DIAGNOSE ROBUSTA DE SISTEMAS A EVENTOS DISCRETOS

Lilian Kawakami Carvalho

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Elétrica.

Orientador: João Carlos dos Santos Basilio

Rio de Janeiro

Março de 2011

DIAGNOSE ROBUSTA DE SISTEMAS A EVENTOS DISCRETOS

Lilian Kawakami Carvalho

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. João Carlos dos Santos Basilio, D.Phil.

Prof. Liu Hsu, Docteur d'Etat

Prof. Amit Bhaya, Ph.D.

Prof. José Eduardo Ribeiro Cury, Docteur d'Etat

Prof. Antônio Marcus Nogueira Lima, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

MARÇO DE 2011

Carvalho, Lilian Kawakami

Diagnose Robusta de Sistemas a Eventos Discretos/Lilian Kawakami Carvalho. – Rio de Janeiro: UFRJ/COPPE, 2011.

XVII, 143 p.: il.; 29, 7cm.

Orientador: João Carlos dos Santos Basilio

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2011.

Referências Bibliográficas: p. 134 – 143.

1. Sistemas a eventos discretos. 2. Diagnose de falha. 3. Robustez. I. Basilio, João Carlos dos Santos. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

"O impossível é possível."

Tiago Roux Oliveira

Agradecimentos

Agradeço aos meus pais Itajaci Costa Carvalho e Helena Keiko Kawakami Carvalho pela dedicação ao longo da minha vida e aos meus irmãos Laura Kawakami Carvalho e Rubens Akira Kawakami Carvalho pelo apoio diário.

Agradeço aos meus tios José Ferreira Pontes e Iraima Costa Carvalho Pontes e minhas primas Aline Carvalho Pontes e Beatriz Carvalho Pontes pelo carinho.

Agradeço ao meu orientador João Carlos Basilio por ter me acompanhado de perto durante a realização da tese e ter me auxiliado na minha formação profissional. Além de sempre me incetivar a realizar novas conquistas.

Agradeço ao Tiago Roux de Oliveira pelo carinho e atenção durante essa jornada.

Agradeço aos meus colegas da COPPE, em especial aos amigos Leonardo Bermeo, Patrícia Gróf, Josiel Gouveia, Eduardo Vieira Leão Nunes, Marcos Vicente Moreira, Oumar Diene e Sérgio Sami pela companhia e ajuda.

Agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico do Brasil (CNPq) e à Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ) pelo suporte financeiro.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

DIAGNOSE ROBUSTA DE SISTEMAS A EVENTOS DISCRETOS

Lilian Kawakami Carvalho

Março/2011

Orientador: João Carlos dos Santos Basilio

Programa: Engenharia Elétrica

Esta tese considera o problema da diagnose robusta de falhas de sistemas a eventos discretos. Inicialmente o problema da diagnose de falhas de sistemas a eventos discretos sujeitos a perdas permanentes de sensores é revisitado, sendo proposto um verificador de complexidade polinomial para a análise da diagnosticabilidade do sistema. Em seguida, o problema da diagnose de falhas em sistemas a eventos discretos sujeitos a perdas intermitentes de sensores é formulado e resolvido. Para resolver esse problema, propõe-se uma nova operação sobre a linguagem gerada pelo sistema, a dilatação, que modela os efeitos das perdas intermitentes de sensores na linguagem observada do sistema. Condições necessárias e suficientes para a diagnosticabilidade robusta em presença de falhas de intermitentes de sensores, expressas em termos de determinados ciclos de estados em diagnosticadores e verificadores, foram também apresentadas. Extensões para a codiagnosticabilidade foram também consideradas. Finalmente, é apresentada uma formulação geral para o problema da diagnose robusta, sendo também apresentadas condições necessárias e suficientes para a diagnosticabilidade robusta generalizada.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

ROBUST DIAGNOSE OF DISCRETE-EVENT SYSTEMS

Lilian Kawakami Carvalho

March/2011

Advisor: João Carlos dos Santos Basilio

Department: Electrical Engineering

This thesis deals with the problem of robust fault diagnosis of discrete-event systems. Initially, the problem of fault diagnosis in discrete-event systems subject to permanent sensor failure is revisited, and a verifier of polynomial complexity for the analysis of system diagnosticability analysis is proposed. In the sequel, the problem of fault diagnosis of discrete-event systems subject to intermittent sensor failure is formulated and solve. In order to solve this problem, a new operation over the language generated by the system (dilation) is proposed, models the effects of intermittent sensor failures in the system observable language. Necessary and sufficient conditions for robust diagnosability in the presence of intermittent sensor failure, expressed in terms of special cycles of states in diagnosers and verifiers, are also presented. Extensions to codiagnosability are also considered. Finally, a general formulation for the robust diagnosis problem is presented, and neccessary and sufficient conditions for the generalized robust diagnosability is also presented.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

Σ	—	Conjunto de eventos
Σ_o	—	Conjunto de eventos observáveis, $\Sigma_o \subset \Sigma$
Σ_{uo}	—	Conjunto de eventos não-observáveis, $\Sigma_{uo} \subset \Sigma$
$\Sigma_f = \{\sigma_f\}$	—	Conjunto de eventos de falha, $\Sigma_f \subseteq \Sigma_{uo}$
Σ^*	—	Fecho de Kleene
L	—	Linguagem gerada por um autômato
$\mathbb{L}$	—	Classe de linguagens $\{L_i : i = 1, 2, \ldots, m\}$
$\mathbb{G}$	—	Classe de autômatos $\{G_i : i1, 2, \ldots, m\}$
$\bar{s}$	—	Fecho do prefixo de uma sequência s
$\|\|t\|\|$	—	Comprimento de uma sequência t
L/s	—	Continuação da linguagem L após uma sequência s
$\Psi(\Sigma_f)$	—	Conjunto de todas as sequências de L que terminam com um evento de Σ_f
$\Sigma_f \in s$	—	$\bar{s} \cap \Psi(\Sigma_f) \neq \emptyset$
s_f	—	Último evento de uma sequência s
2^A	—	Conjunto potência de A
P_o	—	Projeção de elementos de Σ^* em Σ_o^*
P_o^{-1}	—	Projeção inversa de elementos de Σ_o^* em 2^{Σ^*}
$P_{o'}$	—	Projeção de elementos de Σ^* em $\Sigma_o'^*$
$P_{oo'}$	—	Projeção de elementos de Σ_o^* em $\Sigma_o'^*$, sendo $\Sigma_o' \subset \Sigma_o$
Σ_{o_i}	—	Conjuntos de eventos observáveis dos módulos M_i, $i = 1, 2, \ldots, N$
P_{o_i}	—	Projeção de elementos de Σ^* em $\Sigma_{o_i}^*$, $i = 1, 2, \ldots, N$
$P_{o_L}^{-1}(M)$	—	$\{s \in L : (\exists y \in M)[P_o(s) = y]\}$: Projeção inversa de uma linguagem M restrita à linguagem L
$G_1 \| G_2$	—	Composição paralela de G_1 e G_2
$G_1 \times G_2$	—	Composição produto de G_1 e G_2
$Obs(G, \Sigma_o)$	—	Observador de G em relação a Σ_o
G_d	—	Diagnosticador centralizado associado a um autômato G
A_ℓ	—	Autômato rotulador de falhas
G_{d_i}	—	Diagnosticador parcial para o módulo M_i, $i = 1, 2, \ldots, N$

G_{test_N}	—	Diagnosticador teste
R, R_i	—	Funções de renomeação
Σ_R, Σ_{R_i}	—	$R(\Sigma)$, $R_i(\Sigma)$, $i = 1, 2, \ldots, m$
G_N	—	Autômato que modela o comportamento de não falha do sistema
$G_{N,i}$	—	Autômato que modela o comportamento de não falha do sistema renomeado, $i = 1, 2, \ldots, m$
G_{N_i}, $G_{N_{R_i}}$	—	Autômatos de não falha, $i = 1, 2, \ldots, m$
$G^a_{N_i}$, $G_{\mathcal{N}_i}$	—	Autômatos aumentados, $i = 1, 2, \ldots, m$
G_F	—	Autômato que modela o comportamento de falha do sistema
G_{F_i}	—	Autômatos de falha, $i = 1, 2, \ldots, m$
G_V	—	Autômato verificador
G_{V_i}	—	Autômato verificador, $i = 1, 2, \ldots, m$
L_N	—	Linguagem gerada por G_N
L_{N_i}	—	Linguagem gerada por G_{N_i}, $i = 1, 2, \ldots, m$
Σ_{db}	—	$\{\Sigma_{o_1}, \Sigma_{o_2}, \ldots, \Sigma_{o_m}\}$ em que Σ_{o_i} são bases mínimas ou não mínimas para a diagnose de L
Σ_{rob}	—	$\{\Sigma_{uo_1}, \Sigma_{uo_2}, \ldots, \Sigma_{uo_m}\}$ em que $\Sigma_{uo_i} = \Sigma_o \backslash \Sigma_{o_i}$, para $i = 1, \ldots, m$
$\tilde{G}_d(\Sigma_o \backslash \Sigma'_o)$	—	Diagnosticador com marcação de perda de sensores
$\tilde{G}'_{d_i}$	—	Diagnosticadores parciais com marcações de perdas de sensores
$\tilde{G}_{d_0}$	—	Diagnosticador centralizado com marcações de perdas de sensores
$G_{du}(\Sigma_{db})$	—	Diagnosticador união
Σ_{isf}	—	Conjunto de eventos associados às perdas intermitentes de sensores, sendo $\Sigma_{isf} \subset \Sigma_o$
Σ_{nf}	—	Conjunto de eventos associados a sensores que não estão sujeitos a um mau funcionamento
G_{isf}	—	Autômato que modela o comportamento de G quando sujeito a perdas intermitentes de sensores
Σ'_{isf}	—	$\{\sigma' : \sigma \in \Sigma_{isf}\}$: Conjunto de eventos não-observáveis associados aos sensores sujeitos a perdas intermitentes
D_{isf}	—	Operador dilatação
L_{isf}	—	$L(G_{isf}) = D_{isf}(L)$
Σ'	—	$\Sigma \cup \Sigma'_{isf}$
Σ'_o	—	$\Sigma_o \cup \Sigma'_{isf}$
P'_o	—	Projeção de elementos de Σ'^* em Σ^*_o
P'_{oo}	—	Projeção de elementos de Σ'^*_o em Σ^*_o
$G_{isf,d}$	—	Diagnosticador de robustez: $Obs(G_{isf}\|\|A_\ell, \Sigma_o)$
$\hat{G}_{isf,d}$	—	Diagnosticador de robustez: $Obs(G_{d,isf}, \Sigma_o)$
$G_{isf,V}$	—	Verificador de robustez
$G_{isf,t}$	—	Diagnosticador teste de robustez

Capítulo 1

Introdução

Com a inserção de exigências mais rigorosas sobre o desempenho e a confiabilidade de sistemas complexos, torna-se necessário, cada vez mais, desenvolver métodos sistemáticos para a diagnose rápida e precisa de falhas em sistemas. Para esse fim, diversas abordagens têm sido propostas, tais como: i) métodos baseados em modelos matemáticos (Blanke *et al.*, 2006; Gertler, 1998; Hamscher *et al.*, 1992; Patton *et al.*, 2000); ii) métodos baseados em inteligência artificial e em sistemas especialistas (Angeli, 2008; Biswas *et al.*, 2004); iii) métodos baseados em sistemas a eventos discretos (veja Basilio *et al.* (2010) e as correspondentes referências). Os métodos quantitativos utilizam modelos analíticos do processo físico, sendo a diagnose baseada na comparação entre as medições nos sensores e o valores esperados para aquela variável. A diagnose de falhas utilizando sistemas especialistas é baseada na experiência de especialistas no processo. Com base nessa experiência, são construídos mapeamentos que associam as observações com as correspondentes diagnoses, sendo esses mapeamentos utilizados para diagnosticar a ocorrência de falhas. A diagnose de falhas utilizando modelos a eventos discretos pode ser vista como uma abordagem híbrida que utiliza um sistema especialista que é construído com base em modelos do sistema. Por essa razão, desfruta das vantagens e desvantagens das abordagens baseadas em modelos e que utilizam sistemas especialistas.

Nesse trabalho será considerada a diagnose de falhas utilizando modelos a eventos discretos. Essa abordagem tem despertado grande interesse nos últimos anos,

podendo ser aplicada 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). Isso decorre do fato de que SDVC podem também ser modelados como sistemas a eventos discretos (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 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 no capítulo 2 e consiste no desenvolvimento de um conjunto de regras (protocolo) a serem seguidas para a identificação e a diagnose de falhas.

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, o qual permite tanto análise, isto é,

inferir sobre a capacidade de se 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 (Manyari-Rivera *et al.*, 2007; 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).

A seguir serão descritos alguns tópicos relativos à diagnose de falhas em SEDs presentes na literatura .

Diagnose descentralizada. 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 em 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 pode ser feita utilizando-se somente os diagnosticadores locais (*i.e.*, os diagnosticadores projetados para cada um dos subsistemas).

Diagnose de falhas em SEDs estocásticos. 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 aquelas 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.

Diagnose de falhas em SED fuzzy. Outra forma de descrever a incerteza de SEDs é por meio dos chamados autômatos fuzzy (Belohlavek, 2002; Li *et al.*, 2006; Lin e Ying, 2002), nos quais 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 esteja entre zero e 1, não é possível precisar o tipo da falha que ocorreu. Se o grau de diagnosticabilidade for igual a zero, a linguagem não é diagnosticável.

Diagnose de falhas em SEDs temporizados. 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. Nessa formulação, um conjunto de *clocks* com dinâmicas dirigidas pelo tempo são incorporados aos autômatos e as 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 falhas distribuídas 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.

Diagnose segura. O conceito de diagnosticabilidade segura foi introduzido por Paoli e Lafortune (2005). Nessa abordagem, um sistema possui 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) estenderam 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.

Diagnose de falhas intermitentes. Um outro problema frequente em todas as áreas da engenharia são as falhas intermitentes decorrentes de 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.

Verificação da diagnosticabilidade em tempo polinomial. O diagnosticador proposto por Sampath *et al.* (1995), embora intuitivo e adequado para aplicação na diagnose em tempo real de SEDs, tem a sua utilização na verificação (análise) da diagnosticabilidade de SEDs questionada em função do espaço de estados do diagnosticador ter 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), Wang *et al.* (2007) e Moreira *et al.* (2011) estenderam esses verificadores para a codiagnose, levando aos chamados verificadores descentralizados.

Diagnose de falhas em SED modelados por redes de Petri. Um outro formalismo para a modelagem de SEDs são as redes de Petri (David e Alla, 2005; Murata, 1989; Peterson, 1981). Recentemente, o problema da diagnose de falhas em sistemas modelados por redes de Petri tem recebido grande atenção (Basile *et al.*, 2009; Benveniste *et al.*, 2003; Chung *et al.*, 2003; Dotoli *et al.*, 2009; Genc e Lafortune, 2007; Giua e Seatzu, 2005; Lefebvre e Delherm, 2007; Manyari-Rivera *et al.*, 2007; Ramirez-Trevino *et al.*, 2004; Ru e Hadjicostis, 2009; Ushio *et al.*, 1998). 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.

A robustez é uma propriedade importante que garante que o sistema mantém seu desempenho mesmo com variações do ambiente. Essa propriedade tem sido extensivamente estudada no contexto de sistemas dinâmicos de variáveis contínuas em sistemas diagnosticáveis (ver Chen e Patton (1999); Mangoubi (1998)). Apesar do grande número de trabalhos em diagnose de SEDs em arquitetura centralizada e descentralizada, poucos trabalhos abordam explicitamente o problema da diagnosticabilidade robusta de SEDs. O conceito diagnosticabilidade robusta foi introduzido por Basilio e Lafortune (2009) no contexto de diagnosticabilidade decentralizada supondo que a comunicação entre um módulo e o coordenador não é confiável. Mais recentemente no contexto de diagnosticabilidade centralizada, Lima *et al.* (2010) consideram a diagnosticabilidade robusta a perdas permanentes de sensores, isto é, quando um grupo de sensores para de funcionar e não são capazes de se recuperar. Para tanto, são utilizadas as redundâncias que possam existir em conjuntos de eventos observáveis formados pelo conjunto de eventos que garantem a diagnosticabilidade de uma falha para manter a diagnosticabilidade do sistema mesmo quando ocorrem perdas permanentes de sensores. Outra definição de diagnosticabilidade robusta foi proposta por Takai (2010), na qual o sistema é descrito por um conjunto de possíveis modelos que possuem o mesmo conjunto de eventos observáveis. Outros trabalhos em diagnosticabilidade robusta (Athanasopoulou *et al.*, 2010; Thorsley *et al.*, 2008) desenvolvem metodologias probabilísticas para a diagnose de falhas em maquinas de estados finitos baseadas em sequências de observação incertas que podem ocorrer devido à ocorrência de perdas de sensores, provocando, assim, inserção, remoção ou tranposição de símbolos em uma sequência.

Nesse trabalho será considerada a diagnose robusta de sistemas a eventos discretos modelados por autômatos (Cassandras e Lafortune, 2007; Hopcroft *et al.*, 2007). Inicialmente é considerado o trabalho de Lima *et al.* (2010), que utiliza o chamado diagnosticador união para verificar a diagnosticabilidade robusta a perdas permanentes de sensores. Para verificar a diagnosticabilidade robusta será aqui proposto

um verificador de complexidade polinomial.

Além de considerar a robustez a perdas permanentes de sensores, também é considerada a diagnosticabilidade robusta a perdas intermitentes de sensores (Carvalho *et al.*, 2010), que é uma formulação mais geral, uma vez que engloba as perdas permanentes de sensores. Para tanto, propõe-se uma modelagem para perdas intermitentes de sensores em SEDs e obtém-se um diagnosticador baseado nesse modelo que pode ser utilizado na verificação da diagnosticabilidade robusta ou ser utilizado para realizar a diagnose de falhas *online*. Além disso, são também fornecidas condições necessárias e suficientes para a diagnose de uma linguagem sujeita a perdas intermitentes de observabilidade. A verificação da diagnosticabilidade robusta utilizando verificadores é também considerada, assim como a codiagnosticabilidade robusta, que é verificada através do diagnosticador teste e do verificador proposto por Moreira *et al.* (2011) construído a partir do modelo do sistema que leva em conta as perdas intermitentes de sensores.

Na tentativa de generalizar a diagnosticabilidade robusta a perdas permanentes de sensores proposta por Lima *et al.* (2010) e Takai (2010), e a diagnosticabilidade robusta a perdas intermitentes proposta por Carvalho *et al.* (2010), é proposta a diagnosticabilidade robusta generalizada (Carvalho *et al.*, 2011). Essa abordagem considera tanto um sistema modelado por um conjunto de possíveis modelos utilizando o mesmo conjunto de eventos Σ — como em Takai (2010) — com a diferença que cada modelo pode ter conjuntos de eventos observáveis distintos — como em Lima *et al.* (2010). Além disso, as perdas intermitentes de sensores podem ser modeladas por um conjunto de modelos em que cada modelo representa as possíveis combinações de perdas de sensores. Para análise da diagnose robusta generalizada, é proposto um algoritmo em tempo polinomial baseado em Moreira *et al.* (2011).

Esse trabalho está organizado da seguinte forma. No capítulo 2 é apresentada uma breve revisão da teoria de SEDs e é formulado o problema da diagnose de falhas. Além disso, são apresentadas as condições necessárias e suficientes para a diagnose centralizada e descentralizada utilizando diagnosticadores e verificadores.

No capítulo 3 aborda-se o problema da diagnosticabilidade robusta a perdas permanentes de sensores sendo proposto um verificador de complexidade polinomial para a análise da diagnosticabilidade robusta a perdas permanentes de sensores. No capítulo 4, é considerada uma modelagem de sistemas a eventos discretos considerando falhas intermitentes nos sensores e é formulado e resolvido o problema da diagnose robusta a perdas intermitentes de sensores. No capítulo 5 é apresentada a definição de diagnosticabilidade robusta generalizada e é proposto um algoritmo em tempo polinomial para verificar a propriedade da diagnosticabilidade robusta generalizada. Comentários finais e sugestões de tópicos futuros de pesquisa são apresentados no capítulo 6.

Capítulo 2

Diagnose de falhas em sistemas a eventos discretos

O objetivo principal deste capítulo é apresentar uma breve revisão dos conceitos fundamentais para o estudo da diagnose de falhas em SEDs. Inicialmente, uma revisão da teoria de sistemas a eventos discretos será apresentada e, em seguida, será considerado o problema da diagnose de falhas em SEDs utilizando estruturas centralizada e descentralizada.

Este capítulo está organizado da seguinte forma. Na seção 2.1 é apresentada uma breve revisão da teoria de SEDs. Na seção 2.2 é formulado o problema da diagnose de falhas. Na seção 2.3 são apresentadas condições necessárias e suficientes para a diagnose centralizada. Além disso, é considerado o problema de se diagnosticar uma falha utilizando como conjunto de eventos observáveis um subconjunto do conjunto de eventos observáveis original. Ainda nessa seção será abordado o problema da diagnose descentralizada com coordenação. Na seção 2.4 são apresentadas as condições necessárias e suficientes para a diagnose centralizada e descentralizada utilizando verificadores. Finalmente, na seção 2.5 são apresentadas as conclusões.

A maioria dos resultados apresentados nesse capítulo pode ser encontrada também em Basilio *et al.* (2010) exceto os resultados apresentados na seção 2.4 cujos fundamentos são baseados em Moreira *et al.* (2011) e não em Wang *et al.* (2007).

2.1 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, valores discretos tais como valores numéricos pertencentes aos conjuntos $\mathbb{N}$ ou $\mathbb{Z}$, ou ainda 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 sistema, 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 2.1 *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.*

Tabela 2.1: 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	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
	$X_r = \{I, T, H, R\}$	$E_r = \{e_1, c_2, a_2, r_1\}$
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
	$X_2 = \{I_2, P_2\}$	$E_2 = \{a_2, t_2\}$

Caso uma outra peça chegue enquanto a máquina M_1 esteja 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 retornará à máquina M_1. A máquina M_2 recebe a peça do robô e a processa.

A tabela 2.1 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 o evento e_1 ocorrer, a máquina M_1 deverá estar no estado H_1 e o robô no estado I e 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 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. □

2.1.1 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).

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:

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

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 são 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 do prefixo de uma linguagem L (denotado por $\bar{L}$) é o conjunto formado por todos os prefixos das sequências 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 $\bar{L} = \{\varepsilon, a, ab, b, ba\}$. Uma linguagem L tal que $L = \bar{L}$ é dita ser prefixo-fechada.

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

$$\begin{array}{rcccl} P_o & : & \Sigma^* & \rightarrow & \Sigma_o^* \text{ , sendo } \Sigma_o \subseteq \Sigma \\ & & s & \mapsto & P_o(s), \end{array} \tag{2.1}$$

com as seguintes propriedades:

$$\begin{aligned} P_o(\varepsilon) &= \varepsilon, \\ P_o(\sigma) &= \begin{cases} \sigma, \text{ se } \sigma \in \Sigma_o, \\ \varepsilon, \text{ se } \sigma \in \Sigma \backslash \Sigma_o, \end{cases} \\ P_o(s\sigma) &= P_o(s)P_o(\sigma), s \in \Sigma^*, \sigma \in \Sigma. \end{aligned} \tag{2.2}$$

O operador projeção pode ser estendido para linguagens de forma natural aplicando a projeção (2.2) a todas as sequências dessa linguagem. Assim, se $L \subseteq \Sigma^*$ então

$$P_o(L) = \{t \in \Sigma_o^* : (\exists s \in L)[P_o(s) = t]\}. \tag{2.3}$$

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, a projeção pode representar 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 de uma 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:

$$\begin{array}{rcccl} P_o^{-1} & : & \Sigma_o^* & \rightarrow & 2^{\Sigma^*} \\ & & s & \mapsto & P_o^{-1}(s) = \{t \in \Sigma^* : P_o(t) = s\}. \end{array} \tag{2.4}$$

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

$$P_{o_L}^{-1}(M) = \{s \in L : (\exists y \in M)[P_o(s) = y]\}. \tag{2.5}$$

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 $P_o^{-1}[P_o(L)] \supseteq L$.

2.1.2 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

$$G = (X, \Sigma, f, \Gamma, x_0, X_m), \tag{2.6}$$

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[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 2.2 *A figura 2.1 mostra um diagrama de transição de um autômato* $G = (X, \Sigma, f, \Gamma, x_0, X_m)$ *em que* $X = \{0, 1, 2, 3\}$, $\Sigma = \{a, b\}$, $x_o = 0$ *e* $X_m = \{2, 3\}$. *A*

[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 (2007, pp. 60).

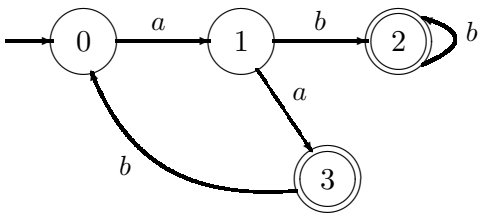


Figura 2.1: Autômato simples.

evolução dinâmica do autômato representado na figura 2.1 se dá da seguinte forma. Quando ligado, o sistema se encontra no estado $x_o = 0$. *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 em 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:

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

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 2.1, tem-se que $L = \overline{\{a\}\{aba\}^*\{b\}\{b\}^*}$ e $L_m = \{a\}\{aba\}^*\{b\}\{b\}^* \bigcup \{aa\}\{baa\}^*$.

Suponha agora 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 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:

$$G_1 \| G_2 \;=\; \mathrm{Ac}(X_1 \times X_2, \Sigma_1 \cup \Sigma_2, f_{1\|2}, \Gamma_{1\|2}, (x_{0_1}, x_{0_2}), X_{m_1} \times X_{m_2}),$$

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 (x_{0_1}, x_{0_2}) 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:

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

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 por $G_1 \times G_2$ 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 2.3 *Para ilustrar o uso da composição paralela, considere a célula de manufatura descrita no exemplo 2.1. Com o auxílio da tabela 2.1, 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 2.2.*

O modelo do sistema que considera o comportamento síncrono das máquinas

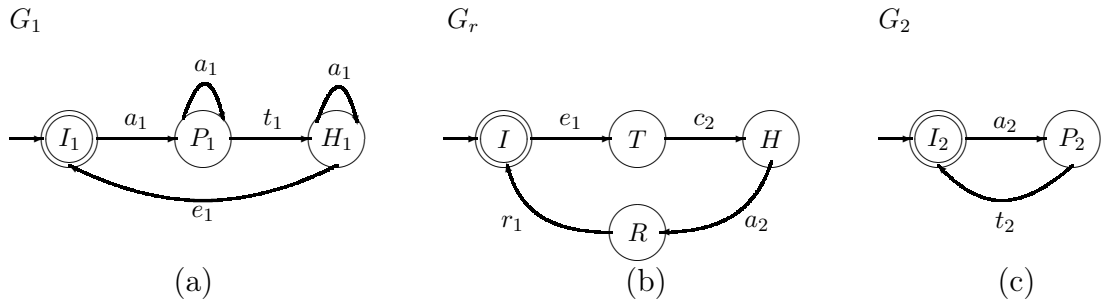


Figura 2.2: Máquina M_1 (a); Robô (b); Máquina M_2 (c).

M_1, M_2 e do robô será obtido pela composição paralela de G_1, G_2 e G_r. A figura 2.3 mostra a composição $G_r||G_2$. Note na figura 2.2 que o evento a_2 é um evento comum dos autômatos G_r e G_2. Dessa forma, esse evento somente poderá ocorrer quando

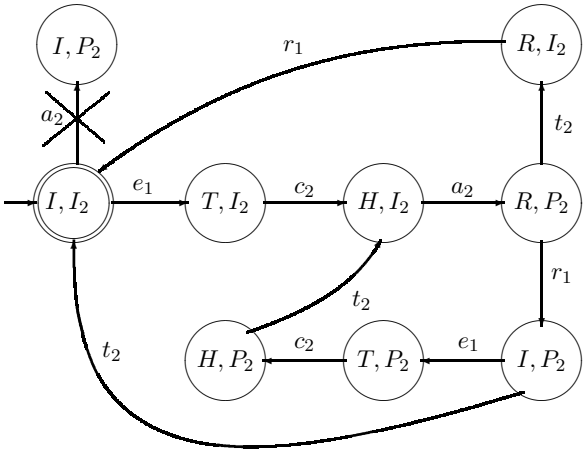


Figura 2.3: Composição síncrona de G_r e G_2.

