \in \\ \Gamma_1(x_1) \cap \Gamma_2(x_2), \\ (f_1(x_1,\sigma), x_2), \text{ se } \sigma \in \Gamma_1(x_1) \setminus \Sigma_2, \\ (x_1, f_2(x_2,\sigma)), \text{ se } \sigma \in \Gamma_2(x_2) \setminus \Sigma_1, \\ \text{não definido, caso contrário.} \end{cases}$$

Supondo que $L_1 = L(G_1)$ e $L_2 = L(G_2)$, pode-se mostrar que as linguagens gerada e marcada por $G_1 \| G_2$ são dadas por:

$$\begin{aligned} L(G_1\|G_2) &= P_1^{-1}(L_1) \cap P_2^{-1}(L_2), \\ L_m(G_1\|G_2) &= P_1^{-1}(L_{m_1}) \cap P_2^{-1}(L_{m_2}), \end{aligned}$$

sendo $P_i : (\Sigma_1 \cup \Sigma_2)^\star \to \Sigma_i^\star,\ i = 1, 2$.

Outra composição importante entre autômatos é a composição produto. Essa composição permite somente transições com eventos comuns e é definida da seguinte forma:

$$\begin{aligned} G_1 \times G_2 &= \text{Ac}(X_1 \times X_2, \Sigma_1 \cup \Sigma_2, f_{1\times 2}, \Gamma_{1\times 2}, \\ &\quad (x_{0_1}, x_{0_2}), X_{m_1} \times X_{m_2}), \end{aligned}$$

sendo

$$f_{1\times 2}((x_1,x_2),\sigma) = \begin{cases} (f_1(x_1,\sigma), f_2(x_2,\sigma)), \text{ se } \sigma \in \\ \Gamma_1(x_1) \cap \Gamma_2(x_2), \\ \text{não definido, caso contrário.} \end{cases}$$

Se $\Sigma_1 = \Sigma_2$, então a composição paralela reduzir-se-á ao produto, já que todos os eventos são comuns.

Pode-se verificar que as linguagens gerada e marcada de $G_1 \times G_2$ são dadas por:

$$\begin{aligned} L(G_1 \times G_2) &= L_1 \cap L_2, \\ L_m(G_1 \times G_2) &= L_{m_1} \cap L_{m_2}. \end{aligned}$$

Exemplo 3 Para ilustrar o uso da composição paralela, considere a célula de manufatura descrita no exemplo 1. Com o auxílio da tabela 1, é possível construir os autômatos das máquinas M_1 e M_2, e do robô, que serão denotados por G_1, G_2 e G_r, respectivamente. Os correspondentes diagramas de transição estão representados na figura 2.

O modelo do sistema que considera o comportamento síncrono das máquinas M_1, M_2 e do robô será obtido pela composição paralela de G_1, G_2 e G_r. A figura 3 mostra a composição $G_r || G_2$. Note na figura 2 que o evento a_2 é um evento comum dos autômatos G_r e G_2. Dessa forma, esse evento somente poderá ocorrer quando G_r e G_2 estiverem em estados cujos conjuntos dos eventos ativos tenham, ambos, o evento a_2 como elemento. Observe que a_2 não pertence ao conjunto dos eventos ativos do estado I (de G_r) e, embora pertença ao conjunto dos eventos ativos de I_2 (de G_2), não poderá ocorrer quando $G_r || G_2$ estiver no estado inicial (I, I_2), conforme mostrado na figura 3. Esta restrição pode ser entendida mais claramente do ponto de vista prático, pois o robô não poderá entregar uma peça à máquina M_2 quando estiver parado à espera de peças na máquina M_1. De fato, conforme mencionado no exemplo 1, o evento a_2 somente ocorrerá quando o robô estiver no estado H e a máquina M_2 estiver no estado I_2, como pode ser visto na figura 3. □

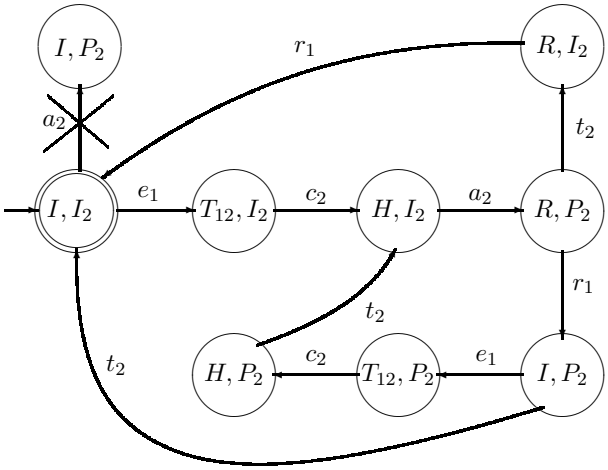


Figura 3: Composição síncrona de G_r e G_2.

Suponha que Σ seja particionado como $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$, isto é, $\Sigma = \Sigma_o \cup \Sigma_{uo}$, $\Sigma_o \cap \Sigma_{uo} = \emptyset$ e $\Sigma_{uo} \neq \emptyset$, sendo Σ_o um conjunto de eventos observáveis e Σ_{uo} um conjunto de eventos não-observáveis. Um evento é observável quando sua ocorrência puder ser registrada através de sensores ou quando estiver associado a comandos. Os eventos não-observáveis, por sua vez, designam aqueles eventos do sistema cuja ocorrência não pode ser observada por sensores (incluindo os eventos de falhas) ou, embora haja sensores para registrá-los, esses eventos não podem ser vistos em função da natureza distribuída do sistema. Quando $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$ tem-se o chamado autômato determinístico com eventos não-observáveis.

O comportamento dinâmico de um autômato determinístico com eventos não-observáveis pode ser descrito por um autômato determinístico, denominado observador, cujo conjunto de eventos é formado pelos eventos observáveis. Os estados do observador são todos os estados em que um autômato determinístico com eventos não-observáveis pode estar após a observação de uma sequência de eventos observáveis. O observador para G, denotado por $\mathit{Obs}(G)$, é definido da seguinte forma:

$$\mathit{Obs}(G) = (X_{obs}, \Sigma_o, f_{obs}, \Gamma_{obs}, x_{0_{obs}}, X_{m_{obs}}), \qquad (7)$$

sendo $X_{obs} \in 2^X$ e $X_{m_{obs}} = \{B \in X_{obs} : B \cap X_m \neq \emptyset\}$. Para a definição de $x_{0_{obs}}$, Γ_{obs} e f_{obs}, é necessário introduzir o conceito de alcance não-observável de um estado $x \in X$ (denotado por $\mathit{UR}(x)$):

$$\mathit{UR}(x) = \{y \in X : (\exists t \in \Sigma_{uo}^*)[f(x,t) = y]\}. \qquad (8a)$$

De forma análoga, o alcance não-observável de um conjunto $B \in 2^X$ é definido como

$$\mathit{UR}(B) = \bigcup_{x \in B} \mathit{UR}(x). \qquad (8b)$$

Usando (8a) e (8b), pode-se definir $x_{0_{obs}}$, Γ_{obs}, f_{obs} e X_{obs} no algoritmo a seguir.

Algoritmo 1 (Construção de observadores)

Passo 1: Defina $x_{0_{obs}} = \mathit{UR}(x_0)$ e faça $X_{obs} = \{x_{0_{obs}}\}$ e $\tilde{X}_{obs} = X_{obs}$.

Passo 2: $\hat{X}_{obs} = \tilde{X}_{obs}$ e $\tilde{X}_{obs} = \emptyset$.

Passo 3: Para cada $B \in \hat{X}_{obs}$,

- $\Gamma_{obs}(B) = \left(\bigcup_{x \in B} \Gamma(x)\right) \cap \Sigma_o$;
- Para cada $e \in \Gamma_{obs}(B)$,

  $f_{obs}(B,e) = \mathit{UR}(\{x \in X : (\exists y \in B)[x = f(y,e)]\})$;

- $\tilde{X}_{obs} \leftarrow \tilde{X}_{obs} \cup f_{obs}(B,e)$.

Passo 4: $X_{obs} \leftarrow X_{obs} \cup \tilde{X}_{obs}$.

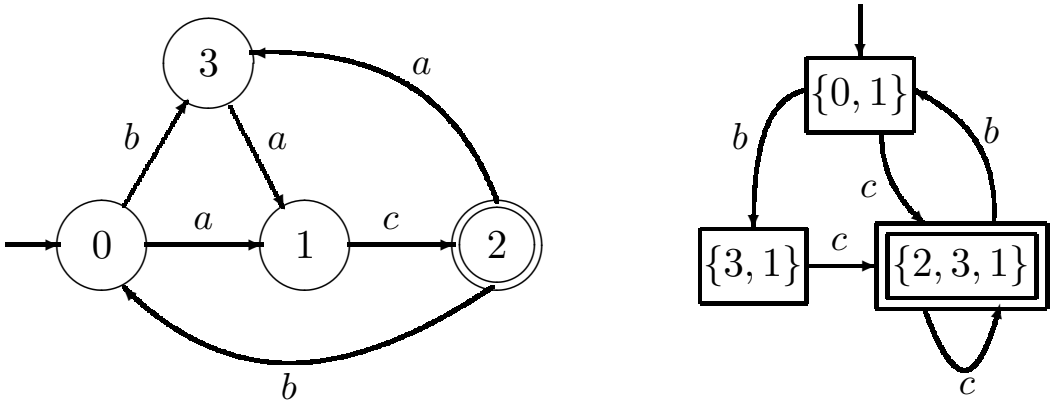
Passo 5: Repita os passos 2 a 4 até que toda a parte acessível de $\mathit{Obs}(G)$ tenha sido construída.

Passo 6: $X_{m_{obs}} = \{B \in X_{obs} : B \cap X_m \neq \emptyset\}$. □

A idéia do algoritmo 1 é calcular o alcance não-observável para cada estado de G alcançado por um evento observável. Assim, no passo 1 calcula-se o alcance não-observável do estado inicial x_0 formando o estado inicial do observador. No passo 3 calculam-se os conjuntos dos eventos ativos dos estados do observador obtidos no passo anterior ou na iteração anterior (o primeiro se refere ao alcance observável do estado inicial e o último aos estados de G alcançados por meio de eventos observáveis juntamente com os respectivos alcances não-observáveis). Além disso são calculados os próximos estados do observador, que correspondem aos alcances não-observáveis dos estados de G alcançados a partir do estado atual do observador por meio de eventos observáveis. Essa sequência é repetida até que todos os estados acessíveis do observador tenham sido encontrados.

A partir da construção do observador, pode-se concluir que a linguagem gerada por $\mathit{Obs}(G)$ é a projeção da linguagem de G sobre o conjunto de eventos observáveis, isto é, $L(\mathit{Obs}(G)) = P_o[L(G)]$.

Exemplo 4 Para ilustrar a construção de observadores, considere o autômato G da figura 4(a) e suponha que o evento a seja não-observável, isto é, tem-se que $\Sigma = \{a, b, c\}$, $\Sigma_o = \{b, c\}$ e $\Sigma_{uo} = \{a\}$. Assim, quando esse autômato inicia sua operação, não é possível precisar se ele ainda está no estado inicial $x_0 = 0$ ou se mudou para o estado $x = 1$, uma vez que a ocorrência do evento a não pode ser registrada; daí o estado inicial do observador mostrado na figura



(a) (b)

Figura 4: Um autômato determinístico com eventos não-observáveis (a) e o seu correspondente observador (b).

4(b) ser $\{0, 1\}$. Caso o próximo evento observável a ocorrer seja o evento b, pode-se, então, afirmar que o autômato estará, inicialmente no estado $x = 3$, porém, como o evento a é não-observável, o autômato pode mudar para o estado $x = 1$, sem que essa mudança seja percebida; por conseguinte, a transição rotulada pelo evento b no observador leva do estado inicial para o estado $\{3, 1\}$. Há ainda transições rotuladas pelo evento c que levam ao estado $\{2, 3, 1\}$ do observador, partindo tanto do estado inicial quanto do estado $\{3, 1\}$, uma vez que o evento c é o único evento observável pertencente aos conjuntos dos eventos ativos dos estados de G que compõem esses dois estados, e os estados $x = 3$ e $x = 1$ pertencem ao alcance não-observável do estado $x = 2$. Finalmente, quando a ocorrência do evento c for registrada, o observador irá permanecer no estado $\{2, 3, 1\}$. Contudo, se o evento b ocorrer, o observador retornará ao seu estado inicial. □

3 O PROBLEMA DA DIAGNOSE DE FALHAS DE SISTEMAS A EVENTOS DISCRETOS

Conforme visto na seção anterior, ao se incorporar eventos não-observáveis no modelo G, torna-se possível considerar tanto o comportamento normal do sistema, descrito pelos eventos não-observáveis que não sejam associados a falhas no sistema, como também as falhas que possam ocorrer. Para esse fim, seja $\Sigma_f \subseteq \Sigma_{uo}$ o conjunto dos eventos associados às falhas do sistema.

Em geral, o conjunto Σ_f é particionado em diferentes subconjuntos Σ_{f_i}, $i = 1, 2, \ldots, m$, não necessariamente unitários, em que cada conjunto Σ_{f_i} é formado por eventos que modelam falhas que são, de alguma forma, correlacionadas; o leitor pode obter maiores detalhes sobre esse tópico nos trabalhos de Sampath et al. (1995), Sampath et al. (1996) e Lafortune et al. (2001). Suponha que $\Pi_f = \{\Sigma_{f_1}, \Sigma_{f_2}, \ldots, \Sigma_{f_m}\}$ denote essa partição. Assim, cada vez que for dito que uma falha f_i ocorreu, deve ser entendido que algum evento do conjunto Σ_{f_i} ocorreu.

Nos trabalhos envolvendo diagnose de falhas em SEDs, as seguintes hipóteses são feitas:

A1. A linguagem gerada por G é viva, *i.e.*, $\Gamma(x_i) \neq \emptyset$ para todo $x_i \in X$;

A2. O autômato G não possui nenhum ciclo formado somente por eventos não-observáveis, *i.e.*, $\forall ust \in L$, $s \in \Sigma_{uo}^*$, $\exists n_0 \in \mathbb{N}$ tal que $\|s\| \leq n_0$, em que $\|s\|$ denota o comprimento da sequência s.

A3. Existe somente um único tipo de falha, *i.e.*, $\Pi_f = \{\Sigma_f\}$, em que $\Sigma_f = \{\sigma_f\}$.

A hipótese A1 é feita tendo em vista que considera-se o sistema em continua operação. A hipótese A2 é necessária para evitar que a ocorrência do evento associado à falha possa vir a não ser detectada caso o sistema fique preso em um ciclo de estados ligados por eventos não observáveis após a sua ocorrência. Essa hipótese será removida ao longo desse tutorial, sendo tais ciclos referidos como escondidos (Basilio e Lafortune, 2009). A hipótese A3 é feita por simplicidade, uma vez que para cada conjunto de eventos de falhas de um mesmo tipo é necessário criar um rótulo diferente; os fundamentos relacionados à análise da diagnosticabilidade são, contudo, os mesmos daqueles empregados para um único tipo de falha.

No estudo de diagnose de falhas de SEDs, as seguintes notações serão utilizadas.

- s_f: último evento de uma sequência s.

- $\Psi(\Sigma_f) = \{s \in L : s_f \in \Sigma_f\}$: conjunto de todas as sequências de L que terminam com o evento σ_f.
- $L/s = \{t \in \Sigma^* : st \in L\}$: continuação da linguagem de L após uma sequência s.

Suponha que $\overline{s}$ denote o fecho de prefixo de s. Com um ligeiro abuso de notação a relação de pertinência $\Sigma_f \in s$ será usada para denotar que $\overline{s} \cap \Psi(\Sigma_f) \neq \emptyset$.

Definição 1 A sequência $s \in L$ é uma sequência que contém uma falha se $\Sigma_f \in s$. □

A definição acima permite, então, apresentar formalmente a definição de diagnosticabilidade de uma linguagem (Sampath et al., 1995).

Definição 2 Seja L uma linguagem gerada por um autômato G e suponha que L seja viva e prefixo-fechada. Então L é diagnosticavél em relação à projeção P_o e $\Sigma_f = \{\sigma_f\}$ se a seguinte condição for verificada:

$$(\exists n \in \mathbb{N})(\forall s \in \Psi(\Sigma_f))(\forall t \in L/s)(\|t\| \geq n \Rightarrow D),$$

sendo a condição de diagnose D expressa por

$$(\forall \omega \in P_{o_L}^{-1}(P_o(st)))(\Sigma_f \in \omega).$$

□

Informalmente, diz-se que a linguagem gerada por um autômato será diagnosticável em relação a um conjunto de eventos observáveis Σ_o e um conjunto de eventos de falhas $\Sigma_f = \{\sigma_f\}$ se a ocorrência de σ_f puder ser detectada após um número finito de transições depois da ocorrência de σ_f usando somente sequências de eventos observáveis.

4 DIAGNOSTICADOR

Com o objetivo de se realizar a diagnose de falhas a partir da observação do comportamento do sistema em tempo real e para verificar se a linguagem gerada por um autômato G é diagnosticável, pode-se utilizar um autômato determinístico denominado diagnosticador. Além disso, dependendo de como as informações sobre a evolução dinâmica do sistema é disponibilizada, isto é, centralizada em um único sistema de aquisição ou distribuída como no caso de redes de comunicação, sistemas de manufaturas, e sistemas elétricos de potência, podem-se definir duas estruturas para a diagnose de falhas em SEDs:

1. Diagnosticador centralizado: utiliza um único diagnosticador que tem acesso a todos os eventos observáveis do sistema;

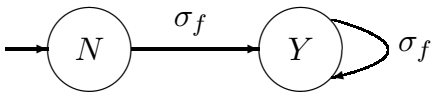


Figura 5: Autômato rotulador.

2. Codiagnosticadores (diagnosticadores descentralizados): a leitura dos sensores não é centralizada, mas sim distribuída em diferentes módulos. Cada módulo observa o comportamento de parte do sistema utilizando um subconjunto do conjunto de eventos observáveis do sistema.

4.1 Diagnose centralizada

O diagnosticador centralizado denotado por G_d é um autômato cujo conjunto de eventos é igual ao conjunto dos eventos observáveis de G e cujos estados são formados adicionando-se os rótulos Y e N aos estados de G para indicar se o evento σ_f ocorreu ou não. Formalmente, G_d é definido como

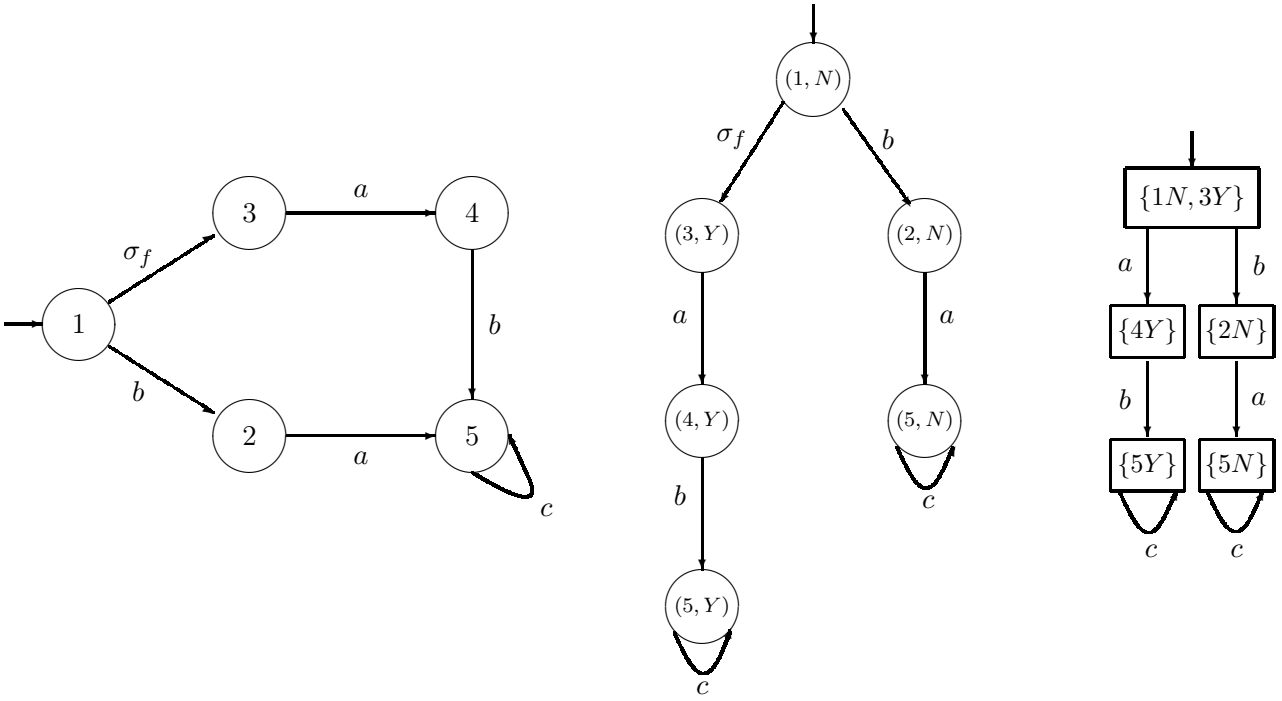
$$G_d = (X_d, \Sigma_o, f_d, \Gamma_d, x_{0_d}), \tag{9}$$

e pode ser construído em dois passos: *(i)* obtenha a composição paralela $G\|A_l$, sendo A_l o autômato rotulador de dois estados mostrados na figura 5; *(ii)* calcule *Obs*$(G\|A_l)$.

É importante observar que o autômato obtido após a composição paralela realizada no passo *(i)*, gera a mesma linguagem que G. Além disso, os estados de $G\|A_l$ são da forma (x, Y) ou (x, N), dependendo se σ_f está ou não na sequência que leva x_0 até x; consequentemente $x_d \in 2^{X \times \{N,Y\}}$.

Exemplo 5 Para ilustrar a construção de diagnosticadores, considere o autômato cujo diagrama de transição de estados está representado na figura 6(a). As figuras 6(b) e (c) mostram, respectivamente, a composição paralela $G\|A_l$ e o diagnosticador $G_d = Obs(G\|A_l)$. Note que o estado 5 de G se divide nos estados $(5, Y)$ e $(5, N)$ de $G\|A_l$ devido à existência de duas sequências distintas ($s_1 = \sigma_f ab$ e $s_2 = ba$) que levam $x_0 = 1$ a $x = 5$, das quais somente a sequência s_1 contém o evento de falha σ_f. Para simplificar a notação, é usual representar os estados de G_d como xN e xY ao invés de (x, Y) e (x, N), conforme mostrado na figura 6(c). □

Um diagnosticador, tal como aquele representado na figura 6(c) é implementado na prática utilizando-se um computador digital (ou um controlador lógico programável). Seu estado inicial é feito igual a x_{0_d}, e após qualquer ocorrência de eventos observáveis, seu estado é atualizado de acordo com a função de transição de estados f_d. Quando o diagnosticador



(a) (b) (c)

Figura 6: Autômato G referente ao exemplo 5 (a); composição paralela entre G e A_l (b); $G_d = Obs(G\|A_l)$ (c).

alcança um estado cujos rótulos são todos iguais a Y, ele se torna certo de que a falha ocorreu. Por exemplo, para o autômato da figura 6(a), suponha que tanto o sistema quanto o diagnosticador estejam nos seus respectivos estados iniciais. O estado inicial do diagnosticador ($\{1N, 3Y\}$) possui ambos os rótulos Y e N, isto é, o evento σ_f, por ser não-observável pode ter ocorrido sem que sua ocorrência seja percebida pelo diagnosticador. Essa possibilidade é levada em consideração pela componente $3Y$, significando que o sistema pode estar no estado 3 e, nesse caso, com a ocorrência do evento de falha. Por outro lado, como o outro evento que pode ocorrer quando o sistema está no estado inicial é observável, então o diagnosticador deverá indicar que o sistema permaneceu no estado 1 e, por essa razão, a falha não ocorreu; essa possibilidade é representada pela componente $1N$. Dessa forma, o diagnosticador não poderá afirmar, com certeza, que a falha ocorreu, isto é, ele está incerto com relação à ocorrência ou não do evento associado à falha. Quando o sistema reporta ao diagnosticador a ocorrência do evento a, o seu estado muda para $4Y$, o que torna o diagnosticador certo da ocorrência de σ_f. Por outro lado, se o sistema reporta a ocorrência do evento b, o diagnosticador se torna certo da não ocorrência da falha, ou equivalentemente, que o sistema está em uma trajetória normal.

É importante ressaltar que, tendo em vista que $G_d = Obs(G\|A_l)$, então uma vez que o diagnosticador tiver certeza da ocorrência da falha, não voltará atrás, isto é, todos os estados seguintes continuarão indicando a falha. Contudo, é possível para um diagnosticador mudar de um estado de não falha para duvidoso ou certo. Nesse contexto, os estados do diagnosticador podem ser classificados, quanto à presença de rótulos Y e N, da seguinte forma (Sampath et al., 1995).

Definição 3 Um estado $x_d \in X_d$ é denominado certo (de falha), se $\ell = Y$ para todo $x\ell \in x_d$, e normal (ou de não falha) se $\ell = N$ para todo $x\ell \in x_d$. Se existir $x\ell, y\tilde{\ell} \in x_d$, x não necessariamente distinto de y tal que $\ell = Y$ e $\tilde{\ell} = N$, então x_d é um estado incerto de G_d. □

Usando as definições 2 e 3, é possível estabelecer as seguintes relações entre os estados do diagnosticador e as sequências da linguagem gerada por G (Sampath et al., 1995).

Lema 1
(i) Seja $x_d = f_d(x_{0_d}, s)$. Se x_d é um estado certo, então para todo $\omega \in P_{o_L}^{-1}(s)$, $\Sigma_f \in \omega$.