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 2.3. 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 2.1, 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 2.3. □

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 $Obs(G, \Sigma_o)$, é definido da seguinte forma:

$$Obs(G, \Sigma_o) = (X_{obs}, \Sigma_o, f_{obs}, \Gamma_{obs}, x_{0_{obs}}, X_{m_{obs}}), \tag{2.7}$$

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 $UR(x)$):

$$UR(x) = \{y \in X : (\exists t \in \Sigma_{uo}^*)[f(x,t) = y]\}. \tag{2.8}$$

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

$$UR(B) = \bigcup_{x \in B} UR(x). \tag{2.9}$$

Usando (2.8) e (2.9), pode-se definir $x_{0_{obs}}$, Γ_{obs}, f_{obs} e X_{obs} de acordo com o algoritmo a seguir.

Algoritmo 2.1 *(Construção de observadores)*

Passo 1: *Defina* $x_{0_{obs}} = UR(x_0)$ *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* $\sigma \in \Gamma_{obs}(B)$,

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

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

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* $Obs(G, \Sigma_o)$ *tenha sido construída.*

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

A ideia do algoritmo 2.1 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 $Obs(G, \Sigma_o)$ é a projeção da linguagem de G sobre o conjunto de eventos observáveis, isto é, $L(Obs(G, \Sigma_o)) = P_o[L(G)]$.

Exemplo 2.4 *Para ilustrar a construção de observadores, considere o autômato G da figura 2.4(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*

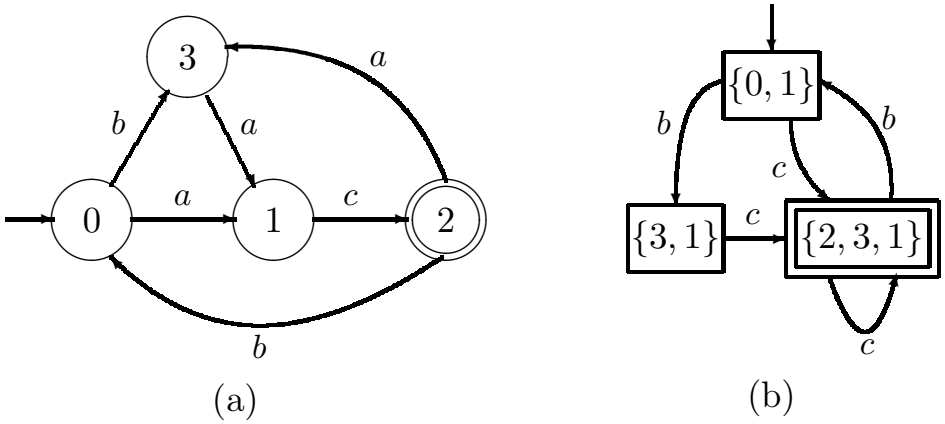


Figura 2.4: Um autômato determinístico com eventos não-observáveis (a) e o seu correspondente observador (b).

sua operação, não é possível precisar se ele ainda está no estado inicial $x_o = 0$ *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 2.4(b) ser* $\{0, 1\}$. *Caso o primeiro 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.* □

2.2 O problema da diagnose de falhas de sistemas a eventos discretos

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 cujos estados são conectados somente por eventos não-observáveis.

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 da tese, 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 do 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 2.1 *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 2.2 *Seja L uma linguagem gerada por um autômato G e suponha que L seja viva e prefixo-fechada. Então L é diagnosticável 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.

2.3 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, pode-se definir duas estruturas para a diagnose de falhas em SEDs:

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

2. Descentralizada: 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.

2.3.1 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

$$G_d = (X_d, \Sigma_o, f_d, \Gamma_d, x_{0_d}), \tag{2.10}$$

e pode ser construído em dois passos: *(i)* obtenha a composição paralela $G\|A_\ell$, sendo A_ℓ o autômato rotulador de falhas de dois estados mostrados na figura 2.5; *(ii)* calcule $Obs(G\|A_\ell, \Sigma_o)$.

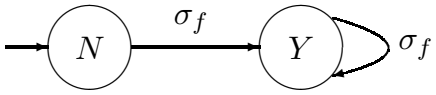


Figura 2.5: Autômato rotulador de falhas.

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 $G\|A_\ell$ são da forma (x, Y) ou (x, N), dependendo se σ_f está ou não na sequência que leva x_0 até x; consequentemente $x_d \in 2^{X\times\{N,Y\}}$. Para simplificar a notação, é usual representar os estados de G_d como xN e xY ao invés de (x, Y) e (x, N).

Exemplo 2.5 *Para ilustrar a construção de diagnosticadores, considere o autômato G cujo diagrama de transição de estados está representado na figura 2.6(a). Suponha que o conjunto de eventos observáveis seja $\Sigma_o = \{a, b, c\}$. As figuras 2.6(b) e (c) mostram, respectivamente, a composição paralela $G\|A_\ell$ e o diagnosticador $G_d = Obs(G\|A_\ell, \Sigma_o)$. Note que o estado 5 de G se divide nos estados $(5, Y)$ e $(5, N)$ de $G\|A_\ell$ 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.* □

Um diagnosticador tal como aquele representado na figura 2.6(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 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 2.6(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

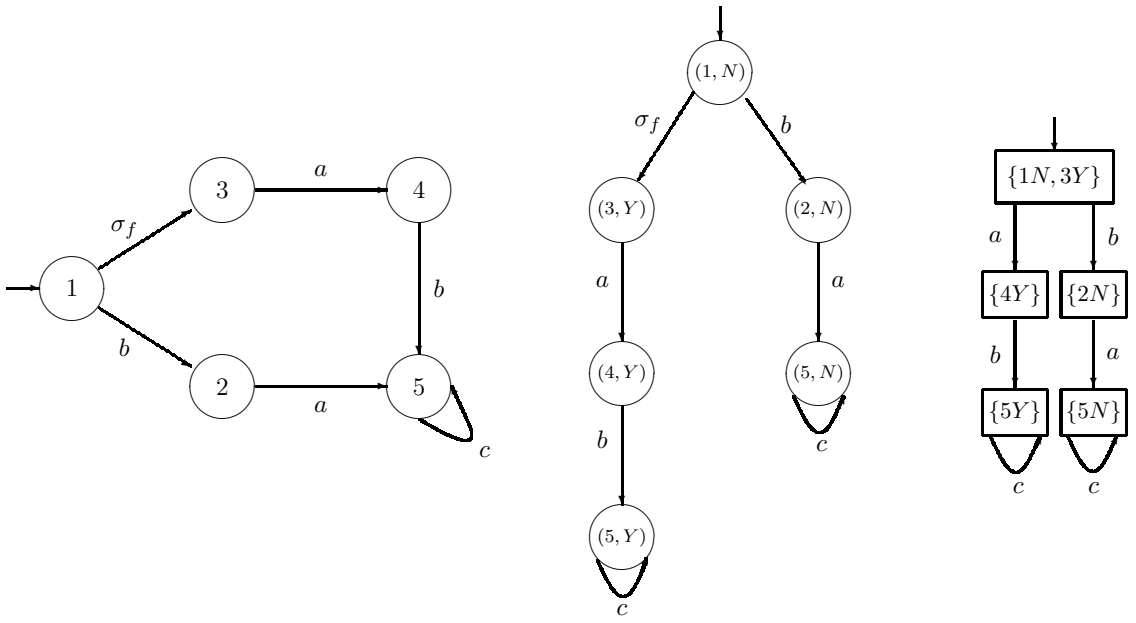


Figura 2.6: Autômato G referente ao exemplo 2.5 (a); composição paralela entre G e A_ℓ (b); $G_d = Obs(G\|A_\ell, \Sigma_o)$ (c).

sem que sua ocorrência tenha sido 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 $G_d = Obs(G\|A_\ell, \Sigma_o)$ 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 2.3 *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 2.2 e 2.3, é possível estabelecer as seguintes relações entre os estados do diagnosticador e as sequências da linguagem gerada por G (Sampath *et al.*, 1995).

Lema 2.1

(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 2.2 e do lema 2.1 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 mantenha o diagnosticador preso indefinidamente em um laço formado por estados incertos. Para explorar mais profundamente esse problema, considere as seguintes definições (Sampath *et al.*, 1995).

Definição 2.4 *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 $\{x_1, x_2, \ldots, x_n\} \subseteq X$ forma um ciclo em um autômato G se existe 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 2.5 *Um conjunto de estados incertos* $\{x_{d_1}, x_{d_2}, \ldots, x_{d_p}\} \subseteq X_d$ *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) $\exists (x_l^{k_l}, Y), (\tilde{x}_l^{r_l}, N) \in x_{d_l}$*, sendo* $x_l^{k_l}$ *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 2.2, 2.4 e 2.5 e o lema 2.1, pode-se enunciar a seguinte condição necessária e suficiente para a diagnose de uma linguagem.

Teorema 2.1 *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 2.6

a) Considere, inicialmente, o autômato da figura 2.6(a). Como o diagnosticador de G *mostrado na figura 2.6(c) não possui ciclos indeterminados, pode-se concluir que* L *é diagnosticável em relação a* P_o *e* $\Sigma_f = \{\sigma_f\}$*.*

b) Considere, agora, o autômato G *cujo diagrama de transição de estados está representado na figura 2.7(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 2.7(b). Observe que o estado* $\{4Y, 6N\}$ *é um estado incerto e como* $f_d(\{4Y, 6N\}, c) = \{4Y, 6N\}$*, então o 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 2.7(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*.* □

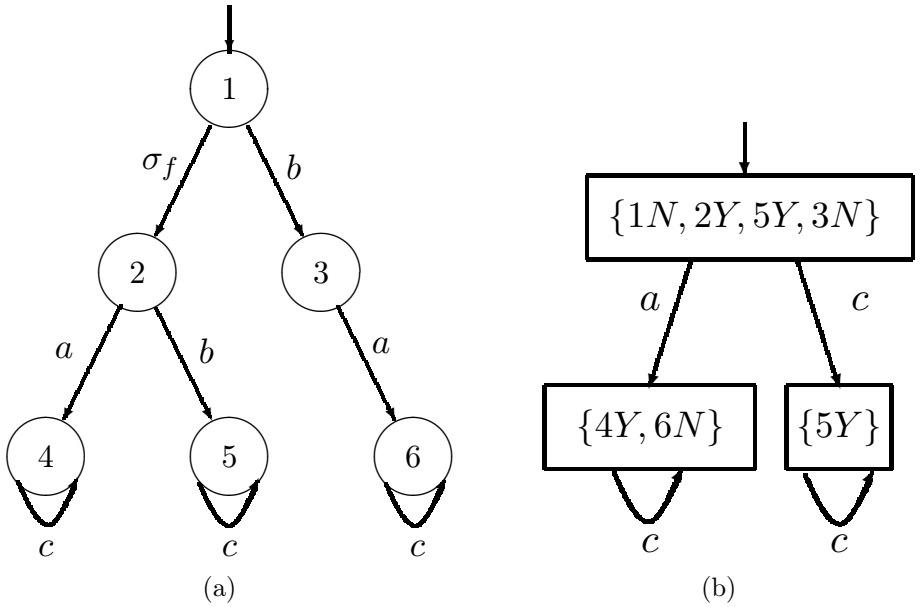


Figura 2.7: Autômato G (a); Diagnosticador centralizado para G (b).

Observação 2.1

(i) De acordo com a definição 2.2, a diagnose da linguagem gerada por G se baseia em um único conjunto de eventos observáveis, ou equivalentemente, na projeção $P_o : \Sigma^* \rightarrow \Sigma_o^*$. *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 o problema da diagnose centralizada, sendo o autômato* G_d *denominado 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 (2007, pp.114).* □

2.3.2 Diagnose centralizada sob observação parcial

O conceito de diagnose de uma linguagem apresentado na definição 2.2 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 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 $P_{o'} : \Sigma^* \to \Sigma_o'^*$, 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 $P_o : \Sigma^* \to \Sigma_o^*$ 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 2.2 *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,* $Obs(G_d, \Sigma_o') = (\hat{X}_d, \Sigma_o', \hat{f}_d, \hat{\Gamma}_d, \hat{x}_{0_d})$ *(o observador de* G_d *em relação à projeção* $P_{oo'} : \Sigma_o^* \to \Sigma_o'^*$*) e* G_d' *são idênticos a menos da seguinte relação de equivalência entre os seus estados:*

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

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

(i) $G_\ell = G \| A_\ell = (X_\ell, \Sigma, f_\ell, \Gamma_\ell, x_{0_\ell})$;

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

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

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

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 $P_{oo'} : \Sigma_o^* \to \Sigma_o'^*$ e $P_{o'} : \Sigma^* \to \Sigma_o'^*$.

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 $x_\ell \in x_{d_i}$ existe uma sequência $s \in L(G_\ell)$ tal que $P_o(s) = s_{d_i}$ e $f_\ell(x_{0_\ell}, s) = x_\ell$. 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 $x_\ell \in x'_d$. Dessa forma, para todo $x_\ell \in x_{d_i} \in \hat{x}_d$ tem-se que $x_\ell \in x'_d$ 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 $s \in L(G_\ell)$, com $s' = P_{o'}(s)$, e $x_\ell = f_\ell(x_{0_\ell}, s)$ tal que $x_\ell \in x'_d$. 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 $x_\ell \in x_{d_i} \in \hat{x}_d$. Consequentemente, $x'_d \subseteq \bigcup_{i=1}^{n} x_{d_i}$. □

Observação 2.2 *O diagnosticador G'_d será referido neste trabalho como diagnosticador centralizado com observação parcial, ou simplesmente diagnosticador parcial.* □

De acordo com o teorema 2.2, 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 $Obs(G_d, \Sigma'_o)$ e substituindo-se cada um dos estados de $Obs(G_d, \Sigma'_o)$ 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 $P_{oo'} : \Sigma_o^* \to \Sigma_o'^*$, 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. 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 (Basilio e Lafortune, 2009).

Definição 2.6 *(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 2.2, todas as sequências s tais que $\Sigma_f \in s$ devem ser diagnosticadas pelo diagnosticador parcial G'_d. Uma sequência s para a qual $\Sigma_f \in s$ que não pode ser diagnosticada pelo diagnosticador parcial é chamado de sequência ambígua.

Definição 2.7 *(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* $\omega \in L$ *tal que* $\Sigma_f \in s$*, porém* $\Sigma_f \notin \omega$*, e* $P_{o'}(s) = P_{o'}(\omega)$*.* □

O teorema a seguir provê uma condição necessária e suficiente para a diagnose de falhas sob observação parcial (Basilio e Lafortune, 2009).

Teorema 2.3 *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 2.7 *Para ilustrar o resultado do teorema 2.3, considere o autômato* $G = (X, \Sigma, f, \Gamma, x_0, X_m)$ *cujo diagrama de transição de estados está representado na figura 2.8. Suponha que* $\Sigma = \{a, b, c, d, \sigma, \sigma_f\}$, $\Sigma_o = \{a, b, c, d\}$, $\Sigma_{uo} = \{\sigma, \sigma_f\}$ *e*

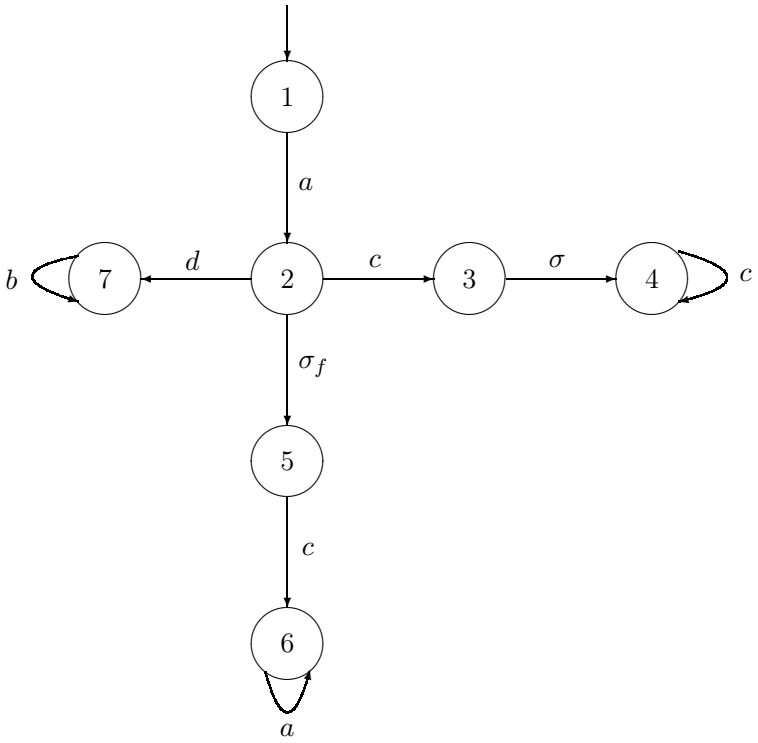


Figura 2.8: Autômato G cuja ocorrência do evento σ_f deve ser diagnosticada.

$\Sigma_f = \{\sigma_f\}$. *O diagnosticador* G_d *associado a* G *pode ser visto na figura 2.9(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* $P_{o'} : \Sigma^* \to \Sigma'^*_o$ *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 2.9(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 das sequências* $s = a\sigma_f ca^n$, $n \in \mathbb{N}$, *que contém o evento de falha* σ_f, *e que possui a mesma projeção*

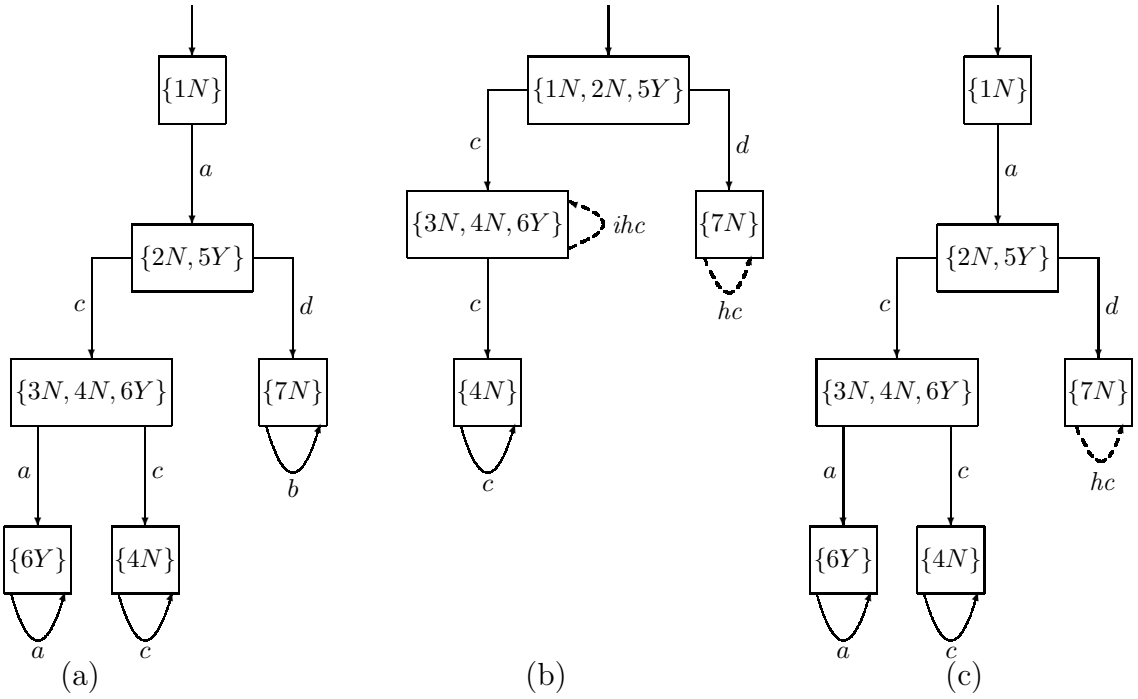


Figura 2.9: Diagnosticador G_d (a) e os diagnosticadores parciais G'_d (b) e G''_d (c) para os conjuntos de eventos observáveis $\Sigma'_o = \{c, d\}$ e $\Sigma''_o = \{a, c, d\}$, respectivamente.

em relação a $P_{o'}$ que uma sequência normal $\omega = ac$, isto é, $P_{o'}(s) = P_{o'}(\omega) = c$; consequentemente s é uma sequência ambígua $\forall n \in \mathbb{N}$. 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 $a \notin \Sigma'_o$ e 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 2.9(c). □

2.3.3 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 2.10. 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

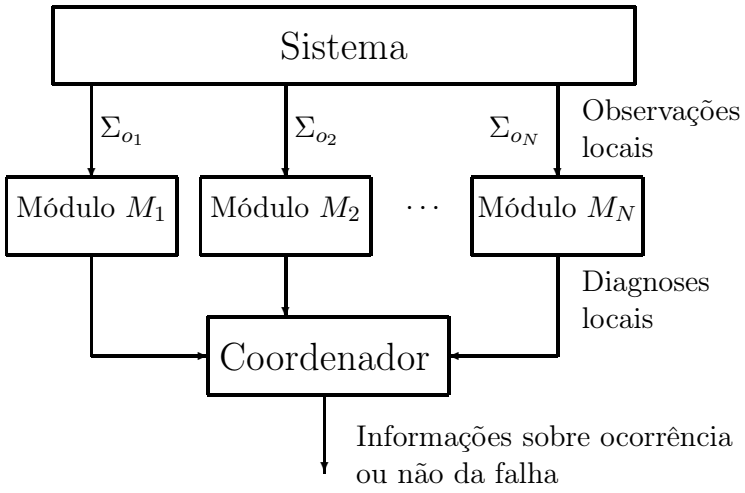


Figura 2.10: Estrutura descentralizada com coordenação.

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. Vale a pena ressaltar 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 2.10 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

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

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 coordenador 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 2.2 pode ser modificada para abranger sistemas descentralizados com coordenação, da forma apresentada a seguir (Debouk *et al.*, 2000).

Definição 2.8 *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:*

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

□

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 nesta tese foi proposto por Debouk *et al.* (2000) e consiste na implementação de um diagnosticador parcial em cada módulo, sendo o estado do diagnosticador parcial, 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 desta tese, 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 imposta: 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 ideia por trás do problema da codiagnose é que toda sequência s tal que $\Sigma_f \in s$ deve ser diagnosticada por, pelo menos, um diagnosticador parcial. Uma sequência s na qual $\Sigma_f \in s$ que não for diagnosticada por nenhum diagnosticador parcial será denominada uma sequência totalmente ambígua.

Definição 2.9 *(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 verificar 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:

$$G_{\text{test}_N} = (\|_{i=1}^{N} G_{d_i}) \| G_d. \tag{2.11}$$

Não é difícil verificar que como

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

então,

$$L(G_{\text{test}_N}) = L(G_d). \tag{2.12}$$

Essa relação 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 2.10 *Um estado x_{t_N} 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 2.11 *Um ciclo de estados incertos em G_{test_N} será indeterminado se todos os correspondentes ciclos em G_{d_i}, $i = 1, 2, \ldots, N$, forem indeterminados.* □

Observação 2.3 *Como os estados de G_{d_i}, $i = 1, 2, \ldots, N$, são formados agrupando-se os estados de G_d conectados por eventos em $\Sigma_o \setminus \Sigma_{o_i}$, e como $L(G_{test_N}) = L(G_d)$, então para qualquer estado $x_{t_N} = (x_{d_1}, x_{d_2}, \ldots, x_{d_N}, x_d)$ de G_{test_N}, $x_d \subseteq x_{d_i}$, $i = 1, 2, \ldots, N$. Dessa forma, se x_d possuir um ciclo escondido então esse ciclo também aparecerá em x_{d_i}, $i = 1, 2, \ldots, N$, embora x_{d_i}, $i = 1, 2, \ldots, N$, possa ter outros ciclos escondidos (indeterminados ou não) que não aparecem em x_d. Como consequência, se x_{t_N} for um estado certo de G_{test_N} então x_{t_N} não terá um ciclo escondido indeterminado, uma vez que pelo menos uma das componentes*

x_{d_i} de x_{t_N} será um estado certo; outros ciclos escondidos indeterminados que possam existir nas componentes incertas de x_{t_N} não impedirão a diagnose da falha. Se x_d for um estado incerto e tiver um ciclo escondido formado por estados normais então diz-se que x_{t_N} possui ciclo escondido (não indeterminado), mesmo que todas as componentes x_{d_i} possuam outros ciclos escondidos indeterminados além daquele que aparece em x_d. Assim, $x_{t_N} = (x_{d_1}, x_{d_2}, \ldots, x_{d_N}, x_d)$ terá um ciclo escondido indeterminado se e somente se x_{t_N} for um estado incerto e todas as componentes x_{d_i}, $i = 1, 2, \ldots, N$ tiverem ciclos escondidos indeterminados. □

De acordo com a definição 2.9, uma sequência totalmente ambígua é uma sequência s tal que $\Sigma_f \in s$ que leva a ciclos indeterminados em todos os diagnosticadores parciais G_{d_i}. Além disso, em face da hipótese A4, para todo s para o qual $\Sigma_f \in s$, 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 2.4 *Uma linguagem L, viva e prefixo-fechada, será codiagnosticável em relação ao conjunto de projeções $P_{o_i} : \Sigma^* \rightarrow \Sigma_{o_i}^*$, $i = 1, 2, \ldots, N$ e $\Sigma_f = \{\sigma_f\}$ se, e somente se, G_{test_N} não possuir ciclos indeterminados (incluindo ciclos indeterminados escondidos).* □

A prova apresentada em Debouk *et al.* (2000) é também válida aqui, uma vez que a definição 2.11 foi modificada para acomodar também os ciclos escondidos indeterminados.

Exemplo 2.8 *Considere o autômato G cujo diagrama de transição de estados está representado na figura 2.7(a) e 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, representado na figura 2.11, mostra que a linguagem L gerada por G é diagnosticável em relação a $P_o : \Sigma^* \rightarrow \Sigma_o^*$ e Σ_f. Suponha agora que nem todos os eventos observáveis estejam disponíveis simultaneamente*

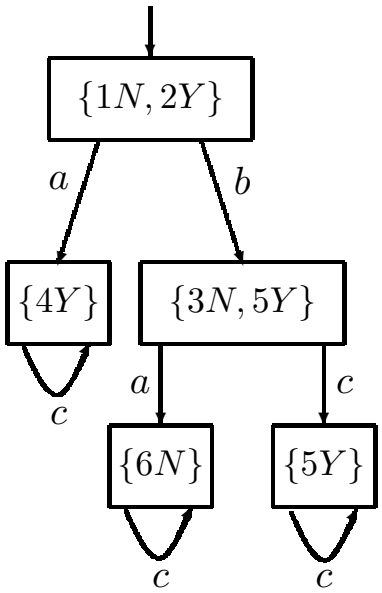


Figura 2.11: Diagnosticador centralizado para G.

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:

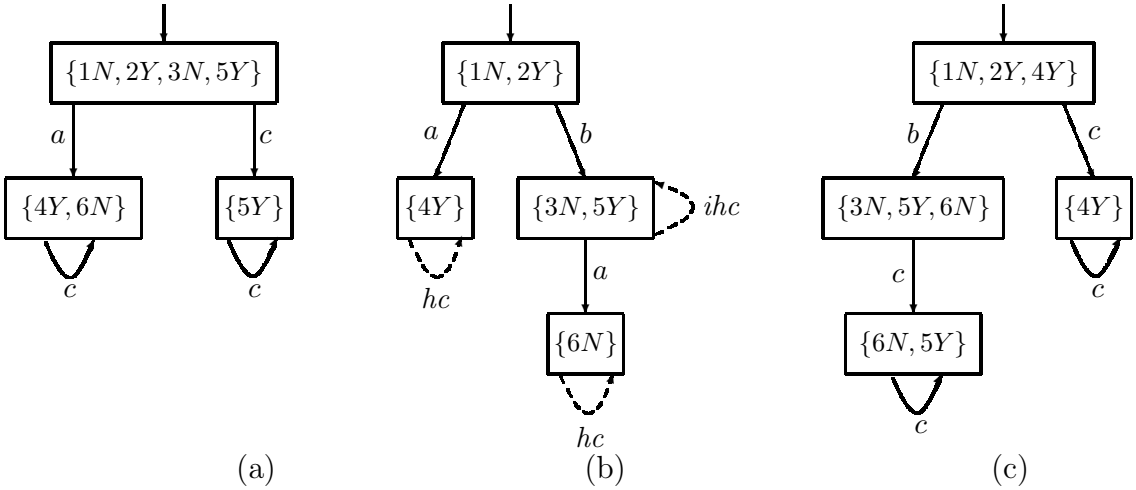


Figura 2.12: Diagnosticadores parciais para o autômato G da figura 2.7(a) para os seguintes conjuntos de eventos observáveis: (a) $\Sigma_{o_1} = \{a, c\}$; (b) $\Sigma_{o_2} = \{a, b\}$; (c) $\Sigma_{o_3} = \{b, c\}$.

1º 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 2.12(a) e (b), e o diagnosticador teste $G_{test_3} = G_{d_1} \| G_{d_2} \| G_d$ pode ser visto nas figura 2.13(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 2.13(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.

2º caso. *Suponha que os conjuntos de eventos observáveis sejam agora $\Sigma_{o_2} = \{a, b\}$ e $\Sigma_{o_3} = \{b, c\}$. Os correspondentes diagnosticadores parciais G_{d_2} e G_{d_3} estão representados nas figuras 2.12(b) e (c), e o diagnosticador teste $G'_{test_3} = G_{d_2} \| G_{d_3} \| G_d$ pode ser visto na figura 2.13(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.*

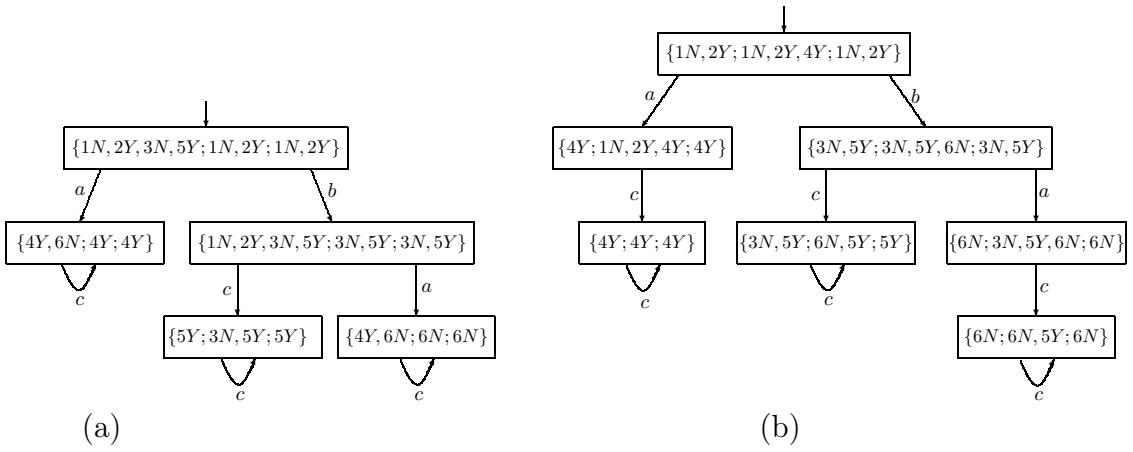


Figura 2.13: Diagnosticadores teste para o autômato G da figura 2.7(a): $G_{\text{test}_3} = G_{d_1} \| G_{d_2} \| G_d$ (a); $G'_{\text{test}_3} = G_{d_2} \| G_{d_3} \| G_d$ (b).

Na prática a diagnose de L em relação a P_{o_1}, P_{o_2} e Σ_f pode ser 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, 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. □

2.4 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,

$$|X_d| = 2^{2|X|}, \tag{2.13}$$

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) para o caso da diagnose centralizada e Qiu e Kumar (2006), Wang *et al.* (2007), Basilio e Lafortune (2009) e Moreira *et al.* (2011) para codiagnose), cujos espaços de estados dos autômatos são polinomiais em relação à cardinalidade do espaço de estados de G. O teste proposto por Moreira *et al.* (2011)

possui a menor complexidade computacional e, além disso, é consideravelmente mais simples e intuitivo que aqueles apresentados em Jiang *et al.* (2001), Yoo e Lafortune (2002), Qiu e Kumar (2006), Wang *et al.* (2007) e Basilio e Lafortune (2009). Por essas razões, será apresentado em detalhes nessa seção.

2.4.1 Verificador centralizado

Seja G o autômato determinístico da equação (2.6). O verificador centralizado proposto por Moreira *et al.* (2011) é construído de acordo com o seguinte algoritmo.

Algoritmo 2.2 *(Construção do verificador centralizado)*

Passo 1: *Construa o autômato G_N que modela o comportamento de não falha do sistema.*

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

2. Calcule o autômato de não falha $G_N := G \times A_N = (X_N, \Sigma, f_N, \Gamma_N, x_{0,N})$;

3. Refine o conjunto de eventos de G_N como $\Sigma_N = \Sigma \backslash \Sigma_f$, isto é, $G_N = (X_N, \Sigma_N, f_N, \Gamma_N, x_{0,N})$;

Passo 2: *Construa o autômato G_F que modela o comportamento de falha do sistema.*

1. Defina $A_\ell = (X_\ell, \Sigma_f, f_\ell, x_{0,\ell})$ em que $X_\ell = \{N, Y\}$, $x_{0,\ell} = \{N\}$, $f_\ell(N, \sigma_f) = Y$ e $f_\ell = (Y, \sigma_f) = Y$, para todo $\sigma_f \in \Sigma_f$;

2. Obtenha $G_\ell = G \| A_\ell$ e marque todos estados de G_ℓ cuja segunda coordenada é igual a Y;

3. Calcule o autômato de falha $G_F = CoAc(G_\ell)$;

Passo 3: *Defina a função* $R : \Sigma_N \longrightarrow \Sigma_R$ *como*[2]*:*

$$R(\sigma) = \begin{cases} \sigma & , \text{ se } \sigma \in \Sigma_o, \\ \sigma_R & , \text{ se } \sigma \in \Sigma_{uo} \backslash \Sigma_f. \end{cases} \tag{2.14}$$

Em seguida, construa o autômato $G_{N,1} = (X_N, \Sigma_R, f_{N,1}, x_{0,N})$ *com* $f_{N,1}(x_N, R(\sigma)) = f_N(x_N, \sigma)$ *para todo* $\sigma \in \Sigma_N$*.*

Passo 4: *Construa o autômato verificador* $G_V = G_{N,1} \| G_F = (X_V, \Sigma_R \cup \Sigma, f_V, x_{0,V})$*. Note que um estado de* G_V *é dado por* $x_V = (x_N, x_F)$*, em que* x_N *e* x_F *são estados de* $G_{N,1}$ *e* G_F*, respectivamente, e* $x_F = (x, x_\ell)$*, sendo* x *e* x_ℓ *estados de* G *e* A_ℓ*, respectivamente.*

Passo 5: *Verifique a existência de um ciclo*

$$cl := (x_V^k, \sigma_k, x_V^{k+1}, \sigma_{k+1}, \ldots, x_V^l, \sigma_l, x_V^k),$$

sendo $l \geq k > 0$*, em* G_V *satisfazendo às seguintes condições:*

$$\exists j \in \{k, k+1, \ldots, l\} \text{ tal que, para algum } x_V^j, (x_\ell^j = Y) \wedge (\sigma \in \Sigma).$$

□

Note que pela construção do verificador através do algoritimo 2.2 todas as sequências de G que não contêm nenhum evento de falha do conjunto Σ_f pertencem à linguagem gerada por G_N (denotada por L_N). Além disso, G_F é um subautômato de G que representa o comportamento de falha do sistema *i.e.*, $L(G_F) = L \setminus L_N$. Note ainda que a função de renomeação dada pela equação (2.14) altera o rótulo dos eventos pertencentes a $\Sigma_{uo} \setminus \Sigma_f$ para que na composição paralela do passo 4 do algoritmo 2.2, somente os eventos observáveis sejam comuns a G_N e G_F. Portanto, como só os eventos observáveis podem ocorrer simultaneamente, então a linguagem

[2]Note que a função R apenas renomeia os rótulos dos eventos em $\Sigma_{uo} \backslash \Sigma_f$. A notação $R(\Sigma_N)$ é usada neste artigo para representar a renomeação dos eventos do conjunto Σ_N como descrito em (2.14). Assim, $\Sigma_R = R(\Sigma_N)$.

gerada no verificador G_V será composta pelas sequências de $L \setminus L_N$ que possuem a mesma projeção que alguma sequência de L_N, conforme ilustrado na figura 2.14.

Figura 2.14: Busca por sequências ambíguas utilizando o verificador construído de acordo com o algoritmo 2.2.

Teorema 2.5 *(Moreira et al., 2011) Sejam L e L_N linguagens prefixo-fechadas geradas por G e G_N, respectivamente. Considere a projeção $P_o : \Sigma^* \rightarrow \Sigma_o^*$ e o conjunto de eventos de falha Σ_f. Então, L não será diagnosticável com relação a P_o e Σ_f se, e somente se, existir um ciclo em G_V, $cl := (x_V^k, \sigma_k, x_V^{k+1}, \sigma_{k+1}, \ldots, x_V^l, \sigma_l, x_V^k)$, sendo $l \geq k > 0$, que satisfaz à seguinte condição:*

$$\exists j \in \{k, k+l, \ldots, l\} \text{ tal que, para algum } x_V^j, (x_\ell^j = Y) \wedge (\sigma \in \Sigma).$$

□

O exemplo a seguir ilustra a construção de verificadores e a sua utilização na análise da diagnose de falhas de um SED.

Exemplo 2.9 *Para ilustrar a construção de verificadores, considere o autômato G da subseção 2.3.3, cujo diagrama de transição de estados está representado na figura 2.7(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 2.2, deve-se inicialmente construir o autômato A_N (representado na figura 2.15(a)) e, em seguida, calcular G_N cujo resultado está mostrado na figura 2.15(b). O próximo passo é obter o autômato G_ℓ (apresentado na fi-*

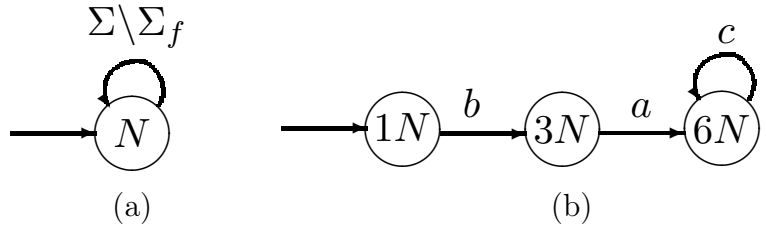


Figura 2.15: Autômato A_N (a); autômato G_N (b).

gura 2.16(b)), que é o resultado da composição síncrona entre G e A_ℓ (apresentado na figura 2.16(a)), e o autômato de falha G_F representado na figura 2.16(c), obtido tomando a parte coacessível de G_ℓ. Como $\Sigma_{uo} \backslash \Sigma_f = \emptyset$, então $\Sigma_R = \Sigma_N$, ou seja, os eventos de G_N não serão renomeados. Portanto, $G_{N,1} = G_N$. De acordo

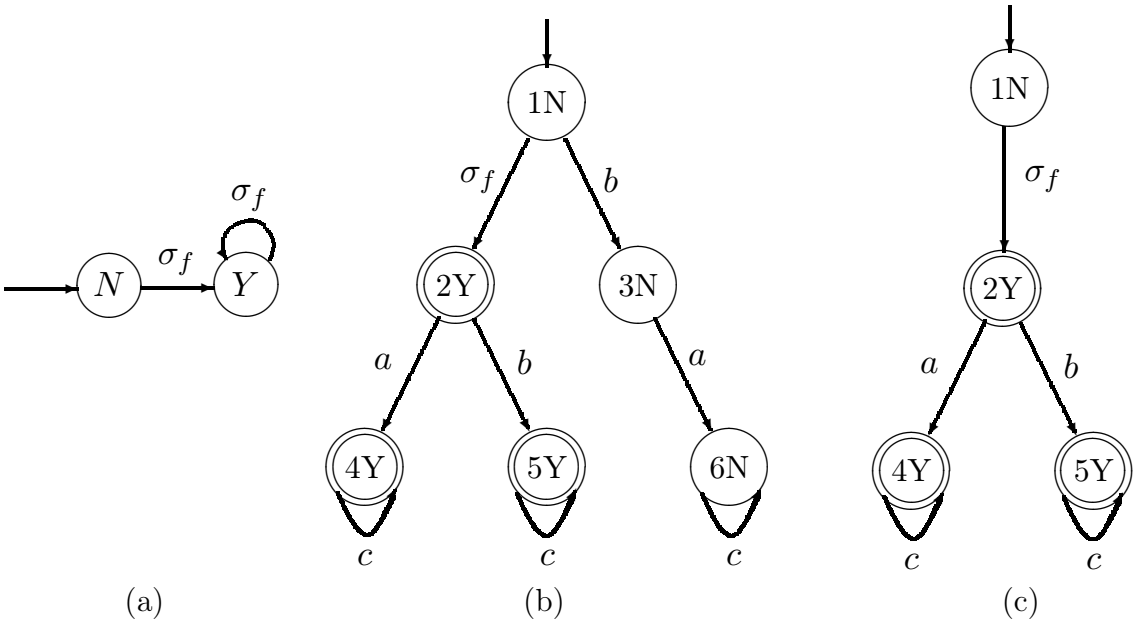


Figura 2.16: Autômato A_ℓ (a); autômato G_ℓ (b); autômato G_F (c).

com o passo 4 do algoritmo 2.2, deve-se calcular a composição paralela entre G_N e G_F para obter o verificador apresentado na figura 2.17. O estado inicial do verificador centralizado G_V tem como componentes os estados iniciais de G_N e G_F; portanto $x_{0_V} = (1N, 1N)$. Em seguida, deve-se observar a função dos eventos ativos de cada componente, isto é, $\Gamma_N(x_{0,N}) = \{b\}$ e $\Gamma_F(x_{0,F}) = \{\sigma_f\}$. Como o evento b é um evento comum a G_N e G_F e só está presente no conjunto dos eventos ativos de G_N, então o evento b não poderá ocorrer. O evento σ_f é um evento privado de G_F, e, portanto, ele pode ser executado, isto é, $\Gamma_V(x_{0,V}) = \{\sigma_f\}$. Assim, $f_V((1N, 1N), \sigma_f) = (1N, f_F(1N, \sigma_f)) = (1N, 2Y)$. No estado $x_V = (1N, 2Y)$, as

funções dos eventos ativos são $\Gamma_N(1N) = \{b\}$ *e* $\Gamma_F(2Y) = \{a, b\}$. *Como os eventos* a *e* b *são comuns a* G_N *e* G_F, *e só o evento* b *está ativo nos dois conjuntos, então somente o evento* b *poderá ocorrer no estado* $(1N, 2Y)$, *levando o verificador para o estado* $(3N, 5Y)$. *Como, as funções dos eventos ativos de* $(3N, 5Y)$ *são* $\Gamma_N(3N) = \{a\}$ *e* $\Gamma_F(5Y) = \{c\}$, *não será possível definir qualquer transição a partir desse estado já que os eventos* a *e* c *são eventos comuns a* G_N *e* G_F. *Observando a figura 2.17 é possível verificar que não existe nenhum ciclo* cl *como definido no teorema 2.5. Portanto, a linguagem* L *gerada por* G *será diagnosticável em relação a* $P_o : \Sigma^* \rightarrow \Sigma_o^*$ *e* Σ_f. *Note que este resultado coincide com o obtido na subseção 2.3.3 com o auxílio de um diagnosticador.* □

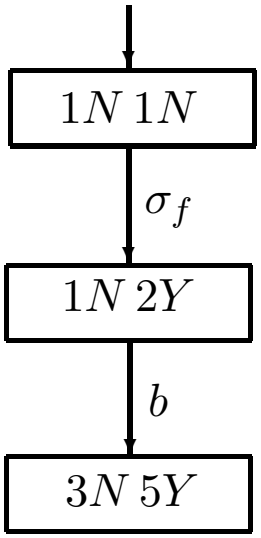


Figura 2.17: Verificador centralizado para G.

2.4.2 Verificador descentralizado

Moreira *et al.* (2011) propuseram também um verificador descentralizado construído de forma semelhante àquele obtido de acordo com o algoritmo 2.2. A ideia básica é construir um autômato de teste que identifique sequências contendo falhas e sequências que não contenham falhas mas que possuam a mesma projeção em cada módulo. Os passos 1 e 2 do algoritmo 2.2 são mantidos integralmente, enquanto os passos 3 e 4 são modificados conforme algoritmo a seguir.

Algoritmo 2.3 *(Construção do verificador descentralizado para n módulos)*

Passo 1: *Idêntico ao Passo 1 do algoritmo 2.2.*

Passo 2: *Idêntico ao Passo 2 do algoritmo 2.2.*

Passo 3: *Defina a função* $R_i : \Sigma_N \longrightarrow \Sigma_{R_i}$, $i = 1 \ldots n$, *como:*

$$R_i(\sigma) = \begin{cases} \sigma & , \text{ se } \sigma \in \Sigma_{o_i}, \\ \sigma_{R_i} & , \text{ se } \sigma \in \Sigma_{uo_i} \backslash \Sigma_f. \end{cases} \tag{2.15}$$

Em seguida, construa o autômato $G_{N,i} = (X_N, \Sigma_{R_i}, f_{N,i}, x_{0,N})$, *para* $i = 1 \ldots n$, *com* $f_{N,i}(x_N, R_i(\sigma)) = f_N(x_N, \sigma)$ *para todo* $\sigma \in \Sigma_N$.

Passo 4: *Construa o autômato verificador* $G_V = G_{N,1} \| \ldots \| G_{N,n} \| G_F = (X_V, \bigcup_{i=1}^{n} \Sigma_{R_i} \cup \Sigma, f_V, x_{0,V})$. *Note que um estado de* G_V *é dado por* $x_V = (x_{N,1}, x_{N,2}, \ldots x_{N,n}, x_F)$, *em que* $x_{N,i}$ *é um estado de* $G_{N,i}$, *para* $i = 1 \ldots n$, *e* x_F *é um estado de* G_F, *e* $x_F = (x, x_\ell)$, *sendo* x *and* x_ℓ *estados de* G *e* A_ℓ, *respectivamente.*

Passo 5: *Verifique a existência de um ciclo*

$$cl := (x_V^k, \sigma_k, x_V^{k+1}, \sigma_{k+1}, \ldots, x_V^l, \sigma_l, x_V^k),$$

em G_V, *sendo* $l \geq k > 0$, *que satisfaça às seguintes condições:*

$$\exists j \in \{k, k+1, \ldots, l\} \text{ tal que, para algum } x_V^j, (x_\ell^j = Y) \wedge (\sigma \in \Sigma).$$

□

Assim como no verificador centralizado, a presença de ciclos *cl* no verificador levam à violação da diagnose conforme mostrado no teorema 2.5.

O exemplo a seguir ilustra a construção e utilização do verificador na análise da codiagnose de falha de SEDs.

Exemplo 2.10 *Considere novamente o autômato G da figura 2.7(a) e suponha que os conjuntos de eventos observáveis de cada módulo sejam $\Sigma_{o_1} = \{a, b\}$ e $\Sigma_{o_2} = \{b, c\}$. Os passos 1 e 2 do algoritmo 2.2 são mantidos no caso descentralizado, e, portanto, G_N e G_F são aqueles autômatos cujos diagramas de transição estão representados nas figuras 2.15(b) e 2.16(c). De acordo com o algoritmo 2.3 e lembrando que existem dois módulos ($n = 2$), obtém-se os autômatos $G_{N,1}$ e $G_{N,2}$ (mostrados nas figuras 2.18(a) e (b)) a partir de G_N pela renomeação dos eventos não observáveis nos conjuntos $\Sigma_{uo_1} \backslash \Sigma_f = \{c\}$ e $\Sigma_{uo_2} \backslash \Sigma_f = \{a\}$, respectivamente. Assim, os conjuntos de eventos de $G_{N,1}$ e $G_{N,2}$ são $\Sigma_{R_1} = \{a, b, c_{R_1}\}$ e $\Sigma_{R_2} = \{a_{R_2}, b, c\}$, respectivamente. Com os autômatos $G_{N,1}$, $G_{N,2}$ e G_F, pode-se obter o verificador des-*

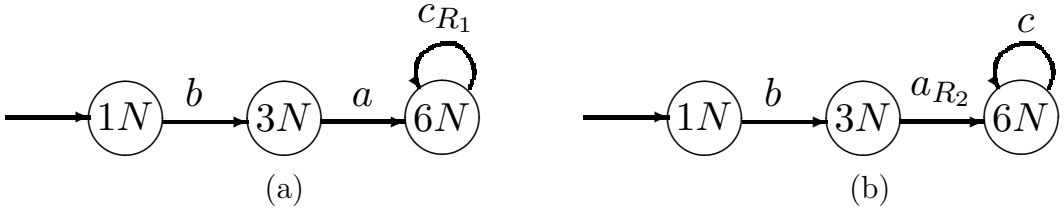


Figura 2.18: Autômato $G_{N,1}$ (a); autômato $G_{N,2}$ (b).

centralizado G_V apresentado na figura 2.19. O estado inicial de G_V tem como componentes os estados iniciais de $G_{N,1}$, $G_{N,2}$ e G_F, e, é portanto $x_{0,V} = (1N, 1N, 1N)$. Para realizar a composição paralela do passo 4 no algoritmo 2.3, deve-se observar a função dos eventos ativos de cada componentes $\Gamma_{N,1}(1N) = \Gamma_{N,2}(1N) = \{b\}$ e $\Gamma_F(1N) = \{\sigma_f\}$, e em seguida, analisar se os eventos são comuns ou privados. Assim, como o evento b é um evento comum a $G_{N,1}$, $G_{N,2}$ e G_F, e não está presente no conjunto de eventos ativos de cada componente, em especial no conjunto de eventos ativos do estado inicial de G_F, então ele não poderá ocorrer. Todavia, como σ_f é um evento privado de G_F, ele ocorrerá, ou seja, $\Gamma_V(x_{0,V}) = \{\sigma_f\}$. Assim, $f_V((1N, 1N, 1N), \sigma_f) = (1N, 1N, f_F(1N, \sigma_f)) = (1N, 1N, 2Y)$. Para o estado $x_V = (1N, 1N, 2Y)$, tem-se que $\Gamma_{N,1}(1N) = \Gamma_{N,2}(1N) = \{b\}$ e $\Gamma_F(2Y) = \{a, b\}$. Como o evento a é um evento comum a $G_{N,2}$ e G_F e não está ativo em $G_{N,1}$, ele não poderá ocorrer. Portanto, o único evento que poderá ocorrer no estado $(1N, 1N, 2Y)$ é evento b, levando o verificador para o estado $(3N, 3N, 5Y)$. Como

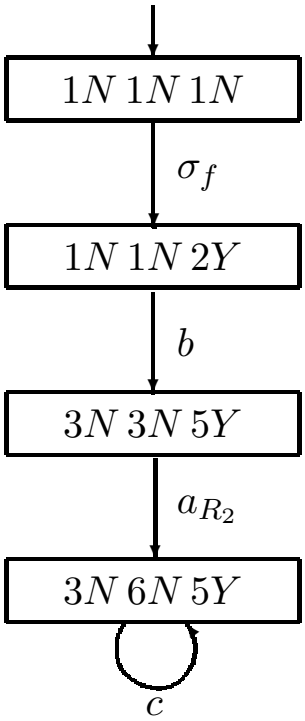


Figura 2.19: Verificador descentralizado para G.

$\Gamma_{N,1}(3N) = \{a\}$, $\Gamma_{N,2}(3N) = \{a_{R_2}\}$ *e* $\Gamma_F(5Y) = \{c\}$, *e* a_{R_2} *é um evento privado de* $G_{N,2}$, *então* $f_V((3N, 3N, 5Y), a_{R_1}) = (3N, 6N, 5Y)$. *Note que como* $\Gamma_{N,1}(3N) = \{a\}$, $\Gamma_{N,2}(6N) = \{c\}$ *e* $\Gamma_F(5Y) = \{c\}$, *e uma vez que o evento* c *é comum a* $G_{N,2}$ *e* G_F *e não é um evento de* $G_{N,1}$, *então* $\Gamma_V((3N, 6N, 5Y)) = \{c\}$ *gerando um autolaço no estado* $(3N, 6N, 5Y)$.

Note que o estado $(3N, 6N, 5Y)$ *de* G_V *tem um ciclo* cl *contendo* $x_F = 5Y$ *e* $\sigma_{V_i} = c \in \Sigma$. *Dessa forma, pode-se concluir que* L *não é diagnosticável em relação a* $P_{o_i} : \Sigma^* \to \Sigma_{o_i}$, $i = 1, 2$ *e* Σ_f. *Isso se deve ao fato da sequência arbitrariamente longa* $s = \sigma_f bc^n$ *poder ser confundida com a sequência* bac^n, *que não contêm* σ_f, *quando projetadas por* $P_{o_2}(s)$. *Dessa forma,* s *é uma sequência totalmente ambígua.*

□

2.5 Comentários finais

O objetivo principal deste capítulo 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 a tese autocontida. Para aqueles que não são familiarizados com a teoria de SEDs, sugere-se consultar Cassandras e Lafortune (2007) e Hopcroft *et al.* (2007).

Capítulo 3

Diagnose robusta a perdas permanentes de sensores

A verificação da diagnosticabilidade da linguagem gerada por um sistema a eventos discretos pode ser realizada utilizando-se diagnosticadores ou verificadores. A decisão se uma linguagem é diagnosticável utilizando o diagnosticador ou o verificador será tomada com base unicamente nos eventos observáveis, isto é, os eventos cujas ocorrências podem ser registradas por meio de sensores. Contudo, se algum sensor que fornece a informação sobre a ocorrência de um evento falhar, a nova linguagem observada do sistema pode não mais ser diagnosticável.

Conforme visto no capítulo anterior, diagnosticadores são autômatos determinísticos cujos estados são conjuntos formados pelos estados da planta acrescido de rótulos que indicam se o traço observado até o momento possui ou não o evento de falha. Portanto, uma vez que o conjunto dos eventos ativos de um estado do diagnosticador é formado pelos próximos eventos observáveis dos estados da planta que aparecem no estado atual do diagnosticador, então, quando um sensor falhar, o diagnosticador pode estacionar em um determinado estado ou, até mesmo, informar de maneira errônea a ocorrência da falha.

Para superar essas deficiências, Lima *et al.* (2010) propuseram um diagnosticador robusto a perdas permanentes nos sensores, isto é, quando o sensor responsável pelo

registro da ocorrência do evento após falhar não mais se recupera da falha. Para a construção desse diagnosticador, foram utilizadas as redundâncias que possam existir em uma base para a diagnose (Basilio *et al.*, 2011) — conjunto de eventos que garantem a diagnosticabilidade de uma falha — visando garantir a diagnose da falha, mesmo quando da perda permanente de sensores . Entretanto, essa abordagem utiliza diagnosticadores que tem o problema da complexidade computacional ser da ordem exponencial em relação a cardinalidade do espaço de estados. Portanto, nesse capítulo será proposto uma nova forma de se verificar a diagnose robusta a falhas permanentes utilizando verificadores sendo o verificador aqui proposto baseado em Moreira *et al.* (2011).

Esse capítulo está estruturado da seguinte forma. Na seção 3.1 são apresentadas a definição de diagnosticabilidade robusta a perdas permanentes de sensores e as propriedades desejadas para um diagnosticador robusto. Na seção 3.2 é proposto um diagnosticador robusto e é apresentada uma condição necessária e suficiente para a diagnosticabilidade robusta expressa em termos de ciclos do diagnosticador. Além disso é apresentado um exemplo para ilustrar os algoritmos e os conceitos introduzidos na seção. Na seção 3.3 é proposto um verificador robusto e é apresentada uma condição necessária e suficiente para a diagnose robusta em termos da existência de determinados ciclos no verificador. Nessa seção também é determinada a complexidade computacional do algoritmo que fornece os passos para a construção do verificador e é apresentado um exemplo que ilustra a construção do verificador e sua utilização na análise da diagnose robusta a perdas permanentes de sensores. Finalmente, conclusões são apresentadas na seção 3.4.