(ii) Se x_d é um estado incerto, então existem $s_1, s_2 \in L$ tais que $\Sigma_f \in s_1$ e $\Sigma_f \notin s_2$, porém $P_o(s_1) = P_o(s_2)$ e $f_d(x_{0_d}, P_o(s_1)) = f_d(x_{0_d}, P_o(s_2)) = x_d$. □

Uma consequência imediata da definição 2 e do lema 1 é que a linguagem gerada por G será diagnosticável com relação a

Σ_f e P_o se, e somente se, o diagnosticador sempre alcançar um estado certo para toda sequência arbitrariamente longa de L que contiver o evento σ_f. Isso não irá ocorrer se, e somente se, existir uma sequência de L que faça o diagnosticador ficar preso indefinidamente em um laço formado por estados incertos. Para explorar mais profundamente esse problema, considere as seguintes definições (Sampath et al., 1995).

Definição 4 Seja $L(G,x) = \{s \in \Sigma^* : f(x,s) \in X\}$, isto é, o conjunto formado por todas as sequências que levam o autômato G do estado $x \in X$ a um outro estado do autômato. Um conjunto de estados $\{x_1, x_2, \ldots, x_n\} \subset X$ forma um ciclo em um autômato G se existir uma sequência $s = \sigma_1\sigma_2\ldots\sigma_n \in L(G,x_1)$ tal que $f(x_l,\sigma_l) = x_{l+1}$, $l = 1,\ldots,n-1$, e $f(x_n,\sigma_n) = x_1$. □

Definição 5 Um conjunto de estados incertos $\{x_{d_1}, x_{d_2}, \ldots, x_{d_p}\} \subset X_d$ forma um ciclo indeterminado se as seguintes condições forem satisfeitas:

1) $x_{d_1}, x_{d_2}, \ldots, x_{d_p}$ forma um ciclo em G_d;

2) $\exists(x_l^{k_l}, Y), (\tilde{x}_l^{r_l}, N) \in x_{d_l}$, $x_l^{k_l}$ não necessariamente distinto de $\tilde{x}_l^{r_l}$, $l = 1,2,\ldots,p$, $k_l = 1,2,\ldots,m_l$, e $r_l = 1,2,\ldots,\tilde{m}_l$ de tal sorte que as sequências de estados $\{x_l^{k_l}\}$, $l = 1,2,\ldots,p$, $k_l = 1,2,\ldots,m_l$ e $\{\tilde{x}_l^{r_l}\}$, $l = 1,2,\ldots,p$, $r_l = 1,2,\ldots,\tilde{m}_l$ podem ser rearranjadas para formar ciclos em G, cujas sequências correspondentes s e $\tilde{s}$, formadas com os eventos que definem a evolução dos ciclos, têm como projeção $\sigma_1\sigma_2\ldots\sigma_p$, em que σ_1, σ_2,...,σ_p são definidos de acordo com o item 1). □

Usando as definições 2, 4 e 5 e o lema 1, pode-se enunciar a seguinte condição necessária e suficiente para a diagnose de uma linguagem.

Teorema 1 Uma linguagem L gerada por um autômato G será diagnosticável com relação à projeção P_o e $\Sigma_f = \{\sigma_f\}$ se, e somente se, o seu diagnosticador G_d não tiver ciclos indeterminados. □

Exemplo 6

1) Considere o autômato da figura 6(a). Note que o diagnosticador de G, mostrado na figura 6(c) não possui ciclos indeterminados. Dessa forma, pode-se concluir que L é diagnosticável em relação a Σ_o e $\Sigma_f = \{\sigma_f\}$.

2) Considere, agora, o autômato G cujo diagrama de transição de estados está representado na figura 7(a). Suponha que $\Sigma = \{a,b,c,\sigma_f\}$, $\Sigma_o = \{a,c\}$, $\Sigma_{uo} = \{b,\sigma_f\}$ e $\Sigma_f = \{\sigma_f\}$. O diagnosticador G_d associado a G pode ser visto na figura 7(b). Observe que o estado $\{4Y,6N\}$ é um estado incerto e como $f_d(\{4Y,6N\},c) = \{4Y,6N\}$, então o

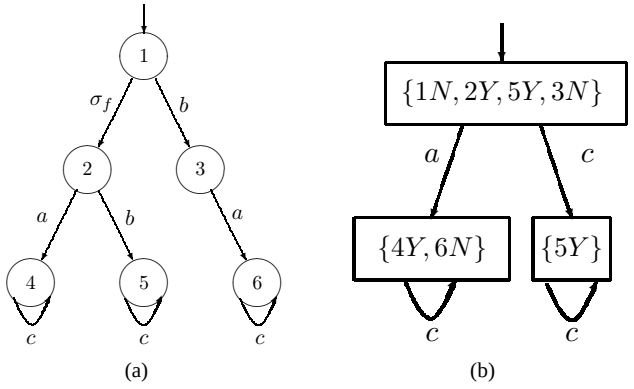


Figura 7: (a) Autômato G; (b) Diagnosticador centralizado para G.

estado incerto $\{4Y,6N\}$ forma um ciclo em G_d. Além disso, associados, respectivamente, às componentes $\{4Y\}$ e $\{6N\}$ de $\{4Y,6N\}$, existem em G dois ciclos formados pelos estado 4 e 6 (ver figura 7(a)). Logo, $\{4Y,6N\}$ forma um ciclo indeterminado em G_d. Pode-se, portanto, concluir que L é não diagnosticável em relação P_o e Σ_f. □

Observação 1 (i) De acordo com a definição 2, a diagnose da linguagem gerada por G se baseia em um único conjunto de eventos observáveis, ou equivalentemente, na projeção $P_o : \Sigma^* \to \Sigma_o^*$. Isso significa que, na prática, a decisão sobre a ocorrência ou não de uma falha é tomada por um diagnosticador central. Por essa razão, esse problema é usualmente referido na literatura como problema da diagnose centralizada e o autômato G_d é chamado diagnosticador centralizado.

(ii) É importante ressaltar que a existência de um ciclo de estados incertos no diagnosticador não necessariamente implica na impossibilidade de se diagnosticar a ocorrência de uma falha. Para que L seja diagnosticável em relação a P_o e Σ_f, não pode existir um ciclo de estados em G após a ocorrência da falha que corresponda ao ciclo de estados incertos no diagnosticador. Um exemplo que ilustra essa situação pode ser encontrado em Cassandras e Lafortune (2008, pp.114). □

4.2 Diagnose centralizada sob observação parcial

O conceito de diagnose de uma linguagem apresentado na definição 2 leva em conta não somente a linguagem gerada por um autômato, mas também o conjunto de eventos observáveis e a partição do conjunto de falhas. A dependência da diagnose da linguagem em relação ao conjunto de eventos observáveis sugere a possibilidade da linguagem gerada por

um autômato ser também diagnosticada com relação a Σ_f e à projeção $P_{o'} : \Sigma^* \rightarrow \Sigma_o'^*$, para algum subconjunto Σ_o' de Σ_o. Para abordar esse problema, além das hipóteses A1–A3, a seguinte hipótese será feita (Basilio e Lafortune, 2009):

A4. L é diagnosticável com relação à projeção $P_o : \Sigma^* \rightarrow \Sigma_o^*$ e Σ_f (diagnose centralizada).

Suponha que $G_d' = (X_d', \Sigma_o', f_d', \Gamma_d', x_{0_d}')$ denote um diagnosticador para L supondo observação parcial, isto é, G_d' é capaz de observar somente os eventos pertencentes a um conjunto $\Sigma_o' \subset \Sigma_o$. O seguinte resultado pode ser enunciado.

Teorema 2 Suponha que $G_d = (X_d, \Sigma_o, f_d, \Gamma_d, x_{0_d})$ e $G_d' = (X_d', \Sigma_o', f_d', \Gamma_d', x_{0_d}')$ denotem diagnosticadores centralizados, supondo observação total e parcial, respectivamente, isto é, $\Sigma_o' \subset \Sigma_o$ e $\Sigma_o' \neq \emptyset$. Então, $\mathit{Obs}(G_d, \Sigma_o') = (\hat{X}_d, \Sigma_o', \hat{f}_d, \hat{\Gamma}_d, \hat{x}_{0_d})$ (o observador de G_d em relação à projeção $P_{oo'} : \Sigma_o^* \rightarrow \Sigma_o'^*$) e G_d' são idênticos a menos da seguinte relação de equivalência entre os seus estados:

$$\begin{aligned} \hat{x}_d &= \{x_{d_1}, x_{d_2}, \ldots, x_{d_n}\} \in \hat{X}_d, x_{d_i} \in X_d \Leftrightarrow \\ x_d' &= \bigcup_{i=1}^{n} x_{d_i} \in X_d'. \end{aligned}$$

Demonstração. Seja $\Sigma = \Sigma_o \cup \Sigma_{uo}$, e considere um conjunto não vazio $\Sigma_o' \subset \Sigma_o$. Defina:

(i) $G_l = G \| A_l = (X_l, \Sigma, f_l, \Gamma_l, x_{0_l})$;

(ii) $G_d = \mathit{Obs}(G_l, \Sigma_o) = (X_d, \Sigma_o, f_d, \Gamma_d, x_{0_d})$;

(iii) $G_d' = \mathit{Obs}(G_l, \Sigma_o') = (X_d', \Sigma_o', f_d', \Gamma_d', x_{0_d}')$;

(iv) $\hat{G}_d = \mathit{Obs}(G_d, \Sigma_o') = (\hat{X}_d, \Sigma_o', \hat{f}_d, \hat{\Gamma}_d, \hat{x}_{0_d})$.

Note que as linguagens geradas por $\hat{G}_d$ e G_d' são iguais, isto é, $L(\hat{G}_d) = L(G_d')$, uma vez que $P_{oo'}[P_o(s)] = P_{o'}(s), \forall s \in L$, sendo $P_{oo'} : \Sigma_o^* \rightarrow \Sigma_o'^*$ e $P_{o'} : \Sigma^* \rightarrow \Sigma_o'^*$.

Seja $\hat{x}_d \in \hat{X}_d$ um estado de $\hat{G}_d$ alcançável por uma sequência $s' \in L(\hat{G}_d)$. Então, para todo $x_{d_i} \in \hat{x}_d$, $i \in \{1, 2, \ldots, n\}$, existe uma sequência $s_{d_i} \in L(G_d)$ tal que $P_{oo'}(s_{d_i}) = s'$ e $f_d(x_{0_d}, s_{d_i}) = x_{d_i}$. Da mesma forma, para todo $x_l \in x_{d_i}$ existe uma sequência $s \in L(G_l)$ tal que $P_o(s) = s_{d_i}$ e $f_l(x_{0_l}, s) = x_l$. Portanto, como G_d' é um autômato determinístico e $P_{oo'}[P_o(s)] = P_{o'}(s)$, então existe $x_d' = f_d'(x_{0_d}', s')$ tal que $x_l \in x_d'$. Dessa forma, para todo $x_l \in x_{d_i} \in \hat{x}_d$ tem-se que $x_l \in x_d'$ e, portanto, $\bigcup_{i=1}^{n} x_{d_i} \subseteq x_d'$.

Sejam, agora, $x_d' \in X_d'$ e $s' \in L(G_d')$ que satisfazem $x_d' = f_d'(x_{0_d}', s')$. Logo, existem $s \in L(G_l)$, com $s' = P_{o'}(s)$, e $x_l = f_l(x_{0_l}, s)$ tal que $x_l \in x_d'$. Como $\hat{G}_d$ é um autômato determinístico e $P_{oo'}[P_o(s)] = s'$, então existem $x_{d_i} = f_d(x_{0_d}, P_o(s))$ e $\hat{x}_d = \hat{f}_d(\hat{x}_{0_d}, s')$ tais que $x_l \in x_{d_i} \in \hat{x}_d$. Consequentemente, $x_d' \subseteq \bigcup_{i=1}^{n} x_{d_i}$. □

Observação 2 O diagnosticador G_d' será referido neste artigo como diagnosticador centralizado com observação parcial, ou simplesmente diagnosticador parcial. □

De acordo com o teorema 2, o diagnosticador parcial G_d' que observa eventos pertencentes ao subconjunto Σ_o' do conjunto dos eventos observáveis Σ_o pode ser construído diretamente a partir do diagnosticador centralizado G_d calculando-se $\mathit{Obs}(G_d, \Sigma_o')$ e substituindo-se cada um dos estados de $\mathit{Obs}(G_d, \Sigma_o')$ pela união dos conjuntos que formam cada um desses estados. Além disso, como $L(G_d') = L(Obs(G_d, \Sigma_o')) = P_{oo'}(L(G_d))$, sendo $P_{oo'} : \Sigma_o^* \rightarrow \Sigma_o'^*$, então embora a linguagem gerada pelo diagnosticador centralizado com observação total seja por hipótese viva, as linguagens geradas por diagnosticadores parciais não são necessariamente vivas. Isso ocorre sempre que os eventos que formam um ciclo em G_d se tornam não-observáveis no diagnosticador parcial; não é dificil verificar que quando isso acontece, esse ciclo se reduz a um único estado em G_d' (Basilio e Lafortune, 2009). Quando, após o sistema alcançar um determinado estado, ocorrer um ciclo de estados ligados por eventos não-observáveis, não haverá mudança de estado no diagnosticador parcial, embora os estados reais do autômato mudem de forma cíclica. Nesse caso, diz-se que esse diagnosticador parcial possui um ciclo escondido nesse estado.

Definição 6 (Ciclos escondidos e ciclos escondidos indeterminados) Suponha que $x_d' \in X_d'$ tenha sido obtido agrupando-se os estados $x_{d_1}, x_{d_2}, \ldots, x_{d_n} \in X_d$. Então, existe um ciclo escondido em x_d' de G_d' se para algum $\{i_1, i_2, \ldots, i_k\} \subseteq \{1, 2, \ldots, n\}$, $\{x_{d_{i_1}}, x_{d_{i_2}}, \ldots, x_{d_{i_k}}\}$ forma um ciclo em G_d. Além disso, se x_d' é um estado incerto e todos os estados $x_{d_{i_1}}, x_{d_{i_2}}, \ldots, x_{d_{i_k}}$ são certos, então o ciclo escondido é denominado indeterminado. □

Os ciclos escondidos serão representados nos diagramas de transição de estados dos diagnosticadores parciais por laços tracejados: os ciclos escondidos indeterminados serão rotulados como *ihc* (do inglês *indeterminate hidden cycle*) e os ciclos escondidos em estados normais, em estados certos ou em estados incertos cujos ciclos escondidos não sejam formados por estados certos de G_d serão rotulados simplesmente como *hc*. Conforme será visto mais à frente, a existência de ciclos escondidos que não sejam indeterminados não leva à perda de diagnosticabilidade da linguagem quando se tem apenas observação parcial dos eventos.

De acordo com a condição para diagnose dada na definição 2, todas as sequências $s \in \Psi(\Sigma_f)$ devem ser diagnosticadas pelo diagnosticador parcial G_d'. Uma sequência $s \in \Psi(\Sigma_f)$

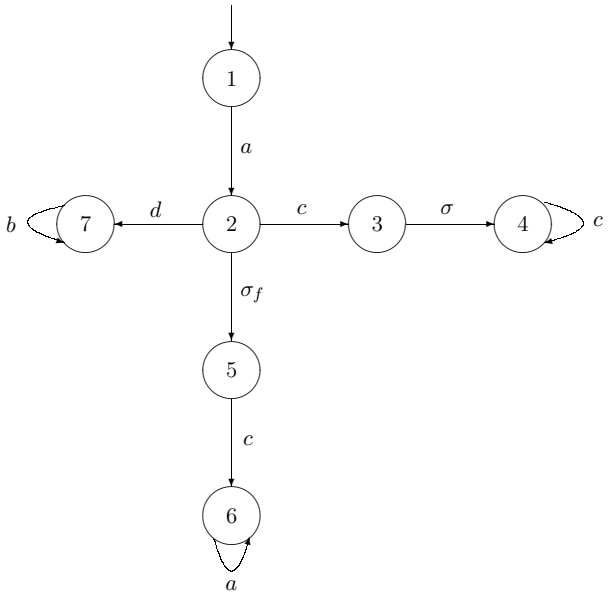


Figura 8: Autômato G cuja ocorrência do evento σ_f deve ser diagnosticada.

que não pode ser diagnosticada pelo diagnosticador parcial é chamado de sequência ambígua.

Definição 7 (Sequência ambígua) Uma sequência $s \in L$ é uma sequência ambígua em relação à projeção $P_{o'} : \Sigma^* \rightarrow \Sigma_o'^*$ e Σ_f se existir uma sequência $s' \in L$ tal que $\Sigma_f \in s$, porém $\Sigma_f \notin s'$, e $P_{o'}(s) = P_{o'}(s')$. □

O teorema a seguir provê uma condição necessária e suficiente para a diagnose de falhas sob observação parcial.

Teorema 3 Suponha que a linguagem L seja diagnosticável em relação à projeção P_o e Σ_f. Então L será também diagnosticável em relação à projeção $P_{o'} : \Sigma^* \rightarrow \Sigma_o'^*$, $\Sigma_o' \subset \Sigma_o$, e $\Sigma_f = \{\sigma_f\}$ se, e somente se, G_d' não tiver nenhum ciclo indeterminado (incluindo ciclos escondidos).

Exemplo 7 Para ilustrar o resultado do teorema 3, considere o autômato $G = (X, \Sigma, f, \Gamma, x_0, X_m)$ cujo diagrama de transição de estados está representado na figura 8. Suponha que $\Sigma = \{a, b, c, d, \sigma, \sigma_f\}$, $\Sigma_o = \{a, b, c, d\}$, $\Sigma_{uo} = \{\sigma, \sigma_f\}$ e $\Sigma_f = \{\sigma_f\}$. O diagnosticador G_d associado a G pode ser visto na figura 9(a), de onde se pode notar que L é diagnosticável em relação a P_o e Σ_f, uma vez que G_d não tem nenhum ciclo indeterminado.

Considere agora o problema de verificar se L é também diagnosticável em relação à projeção $P_{o'} : \Sigma^* \rightarrow \Sigma_o'^*$ e Σ_f, sendo $\Sigma_o' = \{c, d\} \subset \Sigma_o$. O diagnosticador parcial G_d' correspondente ao conjunto de eventos observáveis Σ_o' está representado na figura 9(b), de onde se pode ver que G_d' tem um ciclo escondido indeterminado no estado $\{3N, 4N, 6Y\}$. Como consequência, L é não diagnosticável em relação a $P_{o'}$ e Σ_f. A justificativa para a não diagnose de L com relação a $P_{o'}$ é a existência da sequência $s = a\sigma_f ca^n$, $n \in \mathbb{N}$, que contém o evento de falha σ_f, e que possui a mesma projeção em relação a $P_{o'}$ que uma sequência normal $s' = ac$, isto é, $P_{o'}(s) = P_{o'}(s') = c$; consequentemente s é uma sequência ambígua (s é, na verdade, a única sequência ambígua nesse exemplo). Para se obter um novo subconjunto de Σ_o que faça com que L possa ser diagnosticada sob observação parcial, note que $a \in s$, porém $a \notin \Sigma_o'$. Dessa forma, acrescentando-se o evento a ao conjunto de eventos observáveis Σ_o', e formando um novo conjunto de eventos observáveis $\Sigma_o'' = \{a, c, d\}$, é esperado que L se torne diagnosticável em relação $P_{o''} : \Sigma^* \rightarrow \Sigma_o''^*$ e Σ_f. Isso, de fato, acontece, uma vez que G_d'' não possui ciclos indeterminados (incluindo ciclos escondidos), conforme pode ser visto na figura 9(c). □

4.3 Diagnose descentralizada com coordenadação (codiagnose)

A fim de se contornar o problema da natureza distribuída de alguns sistemas, foi proposta em Debouk et al. (2000) a estrutura descentralizada mostrada na figura 10. Nessa arquitetura descentralizada, as leituras provenientes dos sensores não são centralizadas e sim distribuídas em diferentes módulos M_i, $i = 1, 2, \ldots, N$, cada um observando o comportamento do sistema tendo como base as informações fornecidas pelos sensores conectados a ele, isto é, a partir de conjuntos de eventos observáveis Σ_{o_i}, $i = 1, 2, \ldots, N$. Cada módulo processa a informação recebida (ocorrência de eventos), e, no caso da arquitetura descentralizada proposta por Debouk et al. (2000), somente podem comunicar os seus diagnósticos ao coordenador. A cada instante, cada módulo reporta ao coordenador o seu diagnóstico, e o coordenador processa essa informação de acordo com uma regra pré-estabelecida e toma uma decisão no que diz respeito à ocorrência ou não da falha. É importante observar que o coordenador não tem qualquer conhecimento do modelo do sistema, e tem, em geral, capacidades de memória e processamento limitados.

A diagnose da linguagem L pela estrutura descentralizada com coordenação representada na figura 10 depende, além do conjunto de falhas Σ_f, de outros quatro elementos, quais sejam: *(i)* as regras usadas para gerar os diagnósticos locais; *(ii)* as regras de comunicação entre os módulos e o coordenador; *(iii)* as regras de decisão empregadas pelo coordenador para a diagnose das falhas; *(iv)* as projeções

$$P_{o_i} : \Sigma^* \rightarrow \Sigma_{o_i}^*, i = 1, 2, \ldots, N,$$

associadas a cada módulo M_i. Ao conjunto dos elementos *(i)*, *(ii)* e *(iii)* dá-se o nome de protocolo.

Suponha que C denote a informação passada ao coordena-

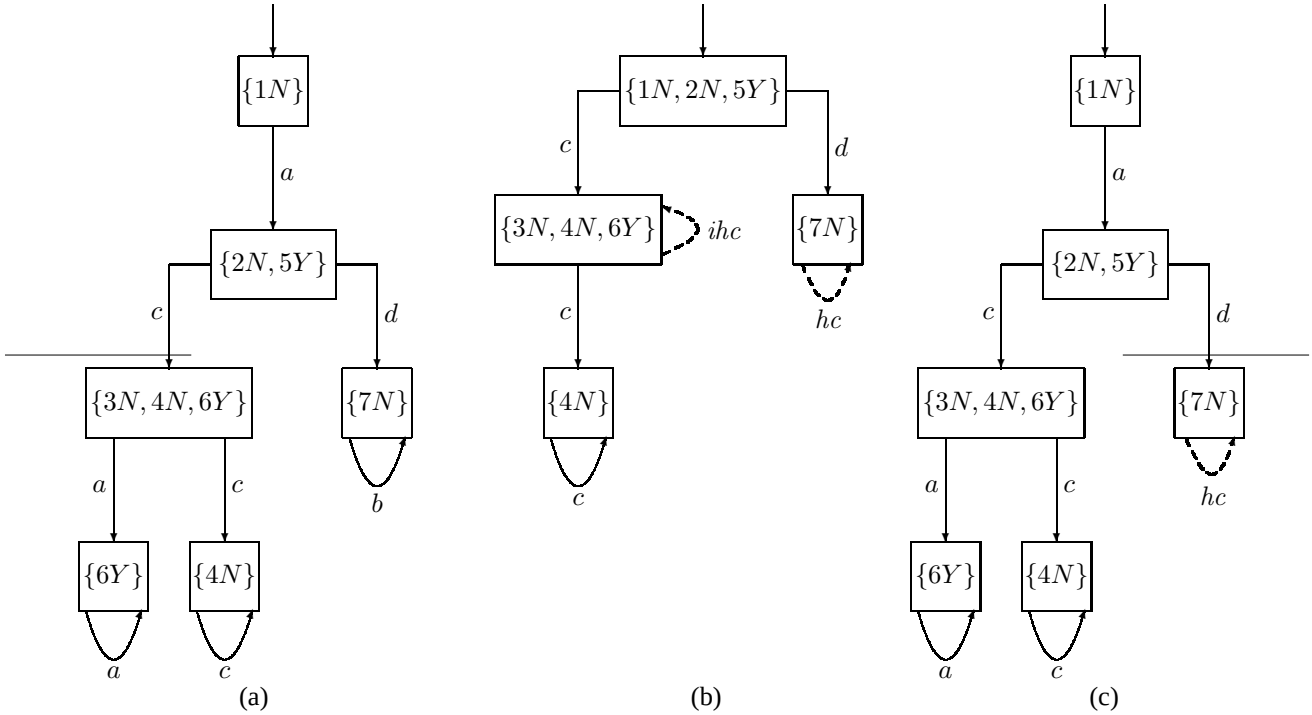


Figura 9: Diagnosticador G_d (a) e os diagnosticadores parciais G'_d (b) e G''_d (c) para os conjuntos de eventos observáveis $\Sigma'_o = \{c, d\}$ e $\Sigma''_o = \{a, c, d\}$, respectivamente.

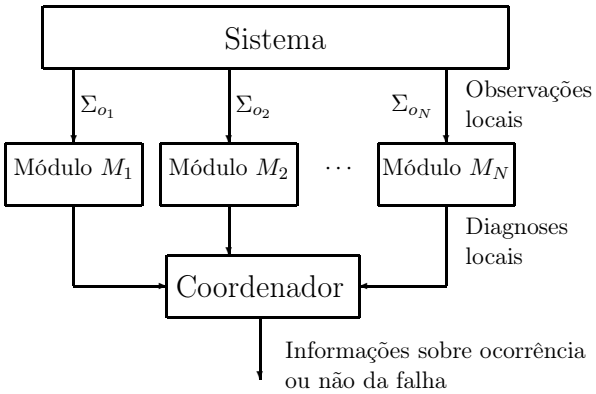


Figura 10: Estrutura descentralizada com coordenação.

dor para que este faça a diagnose. Observe que, para cada trajetória do SED, C é representada por um conjunto de informações que é dependente do protocolo empregado. Então, a informação passada ao coordenador para a diagnose é chamada certa se, baseada em C, o coordenador está certo da ocorrência de uma falha em Σ_f. Consequentemente, a definição 2 pode ser modificada para abranger sistemas descentralizados com coordenação, da forma apresentada a seguir (Debouk et al., 2000).