3.1 Diagnosticabilidade robusta em relação a perdas permanentes de sensores

Quando falhas nos sensores que detectam a ocorrência de eventos observáveis ocorrem, o diagnosticador pode ficar estacionado em um estado, sem evoluir ao longo do

processo de observação e registro de eventos, ou, então evoluir de maneira errônea e, assim, informar de maneira equivocada a ocorrência ou não da falha. Para ilustrar esse problema, considere o autômato da figura 3.1(a) cujo diagnosticador considerando $\Sigma_o = \{a, b, c, d, e\}$ está representado na figura 3.1(b). Suponha que tenha ocorrido uma perda definitiva do sensor que registra a ocorrência do evento c antes da primeira vez que esse evento ocorreu. Suponha que a sequência $s_Y = c\,\sigma_f a e^n$, $n \in \mathbb{N}$, tenha ocorrido. Note que como o evento σ_f é não-observável e a ocorrência

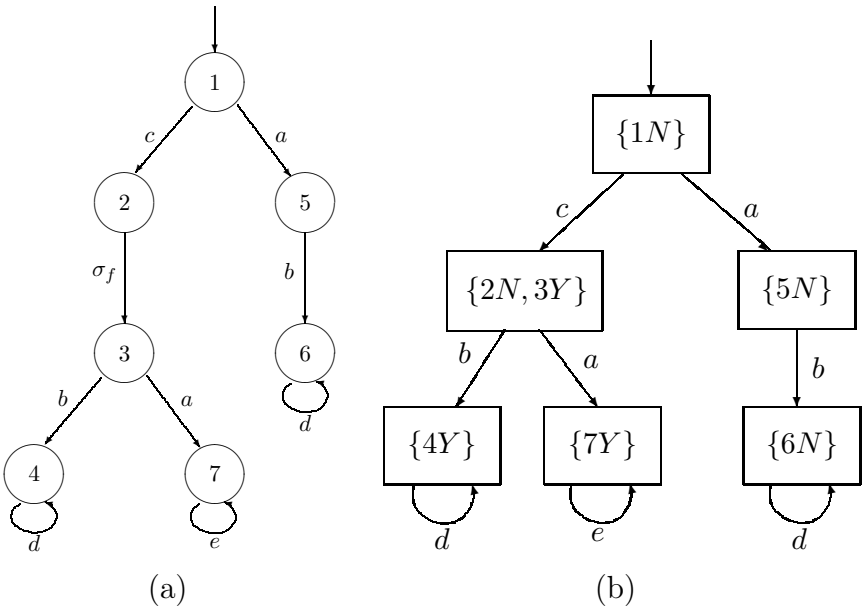


Figura 3.1: Autômato G (a); Diagnosticador centralizado G_d (b).

de c não foi registrada pelo sensor, então o primeiro evento a ter a sua ocorrência reconhecida pelo diagnosticador da figura 3.1(b) é o evento a. Recebida a informação da ocorrência do evento a, o diagnosticador irá atualizar o seu estado, passando, então, ao estado $\{5N\}$. Como o evento e é o próximo evento a ocorrer na sequência s_Y e ele não pertence ao conjunto dos eventos ativos de $\{5N\}$, o diagnosticador permanecerá nesse estado. Dessa forma, o diagnosticador estará certo que a falha não ocorreu, o que é incorreto, uma vez que a sequência que ocorreu, além de possuir um evento de falha, é infinitamente longa. Esse comportamento incorreto do diagnosticador em presença da perda permanente do sensor associado ao evento c sugere que o diagnosticador deve ser modificado para tolerar possíveis perdas de sensores.

Isso leva à formulação do seguinte problema: dado um autômato $G =$

$(X, \Sigma, f, \Gamma, x_0)$ e supondo que L seja diagnosticável em relação a $P_o : \Sigma^* \rightarrow \Sigma_o^*$ (Σ_o conjunto de eventos observáveis de G) e Σ_f, encontre um diagnosticador que seja robusto à perda do maior número possível de sensores associados a conjuntos de eventos pertencentes ao conjunto potência de Σ_o.

Considere a seguinte hipótese.

A7. A perda do sensor, quando ocorrer, dar-se-á antes da primeira ocorrência do evento a ele associado e é permanente, *i.e.*, o sensor não mais se recupera.

Além disso, suponha que, de acordo com a hipótese A4, L é diagnosticável com relação a $P_o : \Sigma^* \rightarrow \Sigma_o^*$ e $\Sigma_f = \{\sigma_f\}$. O problema aqui formulado aborda os chamados sistemas cíclicos, que reiniciam constantemente. Nesse caso a perda do sensor não pode ser considerada em qualquer instante, mas somente quando o sistema é ligado.

Pela hipótese A7, observa-se a necessidade de se buscar um diagnosticador que seja robusto a perdas de sensores, ou seja, um diagnosticador que continue diagnosticando L mesmo que ocorram perdas de sensores. Antes de apresentar esse novo diagnosticador, as seguintes definições são apresentadas (Basilio *et al.*, 2011).

Definição 3.1 *(Bases para a diagnose) Um conjunto $\Sigma'_o \subseteq \Sigma_o$ é uma base para a diagnose de falhas se L for diagnosticável com relação à projeção $P_{o'} : \Sigma^* \rightarrow \Sigma_o'^*$ e $\Sigma_f = \{\sigma_f\}$.* □

Definição 3.2 *(Bases mínimas para a diagnose) Um conjunto $\Sigma'_o \subseteq \Sigma_o$ será uma base mínima para a diagnose de falhas se Σ'_o é uma base para a diagnose de falhas, e se para todo subconjunto não vazio Σ''_o de Σ'_o, L não for diagnosticável com relação à projeção $P_{o''} : \Sigma^* \rightarrow \Sigma_o''^*$ e $\Sigma_f = \{\sigma_f\}$.* □

De acordo com as definições 3.1 e 3.2, a principal diferença entre uma base para diagnose e uma base mínima é que se um evento for retirado da base mínima perde-se a diagnosticabilidade. Uma base para a diangose pode ter eventos redundantes, ou seja, nem todos os eventos são necessários para manter diagnosticabilidade.

Será agora apresentada a definição de diagnosticabilidade robusta com relação a perdas permanentes de sensores (Lima *et al.*, 2010).

Definição 3.3 *(Diagnosticabilidade robusta de SEDs sujeitos a perdas permanentes de sensores) Considere* $\Sigma_{db} = \{\Sigma_{o_1}, \Sigma_{o_2}, \ldots, \Sigma_{o_m}\}$ *no qual* Σ_{o_i}*, para* $i = 1, \ldots, m$*, são bases mínimas ou não mínimas para a diagnose de* L*. Defina o conjunto*

$$\Sigma_{rob} = \{\Sigma_{uo_1}, \Sigma_{uo_2}, \ldots, \Sigma_{uo_m}\} \tag{3.1}$$

em que $\Sigma_{uo_i} = \Sigma_o \backslash \Sigma_{o_i}$*, para* $i = 1, \ldots, m$*. Então,* L *é robustamente diagnosticável a perdas permanentes de sensores associados aos elementos do conjunto* Σ_{rob}*, com relação às projeções* $P_{o_1}, P_{o_2}, \ldots, P_{o_m}$*, em que* $P_{o_i} : \Sigma^* \longrightarrow \Sigma_{o_i}^*$*, e* $\Sigma_f = \{\sigma_f\}$*, se a seguinte condição for verificada:*

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

sendo a condição de diagnosticabilidade D_p *expressa por*

$$(\forall i, j \in \{1, 2, \ldots, m\}, i \neq j)(\nexists \omega_j \in L)[\Sigma_f \notin \omega_j \wedge P_{o_i}(st) = P_{o_j}(\omega_j)]$$

□

O diagnosticador que é capaz de diagnosticar uma falha e satisfazer as condições da definição 3.3 será chamado de diagnosticador robusto a perdas permanentes de sensores ou simplesmente diagnosticador robusto.

A ideia por trás da definição 3.3 é que como L é diagnosticável com relação a $P_{o_i} : \Sigma^* \longrightarrow \Sigma_{o_i}^*$, e $\Sigma_f = \{\sigma_f\}$, e supondo que todos diagnosticadores parciais para Σ_{o_i}, $i = 1, 2, \ldots, m$, sejam executados simultaneamente, e têm acesso a todos os sensores disponíveis, então quando os eventos de Σ_{uo_k} se tornarem não observáveis, o diagnosticador parcial cujos eventos observáveis são elementos de Σ_{o_k}, realizará a diagnose de falhas, uma vez que Σ_{o_k} é, por hipótese, uma base para a diagnose. Suponha, contudo, que existam duas sequências de L, uma de comprimento arbitrariamente longo s_Y que contém o evento de falha e uma outra sequência s_N que não contenha a falha, e suponha que s_Y tenha a mesma projeção em $\Sigma_{o_k}^*$ que s_N em $\Sigma_{o_j}^*$,

para $k \neq j$. Nesse caso, L não é robustamente diagnosticável com relação a perdas de sensores associados aos eventos em Σ_{uo_k} e Σ_{uo_j}, uma vez que o diagnosticador, quando da ocorrência de s_Y, permanecerá indefinidamente em dúvida com relação à ocorrência de falha.

Nesse contexto, as propriedades desejadas para o diagnosticador robusto podem ser enunciadas.

P1. O diagnosticador robusto deve conter o maior número possível de diagnosticadores parciais cujos eventos observáveis são as bases (mínimas e não-mínimas) para a diagnose de falhas do SED considerado;

P2. A linguagem gerada pelo diagnosticador robusto deve ser a união das linguagens geradas pelos diagnosticadores parciais;

P3. O diagnosticador robusto deve manter as marcações Y e N dos estados dos diagnosticadores parciais;

P4. Os estados do diagnosticador robusto devem ter marcações que identifiquem quais sensores falharam e de quais diagnosticadores parciais elas provêm.

3.2 Diagnosticador robusto

O primeiro passo para se obter um diagnosticador robusto que satisfaça as propriedades P3 e P4 deve ser a construção dos diagnosticadores parciais cujos conjuntos de eventos observáveis são bases para a diagnose de falha, e incluir as marcações nos estados que indicam a perdas dos sensores. Suponha que $\Sigma'_o \subset \Sigma_o$ seja uma base para a diagnose e seja $\Sigma'_{uo} = \Sigma_o \backslash \Sigma'_o = \{\sigma'_1, \sigma'_2, \ldots, \sigma'_p\}$.

Definição 3.4 *Suponha que $\bar{\sigma}_i$ e σ_i, $i = 1, 2, \ldots, p$, denotem, respectivamente, a não-ocorrência e a ocorrência do evento σ_i que representa a perda do sensor associado a ele. O conjunto de marcações das perdas dos sensores é definido como:*

$$M = \{S_m : m \in \{\sigma'_1, \bar{\sigma}'_1\} \times \{\sigma'_2, \bar{\sigma}'_2\} \times \cdots \times \{\sigma'_p, \bar{\sigma}'_p\}\}.$$

□

Definição 3.5 *A função de propagação de marcação de perdas de sensores é* $S : X_d \times M \times \Sigma_o \rightarrow M$. *Para* $x_d \in X_d$, $S_m \in M$ *e* $\sigma \in \Gamma(x_d)$, *então*

$$S(x_d, S_m, \sigma) = S_{m'},$$

sendo

$$m' = \begin{cases} m, \ \textit{se } \sigma \notin \Sigma'_{uo} \\ \sigma'_1, \sigma'_2, \ldots, \sigma'_{k-1}, \sigma, \sigma'_{k+1}, \ldots, \sigma'_p, \ \textit{se } \sigma \in \Sigma'_{uo} \wedge (\sigma_k = \sigma \vee \sigma_k = \bar{\sigma}) \end{cases} . \quad (3.2)$$

□

Note que a marcação de perdas de sensores nos estados dos diagnosticadores parciais indica se um estado do diagnosticador foi alcançado através de uma sequência que não contém o evento cujo sensor está sujeito à perda permanente ou se houve a perda do sensor que registra a ocorrência do evento considerado.

Definição 3.6

A. *O diagnosticador com marcação de perda de sensores é definido como:*

$$\tilde{G}_d(\Sigma_o \backslash \Sigma'_o) = (\tilde{X}_d, \Sigma_o, \tilde{f}_d, \tilde{\Gamma}_d, \tilde{x}_{0,d}),$$

em que $\tilde{X}_d \subseteq X_d \times M, \tilde{x}_{0,d} = x_{0,d} S_{\bar{\sigma}'_1, \bar{\sigma}'_2, \ldots, \bar{\sigma}'_p}$, *e* $\tilde{f}_d$ *e* $\tilde{\Gamma}_d$ *são definidos como se segue: para* $\tilde{x}_d = x_d S_m$ *e supondo que* $f_d(x_d, \sigma) = x'_d$, *então* $\tilde{\Gamma}_d(\tilde{x}_d) = \Gamma_d(x_d)$ *e* $\tilde{f}_d(\tilde{x}_d, \sigma) = x'_d S_{m'}$, *sendo* $S_{m'} = S(x_d, S_m, \sigma)$.

B. *O diagnosticador em que os sensores têm um comportamento normal em todos os estados receberão a marcação* S_n, *e é o caso quando* $\Sigma'_o = \Sigma_o$. *Nesse caso, esse diagnosticador será denotado por* $\tilde{G}_d(\emptyset)$. □

De acordo com a definição 3.6, a linguagem gerada por $\tilde{G}_d(\Sigma_o \backslash \Sigma'_o)$ não considera a perda de observabilidade dos eventos pertencentes a $\Sigma_o \backslash \Sigma'_o$, ou seja, $L[\tilde{G}_d(\Sigma_o \backslash \Sigma'_o)] = L(G_d)$, sendo somente construído para identificar as marcações de perdas de sensores. Feito isso, o próximo passo é obter o diagnosticador parcial supondo Σ'_o como

o conjunto de eventos observáveis. O algoritmo a seguir apresenta uma forma sistemática de se obter o diagnosticador parcial com marcação de perdas de sensores, denotado por $\tilde{G}'_d(\Sigma_o \backslash \Sigma'_o)$ a ser utilizado na construção do diagnosticador robusto.

Algoritmo 3.1 *(Cálculo do diagnosticador parcial com marcação de perdas de sensores) Seja Σ_o o conjunto de eventos observáveis e suponha que $\Sigma'_o \subset \Sigma_o$ seja uma base para a diagnose de L.*

Passo 1 *Construa o diagnosticador com marcação de perdas de sensores $\tilde{G}_d(\Sigma_o \backslash \Sigma'_o)$.*

Passo 2 *Calcule o observador de $\tilde{G}_d(\Sigma_o \backslash \Sigma'_o)$ supondo Σ'_o como o conjunto de eventos observáveis, ou seja, $Obs[\tilde{G}_d(\Sigma_o \backslash \Sigma'_o), \Sigma'_o]$.*

Passo 3 *Forme cada estado de $\tilde{G}'_d(\Sigma_o \backslash \Sigma'_o)$ calculando-se a união do conjunto de elementos de cada estado de $Obs[\tilde{G}_d(\Sigma_o \backslash \Sigma'_o), \Sigma'_o]$.*

□

Suponha, agora, que $\Sigma_{mb,i} \subset \Sigma'_o$, $i = 1, 2, \ldots, N_b$ sejam todas as bases mínimas para a diagnose de L (Basilio *et al.*, 2011). Defina o conjunto

$$\Sigma_{red,i} = \Sigma_o \backslash \Sigma_{mb,i},\ i = 1, 2, \ldots, N_b.$$

Então, o conjunto $\Sigma_{db,max}$ que contém todas as bases mínimas e não-mínimas para a diagnose de L pode ser formado da seguinte forma:

$$\Sigma_{db,max} = \bigcup_{i=1}^{N_b} \Sigma_{db,i}\ ,$$

em que

$$\Sigma_{db,i} = \{\tilde{\Sigma} : (\exists \Sigma_{pow} \in 2^{\Sigma_{red,i}})[\tilde{\Sigma} = \Sigma_{mb,i} \cup \Sigma_{pow}]\},$$

sendo $2^{\Sigma_{red,i}}$ o conjunto potência de $\Sigma_{red,i}$

Tendo sido obtidos o diagnosticador centralizado com marcação de perdas de sensores e todos os diagnosticadores parciais com marcação de perdas de sensores

associados a $\Sigma_{db,max}$, então, a fim de satisfazer as propriedades P1 e P2, o próximo passo é construir o diagnosticador cuja linguagem gerada seja a união das linguagens geradas pelos diagnosticadores centralizado e parciais com marcação de perdas de sensores.

Definição 3.7 *(Diagnosticador união) Seja*

$$\Sigma_{db,max} = \{\Sigma_{o_1}, \Sigma_{o_2}, \ldots, \Sigma_{o_q}, \Sigma_o\}$$

o conjuntos de todas as bases (mínimas e não mínimas) para a diagnose de L e denote $\tilde{G}'_{d_i}$, $i = 1, \ldots, q$, os diagnosticadores parciais com marcações de perdas de sensores e $\tilde{G}_{d_0}$ o diagnosticador centralizado com marcações de perdas de sensores. O diagnosticador união, $G_{du}(\Sigma_{db,max})$, é o diagnosticador cuja linguagem gerada é a união das linguagens geradas pelos diagnosticadores $\tilde{G}'_{d_i}$, para $i = 1, \ldots, q$, e $\tilde{G}_{d_0}$. □

A construção do diagnosticador união pode ser feita da seguinte forma (Cassandras e Lafortune, 2007, pp. 94): (i) crie um novo estado inicial e adicione uma transição-ε desse novo estado para todos os estados iniciais de $\tilde{G}_{d_i}$, para $i = 0, \ldots, q$, obtendo um autômato não determinístico; (ii) calcule o observador desse autômato não determinístico com relação a Σ_o, obtendo, então, um autômato determinístico.

Para ilustrar a construção do diagnosticador união considere o seguinte exemplo.

Exemplo 3.1 *Considere novamente o autômato G da figura 3.1(a) cujo conjunto de eventos observáveis é $\Sigma_o = \{a, b, c, d, e\}$. O correspondente diagnosticador centralizado está mostrado na figura 3.1(b). De acordo com Basilio et al. (2011), as bases mínimas para a diagnose de $L(G)$ são:*

$$\Sigma_{mdb} = \{\{a, b, c\}, \{c, d, e\}, \{a, c, d\}, \{a, d, e\}, \{a, b, e\}, \{b, c, e\}\}.$$

As bases não-mínimas para a diagnose de $L(G)$ podem ser encontradas

acrescentando-se às bases mínimas os eventos que não pertencem a elas, mas que estão no conjunto de eventos observáveis do autômato. O seguinte conjunto é, então, formado:

$$\Sigma_{nmdb} = \{\{a, b, c, d\}, \{a, b, c, e\}, \{a, b, d, e\}, \{a, c, d, e\}, \{b, c, d, e\}, \Sigma_o\}.$$

Portanto,

$$\Sigma_{db,max} = \Sigma_{mdb} \cup \Sigma_{nmdb}.$$

O primeiro passo para se obter o diagnosticador união é construir o diagnosticador centralizado com marcação de perda de sensores $\tilde{G}_{d_0}$ conforme mostrado na figura 3.2. Para melhorar a visualização da notação, os estados de G_d foram renomeados como $x_0 = \{1N\}$, $x_1 = \{2N, 3Y\}$, $x_2 = \{5N\}$, $x_3 = \{4Y\}$, $x_4 = \{7Y\}$ e $x_5 = \{6N\}$.

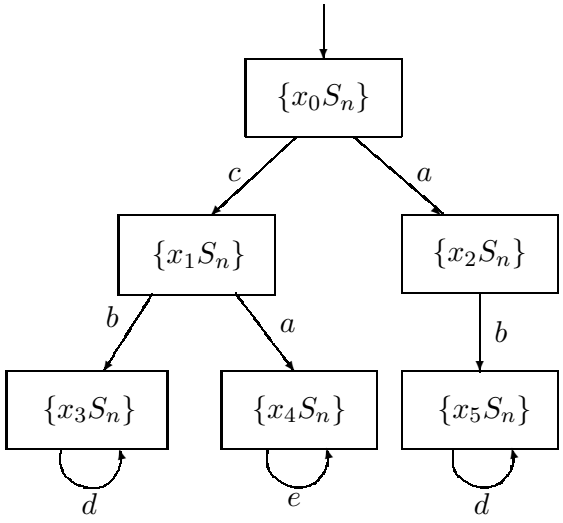


Figura 3.2: Diagnosticador com marcação de perda de sensores $\tilde{G}_{d_0}$.

O próximo passo é obter todos os diagnosticadores parciais com marcação de perda de sensores de acordo com algoritmo 3.1, sendo $\tilde{G}'_d(\{d, e\})$, $\tilde{G}'_d(\{a, b\})$, $\tilde{G}'_d(\{b, e\})$, $\tilde{G}'_d(\{b, c\})$, $\tilde{G}'_d(\{c, d\})$, $\tilde{G}'_d(\{a, d\})$, $\tilde{G}'_d(\{e\})$, $\tilde{G}'_d(\{d\})$, $\tilde{G}'_d(\{c\})$, $\tilde{G}'_d(\{b\})$ e $\tilde{G}'_d(\{a\})$. Para exemplificar a construção do diagnosticador parcial com marcação de perda de sensores, considere as perdas permanentes dos sensores associados aos eventos "b" e "e". Nesse caso a base para diagnose é $\{a, c, d\}$. De acordo com a definição 3.6 e os passos 2 e 3 do algoritmo 3.2 , obtém-se $\tilde{G}'_d(\{b, e\})$ mostrado na figura 3.3. Com

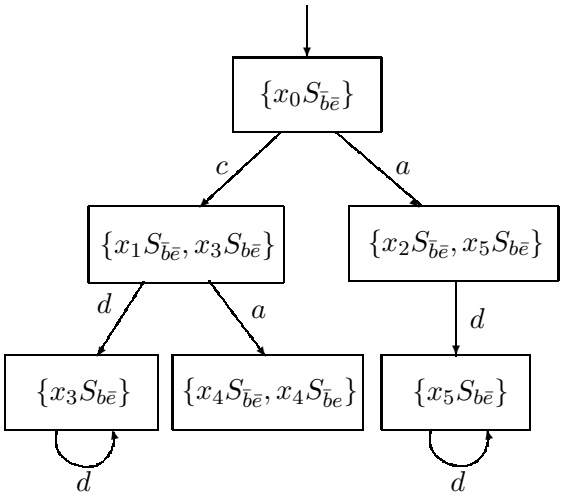


Figura 3.3: Diagnosticador parcial com marcação de perda de sensores $\tilde{G}'_d(\{b, e\})$.

todos diagnosticadores parciais e centralizado construídos, a próxima etapa consiste em gerar o diagnosticador união mostrado na figura 3.4. É importante ressaltar que todas as bases para a diagnose de falhas foram utilizadas na construção desse diagnosticador.

Note que as marcações de perdas de sensores nos diagnosticadores parciais são elementos de Σ'_{uo} *enquanto as transições em* $\tilde{G}'_d$ *são rotuladas com eventos em* Σ'_o*. É, portanto, possível identificar pela inspeção dos estados do diagnosticador união de que diagnosticador parcial com marcação de perda de sensores cada componente vem. A tabela 3.1 mostra a relação entre o conjunto de eventos redundantes* Σ_{rob} *e das base para diagnose* $\Sigma_{db,max}$*.* □

Tabela 3.1: Relação entre os eventos redundantes e as bases para diagnose.

$\Sigma_{red,i}$	$\Sigma_{db,i}$
$\{d, e\}$	$\{a, b, c\}$
$\{a, b\}$	$\{c, d, e\}$
$\{b, e\}$	$\{a, c, d\}$
$\{b, c\}$	$\{a, d, e\}$
$\{c, d\}$	$\{a, b, e\}$
$\{a, d\}$	$\{b, c, e\}$
$\{e\}$	$\{a, b, c, d\}$
$\{d\}$	$\{a, b, c, e\}$
$\{c\}$	$\{a, b, d, e\}$
$\{b\}$	$\{a, c, d, e\}$
$\{a\}$	$\{b, c, d, e\}$

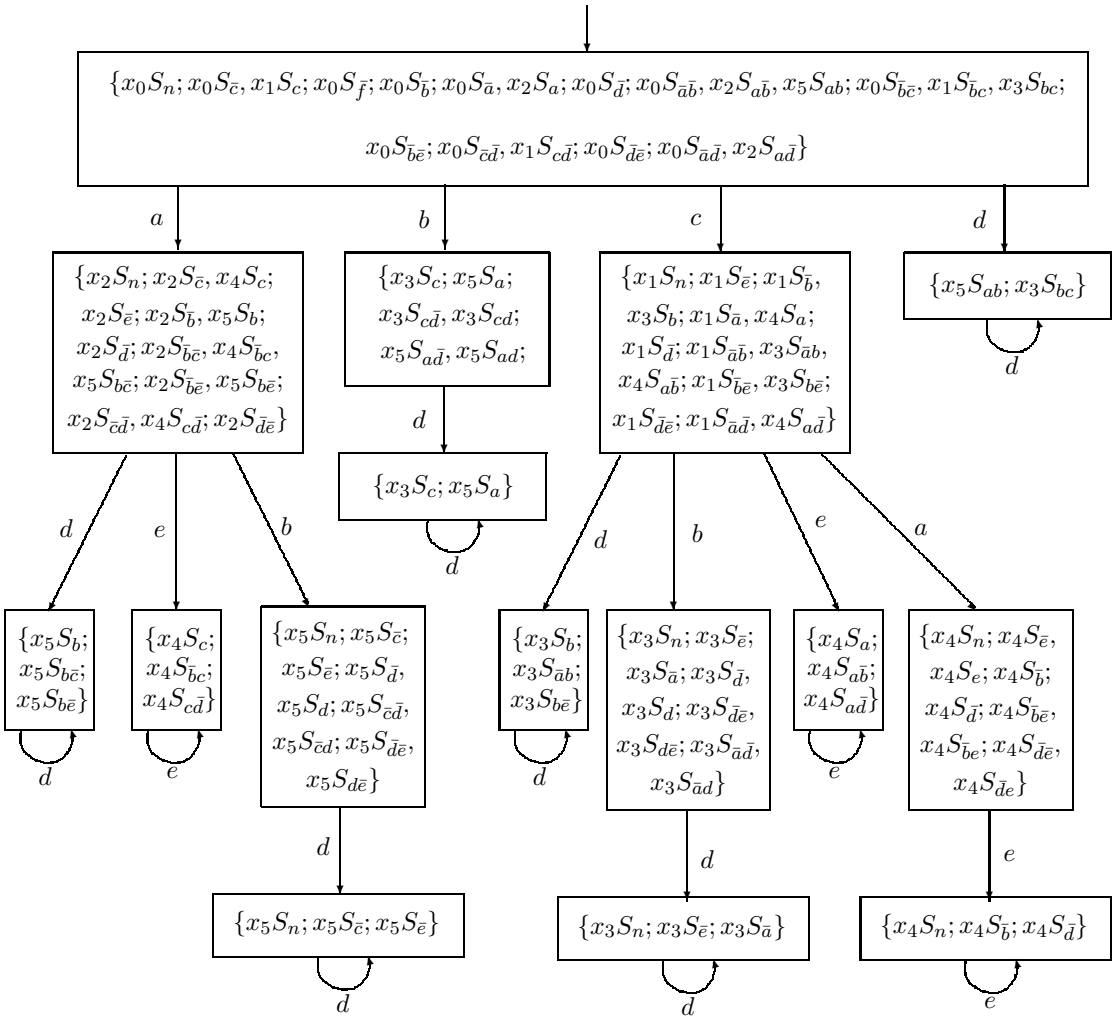


Figura 3.4: Diagnosticador união $G_{du}(\Sigma_{db,max})$.

O diagnosticador união construído utilizando-se todas as bases para a diagnose de um SED pode não ser robusto a perdas de sensores, ou seja, um dado diagnosticador parcial alcança um conjunto de estados certos por uma sequência arbitrariamente longa e um outro diagnosticador parcial alcança um conjunto de estados normais ou incertos por uma sequência de comprimento finito ou infinito e as duas sequências alcançam o mesmo estado no diagnosticador união. Assim, para se chegar a um diagnosticador robusto a partir do diagnosticador união, é necessário remover os diagnosticadores parciais correspondentes às bases que levam a ambiguidades no diagnosticador união. Com esse propósito, torna-se necessário definir formalmente estado incerto e ciclo indeterminado (observado e escondido) do diagnosticador união.