Definição 8 Uma linguagem L viva e prefixo-fechada é diagnosticável em relação a um protocolo, um conjunto de projeções P_{o_i}, $i = 1, \ldots, N$, e um conjunto de falhas $\Sigma_f = \{\sigma_f\}$, se a seguinte condição for verificada:

$$(\exists n \in \mathbb{N})(\forall s \in \Psi(\Sigma_f))(\forall t \in L/s)(\|t\| \geq n \Rightarrow C \text{ está certa}). \quad \square$$

Assim como no caso da diagnose centralizada, a detecção de qualquer falha será efetuada pelo coordenador após um atraso finito de sua ocorrência.

O protocolo a ser considerado neste tutorial foi proposto por Debouk et al. (2000) e consiste na implementação de um diagnosticador parcial em cada módulo, sendo o estado do diagnosticador, após a ocorrência de um evento observável, a informação para diagnose, na qual o módulo irá se basear para inferir sobre a ocorrência de falhas. Quando um módulo observa um evento que leva a um estado certo no seu diagnosticador, ele comunica a ocorrência da falha ao coordenador. Da parte do coordenador, a declaração definitiva da ocorrência de uma falha dar-se-á sempre que pelo menos um módulo reportar a ocorrência da falha; por outro lado ele se manterá "em silêncio" quando nenhum dos módulos reportar a ocorrência de falha. Note que esse protocolo pode ser visto como uma extensão da diagnose centralizada para o caso de diagnose descentralizada com coordenação e, por essa razão,

no decorrer deste artigo, será referida simplesmente como codiagnose.

Para se abordar o problema da codiagnose, além das hipóteses A1 a A3, deve-se supor ainda que:

A5. L não é diagnosticável em relação a P_{o_i}, $i = 1, 2, \ldots, N$.

A6. A comunicação entre os módulos e o coordenador é confiável, isto é, todas as mensagens enviadas por todos os módulos são recebidas pelo coordenador corretamente e na mesma ordem do envio.

A hipótese A5 exclui o caso trivial em que um módulo possa vir a desempenhar o papel de diagnosticador centralizado e a hipótese A6 foi removida por Basilio e Lafortune (2009). É importante ressaltar que em Debouk et al. (2000), uma hipótese adicional bastante restritiva foi feita: G não possui nenhum ciclo de eventos não-observáveis com relação a Σ_{o_i}, $i = 1, 2, \ldots, N$. Essa hipótese foi aqui removida, sendo os ciclos de eventos não-observáveis em relação a Σ_{o_i} considerados como ciclos escondidos.

A idéia por trás do problema da codiagnose é que toda sequência $s \in \Psi(\Sigma_f)$ deve ser diagnosticada por, pelo menos, um diagnosticador parcial. Uma sequência $s \in \Psi(\Sigma_f)$ que não for diagnosticada por nenhum diagnosticador parcial será denominada uma sequência totalmente ambígua, cuja definição é dada a seguir.

Definição 9 (Sequência totalmente ambígua) Uma sequência $s \in L$ será totalmente ambígua em relação às projeções $P_{o_i}, i = 1, 2, \ldots, N$, e à falha σ_f, se existirem N sequências $s_1, s_2, \ldots, s_N \in L$, não necessariamente distintas, tais que

1) $\Sigma_f \in s$, porém $\Sigma_f \notin s_i$, $i = 1, 2, \ldots, N$;

2) $P_{o_i}(s) = P_{o_i}(s_i)$, $i = 1, 2, \ldots, N$. □

Consequentemente, a fim de se verificar se L é diagnosticável em relação às projeções $P_{o_i}, i = 1, 2, \ldots, N$ e $\Sigma_f = \{\sigma_f\}$, basta se certificar da existência de sequências totalmente ambíguas. Com o intuito de se obter um teste para detectar a existência de sequências totalmente ambíguas, seja $G_{d_i} = (X_{d_i}, \Sigma_{o_i}, f_{d_i}, \Gamma_{d_i}, x_{0_{d_i}})$ o diagnosticador parcial para o módulo M_i, $i = 1, 2, \ldots, N$, e suponha que G_d denote o diagnosticador centralizado. Considere o diagnosticador G_{test_N} definido da seguinte forma:

$$G_{\text{test}_N} = (\|_{i=1}^{N} G_{d_i}) \| G_d. \tag{10}$$

Pode-se, portanto, concluir que

$$L(G_{\text{test}_N}) = \{\bigcap_{i=1}^{N} P_{o_i}^{-1}[L(G_{d_i})]\} \bigcap L(G_d).$$

Dessa forma,

$$L(G_{\text{test}_N}) = L(G_d), \tag{11}$$

que mostra que G_{test_N} provê os meios necessários para se identificar o estado atual dos diagnosticadores parciais G_{d_i}, $i = 1, 2, \ldots, N$, após a execução de qualquer sequência de L. Observe ainda que o estado x_{t_N} de G_{test_N} tem a seguinte estrutura:

$$x_{t_N} = (x_{d_1}, x_{d_2}, \ldots, x_{d_n}, x_d),$$

em que $x_{d_i} \in X_{d_i}$ e $x_d \in X_d$.

As definições de estado incerto e ciclo indeterminado são estendidas para codiagnose da seguinte forma.

Definição 10 Um estado x_t de G_{test_N} é certo se x_d é certo e x_{d_i} for certo para algum $i \in \{1, 2, \ldots, N\}$, e é incerto se x_d é certo e x_{d_i} for incerto para todo $i \in \{1, 2, \ldots, N\}$. □

Definição 11 Um ciclo em G_{test_N} será indeterminado se todos os correspondentes ciclos em G_{d_i}, $i = 1, 2, \ldots, N$ (incluindo ciclos escondidos) forem indeterminados. □

De acordo com a definição 9, uma sequência totalmente ambígua s é uma sequência $s \in \Psi(\Sigma_f)$ que leva a ciclos indeterminados em todos os diagnosticadores parciais G_{d_i}. Além disso, em face da hipótese A4, para todo $s \in \Psi(\Sigma_f)$, existirá sempre uma sequência de comprimento finito t tal que st leva a um evento certo de G_d. Consequentemente, conforme mencionado em Debouk et al. (2000), G_{test_N} pode ser usado para verificar a existência de sequências totalmente ambíguas.

Teorema 4 Uma linguagem L, viva e prefixo-fechada, será codiagnosticável em relação ao conjunto de projeções $P_{o_i} : \Sigma^* \to \Sigma_{o_i}^*$, $i = 1, 2, \ldots, N$ e $\Sigma_f = \{\sigma_f\}$ se, e somente se, G_{test_N} não possuir ciclos indeterminados. □

A prova apresentada em Debouk et al. (2000) é também válida aqui, uma vez que a definição 11 foi modificada para acomodar também os ciclos escondidos indeterminados.

Exemplo 8 Considere o autômato G cujo diagrama de transição de estados está representado na figura 7(a). Suponha que os conjuntos de eventos observáveis e não-observáveis de G sejam $\Sigma_o = \{a, b, c\}$ e $\Sigma_{uo} = \Sigma_f = \{\sigma_f\}$, respectivamente. O diagnosticador centralizado G_d para G está representado na figura 11, que mostra que a linguagem L gerada por G pode ser diagnosticada em relação a $P_o : \Sigma^* \to \Sigma_o^*$ e Σ_f. Suponha agora que nem todos os eventos observáveis estejam disponíveis simultaneamente em um mesmo lugar. Assim, para contornar esse problema, um diagnosticador descentralizado com coordenação composto, por exemplo, por dois módulos, deve ser construído para detectar a ocorrência de σ_f. Considere as seguintes situações:

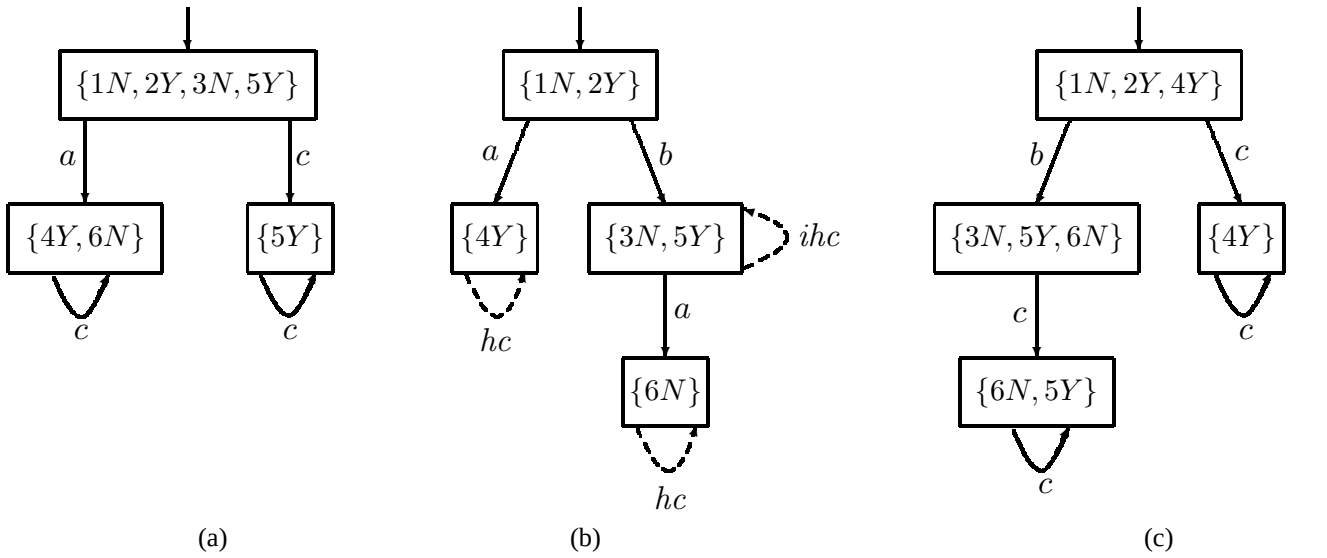


Figura 12: Diagnosticadores parciais para o autômato G da figura 7(a) para os seguintes conjuntos de eventos observáveis: (a) $\Sigma_{o_1} = \{a, c\}$; (b) $\Sigma_{o_2} = \{a, b\}$; (c) $\Sigma_{o_3} = \{b, c\}$.

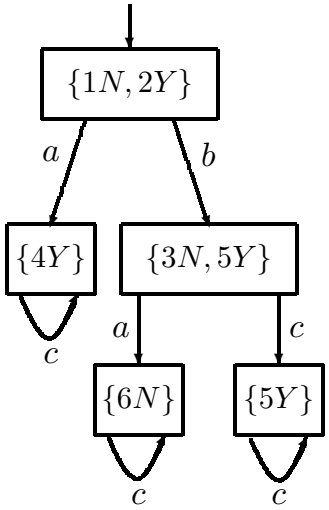


Figura 11: Diagnosticador centralizado para G.

1^{o} *caso*. Os conjuntos de eventos observáveis são $\Sigma_{o_1} = \{a, c\}$ e $\Sigma_{o_2} = \{a, b\}$. Os correspondentes diagnosticadores parciais G_{d_1} e G_{d_2} estão representados nas figuras 12(a) e (b), e o diagnosticador teste $G_{\text{test}_3} = G_{d_1} \| G_{d_2} \| G_d$ pode ser visto nas figura 13(a). Observe que embora as linguagens geradas por G e G_d sejam vivas, a linguagem gerada por G_{d_2} não é viva. Isso ocorre porque a linguagem gerada por G_d se mantém viva após os estados $\{4Y\}$ e $\{6N\}$ pela ocorrência do evento c, que se tornou não-observável para o módulo 2; criando, consequentemente, ciclos escondidos nos estados $\{4Y\}$ e $\{6N\}$. Observe ainda que, além desses ciclos, há ainda em G_{d_2} um outro ciclo escondido no estado $\{3N, 5Y\}$, também devido ao evento c. A presença de um ciclo indeterminado escondido em G_{d_2} não altera a codiagnose do sistema, uma vez que, conforme pode ser visto na figura 13(a), G_{test_3} não possui ciclos indeterminados, e, portanto, L é diagnosticável em relação a $P_{o_i} : E^* \to \Sigma_{o_i}$, $i = 1, 2$, e Σ_f.