Definição 3.8 *Seja Σ_{db} um conjunto cujos elementos são bases para a diagnose de*

L e seja $G_{du}(\Sigma_{db}) = (X_{db}, \Sigma_o, f_{db}, x_{0,db})$ *o diagnosticador união formado por essas bases. Um estado* $x_{du} \in X_{du}$ *será certo se para todo* $x_d S_m \in x_{du}$, x_d *for um estado certo. Se existir* $x_d S_{m'}$, $y_d S_{m''} \in x_{du}$ *tal que* x_d *seja certo e* y_d *seja normal ou incerto, então* x_{du} *será um estado incerto de* $G_{du}(\Sigma_{db})$. □

Definição 3.9 *(Ciclos indeterminados observados e escondidos do diagnosticador união)*

A. Um conjunto de estados incerto $\{x_{du,1}, x_{du,2}, \ldots, x_{du,p}\}$ *de* G_{du} *forma um ciclo indeterminado observado se as seguintes condições forem verificadas:*

U.1) $\{x_{du,1}, x_{du,2}, \ldots, x_{du,p}\}$ *forma um ciclo em* $G_{du}(\Sigma_{db})$*;*

U.2) $\exists x_{d_l}^{k_l} S_m^{k_l}$, $\tilde{x}_{d_l}^{r_l} S_{\tilde{m}}^{r_l} \in x_{du,l}$, *em que* $x_{d_l}^{k_l}$ *é um estado certo e* $\tilde{x}_{d_l}^{k_l}$ *é um estado incerto ou normal, para* $l = 1, 2, \ldots, p$, $k_l = 1, 2, \ldots, q_l$, *e* $r_l = 1, 2, \ldots, \tilde{q}_l$ *tal que as sequências dos estados* $\{x_{d_l}^{k_l} S_m^{k_l}\}$, $l = 1, 2, \ldots, p$, $k_l = 1, 2, \ldots, q_l$, *e* $\{\tilde{x}_{d_l}^{r_l} S_{\tilde{m}}^{r_l}\}$, $l = 1, 2, \ldots, p$, $r_l = 1, 2, \ldots, \tilde{q}_l$ *forme ciclos em dois diferentes diagnosticadores parciais com marcação de perdas de sensores.*

B. Existe um ciclo escondido indeterminado em um estado incerto do diagnosticador união $G_{du}(\Sigma_{db})$ *se uma componente deste estado é um estado certo de um diagnosticador parcial com marcação de perdas de sensores no qual existe um ciclo escondido.* □

Assim como no caso da diagnosticabilidade centralizada sem perda de sensores, é possível obter condições necessárias e suficientes para que a linguagem de um diagnosticador união seja diagnosticável sob perda permanente de sensores. Essas condições são dadas no teorema a seguir (Lima *et al.*, 2010).

Teorema 3.1 *Seja L a linguagem gerada pelo autômato G e suponha que* $\Sigma_{db} = \{\Sigma_{o_1}, \Sigma_{o_2}, \ldots, \Sigma_{o_m}\}$, *em que* Σ_{o_i}, *para* $i = 1, 2, \ldots, m$ *sejam bases mínimas e não-mínimas para a diagnose de L e seja* Σ_{rob} *definido pela equação (3.1). Então, L será robustamente diagnosticável em relação à perda permanente de sensores associado aos conjuntos de eventos pertencentes a* Σ_{rob}, *em relação às projeções* $P_{o_i} : \Sigma^* \rightarrow \Sigma_{o_i}^*$, $i = 1, 2, \ldots, m$ *e* $\Sigma_f = \{\sigma_f\}$ *se e somente se o diagnosticador união* $G_{du}(\Sigma_{db})$ *não tiver nenhum ciclo indeterminado (observado ou escondido).* □

Deve ser ressaltado que, de acordo com o teorema 3.1, as marcações de perda de observabilidade dos eventos indicarão as bases que estão gerando os ciclos indeterminados no diagnosticador robusto. Assim, escolhendo-se algumas bases para serem retiradas, gera-se um novo diagnosticador união cuja linguagem gerada é a união das linguagens geradas pelos diagnosticadores parciais considerando-se as bases restantes como eventos observáveis. Portanto, esse novo diagnosticador será robusto somente à perda permanente dos sensores associados aos conjuntos de eventos redundantes das bases cujos diagnosticadores parciais com marcações foram utilizados na sua construção. O algoritmo a seguir descreve os passos para encontrar o diagnosticador robusto.

Algoritmo 3.2

Passo 1 *Encontre todos os ciclos indeterminados (escondidos e observados) de* $G_{du}(\Sigma_{db,max})$ *e identifique todos os diagnosticadores parciais com marcação de perdas de sensores com componentes normais nos estados que formam ciclos indeterminados.*

Passo 2 *Defina um novo conjunto* $\Sigma_{db} = \Sigma_{db,max} \backslash \Sigma_{ic}$*, em que* $\Sigma_{ic} = \{\Sigma_b \in \Sigma_{db,max} : \tilde{G}'_d(\Sigma_o \backslash \Sigma_b)$ *tem componentes normais no estado que formam ciclos indeterminados de* $G_{du}(\Sigma_{db,max})\}$

Passo 3 *Obtenha outro diagnosticador união formado com os diagnosticadores com marcação de perdas de sensores utilizando o conjunto* Σ_{db} □

Para ilustrar a construção do diagnosticador robusto considere o seguinte exemplo.

Exemplo 3.2 *Considere o diagnosticador união mostrado na figura 3.4 do exemplo 3.1. No diagnosticador união* $G_{du}(\Sigma_{db,max})$*, pode-se notar a presença de autolaços nos estados* $\{x_3S_c; x_5S_a\} = \{4YS_c; 6NS_a\}$ *e* $\{x_5S_{ab}; x_3S_{bc}\} = \{6NS_{ab}; 4YS_{bc}\}$*. Se o diagnosticador união considerado alcançar esses estados, a ocorrência da*

falha não poderá ser determinada, pois o estado x_3 é um estado certo e o estado x_5 é um estado normal de G_d. Portanto, os estados $\{x_3S_c; x_5S_a\}$ e $\{x_5S_{ab}; x_3S_{bc}\}$ formam ciclos observados indeterminados devido às bases $\{a, b, d, e\}$ e $\{b, c, d, e\}$ e, às bases $\{c, d, e\}$ e $\{a, d, e\}$, respectivamente. Além disso, no estado $\{x_3S_c; x_5S_a; x_3S_{c\bar{d}}, x_3S_{cd}; x_5S_{a\bar{d}}, x_5S_{ad}\}$, o conjunto dos eventos ativos das componentes dos diagnosticadores parciais $\tilde{G}'_d(\{c, d\})$ e $\tilde{G}'_d(\{a, d\})$ são ambos vazios devido a ciclos escondidos nos estados $\{x_3S_{c\bar{d}}, x_3S_{cd}\}$ e $\{x_5S_{a\bar{d}}, x_5S_{ad}\}$. Então, as componentes dos diagnosticadores parciais $\tilde{G}'_d(\{c, d\})$ e $\tilde{G}'_d(\{a, d\})$ nunca alcançam um estado certo no caso da falha ter ocorrido e outras componentes do diagnosticador união estarem em um estado normal após a ocorrência do evento b a partir do estado inicial. Portanto, o estado $\{x_3S_c; x_5S_a; x_3S_{c\bar{d}}, x_3S_{cd}; x_5S_{a\bar{d}}, x_5S_{ad}\}$ possui ciclos escondidos indeterminados devido às bases $\{a, b, e\}$, $\{b, c, e\}$ e $\{b, c, d, e\}$.

De acordo com o passo 2 do algoritmo 3.2, deve-se retirar uma das bases que geram cada ciclo observado indeterminado e as bases que geram o ciclo escondido indeterminado de forma a tornar o estado incerto que possui o ciclo escondido um estado normal ou certo. Logo, os diagnosticadores a serem removidos serão: $\tilde{G}'_d(\{a\})$, $\tilde{G}'_d(\{a, b\})$ e $\tilde{G}'_d(\{a, d\})$ cujas as bases $\{b, c, d, e\}$, $\{c, d, e\}$ e $\{b, c, e\}$. Desse modo, as bases para a diagnose robusta serão

$$\Sigma_{bd} = \{\{a, b, c\}, \{a, c, d\}, \{a, d, e\}, \{a, b, e\}, \{a, b, c, d\}, \{a, b, c, e\}, \{a, b, d, e\}, \{a, c, d, e\}, \Sigma_o\}.$$

Assim, como esperado, o diagnosticador robusto $G_{rob}(E_{db})$, mostrado na figura 3.5, não tem ciclos indeterminados . Logo, L é robustamente diagnosticável com respeito ao conjuntos de eventos:

$$\Sigma_{rob} = \{\{b\}, \{c\}, \{d\}, \{e\}, \{c, d\}, \{b, c\}, \{b, e\}, \{d, e\}\},$$

P_{o_i}, em que $P_{o_i} : \Sigma^* \to \Sigma^*_{o_i}$ para todo $\Sigma_{o_i} \in E_{db}$, e Σ_f. □

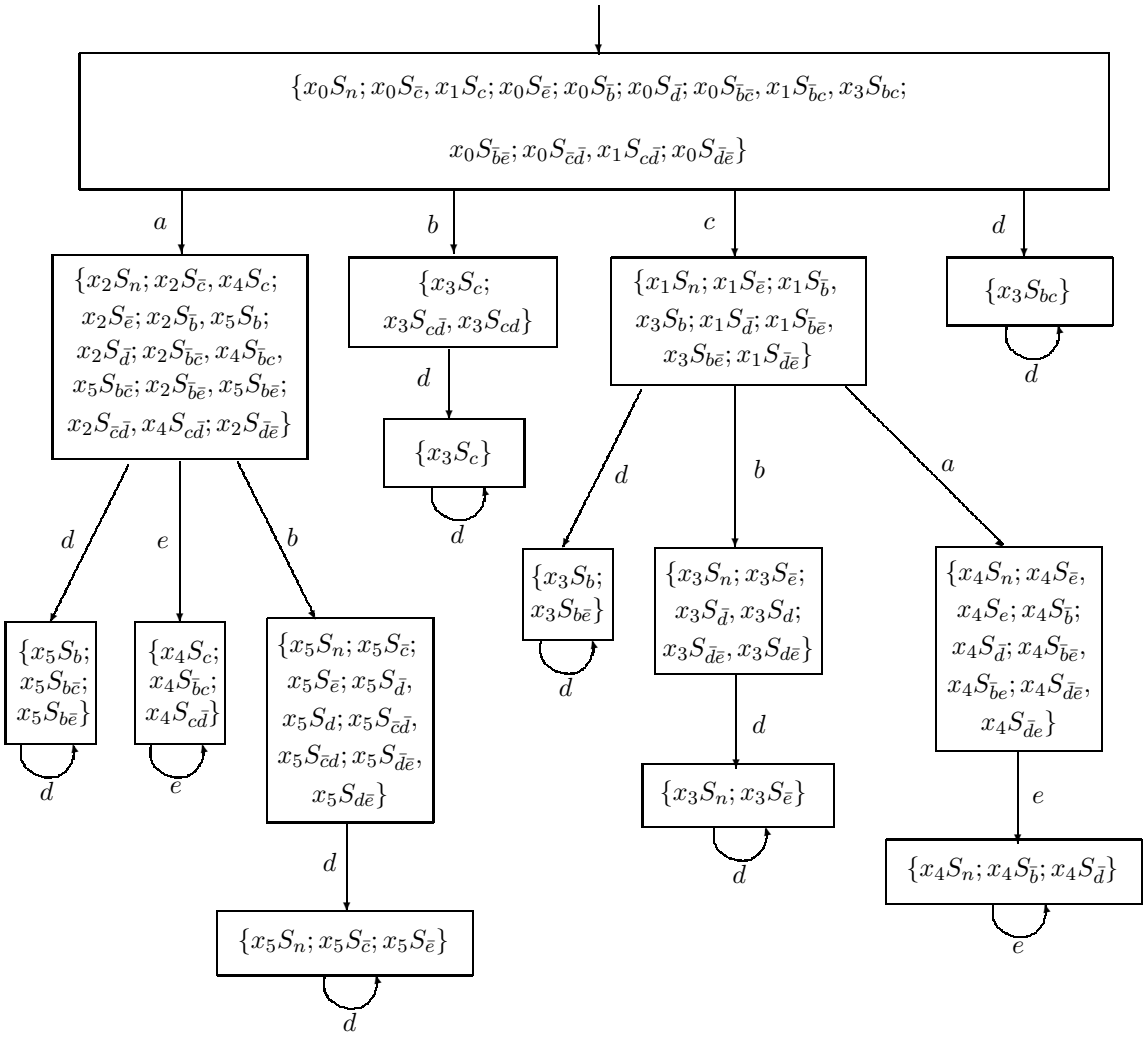


Figura 3.5: Diagnosticador robusto $G_{rob}(E_{db})$.

3.3 Verificador de robustez

Na seção anterior foi abordado o problema de se diagnosticar a ocorrência de uma falha em presença de perdas permanentes de sensores utilizando diagnosticadores. Nessa seção será apresentado um novo algoritmo de execução em tempo polinomial para verificar a diagnosticabilidade robusta de um SED sujeito a perdas permanentes de sensores baseado no algoritmo proposto por Moreira *et al.* (2011) e revisto no capítulo 2. A ideia básica é construir um verificador a partir de modelos que representam as diferentes formas de se descrever o sistema utilizando como eventos observáveis as diferentes bases para diagnose, para buscar as sequências que são ambíguas, ou seja, procurar as sequências de não falha de um modelo que tenham a mesma projeção que uma sequência de falha em outro modelo.

Considere novamente o conjunto formado pelas bases para diagnose $\Sigma_{db,max} = \{\Sigma_{o_1}, \Sigma_{o_2}, \ldots, \Sigma_{o_m}\}$ e suponha que Σ seja particionado em $\Sigma = \Sigma_{o_i} \cup \Sigma_{uo_i}$ sendo Σ_{o_i} e Σ_{uo_i} conjuntos de eventos observáveis e não-observáveis, respectivamente, e defina $\Sigma_o = \bigcup_{i=1}^{q} \Sigma_{o_i}$. Seja G_i um autômato que gera a linguagem L_i sendo obtido a partir da renomeação dos eventos em $\Sigma \setminus \Sigma_{o_i}$ de G pela seguinte função de renomeação $R_i : \Sigma \to \Sigma_i$, para $i = 1, 2, \ldots, m$, definida como:

$$R_i(\sigma) = \begin{cases} \sigma, \text{ se } \sigma \in \Sigma_{o_i} \cup \Sigma_f \\ \sigma_{R_i}, \text{ se } \sigma \in \Sigma_{uo_i} \setminus \Sigma_f \end{cases} . \tag{3.3}$$

Note que R_i somente renomeia os eventos em $\Sigma_{uo_i} \setminus \Sigma_f$. Deve ser ressaltado que o domínio da função R_i pode ser estendido para Σ^* da seguinte forma: $R_i(\varepsilon) = \varepsilon$, e $R_i(s\sigma) = R_i(s)R_i(\sigma)$, $\forall s \in \Sigma^*$ e $\forall \sigma \in \Sigma$. Da mesma forma, R_i pode ser estendida para linguagens $L \subseteq \Sigma^*$ aplicando a função a todas as sequências de L.

Baseado em R_i, pode-se também definir a função renomeação inversa como:

$$\begin{aligned} R_i^{-1} : \quad \Sigma_i \quad &\to \Sigma \\ \sigma_{R_i} \quad &\mapsto \sigma, \end{aligned}$$

em que $\sigma_{R_i} = R_i(\sigma)$. Assim como R_i, R_i^{-1} pode ter a seguinte extensão de domínio em Σ_i^*: $R_i^{-1}(s_{R_i}\sigma_{R_i}) = R_i^{-1}(s_{R_i})R_i^{-1}(\sigma_{R_i})$ para todo $s_{R_i} \in \Sigma_i^*$ e $\sigma_{R_i} \in \Sigma_i$, e $R_i^{-1}(\varepsilon) = \varepsilon$.

Dessa forma, a construção do verificador de robustez pode ser feita de acordo com o seguinte algoritmo.

Algoritmo 3.3 *(Construção do verificador de robustez)*

Passo 1 *Construa os autômatos* $G_i = (X, \Sigma_i, f_i, \Gamma_i, x_0)$, $i = 1, 2, \ldots, m$, *em que* $\Sigma_i = R_i(\Sigma)$, $\Gamma_i(x) = R_i[\Gamma(x)]$, *e* $f_i(x, R_i(\sigma)) = f(x, \sigma)$ *para todo* $x \in X$ *e* $\sigma \in \Gamma(x)$.

Passo 2 *Construa o autômato de falha* G_{F_i}.

- *Passo 2.1: Calcule* $G_{\ell_i} = G_i \| A_\ell$ *em que* A_ℓ *é o autômato rotulador de falhas da figura 2.5,* $i = 1, 2, \ldots, m$*, e marque todos os estados de* G_{ℓ_i} *cujas segundas coordenadas são iguais a* Y*.*
- *Passo 2.2: Obtenha o autômato de falha* $G_{F_i} = CoAc(G_{\ell_i}) = (X_{F_i}, \Sigma_i, f_{F_i}, x_{0,F_i})$*.*
- *Passo 2.3: Refine o conjunto de eventos de* G_{F_i} *para* $\Sigma_{F_i} = \Sigma_i \cup \Sigma_o$*.*

Passo 3 *Construa os autômatos de não falha* G_{N_i}*,* $i = 1, 2, \ldots, m$ *da seguinte forma.*

- *Passo 3.1: Defina* $\Sigma_{N_i} = \Sigma_i \setminus \Sigma_f$*, e obtenha autômato* $A_{N_i} = (\{N\}, \Sigma_{N_i}, f_{N_i}, N)$ *que tenha um único estado com um autolaço rotulado por todos os eventos de* Σ_{N_i}*.*
- *Passo 3.2: Construa* $G_{N_i} = G_i \times A_{N_i} = (X_{N_i}, \Sigma_i,\ f_{N_i}, \Gamma_{N_i}, x_{0,N_i})$*.*
- *Passo 3.3: Refine o conjunto de eventos de cada* G_{N_i} *para* $\Sigma_{N_i} = \Sigma_i \setminus \Sigma_f$*.*

Passo 4 *Construa os autômatos aumentados* $G^a_{N_i} = (X^a_{N_i},\ \Sigma^a_{N_i}, f^a_{N_i}, x^a_{0,N_i})$ *a partir do autômato* G_{N_i} *da seguinte forma.*

- *Passo 4.1: Defina* $\Sigma^a_{N_i} = \Sigma_{N_i} \cup \Sigma_o$*.*
- *Passo 4.2: Defina* $x^a_{0,N_i} = x_{0,N_i}$*.*
- *Passo 4.3: Adicione um novo estado* D_i *no espaço de estados de* G_{N_i}*. Então,* $X^a_{N_i} = X_{N_i} \cup \{D_i\}$*.*
- *Passo 4.4: Para cada* $x^a_{N_i} \in X_{N_i}$ *defina:*

$$f^a_{N_i}(x^a_{N_i}, \sigma) = \begin{cases} f_{N_i}(x^a_{N_i}, \sigma),\ \textit{se } \sigma \in \Gamma_{N_i}(x_{N_i}) \\ D_i,\ \textit{se } \sigma \in \Sigma_o \setminus \Gamma_{N_i}(x_{N_i}) \\ \textit{não definido, caso contrário} \end{cases}, \tag{3.4}$$

e para $x^a_{N_i} = D_i$ *defina:*

$$f^a_{N_i}(x^a_{N_i}, \sigma) = \begin{cases} D_i, \text{ para todo } \sigma \in \Sigma_o \\ \text{não definido, caso contrário} \end{cases}. \tag{3.5}$$

PASSO 5 *Para* $i = 1, 2, \ldots, m$, *calcule o autômato verificador* G_{V_i} *cujo* j*-ésimo estado* $x_{V_{ij}} \in X_{F_i} \times (\times^m_{p=1,p\neq i} X^a_{N_i})$ *e o* j*-ésimo estado de* X_{F_i} *é* $x_{F_{ij}} \in X_i \times X_\ell$, *através de uma composição de* G_{F_i}, $G^a_{N_1}$, ..., $G^a_{N_{i-1}}$, $G^a_{N_{i+1}}$, ..., $G^a_{N_m}$, *que é análoga à composição paralela* $G_{F_i} \| (\|^m_{j=1,j\neq i} G^a_{N_j})$, *exceto quando o estado alcançado seja* $(x_{F_i}, D_1, \ldots, D_{i-1}, D_{i+1}, \ldots, D_m)$, *sendo* $x_{F_i} \in X_{F_i}$, *então o seu conjunto de eventos ativos é forçado a ser vazio*[1], i.e,

$$\Gamma_{V_i}(x_{F_i}, D_1, \ldots, D_{i-1}, D_{i+1}, \ldots, D_m) = \emptyset.$$

PASSO 6 *Verifique a existência de um ciclo*

$$cl_i = (x_{V_{ik}}, \sigma_k, x_{V_{ik+1}}, \sigma_{k+1}, \ldots, \sigma_l, x_{V_{il}}),$$

sendo $l \geq k > 0$, *em* G_{V_i} *que satisfaça a seguinte condição:*

$$\exists j \in \{k, k+1, \ldots, l\} \text{ tal que } (\sigma_j \in \Sigma_i) \wedge (x_{F_{ij}} = \{x_{ij}, Y\}).$$

Se tal ciclo existir, então L *não é robustamente diagnosticável às perdas de sensores em* $\Sigma_{red} = \{\Sigma \setminus \Sigma_{o_1}, \Sigma \setminus \Sigma_{o_2}, \ldots, \Sigma \setminus \Sigma_{o_m}\}$. *Caso contrário,* L *é robustamente diagnosticável.* □

Note que na composição entre os autômatos G_{F_i} e $G^a_{N_j}$, $j = 1, 2, \ldots, m$, $j \neq i$ os eventos comuns a G_{F_i} e $G^a_{N_j}$ são Σ_o, uma vez que pela construção do autômato aumentado $G^a_{N_i}$ no passo 4 do algoritmo 3.3, para $i = 1, 2, \ldots, m$, a linguagem gerada é $L(G^a_{N_i}) \supseteq \Sigma^*_o$, ou seja, todos autômatos aumentados geram Σ^*_o, e, portanto, as

[1]Isso equivale a considerar o estado $(x_{F_i}, D_1, \ldots, D_{i-1}, D_{i+1}, \ldots, D_m)$ como um estado de bloqueio.

sequências geradas por G_{F_i} definem quando os eventos comuns ocorrerem. Portanto, o objetivo do autômato aumentado é possibilitar a busca concorrente das sequências ambíguas entre G_{F_i} e os autômatos $G^a_{N_i}$, $i = 1, 2, \ldots, m$, no verificador.

Vale a pena ressaltar que poderia ser utilizada a composição paralela entre os autômatos no passo 5 do algoritmo 3.3. Entretanto, quando se alcança o estado $(x_{F_i}, D_1, \ldots, D_{i-1}, D_{i+1}, \ldots, D_m)$, as sequências geradas a partir do estado D_i não são sequências de G_i, $i = 1, 2, \ldots, m$, e, portanto a busca de sequências ambíguas entre sequências de falhas de G_i e de não falha de G_j, $j = 1, , 2, \ldots, m$, $j \neq i$, perde o sentido.

A figura 3.6 ilustra a busca de sequências ambíguas para o caso de três módulos, sendo os conjuntos de eventos observáveis denotados por Σ_{o_i}, $i = 1, 2, 3$. Na construção do verificador G_{V_3} buscam-se sequências em $L_3 \setminus L_{N_3}$ que possuem a mesma projeção em $\Sigma^*_{o_3}$ que alguma em sequência de L_{N_1} ou L_{N_2} em $\Sigma^*_{o_1}$ e $\Sigma^*_{o_2}$, respectivamente. Note que, caso uma sequência de L_{N_1} alcance o estado D_1, então as sequências geradas por $G^a_{N_1}$ que levam ao estado D_1 não pertencem a L_1 e, portanto, não podem levar a ambiguidades.

O teorema a seguir prova o algoritmo 3.3.

Teorema 3.2 *L não é robustamente diagnosticável com relação a P_{o_i}, $i = 1, 2, \ldots, m$, e Σ_f se e somente se existe um ciclo $cl_i = (x_{V_{ik}}, \sigma_k, x_{V_{ik+1}}, \sigma_{k+1}, \ldots, \sigma_l, x_{V_{il}})$, sendo $l \geq k > 0$, em ao menos um verificador G_{V_i}, $i \in I_m := \{1, 2, \ldots, m\}$, que satisfaz a seguinte condição:*

$$\exists j \in \{k, k+1, \ldots, l\} \text{ tal que } (\sigma_j \in \Sigma_i) \wedge (x_{F_{ij}} = \{x_{ij}, Y\}). \tag{3.6}$$

□

Demonstração: ($\Longleftarrow$) Suponha que exista um ciclo $cl_i = (x_{V_{ik}}, \sigma_k, x_{V_{ik+1}}, \sigma_{k+1}, \ldots, \sigma_l, x_{V_{il}})$, sendo $l \geq k > 0$, no verificador G_{V_i} que satisfaça a condição (3.6). Uma vez que $x_{F_{ij}} = \{x_{i_j}, Y\}$ para algum $j \in \{k, k+1, \ldots, l\}$, então, pela construção de G_{V_i}, tem-se que $x_{F_{ij}} = \{x_{i_j}, Y\}$ para todo $j \in \{k, k+1, \ldots, l\}$.

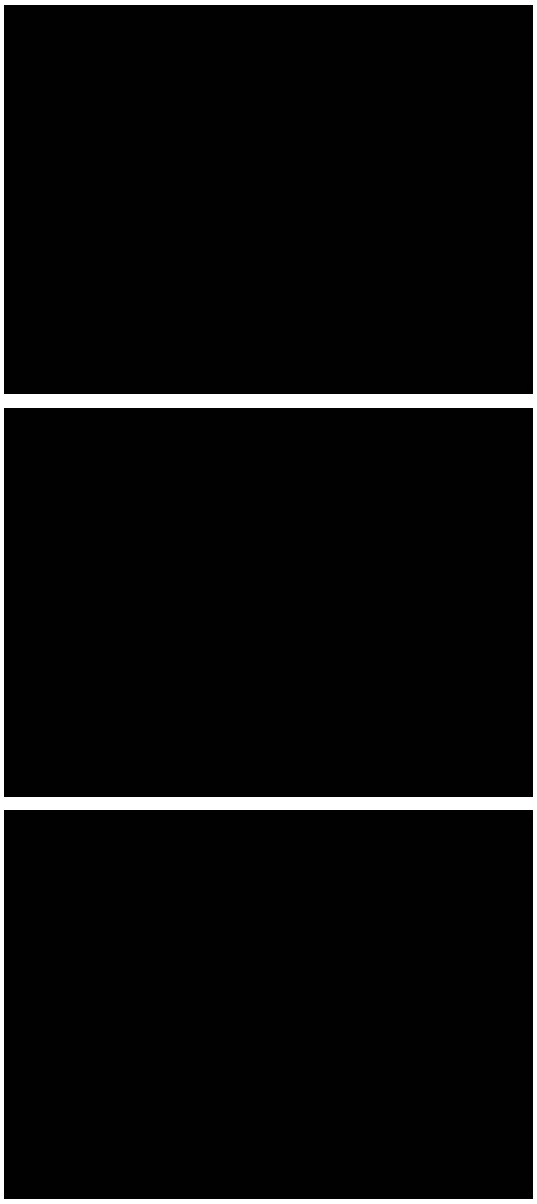


Figura 3.6: Busca por sequências ambíguas utilizando o verificador construído de acordo com o algoritmo 2.2.

Logo, existe uma sequência $s't' \in L(G_{V_i})$, tal que s' contém o evento de falha, e $t' = (\sigma_k \sigma_{k+1} \dots \sigma_l)^n$, $\forall n \in \mathbb{N}$.