2^{o} *caso*. Os conjuntos de eventos observáveis são $\Sigma_{o_2} = \{a, b\}$ e $\Sigma_{o_3} = \{b, c\}$. Os correspondentes diagnosticadores parciais G_{d_2} e G_{d_3} estão representados nas figuras 12(b) e (c), e o diagnosticador de teste $G'_{\text{test}_3} = G_{d_2} \| G_{d_3} \| G_d$ pode ser visto na figura 13(b). Note que o estado $\{3N, 5Y; 6N, 5Y; 5Y\}$ de G'_{test_3} tem um ciclo indeterminado formado com o estado $\{6N, 5Y\}$ de G_{d_3}, e um ciclo indeterminado escondido formado com o estado $\{3N, 5Y\}$ de G_{d_2}. Dessa forma, pode-se concluir que L é não diagnosticável em relação a $P_{o_i} : \Sigma^* \to \Sigma_{o_i}^*$, $i = 2, 3$ e Σ_f. Isso se deve ao fato de, das duas sequências arbitrariamente longas $s_1 = \sigma_f a c^n$ e $s_2 = \sigma_f b c^n$ ($n \in \mathbb{N}$) que contêm σ_f, somente a sequência s_1 pode ser detectada nessa configuração; observe que enquanto $P_{o_2}(s_1)$ e $P_{o_3}(s_1)$ levam ambos a estados certos de G_{d_2} e G_{d_3}, $P_{o_2}(s_2)$ e $P_{o_3}(s_2)$ levam a estados incertos tanto em G_{d_2} quanto em G_{d_3}. Dessa forma, s_2 é uma sequência totalmente ambígua.

Conforme visto acima a linguagem gerada pelo autômato da figura 7(a) é codiagnosticável em relação a Σ_f, P_{o_1}, P_{o_2}. Na prática a diagnose de $L(G)$ é feita da seguinte forma. Constroem-se os diagnosticadores parciais G_{d_1} e G_{d_2} e um coordenador central que recebe informações dos diagnosticadores parciais. Deve-se observar que os diagnosticadores G_{d_1} e G_{d_2} não se comunicam entre si. Caso ocorra,

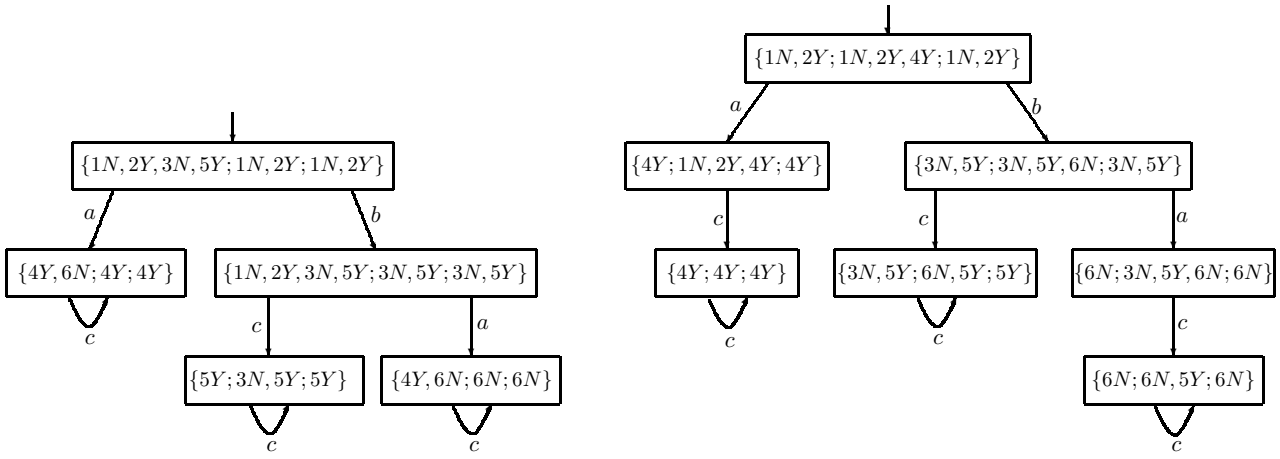


Figura 13: Diagnosticadores de teste para o autômato G da figura 7(a): $G_{\text{test}_3} = G_{d_1} \| G_{d_2} \| G_d$ (a); $G'_{\text{test}_3} = G_{d_2} \| G_{d_3} \| G_d$ (b).

por exemplo, o evento b, ambos os diagnosticadores parciais permanecerão em silêncio visto que estarão nos estados $\{1N, 2Y, 3N, 5Y\}$ e $\{3N, 5Y\}$. Caso o próximo evento a ocorrer seja o evento a, ambos os diagnosticadores parciais permanecerão em silêncio pois G_{d_1} mudará para o estado $\{4Y, 6N\}$ (incerto), e G_{d_2} passará para o estado $\{6N\}$, assim permanecendo indefinidamente. Se o segundo evento a ocorrer for o evento c, o diagnosticador parcial G_{d_1} mudará para o estado certo $\{5Y\}$ e comunicará a ocorrência da falha ao coordenador, que, então, declarará a ocorrência da falha.

□

5 VERIFICADOR

O teste desenvolvido na seção anterior para verificação da diagnosticabilidade de linguagens geradas por SEDs se baseia na construção de um autômato determinístico denominado diagnosticador. Uma vez construído o diagnosticador, a diagnosticabilidade da linguagem pode ser testada com complexidade polinomial em relação à cardinalidade do espaço de estado do diagnosticador. Contudo, a cardinalidade do espaço de estados do diagnosticador é, no pior caso, exponencial, tendo em vista que ao agrupar estados de G para chegar a um estado de G_d obtém-se, no pior caso,

$$|X_d| = 2^{2|X|}, \tag{12}$$

ou seja, $O(2^{|X|})$. Para solucionar o problema da complexidade exponencial do espaço de estados, foram propostos os chamados autômatos verificadores (Jiang et al. (2001), Yoo e Lafortune (2002) e Qiu e Kumar (2006) para o caso da diagnose centralizada e Qiu e Kumar (2006), Wang et al. (2007) e Basilio e Lafortune (2009) para codiagnose), cujos espaços de estados dos autômatos são polinomiais em relação à cardinalidade do espaço de estados de G. O teste proposto por Qiu e Kumar (2006) é consideravelmente mais simples e intuitivo que aqueles apresentados em Jiang et al. (2001), Yoo e Lafortune (2002), Wang et al. (2007) e Basilio e Lafortune (2009) e, por essa razão, será apresentado em detalhes nessa seção.

5.1 Verificador centralizado

Seja G o autômato determinístico da equação (6). O verificador centralizado proposto por Qiu e Kumar (2006) é construído de acordo com o seguinte algoritmo.

Algoritmo 2 (Construção do verificador centralizado)

Passo 1: Construção do autômato H^C.

1. Construa um autômato H_0 que tenha um único estado e um único laço com todos os eventos de Σ exceto os eventos do conjunto de falhas Σ_f, isto é, $\Sigma \backslash \Sigma_f$;

2. Calcule $H := G \times H_0$, em que $H = (X_H, \Sigma, f_H, \Gamma_H, x_{0_H})$;

3. Obtenha o autômato[2] $H^C = (X_{H^C}, \Sigma, f_{H^C}, \Gamma_{H^C}, x_{0_{H^C}})$ em que $X_{H^C} = X_H \cup \{F\}$, sendo que F denota o estado de falha e cuja função de transição f_{H^C} é definida da seguinte forma:

$$f_{H^C}(x_{H^C}, \sigma) = \begin{cases} f_H(x_H, \sigma) & \text{, se } \sigma \in \Gamma(x_H), \\ F & \text{, caso contrário.} \end{cases}$$

$f_{H^C}(F, \sigma) = F$ para todo $\sigma \in \Sigma$.

[2]Note que o autômato H^C somente não representa o complementar de H pelo simples fato do estado F não ser marcado.

Passo 2: Construção do verificador G_V.
O verificador G_V é uma sêxtupla definida como:

$$G_V = (X_V, \Sigma_V, f_V, \Gamma_V, x_{0_V}), \qquad (13)$$

em que

- $X_V = X \times X_{H^C} \times X_H$;
- $x_{0_V} = (x_0, x_{0_{H^C}}, x_{0_H})$;
- $\Sigma_V = \bar{\Sigma} \times \bar{\Sigma}_H$, sendo $\bar{\Sigma} = \Sigma \cup \{\epsilon\}$ e $\bar{\Sigma}_H = \Sigma_H \cup \{\epsilon\}$;
- sendo $x_V = (x, x_{H^C}, x_H) \in X_V$ e $\sigma_V = (\sigma, \sigma_H) \in \Sigma_V \setminus \{\epsilon\epsilon\}$, então $f_V : X_V \times \Sigma_V \to X_V$ é definida como

  $$f_V(x_V, \sigma_V) := (f(x, \sigma), f_{H^C}(x_{H^C}, \sigma), f_H(x_H, \sigma_H))$$

  se e somente se

  $$(P_o(\sigma) = P_o(\sigma_H)) \wedge (f(x, \sigma) \neq \emptyset \vee f_H(x_H, \sigma_H) \neq \emptyset).$$

Note inicialmente que o autômato H é um subautômato de G que possui todas sequências de $L(G)$ que não contêm o evento σ_f. Assim, toda sequência de $L(G)$ que não possuir a falha σ_f deverá pertencer a $L(H)$. Além disso, de acordo com a construção de G_V (passo 2 do algoritmo 2), o seu estado possui três componentes (x, x_{H^C}, x_H) e a função de transição de estados é definida utilizando-se uma combinação de eventos $\sigma\sigma_H$, em que $\sigma \in G$ e $\sigma_H \in H$, não necessitando um evento de H^C, uma vez que f_{H^C} é total em seu domínio. Portanto, as sequências geradas em G_V são as sequências u e v tais que $u \in L(G)$ e $v \in L(H)$. Como, de acordo com o passo 2, deve-se ter $P_o(u) = P_o(v)$, então $L(G)$ não será diagnosticável com relação a P_o e Σ_f se existir uma sequência $v \in L(H)$ que não contenha a falha σ_f e que tenha a mesma projeção que uma outra sequência $u \in L(G)$ tal que $\sigma_f \in u$. Isso permite verificar a diagnosticabilidade da linguagem gerada por G com relação a P_o e Σ_f conforme mostra o resultado a seguir.

Teorema 5 (Qiu e Kumar, 2006) Sejam $G = (X, \Sigma, \Gamma, f, x_0)$ e $H = (X_H, \Sigma, f_H, \Gamma_H, x_{0_H})$ autômatos que geram as seguintes linguagens $L(G) = L = \bar{L}$ e $L(H) = K = \bar{K}$, respectivamente e $K \subseteq L$. A ocorrência de sequências pertencentes a $L - K$ não será diagnosticável com relação a P_o, se e somente se existir um ciclo $cl^T = (x_{V_k}, \sigma_{V_k}, x_{V_{k+1}}, \ldots, x_{V_l}, \sigma_{V_l}, x_{V_k})$ no autômato verificador $G_V = (X_V, \Sigma_V, f_V, \Gamma_V, x_{0_V})$ tal que

$$\{\exists i \in [k, k+1, \ldots, l] : (x_{i_{H^C}} = F) \wedge (\sigma_i \neq \epsilon)\}$$

em que $l > k \geq 0$ e $x_{i_{H^C}}$ é a segunda coordenada do estado x_{i_V}, e σ_i é a primeira coordenada de $\sigma_{V_i} \in \Sigma_V$. □

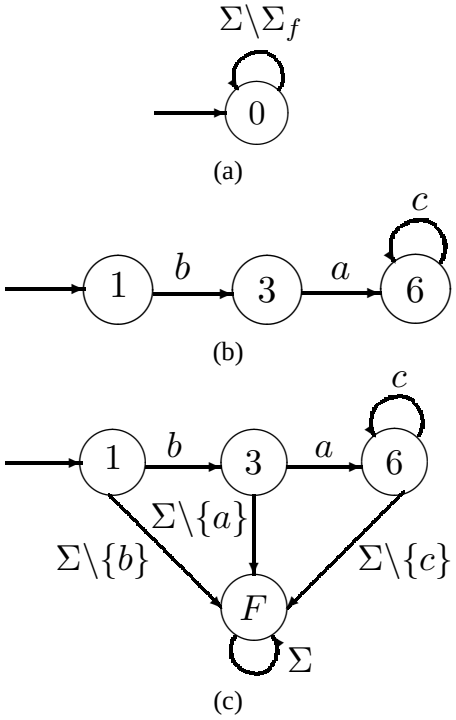


Figura 14: Autômato H_0 (a), autômato $H = G \times H_0$ (b) e autômato H^C (c).

Exemplo 9 Para ilustrar a construção dos verificadores, considere o autômato G da subseção 4.3, cujo diagrama de transição de estados está representado na figura 7(a). Suponha que o conjunto de eventos observáveis e não-observáveis de G sejam $\Sigma_o = \{a, b, c\}$ e $\Sigma_{uo} = \Sigma_f = \{\sigma_f\}$, respectivamente. De acordo com o *passo 1* do algoritmo 2, deve-se inicialmente construir o autômato H_0 mostrado na figura 14(a) e, em seguida, calcular H cujo resultado está mostrado na figura 14(b), para finalmente obter o diagrama de transição de H^C representado na figura 14(c). Uma vez obtidos os autômatos H e H^C, pode-se obter o verificador apresentado na figura 15. O estado inicial do verificador centralizado G_V tem como componentes os estados iniciais de G, H^C e H; portanto $x_{0_V} = \{1, 1, 1\}$. Para se obter o conjunto de eventos ativos do estado inicial, deve-se observar a função dos eventos ativos de cada componente, isto é, $\Gamma(x_0) = \{\sigma_f, b, \epsilon\}$ e $\Gamma_H(x_{0_H}) = \{b, \epsilon\}$. Para satisfazer a propriedade $P_o(\sigma) = P_o(\sigma_H)$, imposta pelo *passo 2* do algoritmo 2, deve-se ter $\Gamma_{G_V}(x_{0_V}) = \{\sigma_f\epsilon, bb\}$. Assim, $f_V(\{1, 1, 1\}, \sigma_f\epsilon) = \{f(1, \sigma_f), f_{H^C}(1, \sigma_f), f_H(1, \epsilon)\} = \{2, F, 1\}$ e $f_V(\{1, 1, 1\}, bb) = \{3, 3, 3\}$. No estado $x_V = \{2, F, 1\}$, as funções dos eventos ativos são $\Gamma(2) = \{a, b, \epsilon\}$ e $\Gamma_H(1) = \{b, \epsilon\}$. Como o evento a é observável e não está ativo em $x_H = \{1\}$, então não pode pertencer a $\Gamma_V(\{2, F, 1\})$. Além disso como $b \in \bar{\Sigma}$ e $b \in \Gamma_H(1)$, então a única combinação de eventos que poderá ocorrer no estado $\{2, F, 1\}$ é bb, levando o verificador para o estado

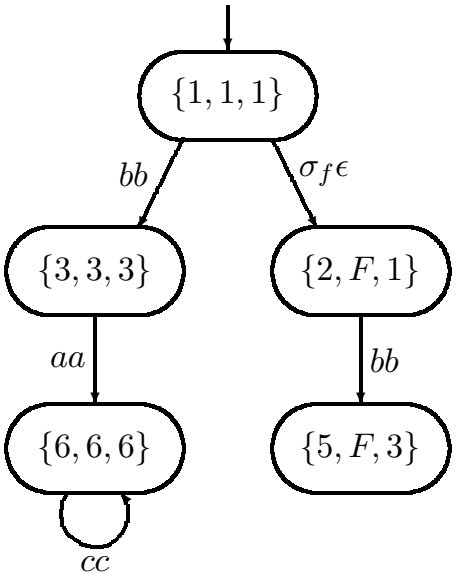


Figura 15: Verificador centralizado para G.

$\{5, F, 3\}$. Como, as funções dos eventos ativos de $\{5, F, 3\}$ são $\Gamma(5) = \{c, \epsilon\}$ e $\Gamma_H(3) = \{a, \epsilon\}$, não será possível satisfazer a propriedade $P_o(\sigma) = P_o(\sigma_H)$ no estado $\{5, F, 3\}$, e dessa forma não será possível definir qualquer transição a partir desse estado. Considere agora, o estado $\{3, 3, 3\}$. É fácil verificar que a única combinação de eventos possível é aa, levando assim ao estado $\{6, 6, 6\}$. Finalmente, a única combinação de eventos que pode ser definida a partir do estado $\{6, 6, 6\}$ é cc que, quando ocorrer, fará com que o verificador permaneça nesse mesmo estado.

Observando a figura 15 é possível ver que não existe um ciclo cl^T contendo $x_{i_{H^C}} = F$. Portanto, a linguagem L gerada por G será diagnosticável em relação a $P_o : \Sigma^* \rightarrow \Sigma_o^*$ e Σ_f. Note que esse resultado coincide com o obtido na subseção 4.3 com o auxílio de um diagnosticador. □

5.2 Verificador descentralizado

Qiu e Kumar (2006) propuseram também um verificador descentralizado que é construído de forma semelhante ao algoritmo 2. A idéia básica é construir um autômato de teste que identifique sequências contendo falhas e as correspondentes sequências que não contenham falhas para cada módulo mas que possuam a mesma projeção. O *passo 1* é mantido integralmente. Já o *Passo 2* é modificado conforme algoritmo a seguir.

Algoritmo 3 (Construção do verificador descentralizado)

Passo 1: Idêntico ao *Passo 1* do algoritmo 2.

Passo 2: Construção do verificador descentralizado G_V.
O verificador G_V é uma sêxtupla definida da seguinte forma:

$$G_V = (X_V, \Sigma_V, f_V, \Gamma_V, x_{0_V}, X_{m_V}), \qquad (14)$$

em que

- $X_V = X \times X_{H^C} \times X_H^1 \times \ldots \times X_H^N$;
- $x_{0_V} = (x_0, x_{0_{H^C}}, x_{0_H}^1, \ldots, x_{0_H}^N)$;
- $\Sigma_V = \bar{\Sigma}_H^{N+1}$;
- sendo $x_V = (x, x_{H^C}, x_H^1, \ldots, x_H^N) \in X_V$ e $\sigma_V = (\sigma, \sigma_H^1, \ldots, \sigma_H^N)$, então $f_V : X_V \times \Sigma_V \setminus \{\epsilon, \underbrace{\epsilon, \ldots, \epsilon}_{N}\} \rightarrow X_V$ é definida como:

  $$f_V(x_V, \sigma_V) := (f(x, \sigma), f_{H^C}(x_{H^C}, \sigma), f_H(x_H^1, \sigma_H^1), \ldots, f_H(x_H^N, \sigma_H^N))$$

  se e somente se

  $P_{o_i}(\sigma) = P_{o_i}(\sigma_H^i) \wedge (f(x, \sigma) \neq \emptyset \vee f_H(x_H^i, \sigma_H^i) \neq \emptyset, i = 1, \ldots, N)$.

Assim como no verificador centralizado, a presença de ciclos que contenham F na segunda componente no verificador levam à violação da diagnose conforme mostrado no teorema 5.

Exemplo 10 Para ilustrar a construção do verificador descentralizado, considere novamente o autômato G da figura 7(a) e suponha que os conjuntos de eventos observáveis de cada módulo sejam $\Sigma_{o_1} = \{a, b\}$ e $\Sigma_{o_2} = \{b, c\}$. O *passo 1* do algoritmo 2 é mantido no caso descentralizado, portanto H_0, H, H^C são aqueles autômatos cujos diagramas de transição estão representados na figura 14.

Com os autômatos H e H^C, pode-se obter o verificador descentralizado G_V apresentado na figura 16. O estado inicial de G_V tem como componentes os estados iniciais de G, H^C e H, e lembrando que existem dois módulos, então $N = 2$ e, portanto, $x_{0_V} = \{1, 1, 1, 1\}$. Para se obter o conjunto de eventos ativos do estado inicial, deve-se observar a função dos eventos ativos das componentes $1, 3$ e 4, uma vez que para todo estado x_{H^C} de H^C, $\Gamma_{H^C}(x_{H^C}) = \Sigma$. Dessa forma, $\Gamma(x_0) = \{\sigma_f, b, \epsilon\}$ e $\Gamma_H(x_{0_H}^1) = \Gamma_H(x_{0_H}^2) = \{b, \epsilon\}$. Para satisfazer as condições $P_{o1}(\sigma) = P_{o1}(\sigma_H^1)$ e $P_{o2}(\sigma) = P_{o2}(\sigma_H^2)$, impostas no *passo 2* do algoritmo 3, deve-se ter $\Gamma_{G_V}(x_{0_V}) = \{\sigma_f\epsilon\epsilon, bbb\}$. Assim, $f_V(\{1, 1, 1, 1\}, \sigma_f\epsilon\epsilon) = \{f(1, \sigma_f), f_{H^C}(1, \sigma_f), f_H(1, \epsilon), f_H(1, \epsilon)\} = \{2, F, 1, 1\}$ e $f_V(\{1, 1, 1, 1\}, bbb) = \{3, 3, 3, 3\}$. Para o estado $x_V = \{2, F, 1, 1\}$, tem-se que $\Gamma(2) = \{a, b, \epsilon\}$ e $\Gamma_H(1) = \{b, \epsilon\}$. Como o evento a não está ativo em H e é observável no

primeiro módulo, não se conseguirá atender à propriedade $P_{o1}(\sigma) = P_{o1}(\sigma_H^1)$. Portanto, a única combinação de eventos que poderá ocorrer no estado $\{2, F, 1, 1\}$ é bbb, levando o verificador do estado $\{2, F, 1, 1\}$ para o estado $\{5, F, 3, 3\}$. Como $\Gamma(5) = \{c, \epsilon\}$ e $\Gamma_H(3) = \{a, \epsilon\}$, e como c não está ativo em H e é observável no primeiro módulo, não se conseguirá atender à propriedade $P_{o1}(\sigma) = P_{o1}(\sigma_H^1)$. Além disso, como a é não-observável no segundo módulo e para atender as propriedades $P_{o1}(\sigma) = P_{o1}(\sigma_H^1)$ e $P_{o2}(\sigma) = P_{o2}(\sigma_H^2)$, deve-se ter $\Gamma_{G_V}(\{5, F, 3, 3\}) = \{\epsilon\epsilon a\}$. Assim, $f_V(\{5, F, 3, 3\}, \epsilon\epsilon a) = \{5, F, 3, 6\}$. Note que como $\Gamma(5) = \{c, \epsilon\}$, $\Gamma_H(3) = \{a, \epsilon\}$ e $\Gamma_H(6) = \{c, \epsilon\}$, e uma vez que o evento c está ativo em G e só é observável no segundo módulo, então para satisfazer as propriedades $P_{o1}(\sigma) = P_{o1}(\sigma_H^1)$ e $P_{o2}(\sigma) = P_{o2}(\sigma_H^2)$, deve-se ter $\Gamma_{G_V}(\{5, F, 3, 6\}) = \{c\epsilon c\}$ gerando um autolaço no estado $\{5, F, 3, 6\}$. Considere agora, o estado $\{3, 3, 3, 3\}$. Como $\Gamma(3) = \{a, \epsilon\}$ e $\Gamma_H(3) = \{a, \epsilon\}$, então as combinações de eventos possíveis que satisfazem as propriedades $P_{o1}(\sigma) = P_{o1}(\sigma_H^1)$ e $P_{o2}(\sigma) = P_{o2}(\sigma_H^2)$ são $\epsilon\epsilon a$, aaa, $aa\epsilon$, levando aos estados $\{3, 3, 3, 6\}$, $\{6, 6, 6, 6\}$ e $\{6, 6, 6, 3\}$, respectivamente. Em $\{3, 3, 3, 6\}$, a única combinação de eventos possível é $aa\epsilon$, que leva ao estado $\{6, 6, 6, 6\}$. Em $\{6, 6, 6, 3\}$, podem ocorrer $\epsilon\epsilon a$ que leva ao estado $\{6, 6, 6, 6\}$ e $\epsilon c\epsilon$ que leva ao próprio estado. Finalmente, as combinações de eventos que podem ser definidas a partir do estado $\{6, 6, 6, 6\}$ são ccc, $c\epsilon c$ e $\epsilon c\epsilon$ que, quando ocorrerem, farão com que o verificador permaneça no mesmo estado.

Note que o estado $\{5, F, 3, 6\}$ de G_V tem um ciclo cl^T contendo $x_{i_{H^C}} = F$ e $\sigma_{V_i} = c\epsilon c$. Dessa forma, pode-se concluir que L não é diagnosticável em relação a $P_{o_i} : \Sigma^* \rightarrow \Sigma_{o_i}$, $i = 1, 2$ e Σ_f. Isso se deve ao fato da sequência arbitrariamente longa $s = \sigma_f bc^n$ quando projetada em $P_{o_1}(s)$ e $P_{o_2}(s)$ poder ser confundida com as sequências b e bac^n que não contêm σ_f. Dessa forma, s é uma sequência totalmente ambígua. □

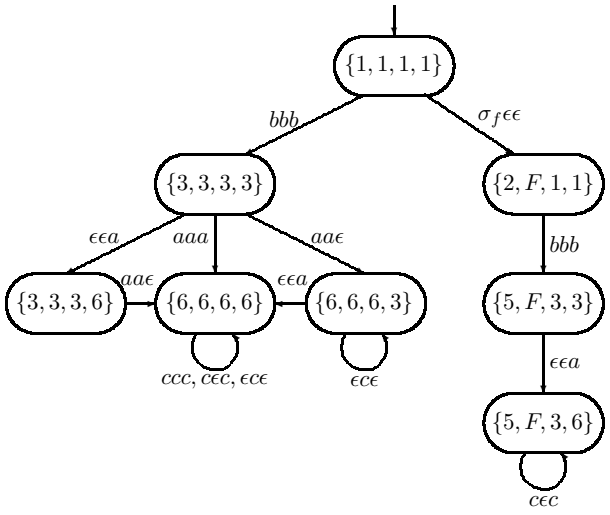


Figura 16: Verificador descentralizado para G.