De acordo com algoritmo 3.3, os estados de G_{V_i} que são iguais a $(x_{F_i}, D_1, \dots, D_{i-1}, D_{i+1}, \dots, D_m)$ são estados de bloqueio. Portanto, se o verificador G_{V_i} tiver um ciclo cl_i que satisfaz a condição (3.6), então existirá $q \in I_{m_i} := \{1, \dots, i-1, i+1, \dots, m\}$ tal que $x^a_{N_{q_j}} \neq D_q$ para todo $j \in \{k, k+1, \dots, l\}$.

Seja

$$\Sigma_R = \bigcup_{\nu=1}^{m} \Sigma_\nu,$$

e defina as seguintes projeções:

$$P_{F_i} \quad : \quad \Sigma_R^* \rightarrow \Sigma_{F_i}^*, \tag{3.7}$$

$$P_{N_p}^a \quad : \quad \Sigma_R^* \rightarrow \Sigma_{N_q}^{a\,*}, \tag{3.8}$$

para $p \in I_{m_i}$. De acordo com o algoritmo 3.3, G_{V_i} é construído por uma operação entre autômatos que é equivalente à composição paralela exceto nos estados de bloqueio. Portanto,

$$L(G_{V_i}) \subseteq P_{F_i}^{-1}[L(G_{F_i})] \cap \left[\bigcap_{p=1, p\neq i}^{m} P_{N_p}^{a\,-1}[L(G_{N_p}^a] \right],$$

o que implica que $s't' \in P_{F_i}^{-1}[L(G_{F_i})]$ já que, conforme mostrado acima, $s't' \in L(G_{V_i})$.

Seja $\tilde{s}\tilde{t} = P_{F_i}(s't')$, em que $\tilde{s} = P_{F_i}(s')$ e $\tilde{t} = P_{F_i}(t')$. Assim, uma vez que $P_{F_i}\left[P_{F_i}^{-1}(L(G_{F_i}))\right] = L(G_{F_i})$, então $\tilde{s}\tilde{t} \in L(G_{F_i})$. Note que, como $t' = (\sigma_k \sigma_{k+1} \ldots \sigma_l)^n$, $\forall n \in \mathbb{N}$, e, por hipótese, existe um evento $\sigma_j \in \Sigma_i$ para $j \in \{k, k+1, \ldots, l\}$, então a sequência $\tilde{t} = P_{F_i}(t')$ pode ser feita arbitrariamente longa após a ocorrência de um evento de falha. Observe que G_{F_i} é idêntico ao autômato G_i, exceto pelo conjunto dos eventos, que é expandido. Portanto, a sequência $\tilde{s}\tilde{t} \in L(G_i)$ pode ser feita arbitrariamente longa após a ocorrência do evento de falha. Além disso, como G_i é obtido de G pela renomeação de seus eventos Σ pela função de renomeação R_i, definida na equação (3.3), então existe uma sequência $st \in L$ associada com $\tilde{s}\tilde{t} \in L(G_i)$ que é obtida por

$$st = R_i^{-1}(\tilde{s}\tilde{t}).$$

Seja $\tilde{s}_q = P_{N_q}^a(s't')$. É sabido que $s't' \in P_{N_q}^{a\,-1}[L(G_{N_q}^a)]$ uma vez que $s't' \in L(G_{V_i})$. Além disso, como $P_{N_q}^a[P_{N_q}^{a\,-1}(L(G_{N_q}^a))] = L(G_{N_q}^a)$, então $\tilde{s}_q \in L(G_{N_q}^a)$. Pela hipótese inicial, o estado $x_{N_{q_j}}^a$, é diferente de D_q para todo $j \in \{k, k+1, \ldots, l\}$, e portanto $\tilde{s}_q \in L(G_{N_q})$. Note que G_{N_q} (autômato de não falha) é obtido de G após a renomeação do conjunto de eventos Σ pela função R_q. Logo, associado com $\tilde{s}_q$ existe uma sequência

de não falha $s_q \in L(G)$ dada por:

$$s_q = R_q^{-1}(\tilde{s}_q).$$

Finalmente, observe que pela construção de G_{F_i}, $G^a_{N_q}$ e G_{V_i} — passos 2, 4 e 5, respectivamente, do algoritmo 3.3 — e pelo fato de que $x^a_{N_{q_j}} \neq D_q$ para todo $j \in \{k, k+1, \ldots, l\}$ no ciclo cl_i, então todos os eventos em Σ_o que estão na sequência $s't'$ — e consequentemente, em st e s_q — são de $\Sigma_{o_i} \cap \Sigma_{o_q}$. Portanto, $P_{o_i}(st) = P_{o_q}(s_q)$, o que viola a condição de diagnosticabilidade robusta da definição 3.3.

($\Longrightarrow$) Suponha que L não seja robustamente diagnosticável com relação a P_{o_i}, $i = 1, \ldots, m$, e $\Sigma_f = \{\sigma_f\}$. Considere que $L_N \subseteq L$ representa as sequências de não falha de L. Assim, existe $i, q \in I_m$, $i \neq q$, tal que para toda sequência $s_i t_i \in L \setminus L_N$, com $s_i \in L \setminus L_N$ e $|t_i| \geq n_i$, $n_i \in \mathbb{N}$, e para todo $w_q \in L_N$ (w_q não necessariamente ilimitada), $P_{o_i}(s_i t_i) = P_{o_q}(w_q)$.

Note que, de acordo com o algoritmo 3.3, G_i e G_q são obtidos através da renomeação dos eventos não-observáveis de G pelas funções de renomeação R_i e R_q. Portanto, as seguintes conclusões podem ser deduzidas:

C1. Existe uma sequência $\tilde{s}_i\tilde{t}_i = R_i(s_i t_i)$, com $\tilde{s}_i = R_i(s_i)$ e $\tilde{t}_i = R_i(t_i)$, $\tilde{s}_i\tilde{t}_i \in L_i$. Além disso, como $s_i t_i \in L \setminus L_N$, *i.e.* $s_i t_i$ é uma sequência de falha, então $\tilde{s}_i\tilde{t}_i \in L(G_{F_i})$. Note que como t_i é uma sequência de comprimento arbitrariamente longa, então $\tilde{t}_i$ também é.

C2. Existe uma sequência $\tilde{w}_q \in L_q$ tal que $\tilde{w}_q = R_q(w_q)$. Uma vez que w_q é uma sequência de não falha de G e $\tilde{w}_q \in L(G_{N_q})$, logo $\tilde{w}_q \in L(G^a_{N_q})$.

C3. $f_{F_i}(x_{0,F_i}, \tilde{s}_i t) = \{x_i, Y\}$, em que $x_i \in X_i$, para todo $t \in pr(\tilde{t}_i)$, sendo $pr(.)$ o fecho do prefixo.

C4. $f^a_{N_q}(x^a_{0,N_q}, w) \neq D_q$ para todo $w \in pr(\tilde{w}_q)$.

De acordo com a hipótese A4, L é diagnosticável com relação à projeção P_{o_i} e Σ_f, para todo $i = 1, 2, \ldots, m$. Uma vez que $P_{o_i}(s_i t_i) \neq \varepsilon$ e $P_{o_i}(s_i t_i) = P_{o_q}(w_q)$,

é possível ver que $\Sigma_{o_i} \cap \Sigma_{o_q} \neq \emptyset$. Portanto, como $L(G_{V_i}) \subseteq P_{F_i}^{-1}[L(G_{F_i})] \cap [\bigcap_{p=1,p\neq i}^{m} P_{N_p}^{a\,-1}[L(G_{N_p}^a)]]$, $P_{F_i}\left[P_{F_i}^{-1}(L(G_{F_i}))\right] = L(G_{F_i})$ e $P_{N_q}^a[P_{N_q}^{a\,-1}(L(G_{N_q}^a))] = L(G_{N_q}^a)$, então é possível encontrar uma sequência $s't' \in L(G_{V_i})$ tal que $\tilde{s}_i\tilde{t}_i = P_{F_i}(s't')$ e $w_q = P_{N_q}^a(s't')$, no qual $\tilde{s}_i = P_{F_i}(s')$ e $\tilde{t}_i = P_{F_i}(t')$, com t' arbitrariamente longa.

Note que o verificador G_{V_i} é um autômato finito e sendo t' uma sequência de comprimento arbitrariamente longo, então $t' = t'_f(\sigma_k\sigma_{k+1}\ldots\sigma_l)^n$, $n \in \mathbb{N}$, em que t'_f é uma sequência de comprimento finito e $\sigma_k\sigma_{k+1}\ldots\sigma_l$ os eventos de um ciclo $cl_i = (x_{V_{ik}}, \sigma_k, x_{V_{ik+1}}, \sigma_{k+1}, \ldots, \sigma_l, x_{V_{il}})$, $l \geq k > 0$, no qual, $\sigma_j \in \Sigma_i$ para todo $j \in \{k, k+1, \ldots, l\}$. Além disso, considerando as conclusões C3) e C4), é possível ver que para todas as sequências $s't'_ft$, $t \in pr((\sigma_k\sigma_{k+1}\ldots\sigma_l)^n)$, e, portanto, $f_{V_i}(x_{0,V_i}, s't'_ft) = x_{V_{ij}}$, em que a primeira componente de $x_{V_{ij}}$ é $\{x_{ij}, Y\}$, $x_{ij} \in X_i$, e a q-ésima componente de $x_{V_{ij}}$ é diferente de D_q. □

3.3.1 Complexidade computacional do algoritmo 3.3

A complexidade computacional do algoritmo 3.3 é determinada pelo número de estados e transições de cada autômato construído no algoritmo 3.3. Na tabela 3.2 é mostrado o número máximo de estados e transições de todos os autômatos que devem ser computados de acordo com o algoritmo 3.3 para obter o autômato G_{V_i} supondo m possíveis modelos G_i para G.

O primeiro passo do algoritmo 3.3 é a construção do autômato renomeado G_i a partir de G. Como G_i é exatamente igual a G exceto pela renomeação dos eventos dos conjuntos Σ_{uo_i}, o número dos estados e transições de G_i são os mesmos de G.

O segundo passo do algoritmo 3.3 é a construção do autômato G_{F_i}. Para tanto, é necessário primeiro construir a autômato A_ℓ com dois estados, N e Y, cujas transições são rotuladas somente por eventos de falha e, em seguida, obter $G_{\ell_i} = G_i \| A_\ell$. Note que $L(G_{\ell_i}) = L(G_i)$ e que os estados de G_{ℓ_i} são da forma (x, N) ou (x, Y), com $x \in X$. Portanto, o número máximo de estados de G_{ℓ_i} é $2|X|$. Finalmente, G_{F_i} é obtido tomando-se a parte coacessível de G_{ℓ_i} ($G_{F_i} = CoAc(G_{\ell_i})$), e então, o número

Tabela 3.2: Complexidade computacional do algoritmo 3.3

	No. de estados	No. de transições
G_i	$\|X\|$	$\|X\|\|\Sigma\|$
A_l	2	2
G_{ℓ_i}	$2\|X\|$	$2\|X\|\|\Sigma\|$
G_{F_i}	$2\|X\|$	$2\|X\|\|\Sigma\|$
A_{N_i}	1	$\|\Sigma\| - \|\Sigma_f\|$
G_{N_i}	$\|X\|$	$\|X\|(\|\Sigma\| - \|\Sigma_f\|)$
$G^a_{N_i}$	$\|X\| + 1$	$\|X\|(\|\Sigma_{uo_i}\| - \|\Sigma_f\| + \|\Sigma_o\|) + \|\Sigma_o\|$
G_{V_i}	$N_{G_{V_i}} = 2\|X\| \prod_{j\neq i}(\|X\|+1)$	$N_{G_{V_i}}\,[m(\|\Sigma\| - \|\Sigma_f\|) + \|\Sigma_f\|]$
Complexidade		$O\Big(m\|X\| \prod_{j\neq i} \|X\|(\|\Sigma\| - \|\Sigma_f\|)\Big)$

máximo de estados e transições de G_{F_i} são $2|X|$ e $2|X||\Sigma|$, respectivamente.

No passo 3, um autômato composto de um único estado A_{N_i} é utilizado para encontrar o autômato de não falha $G_{N_i} = G_i \times A_{N_i}$. Portanto, como A_{N_i} é um autômato de um único estado com um autolaço rotulado por $\Sigma_i \setminus \Sigma_f$, o número máximo de estados e transições de G_{N_i} são $|X|$ e $|X|(|\Sigma| - |\Sigma_f|)$, respectivamente.

No passo 4 o autômato aumentado $G^a_{N_i}$ é obtido a partir de G_{N_i} pela adição do estado D_i e transições dos estados de G_{N_i} para D_i rotulados com eventos de Σ_o que não pertencem ao conjunto dos eventos ativos de cada estado. Portanto, o número máximo de estados e transições de $G^a_{N_i}$ são, respectivamente, $|X| + 1$ e $|X|(|\Sigma_{uo_i}| - |\Sigma_f| + |\Sigma_o|) + |\Sigma_o|$.

Finalmente, no passo 5, o verificador G_{V_i} é obtido a partir de uma nova composição entre G_{F_i} e $G^a_{N_j}$, para todo $j \in I_{m_i}$, baseado na composição paralela com uma regra adicional que estabelece que alguns estados alcançados pela composição paralela são forçados a ser estados de bloqueio. Portanto, o número de estados e transições de G_{V_i} são no pior caso igual a $2|X| \prod_{j=1,j\neq i}^{m}(|X| + 1)$ e $[2|X| \prod_{j=1,j\neq i}^{m}(|X| + 1)][m(|\Sigma| - |\Sigma_f|) + |\Sigma_f|]$, respectivamente. Logo, a complexidade do algoritmo 3.3 é $O(m|X| \prod_{j=1,j\neq i}^{m} |X|(|\Sigma| - |\Sigma_f|))$. □

O exemplo a seguir ilustra a construção do verificador de robustez.

Exemplo 3.3 *Considere novamente o autômato G da figura 3.1(a) em que $\Sigma = \{a, b, c, d, e, \sigma_f\}$ e $\Sigma_o = \{a, b, c, d, e\}$ cujas bases mínimas para a diagnose de L são (Basilio et al., 2011):*

$$\Sigma_{mdb} = \{\{a, b, c\}, \{c, d, e\}, \{a, c, d\}, \{a, d, e\}, \{a, b, e\}, \{b, c, e\}\}. \tag{3.9}$$

Para ilustrar a construção do verificador de robustez, considere, para efeito de simplicidade, os seguintes elementos do conjunto Σ_{mdb} definido na equação (3.9): $E_{o_1} = \{a, b, c\}$, $E_{o_2} = \{c, d, e\}$ e $E_{o_3} = \{a, d, e\}$. Para tanto, com base no autômato

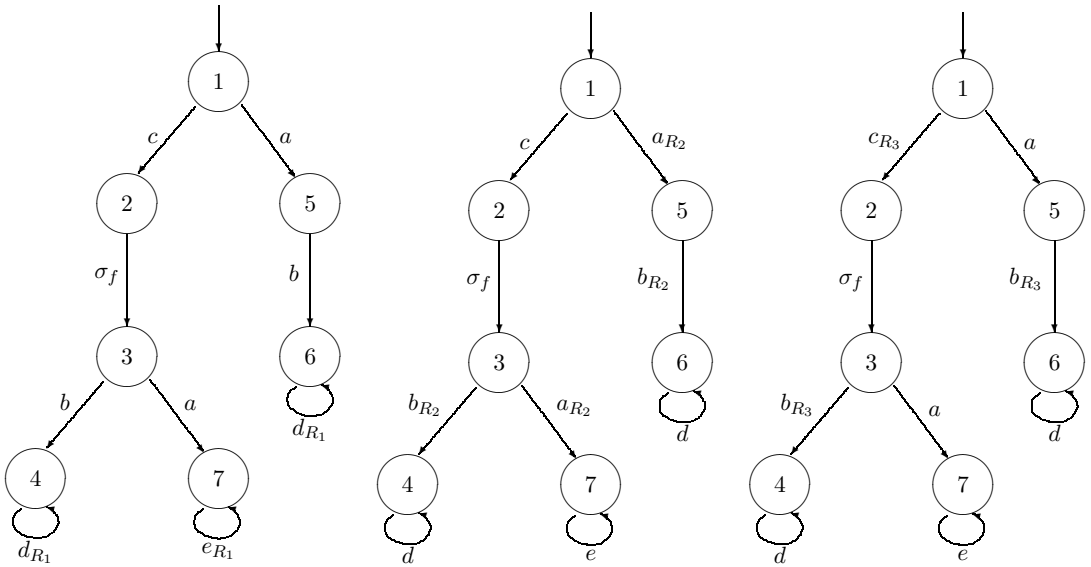


Figura 3.7: Autômato G_1 (a); Autômato G_2 (b); Autômato G_3 (c).

G representado na figura 3.1(a) , deve-se construir os autômatos renomeados G_1, G_2 e G_3 mostrados na figura 3.7. Note que, como $E_{o_1} = \{a, b, c\}$, então os eventos "d" e "e" são renomados para d_{R_1} e e_{R_1}, respectivamente, no autômato G_1, os eventos "a" e "b" para a_{R_2} e b_{R_2} em G_2 uma vez que $E_{o_2} = \{c, d, e\}$, e os eventos "b" e "c" para b_{R_3} e c_{R_3} em G_3, pois $E_{o_3} = \{a, d, e\}$.

O próximo passo do algoritmo 3.3 é encontrar o autômato de falha G_{F_i} para $i = 1, 2, 3$. Seguindo os passos 2.1 a 2.3, são obtidos os autômatos G_{F_1}, G_{F_2} e G_{F_3} representados na figura 3.8. Note que $\Sigma_1 = R_1(\Sigma) = \{a, b, c, d_{R_1}, e_{R_1}, \sigma_f\}$, $\Sigma_2 = R_2(\Sigma) = \{a_{R_2}, b_{R_2}, c, d, e, \sigma_f\}$ e $\Sigma_3 = R_3(\Sigma) = \{a, b_{R_3}, c_{R_3}, d, e, \sigma_f\}$ e, $\Sigma_o =$

$\bigcup_{i=1}^{3} \Sigma_{o_i} = \{a, b, c, d, e\}$, *portanto* $\Sigma_{F_1} = \Sigma_1 \cup \Sigma_o = \{a, b, c, d, e, d_{R_1}, e_{R_1}, \sigma_f\}$, $\Sigma_{F_2} = \Sigma_2 \cup \Sigma_o = \{a, b, c, d, e, a_{R_2}, b_{R_2}, \sigma_f\}$ *e* $\Sigma_{F_3} = \Sigma_3 \cup \Sigma_o = \{a, b, c, d, e, b_{R_3}, c_{R_3}, \sigma_f\}$.

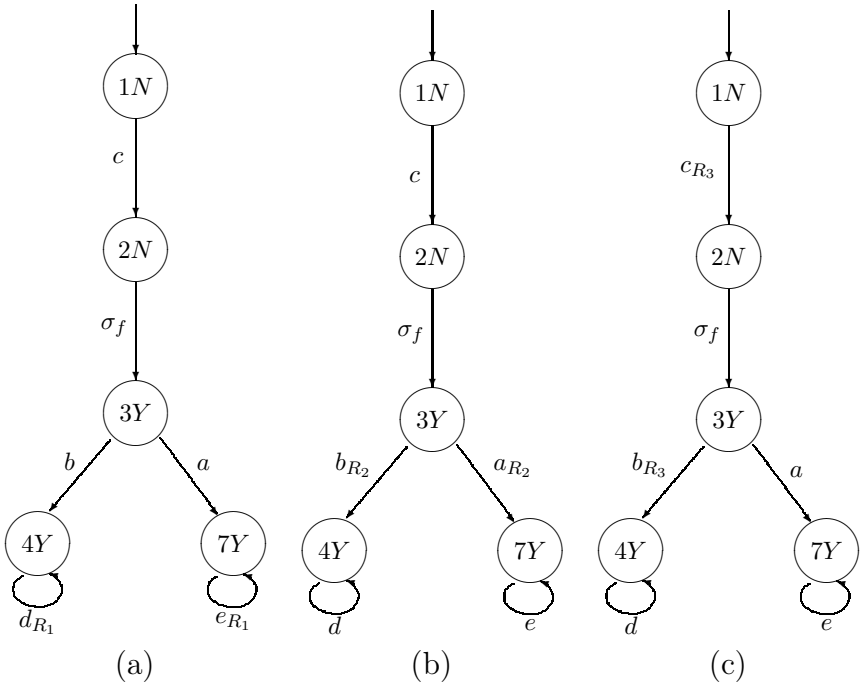


Figura 3.8: Autômatos G_{F_1} (a), G_{F_2} (b), G_{F_3} (c).

O passo 3 do algoritmo 3.3 é obter os autômatos de não falha G_{N_1}, G_{N_2} *e* G_{N_3} *que contêm o comportamento de não falha de* G_1, G_2 *e* G_3, *respectivamente. Os autômatos* G_{N_1}, G_{N_2} *e* G_{N_3} *estão mostrados na figura 3.9.*

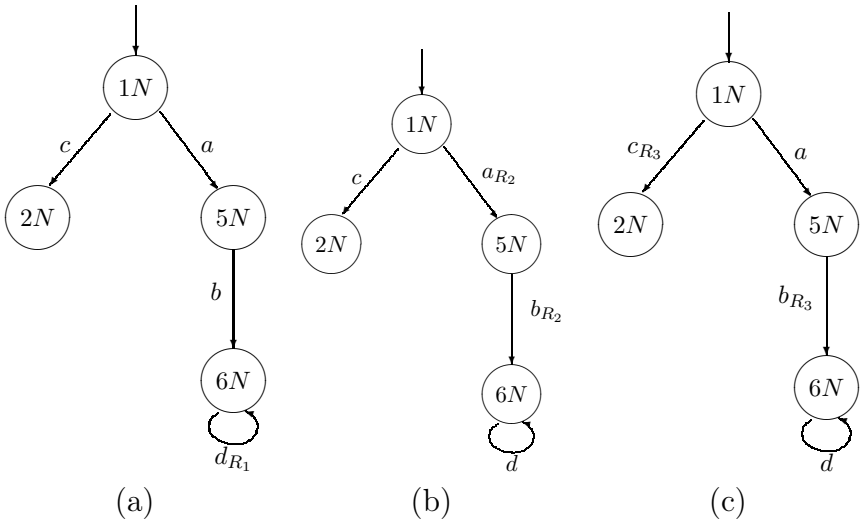


Figura 3.9: Autômatos G_{N_1} (a), G_{N_2} (b), G_{N_3} (c).

A próxima etapa é adicionar um estado a G_{N_1}, G_{N_2} *e* G_{N_3} *como descrito no passo 4 e obter o autômato aumentado* $G^a_{N_1}$, $G^a_{N_2}$ *e* $G^a_{N_3}$ *mostrado na figura 3.10.*

Note que $\Sigma^a_{N_1} = \Sigma_{F_1} \setminus \{\sigma_f\}$, $\Sigma^a_{N_2} = \Sigma_{F_2} \setminus \{\sigma_f\}$ *e* $\Sigma^a_{N_3} = \Sigma_{F_3} \setminus \{\sigma_f\}$.

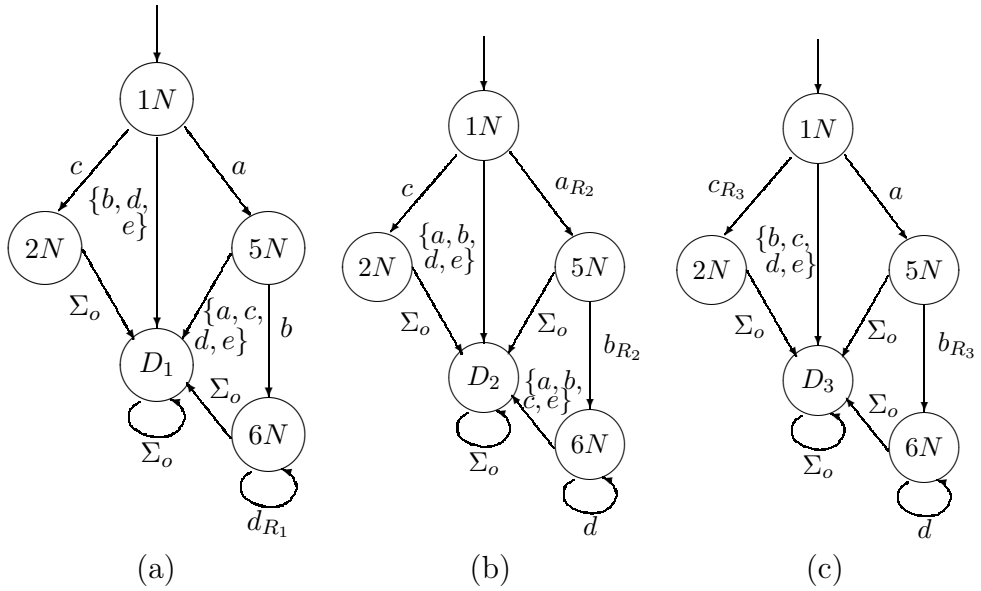


Figura 3.10: Autômatos $G^a_{N_1}$ (a), $G^a_{N_2}$ (b), $G^a_{N_3}$ (c).

Conforme mencionado no passo 5 do algoritmo 3.3, $G_{V_i} = G_{F_i} \| G^a_{N_j} \| G^a_{N_k}$, $i = 1, 2, 3$, $j, k \in \{1, 2, 3\} \setminus \{i\}$, *exceto quando* G_{V_i} *alcançar um estado de* $x_{F_i}\, D_j\, D_k$ *gerando, então,* $\Gamma(x_{F_i}\, D_j\, D_k) = \emptyset$. *Isso pode ser visto nos estados* $\{7Y\, D_2\, D_3\}$ *e* $\{4Y\, D_2\, D_3\}$ *de* G_{V_1} *e* G_{V_2} *e* $\{4Y\, D_1\, D_2\}$ *de* G_{V_3}. *As figuras 3.11, 3.12 e 3.13 mostram os diagramas de transições de estados dos verificadores* G_{V_1}, G_{V_2} *e* G_{V_3}, *respectivamente.*

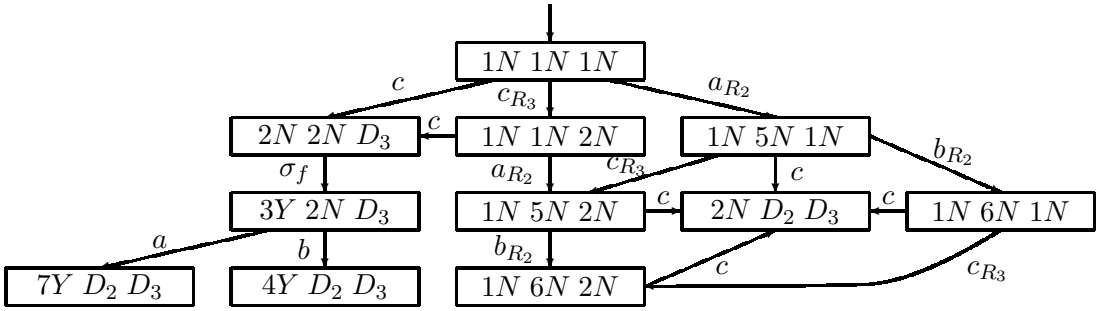


Figura 3.11: Autômato verificador G_{V_1}.