6 COMENTÁRIOS FINAIS

O objetivo principal deste tutorial foi apresentar os conceitos básicos e os resultados principais para o estudo da diagnose de falhas de SEDs. Uma breve revisão dos conceitos de linguagem e autômatos foi também realizada com vistas a tornar o artigo autocontido. Para aqueles que não são familiarizados com a teoria de SEDs, sugere-se consultar Cassandras e Lafortune (2008) e Hopccroft et al. (2007). A revisão bibliográfica que foi realizada teve como intuito principal dar uma visão geral dos principais problemas que estão sendo abordados no que se refere à diagnose de falhas de SEDs e prover o leitor com as principais referências bibliográficas necessárias, caso ele deseje se aprofundar em qualquer um dos assuntos descritos na introdução. Por essa razão, deu-se preferência às publicações apresentadas em revistas científicas, estando assim longe de poder ser esgotada neste tutorial.

Entre os principais conceitos aqui apresentados, merecem destaque as definições de diagnosticabilidade e codiagnosticabilidade. Nesse sentido, condições necessárias e suficientes tanto para a diagnosticabilidade quanto para a codiagnosticabilidade foram apresentadas utilizando-se diagnosticadores e verificadores. Resultados recentes sobre diagnose com observação parcial foram também apresentados, destacando-se os conceitos de ciclos escondidos e uma metodologia para a obtenção de diagnosticadores centralizados com observação parcial a partir do diagnosticador centralizado que utiliza todos os eventos observáveis do sistema (Basilio e Lafortune, 2009). Os diagnosticadores, embora tenham espaço de estado que, no pior caso, tem cardinalidade exponencial em relação à cardinalidade do espaço de estado da planta podem ser usados para a diagnose online (Sampath et al., 1996; Simsek et al., 1999; Sampath, 2001), enquanto que os verificadores podem somente ser usados na análise da diagnosticabilidade. Vale a pena ressaltar que o verificador aqui apresentado é um autômato não-determinístico; um novo verificador que, além de ser determinístico e ter menor complexidade computacional que os algoritmos encontrados na literatura, foi recentemente desenvolvido por Moreira et al. (2010).

Diagnose de falhas em SEDs representa um campo frutífero para investigação. Entre possíveis tópicos de pesquisa nessa área poder-se-ia citar: *(i)* a diagnose robusta em SEDs em relação a perdas de observabilidade de eventos, já que neste tutorial foi suposto que todos os sensores que registram as ocorrências dos eventos funcionam perfeitamente. Caso alguns dos sensores que provejam informações sobre ocorrências de eventos falharem, o diagnosticador pode estacionar

em um determinado estado ou, até mesmo, dar uma informação incorreta em relação à ocorrência da falha. Pesquisas nessa área estão sendo realizadas pelos autores e os resultados já obtidos deram origem a artigos submetidos para publicação (Lima et al., 2010; Carvalho et al., 2010); *(ii)* alocação ótima de sensores, visando minimizar custos, sem, contudo, violar a especificação de tempo admissível entre a ocorrência da falha e a sua identificação pelo diagnosticador; *(iii)* diagnose em SEDs estocásticos, como continuação do trabalho iniciado por Thorsley e Teneketzis (2005); e *(iv)* controle supervisório em SEDs com tolerância a falhas (Paoli et al., 2008).

AGRADECIMENTOS

Os autores gostariam de agradecer aos revisores anônimos pela revisão cuidadosa e pelas sugestões para o aprimoramento desse artigo, ao CNPq e à FAPERJ pelo apoio financeiro.

REFERÊNCIAS

Alur, R. e Dill, D. L. (1994). A theory of timed automata, *Theoretical Computer Science* **126**(2): 183–235.

Basile, F., Chiacchio, P. e De Tommasi, G. (2009). An efficient approach for online diagnosis of discrete event systems, *IEEE Transactions on Automatic Control* **54**(4): 748–759.

Basilio, J. C. e Lafortune, S. (2009). Robust codiagnosability of discrete event systems, *Proceedings of the American Control Conference*, St. Louis, Missouri, pp. 2202–2209.

Belohlavek, R. (2002). Determinism and fuzzy automata, *Information Sciences* **143**(1-4): 205–209.

Benveniste, A., Fabre, E., Haar, S. e Jard, C. (2003). Diagnosis of asynchronous discrete-event systems: a net unfolding approach, *IEEE Transactions on Automatic Control* **48**(5): 714 – 727.

Cabasino, M., Giua, A., Lafortune, S. e Seatzu, C. (2009). Diagnosability analysis of unbounded petri nets, *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pp. 1267 –1272.

Carvalho, L. K., Basilio, J. C. e Moreira, M. V. (2010). Diagnose falhas de sistemas a eventos discretos sujeitos a perdas intermitentes de sensores, *XVIII Congresso Brasileiro de Automática*, Bonito, MS. (submetido para apresentação).

Cassandras, C. G. e Lafortune, S. (2008). *Introduction to Discrete Event Systems*, 2nd edn, Spirnger, Boston.

Chen, Y.-L. e Provan, G. (1997). Modeling and diagnosis of timed discrete event systems - a factory automation example, *Proceedings of the American Control Conference*, Vol. 1, Albuquerque, New Mexico, pp. 31–36.

Chung, S.-L., Wu, C.-C. e Jeng, M. (2003). Failure diagnosis: A case study on modeling and analysis by Petri nets, *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Vol. 3, Washington, pp. 2727–2732.

Contant, O., Lafortune, S. e Teneketzis, D. (2004). Diagnosis of intermittent faults, *Discrete Event Dynamic Systems: Theory And Applications* **14**(2): 171–202.

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

David, R. e Alla, H. (2005). *Discrete, Continuous, and Hybrid Petri Nets*, Springer, New York, NY.

Debouk, R., Lafortune, S. e Teneketzis, D. (2000). Coordinated decentralized protocols for failure diagnosis of discrete event systems, *Discrete Event Dynamic Systems: Theory and Applications* **10**: 33–86.

Dotoli, M., Fanti, M. P., Mangini, A. M. e Ukovich, W. (2009). On-line fault detection in discrete event systems by petri nets and integer linear programming, *Automatica* **45**(11): 2665–2672.

Gaubert, S. e Giua, A. (1999). Petri net languages and infinite subsets of n-m, *Journal Of Computer And System Sciences* **59**(3): 373–391.

Genc, S. e Lafortune, S. (2007). Distributed diagnosis of place-bordered Petri nets, *IEEE Transactions on Automation Science and Engineering* **4**(2): 206–219.

Giua, A. e Seatzu, C. (2005). Fault detection for discrete event systems using Petri nets with unobservable transitions, *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference, CDC-ECC 2005*, Vol. 2005, Seville, Spain, pp. 6323–6328.

Holloway, L. E. e Chand, S. (1996). Distributed fault monitoring in manufacturing systems using concurrent discrete-event observations, *Integrated Computer-Aided Engineering* **3**(4): 244–254.

Hopccroft, J. E., Motwani, R. e Ullman, J. D. (2007). *Introduction to automata theory, languages, and computation*, 3rd edn, Addison Wesley, Boston.

Jiang, S., Huang, Z., Chandra, V. e Kumar, R. (2001). A polynomial algorithm for testing diagnosability of discrete-event systems, *IEEE Transactions on Automatic Control* **46**(8): 1318–1321.

Jiang, S., Kumar, R. e Garcia, H. E. (2003). Diagnosis of repeated/intermittent failures in discrete event systems, *IEEE Transactions on Robotics and Automation* **19**(2): 310–323.

Kilic, E. (2008). Diagnosability of fuzzy discrete event systems, *Information Sciences* **178**(3): 858–870.

Lafortune, S., Teneketzis, D., Sampath, M., Sengupta, R. e Sinnamohideen, K. (2001). Failure diagnosis of dynamic systems: an approach based on discrete event systems, *Proceedings of the American Control Conference*, Arlington, VA, pp. 2058–2071.

Lefebvre, D. e Delherm, C. (2007). Diagnosis of des with petri net models, *IEEE Transactions on Automation Science and Engineering* **4**(1): 114–118.

Li, Z. H., Li, P. e Li, Y. M. (2006). The relationships among several types of fuzzy automata, *Information Sciences* **176**(15): 2208–2226.

Lima, S. T. S., Basilio, J. C., Lafortune, S. e Moreira, M. V. (2010). Diagnose centralizada de falhas de sistemas a eventos discretos robusta à perda permanente de sensores, *XVIII Congresso Brasileiro de Automática*, Bonito, MS. (submetido para apresentação).

Lin, F. (1994). Diagnosability of discrete event systems and its applications, *Discrete Event Dynamic Systems: Theory and Applications* **4**: 197–212.

Lin, F. e Wonham, W. M. (1990). Supervisory control and coordination of discrete-event systems with partial observation, *IEEE Transactions on Automatic Control* **35**: 1330–1337.

Lin, F. e Ying, H. (2002). Modeling and control of fuzzy discrete event systems, *IEEE Transactions On Systems Man And Cybernetics Part B-Cybernetics* **32**(4): PII S 1083–4419(02)03116–3.

Liu, F., Qiu, D., Xing, H. e Fan, Z. (2008). Decentralized diagnosis of stochastic discrete event systems, *IEEE Transactions on Automatic Control* **53**(2): 535–546.

Lunze, J. e Schroder, J. (2001). State observation and diagnosis of discrete-event systems described by stochastic automata, *Discrete Event Dynamic Systems: Theory And Applications* **11**(4): 319–369.

Manyari-Rivera, M., Basilio, J. C. e Bhaya, A. (2007). Integrated fault diagnosis based on Petri net models, *Proceedings of the 16th IEEE International Conference on Control Applications*, Singapore, pp. 958–963.

Moreira, M. V., Jesus, T. C. e Basilio, J. C. (2010). Polynomial time verification of decentralized diagnosability of discrete event systems, *Proc. of the American Control Conference*, Baltimore, Maryland, USA. Aceito para apresentação.

Murata, T. (1989). Petri nets - properties, analysis and applications, *Proceedings of the IEEE* **77**(4): 541–580.

Paoli, A. e Lafortune, S. (2005). Safe diagnosability for fault-tolerant supervision of discrete-event systems, *Automatica* **41**(8): 1335–1347.

Paoli, A., Sartini, M. e Lafortune, S. (2008). A fault tolerant architecture for supervisory control of discrete event systems, *Proceedings of the 17th IFAC World Congress*, Vol. 17, South Korea.

Peterson, J. (1981). *Petri net theory and the modeling of systems*, Prentice Hall, Englewood Cliffs, NJ.

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

Qiu, W., Wen, Q. e Kumar, R. (2009). Decentralized diagnosis of event-driven systems for safely reacting to failures, *IEEE Transactions on Automation Science and Engineering* **6**(2): 362–366.

Ramadge, P. J. e Wonham, W. M. (1989). The control of discrete-event systems, *Proceedings of the IEEE* **77**: 81–98.

Ramirez-Trevino, A., Ruiz-Beltran, E., Rivera-Rangel, I. e Lopez-Mellado, E. (2004). Diagnosability of discrete event systems. A Petri net based approach, *Proceedings of IEEE International Conference on Robotics and Automation*, Vol. 2004, New Orleans, LA, pp. 541–546.

Ru, Y. e Hadjicostis, C. N. (2009). Fault diagnosis in discrete event systems modeled by partially observed petri nets, *Discrete Event Dynamic Systems: Theory And Applications* **19**(4): 551–575.

Sampath, M. (2001). A hybrid aproach to failure diagnosis of industrial systems, *Proc. of the American Control Conference*, Arlington, VA, pp. 2077–2082.

Sampath, M., Lafortune, S., e Teneketzis, D. (1998). Active diagnosis of discrete-event systems, *IEEE Trans. on Automatic Control* **43**: 908–929.

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K. e Teneketzis, D. (1995). Diagnosability of discrete-event systems, *IEEE Trans. on Automatic Control* **40**: 1555–1575.

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

Simsek, H. T., Sengupta, R. e Eskafi, F. (1999). Fault diagnosis for intra-platoon communications, *Proc. of the 38th IEEE Conference on Decision and Control*, Piscataway, NJ, USA, pp. 3520–3525.

Thorsley, D. e Teneketzis, D. (2005). Diagnosability of stochastic discrete-event systems, *IEEE Trans. on Automatic Control* **50**: 476–492.

Tripakis, S. (2002). Fault diagnosis for timed automata, *in* Springe-Verlang (ed.), *Lecture notes in computer sciences, In Formal Techniques in Real Time and Fault Tolerant Systems (FTRTFT)*, Vol. 2469.

Ushio, T., Onishi, I. e Okuda, K. (1998). Fault detection based on Petri net models with faulty behaviors, *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Vol. 1, San Diego, pp. 113–118.

Wang, Y., Yoo, T. S. e Lafortune, S. (2007). Diagnosis of discrete event systems using decentralized architectures, *Discrete Event Dynamic Systems: Theory And Applications* **17**(2): 233–263.

Yoo, T.-S. e 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. e Wonham, W. (2005). Fault diagnosis in discrete-event systems: incorporating timing information, *Automatic Control, IEEE Transactions on* **50**(7): 1010–1015.

Zad, S., Kwong, R. e Wonham, W. (1999). Fault diagnosis in timed discrete-event systems, *Proceedings of the IEEE Conference on Decision and Control*, Vol. 2, Phoenix, Arizona,USA, pp. 1756–1761.