Observe que L *não é robustamente diagnosticável com respeito a* Σ_{rob} *e* Σ_f *uma vez que existe um ciclo* $(4Y\, D_1 6N, d, 4Y\, D_1 6N)$ *em* G_{V_3} *formado por um evento em* Σ_3. *Note que existe uma sequência* $s_{F_3} = c_{R_3}\sigma_f b_{R_3} d^n$, $n \geq 1$, *contendo a falha em* G_3 *e uma outra sequência* $s^a_{N_2} = a_{R_2} b_{R_2} d^m$, $m \in \mathbb{N}$, *não contendo a falha em* G_2, *e após*

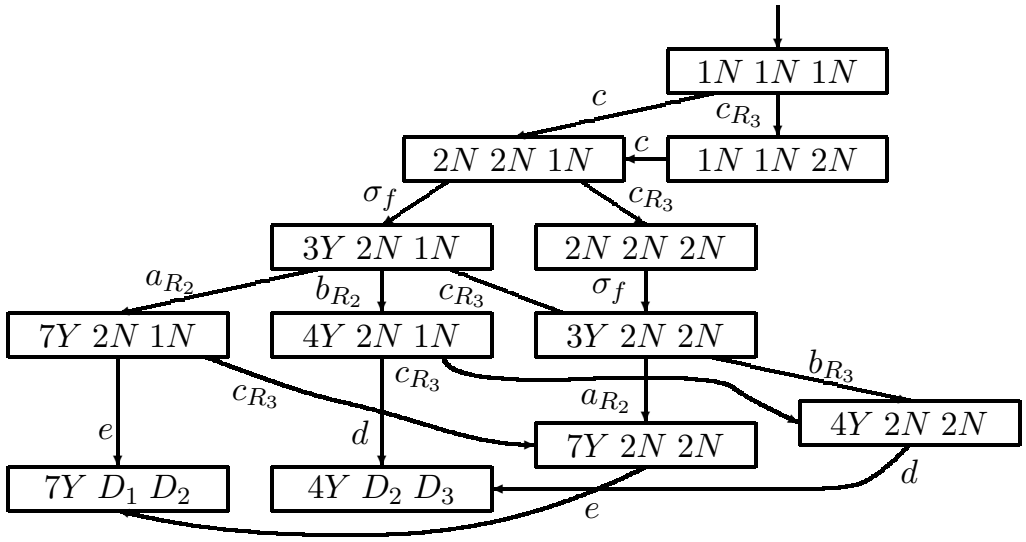


Figura 3.12: Autômato verificador G_{V_2}.

a função renomeação inversa, obtêm-se $s = c\sigma_f bd^n$ *e* $w = abd^m$*, respectivamente. Além disso,* $P_{o_3}(s) = P_{o_2}(w) = d^n$ *que implica que quando a sequência de falha* s *ocorrer não será possível concluir se o sistema está no estado* 4 *de* G *pela observação de* P_{o_3}*, ou no estado* 6 *de* G *pela observação de* P_{o_2}*, que é um estado alcançado por um caminho sem falha.*

Na realidade, para verificar se a linguagem L *é robustamente diagnosticável a perdas permanentes de sensores e encontrar as bases que geram todos os ciclos indeterminados é necessário construir a partir das 12 bases de* $\Sigma_{db,max}$ *como descrito no exemplo 3.1, todos os* 12 *verificadores* G_{V_i}*,* $i = 1, 2, \ldots, 12$*. Entretanto, o resultado obtido pelos verificadores* G_{V_1}*,* G_{V_2} *e* G_{V_3} *coincide com o obtido no diagnosticador união mostrado na figura 3.4 que as sequências* $c\sigma_f bd^n$*,* $n \in \mathbb{N}$*, e* abd^m*,* $m \in \mathbb{N}$*, projetados, respectivamente, nas bases* $\{c, d, e\}$ *e* $\{a, d, e\}$ *são sequências ambíguas e que* L *não é robustamente diagnosticável com respeito a* Σ_{rob}*.* □

3.4 Comentários finais

Nesse capítulo foi proposto um verificador de robustez que permite afirmar se a linguagem é robustamente diagnosticável a perdas permanentes de sensores. A principal vantagem da abordagem aqui proposta está no fato de que, como as possíveis

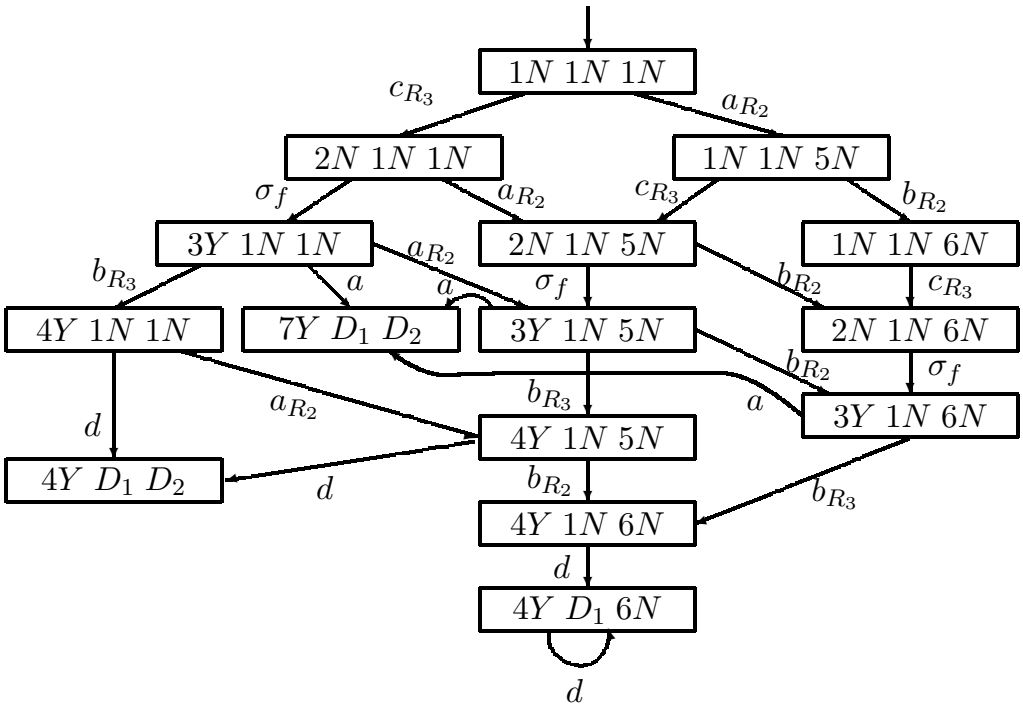


Figura 3.13: Autômato verificador G_{V_3}.

causas de ambiguidades na diagnose de falhas devido à perda de sensores são previstas a priori, é possível projetar outras formas de redundâncias para os sensores que levam a ambiguidades com vistas a melhorar a confiabilidade do sistema de detecção de falhas.

No próximo capítulo, dando continuidade ao tópico de diagnose robusta a perdas de sensores, será abordado o problema de diagnosticabilidade robusta considerando o caso de perdas intermitente de sensores.

Capítulo 4

Diagnose de falhas em SEDs sujeitos a perdas intermitentes de sensores

Na prática, ao se utilizar um diagnosticador para inferir sobre ocorrência de uma falha em um sistema modelado por eventos discretos é suposto que o conjunto de sensores sempre comunica a ocorrência dos eventos a eles associados. Entretanto, conforme visto no capítulo anterior caso um ou mais sensores venham a falhar permanentemente, é possível que o diagnosticador dê um diagnóstico incorreto, isto é, pode ser possível que uma falha tenha ocorrido, porém o diagnosticador interprete como que a sequência de eventos registrada seja de um caminho que não contenha o evento de falha. Isso também se verifica quando ao invés de perdas permanentes de sensores, estes estiverem sujeitos a falhas intermitentes. Por exemplo, considere o autômato G cujo diagrama de transição de estados está mostrado na figura 4.1(a), sendo $\Sigma = \{a, b, c, d, e, \sigma_f\}$, $\Sigma_o = \{a, b, c, d, e\}$ e $\Sigma_f = \{\sigma_f\}$. O correspondente diagnosticador G_d, está representado na figura 4.1(b). De acordo com o teorema 2.2, como G_d não tem ciclos indeterminados, a linguagem gerada por G é diagnosticável com relação a P_o e Σ_f. Considere, inicialmente, a sequência $s'_Y = c\sigma_f abd^n$ $(n \in \mathbb{N})$ e suponha que devido a um defeito intermitente do sensor que registra o evento c este tenha se tornado não-observado. Assim, o primeiro evento a ser registrado por G_d é a, levando o diagnosticador para o estado $\{5N\}$. Quando o próximo evento de

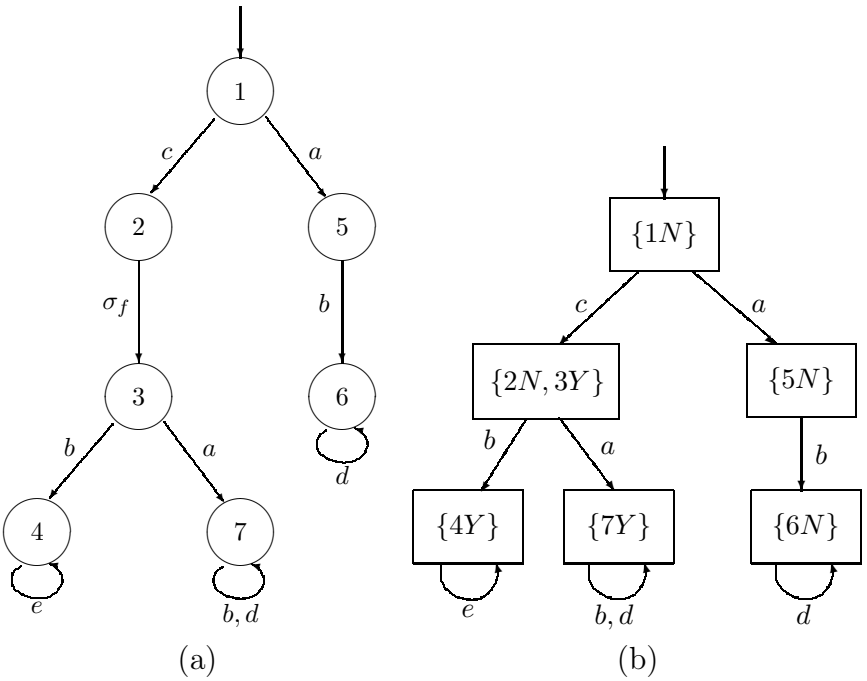


Figura 4.1: Autômato G (a) e seu diagnosticador G_d (b).

s_Y ocorrer, o diagnosticador irá para o estado $\{6N\}$, onde permanecerá mesmo que o evento d ocorra. Considere, agora, a sequência $s''_Y = c\sigma_f be^n$ $(n \in \mathbb{N})$ e suponha que o evento c tenha se tornado não observável devido ao mau funcionamento dos sensores. Nesse caso, o evento b é o primeiro evento a ser registrado por G_d, e como ele não pertence ao conjunto dos eventos ativos de $\{1N\}$, G_d permanece no estado inicial. O próximo evento a ocorrer é o evento e que também não pertence ao conjunto dos eventos ativos de $\{1N\}$, e portanto, o diagnosticador permanece em um estado normal, e, mais uma vez, o diagnosticador fornece indicação errada sobre a ocorrência da falha. Esse comportamento inadequado sugere que o diagnosticador deve também ser modificado a fim de considerar perdas intermitentes de sensores.

Esse capítulo aborda o problema da diagnose de falhas em presença de perdas intermitentes de sensores. Essa abordagem é uma formulação mais geral, uma vez que perdas permanentes de sensores podem ser vistas como perdas intermitentes que durem para sempre. Será proposto um modelo por autômatos para SEDs sujeitos a perdas intermitentes de sensores. Esse modelo terá como base uma nova operação, a dilatação (Carvalho *et al.*, 2010). Além disso, são apresentadas condições necessárias e suficientes para a diagnose robusta a perdas intermitentes de sensores centralizada

e descentralizada utilizando tanto diagnosticadores como verificadores.

Esse capítulo está estruturado da seguinte forma. Na seção 4.1 é proposta uma modelagem por autômatos para SEDs sujeitos a perdas intermitentes de sensores. Na seção 4.2 é considerado o problema da diagnose de falhas em presença de perdas intermitentes de sensores. Na seção 4.3 são apresentadas condições necessárias e suficientes para a diagnose de falha a perdas intermitentes de sensores utilizando diagnosticadores. Serão propostos dois caminhos para a construção do diagnosticador que verifica a diagnosticabilidade de uma linguagem sujeita a perdas intermitentes de sensores. Na seção 4.4 é considerada a verificação de diagnosticabilidade robusta utilizando verificadores e na seção 4.5 é feita a extensão dos resultados obtidos nas seções anteriores para codiagnose robusta utilizando diagnosticadores e verificadores. Por fim, na seção 4.6 as conclusões são apresentadas.

4.1 Modelagem de SEDs sujeito a perdas intermitentes de sensores

Uma abordagem mais geral para o problema de diagnóstico de falhas em presença de perdas de sensores é considerar falhas intermitentes nesse componentes. Essa abordagem tem a vantagem de se levar em conta tanto as perdas intermitentes quanto as permanentes, uma vez que essa última pode ser vista como uma perda intermitente de duração infinita.

Para ilustrar a ocorrência das perdas intermitentes de sensores, considere novamente o autômato da figura 4.1(a) e suponha a seguinte partição de Σ_o: $\Sigma_o = \Sigma_{isf} \dot{\cup} \Sigma_{nf}$, em que Σ_{isf} é o subconjunto de Σ_o cujos eventos estão associados às perdas intermitentes de sensores e Σ_{nf} é o conjunto de eventos associados a sensores que não estão sujeitos a um mau funcionamento. Supondo inicialmente que $\Sigma_{isf} = \emptyset$, então $L(G) = \overline{L_G}$ em que $L_G = \{a\}\{b\}\{d\}^* \cup \{c\}\{\sigma_f\}(\{b\}\{e\}^* \cup \{a\}\{b,d\}^*)$. Suponha agora que $\Sigma_{isf} = \{c\}$ e defina $\Sigma'_{isf} = \{c'\}$, em que c' é um evento não observável que modela a perda de observabilidade de c. Então a linguagem que modela o compor-

tamento de G sujeito à perda intermitente de observabilidade de c não é mais $L(G)$ mas $L_{isf} = L(G) \cup \overline{L_A}$, sendo $L_A = \{c'\}\{\sigma_f\}(\{b\}\{e\}^* \cup \{a\}\{b,d\}^*)$. O diagrama de transição de estados de um autômato que gera L_{isf} está representado na figura 4.2.

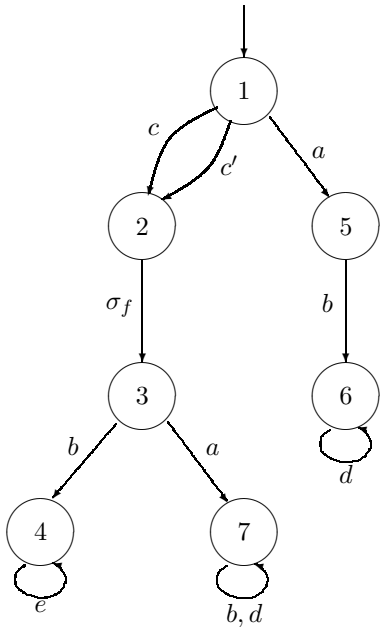


Figura 4.2: Autômato G_{isf}

Como pode ser visto no exemplo acima, a diagnosticabilidade de uma linguagem gerada por um SED sujeito a perdas intermitentes de observabilidade não deve ser analisada em termos de L, mas em termos de L_{isf}. Como consequência, é necessário obter um autômato G_{isf} cuja linguagem gerada não represente apenas o comportamento normal de G, mas que também considere a influencia das perdas intermitentes de sensores em L. Isso pode ser feito, conforme visto na figura 4.2, adicionando-se uma transição em paralelo com a transição associada ao evento cujo sensor está sujeito ao mau funcionamento e rotulada por um evento não observável. Como consequência, o autômato resultante G_{isf} será determinístico.

Antes de definir formalmente G_{isf}, considere a seguinte operação.

Definição 4.1 *(Dilatação) Suponha que* $\Sigma = \Sigma_{isf} \dot{\cup} \Sigma_{nf} \dot{\cup} \Sigma_{uo}$ *em que* Σ_{isf} *é o conjunto de eventos observáveis associados a sensores que falham de forma intermitente e* Σ_{nf} *é o conjunto de eventos observáveis associados a sensores que mantêm um comportamento normal durante todo o período de funcionamento e seja*

$\Sigma'_{isf} = \{\sigma' : \sigma \in \Sigma_{isf}\}$. *A dilatação* D_{isf} *é um mapeamento definido como:*

$$\begin{array}{rcccl} D_{isf} & : & \Sigma^* & \rightarrow & 2^{(\Sigma \cup \Sigma'_{isf})^*} \\ & & s & \mapsto & D_{isf}(s), \end{array} \tag{4.1}$$

em que

$$\begin{aligned} D_{isf}(\varepsilon) &= \{\varepsilon\}, \\ D_{isf}(\sigma) &= \begin{cases} \{\sigma\}, \ se\ \sigma \in \Sigma \backslash \Sigma_{isf}, \\ \{\sigma, \sigma'\}, \ se\ \sigma \in \Sigma_{isf}, \end{cases} \\ D_{isf}(s\sigma) &= D_{isf}(s) D_{isf}(\sigma), s \in \Sigma^*, \sigma \in \Sigma. \end{aligned} \tag{4.2}$$

□

A operação dilatação D_{isf} pode ser estendida para linguagens aplicando-se o operador D_{isf} a todas as sequências da linguagem, isto é,

$$D_{isf}(L) = \bigcup_{s \in L} D_{isf}(s). \tag{4.3}$$

Considere $\Sigma' = \Sigma \cup \Sigma'_{isf}$ e $\Sigma'_o = \Sigma_o \cup \Sigma'_{isf}$, e defina as seguintes projeções: $P'_o : \Sigma'^* \rightarrow \Sigma_o^*$ e $P'_{oo} : \Sigma'^*_o \rightarrow \Sigma_o^*$. Podemos apresentar o seguinte resultado.

Lema 1

A. Para todo evento $\sigma \in \Sigma$, $P'_{oo}[D_{isf}[P_o(\sigma)]] = P'_o[D_{isf}(\sigma)]$.

B. Para toda linguagem L, *definida em* Σ^*, $P'_{oo}[D_{isf}[P_o(L)]] = P'_o[D_{isf}(L)]$.

Demonstração:

A demostração para A.) segue diretamente da definição de dilatação e projeção, e será portanto omitida.

A demostração para B.) é por indução no comprimento das sequências nas duas linguagens.

• O caso base é para a sequência de comprimento 0. Para $s = \varepsilon$, tem-se que, pela definição, $P'_{oo}[D_{isf}[P_o(\varepsilon)]] = \varepsilon$ e $P'_o[D_{isf}(\varepsilon)] = \varepsilon$.

• A hipótese de indução é que para toda sequência $s_N \in L$, $\|s_N\| \leq N$, $P'_{oo}[D_{isf}[P_o(s_N)]] = P'_o[D_{isf}(s_N)]$.

• Considere, agora, a sequência $s_{N+1} = s_N\sigma$, em que $\sigma \in \Sigma$. Então,

$$\begin{aligned} P'_{oo}[D_{isf}[P_o(s_{N+1})]] &= P'_{oo}[D_{isf}[P_o(s_N\sigma)]] \\ &= P'_{oo}[D_{isf}[P_o(s_N)]]P'_{oo}[D_{isf}[P_o(\sigma)]] \\ &= P'_o[D_{isf}(s_N)]P'_o[D_{isf}(\sigma)] \\ &= P'_o[D_{isf}(s_N\sigma)] \\ &= P'_o[D_{isf}(s_{N+1})]. \end{aligned} \tag{4.4}$$

□

Com a ajuda da definição 4.1, pode se definir G_{isf} como segue:

$$G_{isf} = (X, \Sigma', f_{isf}, \Gamma_{isf}, x_0), \tag{4.5}$$

em que $\Sigma' = \Sigma \cup \Sigma'_{isf}$, $\Gamma_{isf}(x) = D_{isf}[\Gamma(x)]$, e f_{isf} é definido como segue: $\forall \sigma_{isf} \in \Gamma_{isf}(x) : \sigma_{isf} \in D_{isf}(\sigma)$, $f_{isf}(x, \sigma_{isf}) = f(x, \sigma)$.

Observação 4.1 *É importante observar que o conjunto dos eventos observáveis de G_{isf} permanece Σ_o, como em G.* □

O seguinte resultado mostra que G_{isf} modela o comportamento de G quando sujeito a perdas intermitentes de sensores.

Teorema 4.1 *Seja G_{isf} um autômato determinístico obtido a partir de G de acordo com equação (4.5). Então, $L_{isf} = L(G_{isf}) = D_{isf}(L)$.*

Demonstração: A prova é por indução:

• O caso base é para a sequência de comprimento 0. Observe que por definição $\varepsilon \in L_{isf}$, e, como $\varepsilon \in L$ e $D_{isf}(\varepsilon) = \{\varepsilon\}$, então $\varepsilon \in D_{isf}(L)$.

• A hipótese de indução é que: $\forall s_n \in L_{isf}$, $\|s_n\| \leq n \Leftrightarrow s_n \in D_{isf}(L) \Leftrightarrow \exists s'_n \in L : s_n \in D_{isf}(s'_n)$.

• Seja $s_{n+1} = s_n\sigma_{isf}$.

(1) Considere primeiro que $s_{n+1} = s_n\sigma_{isf} \in L_{isf}$. Por construção de G_{isf}, $\sigma_{isf} \in \Sigma'$ e então $\exists \sigma \in \Sigma : \sigma_{isf} \in D_{isf}(\sigma) = \{\sigma, \sigma'\}$, que implica que $\sigma_{isf} = \sigma$ ou

$\sigma_{isf} = \sigma'$. Assim, utilizando a hipótese de indução, tem-se que $\exists s'_{n+1} = s'_n\sigma \in L$, em que $s_n \in D_{isf}(s'_n)$, e, então $s_{n+1} = s_n\sigma_{isf} \in D_{isf}(s'_n\sigma) \subseteq D_{isf}(L)$, que implica que $s_{n+1} \in D_{isf}(L)$.

(2) Considere agora que $s_{n+1} = s_n\sigma_{isf} \in D_{isf}(L)$. A partir da hipótese de indução $\exists s'_n \in L : s_n \in D_{isf}(s'_n)$ e como $\sigma_{isf} \in \Sigma'$ então $\exists \sigma \in \Sigma : \sigma_{isf} \in D_{isf}(\sigma)$. Portanto,

$$\begin{aligned} D_{isf}(s'_n\sigma) &= D_{isf}(s'_n)D_{isf}(\sigma) \supset \{s_n\}D_{isf}(\sigma) \\ &= \begin{cases} \{s_n\sigma\}, \sigma \notin \Sigma_{isf} \\ \{s_n\sigma, s_n\sigma'\}, \sigma \in \Sigma_{isf} \end{cases}, \end{aligned}$$

que implica, pela construção de G_{isf} que $s_{n+1} = s_n\sigma_{isf} \in L_{isf}$. □

4.2 Diagnosticabilidade robusta de SEDs sujeitos a perdas intermitentes de sensores

A definição de diagnosticabilidade (Sampath *et al.*, 1995) é formalizada em termos da linguagem observada. Entretanto, como dito na seção anterior, embora a linguagem gerada por uma autômato sujeito a perdas intermitentes de sensores permanece inalterada, sua linguagem observada é dilatada. Então, a fim de se considerar perdas intermitentes de sensores, a definição de diagnosticabilidade precisa ser modificada.

Definição 4.2 *(Diagnosticabilidade robusta de SEDs sujeito a perdas intermitentes de sensores) Uma linguagem L viva e prefixo-fechada, gerada por um autômato G, é robustamente diagnosticável em relação à dilatação D_{isf}, à projeção P'_o e a $\Sigma_f = \{\sigma_f\}$ se a seguinte condição for verificada:*

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

sendo a condição de diagnosticabilidade robusta R_D expressa por

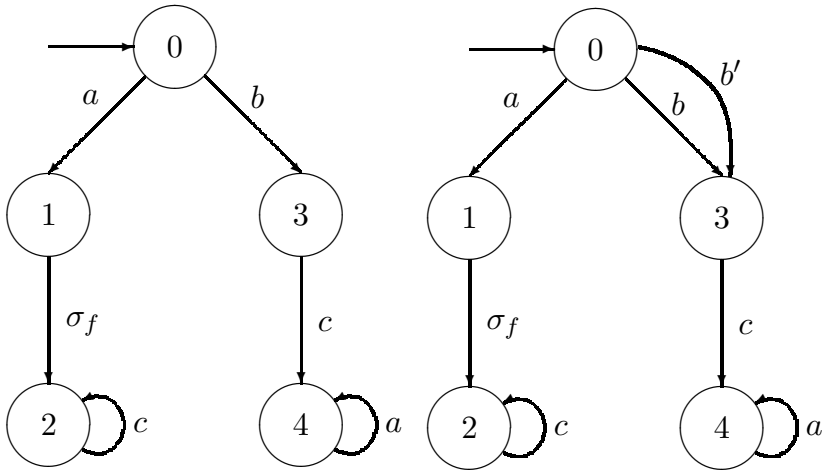
$$(\forall \omega \in P_o'^{-1}(P_o'(D_{isf}(st))) \cap L_{isf})(\Sigma_f \in \omega), \tag{4.6}$$

em que $P_o' : \Sigma'^* \rightarrow \Sigma_o^*$, $\Sigma' = \Sigma \cup \Sigma_{isf}'$. □

Observação 4.2 *Note que se $\Sigma_{isf} = \emptyset$, então $L_{isf} = L$, $D_{isf}(st) = \{st\}$ e P_o' reduz-se a P_o. Nesse caso, a definição 4.2 se reduz à definição usual de diagnosticabilidade introduzida por Sampath et al. (1995).* □

O exemplo a seguir ilustra a verificação da diagnosticabilidade robusta de acordo com a definição 4.2.

Exemplo 4.1 *Considere os autômatos G_1 e G_2 cujos diagramas de transição estão representados nas figuras 4.3(a) e 4.4(a), respectivamente. Para ambos os autômatos, suponha que $\Sigma_o = \{a, b, c\}$, $\Sigma_{isf} = \{b\}$ e $\Sigma_{uo} = \Sigma_f = \{\sigma_f\}$. O objetivo aqui é verificar se as linguagens geradas por G_1 e G_2 (L_1 e L_2, respectivamente) são robustamente diagnosticável com relação a D_{isf}, P_o' e $\Sigma_f = \{\sigma_f\}$.*



(a) Autômato G_1 (b) Autoômato $G_{1_{isf}}$

Figura 4.3: Autômatos G_1 e $G_{1_{isf}}$ do exemplo 4.1.

Considere, inicialmente, o autômato G_1. Observe na figura 4.3(a) que a única sequência que contém o evento de falha de L_1 é $s_Y = a\sigma_f c^n$, $n \in \mathbb{N}$. Seguindo os

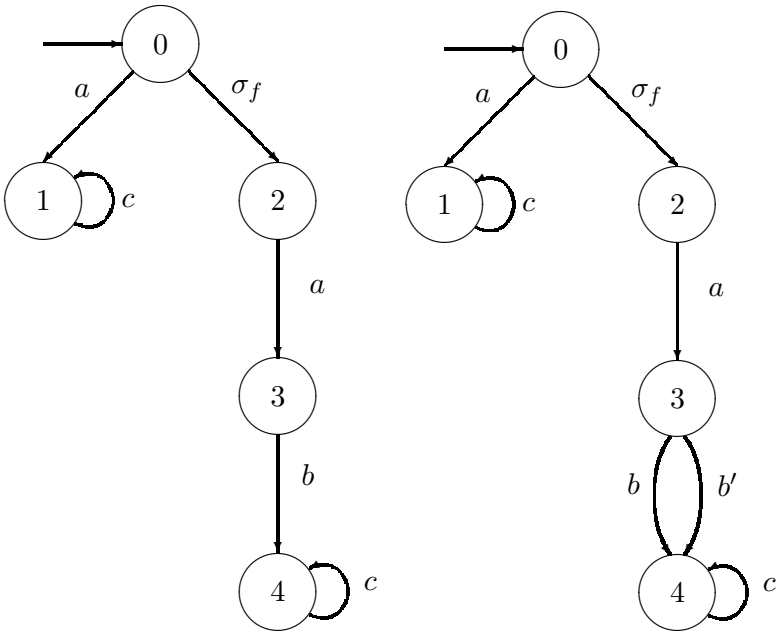
passos da condição de diagnosticabilidade robusta R_D, tem-se que:

$$D_{isf}(s_Y) = \{a\sigma_f c^n\} \Rightarrow P'_o[D_{isf}(s_Y)] = \{ac^n\}.$$

Seja $L_{1_{isf}}$ a linguagem gerada pelo autômato $G_{1_{isf}}$, mostrado na figura 4.3(b). Não é difícil verificar que:

$$P_o'^{-1}\{P'_o[D_{isf}(s_Y)]\} \cap L_{1_{isf}} = \{a\sigma_f c^n\}.$$

Portanto, como $P_o'^{-1}\{P'_o[D_{isf}(s_Y)]\} \cap L_{1_{isf}}$ produz somente sequências que contêm o evento de falha, pode-se concluir que L_1 é diagnosticável em relação a D_{isf}, P'_o e Σ_f.



(a) Autômato G_2 (b) Autômato $G_{2_{isf}}$

Figura 4.4: Autômatos G_2 e $G_{2_{isf}}$ do exemplo 4.1.

Considere agora o autômato G_2. Nesse caso, a única sequência que contém o evento de falha de L_2 é $s_Y = \sigma_f abc^n$, $n \in \mathbb{N}$. Procedendo de acordo com a condição de diagnosticabilidade robusta R_D, tem-se:

$$D_{isf}(s_Y) = \{\sigma_f abc^n, \sigma_f ab'c^n\} \Rightarrow P'_o[D_{isf}(s_Y)] = \{abc^n, ac^n\}.$$

A partir do autômato $G_{2_{isf}}$, representado na figura 4.4(b), pode-se obter $L_{2_{isf}}$, e portanto:

$$P_o'^{-1}\left\{P_o'\left[D_{isf}(s_Y)\right]\right\} \cap L_{2_{isf}} = \{\sigma_f abc^n, \sigma_f ac^n, ac^n\}.$$

Como existe uma sequência normal em $P_o'^{-1}\left\{P_o'\left[D_{isf}(s_Y)\right]\right\} \cap L_{2_{isf}}$, pode-se concluir que L_2 não é diagnosticável em relação a D_{isf}, P_o' e Σ_f. □

4.3 Verificação da diagnosticabilidade robusta utilizando diagnosticadores

4.3.1 Diagnosticador para verificação de robustez de SEDs sujeitos a perdas intermitentes de sensores

A condição de diagnosticabilidade robusta R_D, dada pela equação (4.6), substitui G, $L = L(G)$ e $P_o : \Sigma^* \to \Sigma_o^*$ por G_{isf}, $L_{isf} = D_{isf}(L) = L(G_{isf})$ e $P_o' : \Sigma'^* \to \Sigma_o^*$ ($\Sigma' = \Sigma \cup \Sigma_{isf}'$), respectivamente. Isto sugere que a diagnosticabilidade de uma linguagem sob perdas intermitentes de sensores pode ser verificada utilizado-se a mesma estratégia de Sampath *et al.* (1995).

Considere o seguinte diagnosticador obtido a partir de G_{isf}:

$$G_{isf,d} = Obs(G_{isf}||A_\ell, \Sigma_o), \tag{4.7}$$

em que A_ℓ é o autômato rotulador de falhas mostrado na figura 2.5. Inspirado no teorema 2.1, será apresentada uma condição necessária e suficiente para a diagnosticabilidade de uma linguagem sujeita a perdas intermitentes de observabilidade. Para tanto, considere a seguinte definição de ciclos escondidos.

Definição 4.3 *(Ciclos escondidos e ciclos escondidos indeterminados de G_d) Seja $x_d = \{x_1\ell_1, x_2\ell_2, \ldots, x_n\ell_n\}$ um estado de G_d. Então, existe um ciclo escondido em x_d se para algum $\{i_1, i_2, \ldots, i_k\} \subseteq \{1, 2, \ldots, n\}$, as seguintes condições forem verificadas:*

HC.1) *$x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ forma um ciclo em G;*

HC.2) *Existe um conjunto de eventos $\{\sigma_{i_1}, \sigma_{i_2}, \ldots, \sigma_{i_k}\} \subseteq \Sigma_{uo}$, em que $\sigma_{i_1}, \sigma_{i_2}, \ldots, \sigma_{i_k}$ são tais que $f(x_{i_j}, \sigma_{i_j}) = x_{i_{j+1}}$, $j = 1, 2, \ldots, k-1$, e $f(x_{i_k}, \sigma_{i_k}) = x_{i_1}$.*

Se x_d for um estado incerto de G_d, e além das condições HC.1) e HC.2), a seguinte condição também for verificada,

HC.3) *$\ell_{i_j} = Y$, $j = 1, 2, \ldots, k$,*

então x_d terá um ciclo escondido indeterminado. □

Observação 4.3 *Note que a definição 2.6 supõe que o sistema já seja diagnosticável com relação a P_o e Σ_f (hipótese A.4) e os ciclos escondidos são formados em G'_d pelo agrupamento dos estados do diagnosticador G_d devido à perda de observabilidade dos eventos de $\Sigma \setminus \Sigma'_o$ uma vez que o novo conjunto de eventos observáveis é $\Sigma'_o \subset \Sigma_o$ (hipótese A.2). Entretanto a definição 4.3 considera que os ciclos escondidos em G_d provêm do agrupamento dos estados de G devido a ciclos não observáveis em G. Note que de acordo com a definição 2.6, existe um ciclo escondido no diagnosticador G'_d em $x'_d = \{x_{d_1}, x_{d_2}, \ldots, x_{d_n}\}$ 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. Consequentemente existirá um ciclo de estados em G tais que os eventos que ligam esses estados são não-observáveis em relação a $\Sigma \setminus \Sigma'_o$, que é a definição de ciclos escondidos na definição 4.3. Por outro lado, como a definição 2.6 parte de G_d e considera a hipótese A.2, não é possível que se tenha em G_d um estado x_d no qual as condições HC.1) e HC.2) sejam satisfeitas. Note ainda que de acordo com a definição 2.6 de ciclos escondidos indeterminados 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. Vale a pena observar que estados certos no diagnosticador ocorrem quando o diagnosticador alcança um estado cujos rótulos são todos iguais a Y. Isso equivale a dizer que os rótulos de todos os estados que estão no ciclo escondido são Y, e, portanto, a condição HC.3) não pode também*

ser verificada, uma vez que a definição 2.6 considera a hipótese A.4. Dessa forma, pode-se concluir que a definição 4.3 é mais geral qua a definição 2.6. □

Teorema 4.2 *Uma linguagem L é robustamente diagnosticável com relação a D_{isf}, P'_o e Σ_f se, e somente se, o diagnosticador $G_{isf,d}$ não tiver nenhum ciclo indeterminado (observado ou escondido).*

Demonstração: ($\Leftarrow$) Suponha que L não seja robustamente diagnosticável com relação a D_{isf}, P'_o e Σ_f. Então, de acordo com a definição de diagnosticabilidade robusta, para alguma sequência $s \in \Psi(\Sigma_f)$ e algum $t \in L_{isf}/s$, $\|t\| \geq n$, n arbitrariamente grande, existe uma sequência $w \in P_o'^{-1}(P'_o(D_{isf}(st))) \cap L_{isf}$ tal que $\Sigma_f \neq w$, em que w pode ter comprimento limitado ou ilimitado.

Considere, inicialmente, que w tenha comprimento limitado. Como $P'_o[D_{isf}(w)] = P'_o[D_{isf}(st)]$ e $G_{isf,d}$ é um autômato determinístico, então existe um estado incerto $x_{isf,d} \in X_{isf,d}$ tal que

$$x_{isf,d} = f_{isf,d}(x_{0,isf,d}, P'_o[D_{isf}(st)]) = f_{isf,d}(x_{0,isf,d}, P'_o[D_{isf}(w)]).$$

Suponha que $|X_{isfd}| = N_{isf}$. Se $n > N_{isf}$, então existirá um ciclo de estados cujos eventos associados às transições dos estados são observáveis ou não-observáveis. Se os eventos forem observáveis, $P'_o[D_{isf}(w)] \neq P'_o[D_{isf}(st)]$, que contradiz a hipótese inicial. Por outro lado, se os eventos forem não-observáveis, então existirá um ciclo indeterminado escondido em $x_{isf,d} \in X_{isf,d}$.

Suponha, agora, que w seja ilimitada. Se $\|st\| > N_{isf}$, então st e w necessariamente formarão ciclos de estados e, como $P'_o[D_{isf}(st)] = P'_o[D_{isf}(w)]$, ciclos indeterminados observados serão formados em $G_{isf,d}$.

($\Rightarrow$) Suponha, inicialmente, que existe um ciclo indeterminado escondido em $G_{isf,d}$, em um estado $x_{isf,d} = \{x_1\ell_1, x_2\ell_2, \ldots, x_n\ell_n\} \in X_{isf,d}$, no qual x_i é um estado de G_{isf}. Portanto, de acordo com a definição 4.3, existe $\{i_1, i_2, \ldots, i_k\} \subseteq \{1, 2, \ldots, n\}$ tal que $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ forma um ciclo em G_{isf}, e $\ell_{i_j} = Y$, $j = 1, 2, \ldots, k$. Então, não é difícil ver que existe uma sequência $stu_p \in L_{isf}$ que satisfaz as seguintes condições:

1) $\Sigma_f \in s$ e $f_{isf,d}(x_{0_isf,d}, P'_o(s)) = x_{isf,d}$;

2) $t \in (\Sigma'_{isf} \cup \Sigma_{uo})^*$ tal que $f_{isf,d}(x_{0_isf,d}, P'_o(st)) = x_{isf,d}$;

3) $u_p \in (\Sigma'_{isf} \cup \Sigma_{uo})^*$, $\|u_p\| = p$, com p arbitrariamente longo, é tal que $f_{isf,d}(x_{0_isf,d}, P'_o(stu_p)) = x_{isf,d}$ e $f_{isf}(x_{i_1}\ell_{i_1}, u_p) = x_{i_j}\ell_{i_j}$, em que $j = (p \mod k) + 1$. Além disso, como $x_{isf,d}$ é um estado incerto, existe $w \in L$ tal que $\Sigma_f \notin w$ e $P'_o(w) = P'_o(stu_p)$, o que viola a condição de diagnosticabilidade

Considere agora que existe um ciclo indeterminado em $G_{isf,d}$ formado pelos estados $x_1, x_2, \ldots, x_p$. Nesse caso, o resultado é obtido utilizando-se o fato da diagnosticabilidade robusta de L com relação a D_{isf}, P'_o e Σ_f ser equivalente à diagnosticabilidade de $L_{isf} = D_{isf}(L)$ (de acordo com o teorema 4.1) com relação a P'_o e Σ_f. O resultado segue então do teorema 2.1, substituindo-se L por L_{isf}. □

Observação 4.4 *Na modelagem de perdas intermitentes de sensores podem ocorrer ciclos de eventos não-observáveis em G_{isf} que, de acordo com a hipótese A2, não existiam em G. Para entender esse fato, suponha que G tenha um ciclo de estados conectados por eventos não-observáveis, exceto por um único evento observável. Assim, se o sensor associado a esse evento for sujeito a falhas intermitentes, G_{isf} terá um ciclo de estados conectados por eventos não-observáveis, violando, portanto, a hipótese A2. Por essa razão, foi necessário introduzir o conceito de ciclos escondidos (definição 4.3) para enunciar o teorema 4.2. Note que, com a inclusão do conceito de ciclos escondidos, o teorema 2.1 pode ser visto como um caso particular do teorema 4.2, conforme enunciado a seguir.* □

Corolário 4.1 *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 (observados ou escondidos).* □

Para ilustrar o resultado do teorema 4.2, considere o exemplo a seguir.

Exemplo 4.2

(a) *Considere novamente o autômato G da figura 4.1(a), e suponha que o sensor associado ao evento c falhe intermitentemente,* i.e., *$\Sigma_{isf} = \{c\}$. O autômato G_{isf} que*

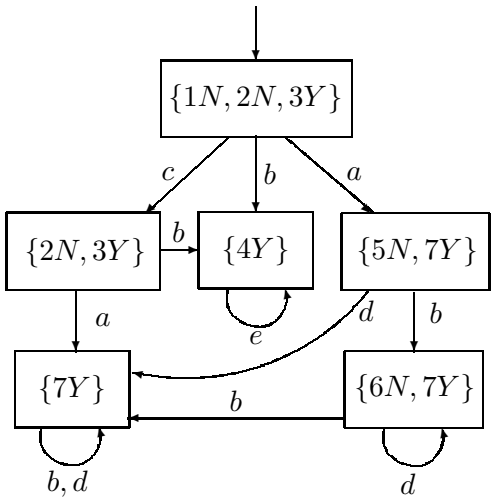


Figura 4.5: Diagnosticador para autômato G da figura 4.1 supondo $\Sigma_{isf} = \{c\}$.

modela o comportamento de G sujeito a perdas intermitentes de sensores é mostrado na figura 4.2 e o seu diagnosticador $G_{isf,d}$ está representado na figura 4.5. Pode-se observar que $G_{isf,d}$ tem um ciclo indeterminado no estado $\{6N, 7Y\}$, e, portanto L não é robustamente diagnosticável com relação a D_{isf}, P'_o and Σ_{isf}. Não é difícil ver que se sequência $s_Y = c\sigma_f abd^n$, n arbitrariamente longo, ocorrer e, além disso, o sensor associado ao evento c falhar, então o diagnosticador $G_{isf,d}$ permanecerá indefinidamente no estado $\{6N, 7Y\}$, portando em dúvida sobre a ocorrência do evento de falha σ_f. Note que a não ocorrência de c é equivalente à ocorrência de c' em G_{isf}, e portanto $s'_Y = c'\sigma_f abd^n$ é uma sequência ambígua de L_{isf} já que tem a mesma projeção da sequência normal $s'_N = abd^n$.

(b) *Suponha, agora, que para o mesmo autômato G da figura 4.1(a), a seja um evento não-observável e b é o evento cujo correspondente sensor estaja sujeito a falhas intermitentes,* i.e., *$\Sigma_{uo} = \{a, \sigma_f\}$ e $\Sigma_{isf} = \{b\}$. O diagnosticador G_d para G é mostrado na figura 4.6(a), na qual é fácil concluir que L é diagnosticável em relação a P_o e Σ_f. O diagnosticador $G_{isf,d}$ que considera a perda intermitente de observabilidade do evento b é representado na figura 4.6(b). Note que como $G_{isf,d}$ tem um ciclo indeterminado escondido no estado $\{2N, 3Y, 4Y, 7Y\}$, pode-se concluir que L não é robustamente diagnosticável em relação a D_{isf}, P'_o e Σ_{isf}. Portanto, se a sequência $s_Y = c\sigma_f ab^n$, n arbitrariamente longo, ocorrer, e o sensor associado a b*

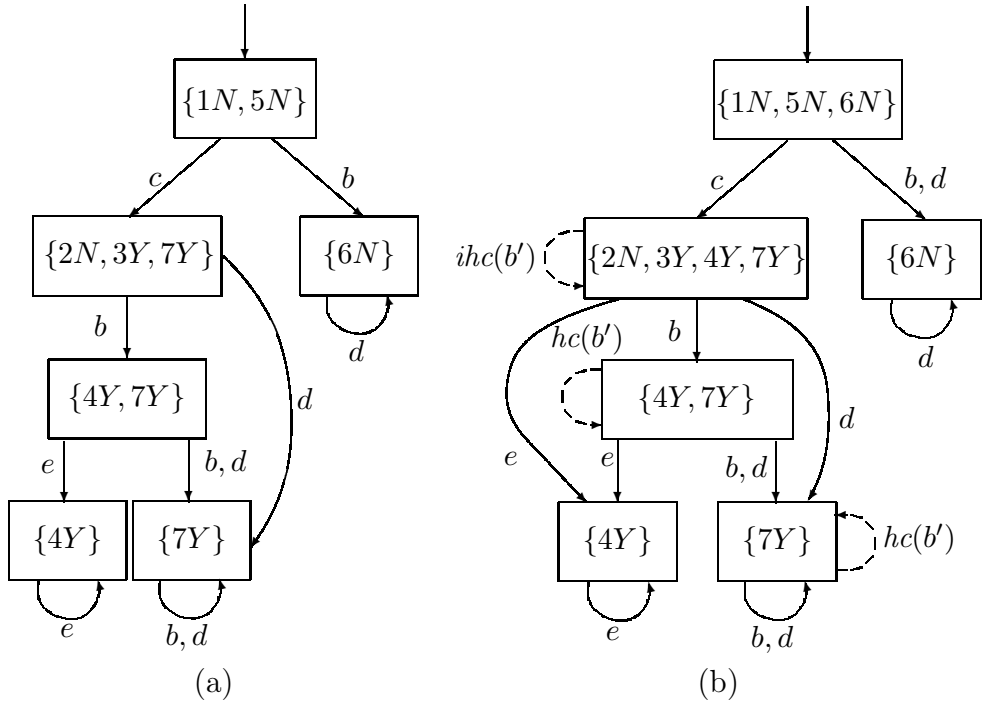


Figura 4.6: Diagnosticador para autômato G da figura 4.1 supondo $\Sigma_{uo} = \{a, \sigma_f\}$ (a), e $\Sigma_{uo} = \{a, \sigma_f\}$ e $\Sigma_{isf} = \{b\}$ (b).

falhar intermitentemente, então $G_{isf,d}$ ficará no estado $\{2N, 3Y, 4Y, 7Y\}$. É importante observar que a falha permanente de b é equivalente a ocorrência infinita de b', então a ocorrência de s_Y em G é equivalente a ocorrência de $s'_Y = c\sigma_f a b'^n$ em G_{isf}, que é uma sequência ambígua à sequência $s_N = c$, já que $P'_o(s'_Y) = P'_o(s_N) = c$. Note que se o sensor associado ao evento b voltar a funcionar, então $G_{isf,d}$ moverá para o estado $\{4Y, 7Y\}$, indicando que o evento de falha σ_f ocorreu; Isso é consistente com o que se espera de um diagnosticador que considera perdas intermitentes de sensores. □

Observação 4.5 *Se a linguagem gerada por um autômato G for robustamente diagnosticável em relação a D_{isf}, P'_o e Σ_f, então o diagnosticador $G_{isf,d}$ não só fornece um teste "off-line" para diagnosticabilidade robusta mas também pode ser utilizado em tempo real para diagnose de um SED sujeito a perdas intermitentes de sensores. Nesse caso, esse diagnosticador será denominado diagnosticador robusto de falhas de SEDs sujeitos a perdas intermitentes de sensores ou simplesmente diagnosticador robusto quando o contexto permitir. Esse fato será ilustrado no exemplo a seguir.* □

Exemplo 4.3 *Considere mais uma vez o autômato G da figura 4.1(a), e suponha que $\Sigma_o = \{a, b, c, d, e\}$. O correspondente diagnosticador G_d é mostrado na figura 4.1(b) de onde se pode concluir que L é diagnosticável em relação a P_o e Σ_f. Suponha que o evento b esteja sujeito a perdas intermitentes de observabilidade,* i.e.*, $\Sigma_{isf} = \{b\}$. O correspondente diagnosticador $G_{isf,d}$ para verificação de diagnose em SEDs sujeitos a perdas intermitentes de sensores está mostrado na figura 4.7. Note que $G_{isf,d}$ não tem ciclos indeterminados (observados ou escondidos). Dessa forma, L é robustamente diagnosticável em relação a D_{isf}, P_o e Σ_f e o diagnosticador $G_{isf,d}$ pode ser referido como robusto.*

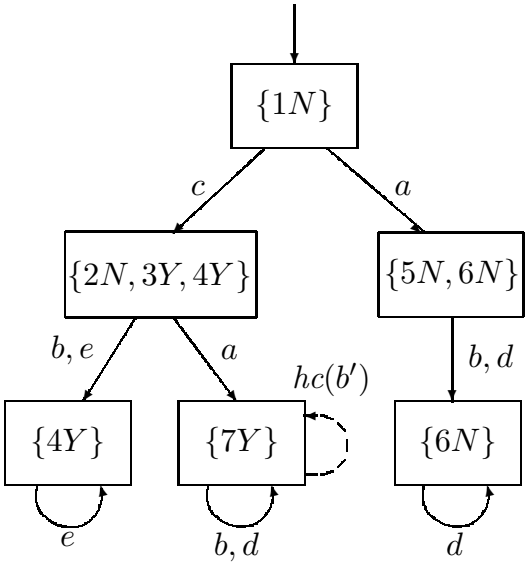


Figura 4.7: Diagnosticador robusto sujeito a perdas intermitentes de sensores do exemplo 4.3.

Considere agora o efeito da perda intermitente do sensor associado ao evento b na detecção de falha "online". Suponha que a sequência de falha $s_Y = c\sigma_f be^n$ ocorra. Utilizando o diagnosticador G_d da figura 4.1(b), tem-se que quando o sensor associado ao evento b falhar, G_d ficará estacionado no estado $\{2N, 3Y\}$, ficando, definitivamente incerto sobre a ocorrência da falha. Ao utilizar o diagnosticador robusto $G_{isf,d}$ da figura 4.7, tem-se que após a ocorrência do evento c, $G_{isf,d}$ move-se para o estado $\{2N, 3Y, 4Y\}$. Suponha que o sensor que identifica a ocorrência do evento b falhe. Neste caso, o próximo evento a ser observado por $G_{isf,d}$ é o evento e. Quando e ocorrer, o diagnosticador robusto irá para o estado $\{4Y\}$, estando portanto

certo da ocorrência da falha. Por outro lado, se o sensor associado ao evento b não falhar, então b será o primeiro evento reconhecido por $G_{isf,d}$ após a ocorrência do evento c; portanto $G_{isf,d}$ move-se para o estado $\{4Y\}$, e permanece nele uma vez que o próximo evento a ocorrer em s_Y é o evento e.

Note que em ambos os casos, quando o sensor que está associado ao evento b falhar ou não, o diagnosticador robusto não só detectará a ocorrência da falha como também indicará o estado correto em que o autômato original estará após a ocorrência de s_Y. □

4.3.2 Construção do diagnosticador de robustez diretamente do diagnosticador G_d

Na subseção anterior, foi apresentado um diagnosticador que testa a diagnosticabilidade robusta de L construído diretamente do autômato G_{isf}. Como a diagnosticabilidade de L com relação a D_{isf}, P'_o e Σ_f requer que L já seja diagnosticável em relação a P_o e Σ_f, cuja verificação pode ser realizada por um diagnosticador, a seguinte pergunta surge naturalmente: é possível obter um diagnosticador para G_{isf} diretamente a partir de G_d pela dilatação da linguagem gerada por G_d, e em seguida obtendo o observador para o diagnosticador dilatado? Esta possibilidade está representada na figura 4.8 (ramo direito) junto com a abordagem apresentada na subseção anterior (ramo esquerdo).

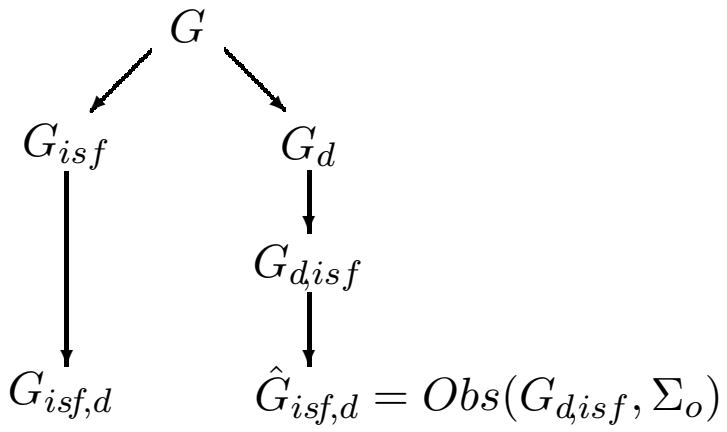


Figura 4.8: Dois modos de se obter um diagnosticador de robustez.

Note que ambos os autômatos na figura 4.8, $G_{isf,d}$ e $\hat{G}_{isf,d}$, são determinísticos, e, portanto, para provar que são equivalentes, basta demostrar que geram a mesma linguagem. Entretanto, os estados do diagnosticador desempenham um papel importante na análise da diagnosticabilidade, e portanto, também é necessário estabelecer uma relação entre seus estados.

Teorema 4.3 *Autômatos $G_{isf,d}$ e $\hat{G}_{isf,d}$ são equivalentes quanto às linguagens,* i.e $L(G_{isf,d}) = L(\hat{G}_{isf,d})$. *Além disso, os estados de $G_{isf,d}$ e $\hat{G}_{isf,d}$ são idênticos a menos da seguinte renomeação:*

$$\hat{x}_{isf,d} = \{x_{d,1}, x_{d,2}, \ldots, x_{d,q}\} \in \hat{X}_{isf,d} \Leftrightarrow x_{isf,d} = \bigcup_{i=1}^{q} x_{d,i} \in X_{isf,d}, \tag{4.8}$$

em que $x_{d,i} \in X_d$, $x_{d,i} = \{x_{i1}\ell_{i1}, x_{i2}\ell_{i2}, \ldots, x_{ip_i}\ell_{ip_i}\}$, $x_{ij} \in X$, $\ell_{ij} \in \{Y, N\}$, $i = 1, 2, \ldots, q$, $j = 1, 2, \ldots, p_i$, e $X_{isf,d}$ e $\hat{X}_{isf,d}$ são, respectivamente, os espaços de estados de $G_{isf,d}$ e $\hat{G}_{isf,d}$. Além disso, todo ciclo escondidos em um estado $x_{isf,d}$ é também é ciclo escondido em $\hat{x}_{isf,d}$ e vice versa.

Demonstração: Considere inicialmente o ramo esquerdo da figura 4.8. Como $G_{isf,d} = Obs(G_{isf} \| A_\ell, \Sigma_o)$, pode-se concluir que $L(G_{isf,d}) = P'_o[D_{isf}(L)]$ em que $P'_o : (\Sigma \cup \Sigma'_{isf})^* \longrightarrow \Sigma_o^*$. Considere agora o ramo direito da figura 4.8. Pode-se observar que como $L(G_d) = P_o(L)$ então, utilizando o teorema 4.1, $L(G_{d,isf}) = D_{isf}[L(G_d)] = D_{isf}[P_o(L)]$. Além disso, $\hat{G}_{isf,d} = Obs(G_{d,isf}, \Sigma_o)$, e, portanto, $L(\hat{G}_{isf,d}) = P'_{oo}[L(G_{d,isf})] = P'_{oo}[D_{isf}[P_o(L)]]$ em que $P'_{oo} : (\Sigma_o \cup \Sigma'_{isf})^* \longrightarrow \Sigma_o^*$. Finalmente, de acordo com o lema 1, $P'_o[D_{isf}(L)] = P'_{oo}[D_{isf}[P_o(L)]]$, que demonstra a equivalência das linguagens de $G_{isf,d}$ e $\hat{G}_{isf,d}$.

Considere agora a equivalência de estados. Para tanto, suponha que $\hat{x}_{isf,d} \in \hat{X}_{isf,d}$ seja um estado de $\hat{G}_{isf,d}$ alcançado por uma sequência $s' \in L(\hat{G}_{isf,d})$. Então, para todo $x_{d,i} \in \hat{x}_{isf,d}$, $i \in \{1, 2, \ldots, q\}$, existe uma sequência $s_{d,i} \in L(G_d)$ tal que $s' \in P'_{oo}[D_{isf}(s_{d,i})]$ e $f_d(x_{0,d}, s_{d,i}) = x_{d,i}$. Da mesma forma, para todo $x_{ij}\ell_{ij} \in x_{d,i}$, $j = 1, 2, \ldots, p_i$, existe uma sequência $s_{ij} \in L(G)$ tal que $P_o(s_{ij}) = s_{d,i}$, $f(x_0, s_{ij}) = x_{ij}$ e $\ell_{ij} = Y$, se $\Sigma_f \in s_{ij}$, ou $\ell_{ij} = N$, se $\Sigma_f \notin s_{ij}$. Portanto, como $G_{isf,d}$ é um

autômato determinístico e $P'_{oo}[D_{isf}[P_o(s_{ij})]] = P'_o[D_{isf}(s_{ij})]$, então existe $x_{isf,d} = f_{isf,d}(x_{0,isf,d}, s')$ tal que $x_{ij}\ell_{ij} \in x_{isf,d}$. Dessa forma, $x_{ij}\ell_{ij} \in x_{d,i} \in \hat{x}_{isf,d}$, e portanto $\bigcup_{i=1}^{q} x_{d,i} \subseteq \hat{x}_{isf,d}$.

Suponha, agora, $x_{isf,d} \in X_{isf,d}$ tal que $x_{isf,d} = f_{isf,d}(x_{0,isf,d}, s')$ para algum $s' \in L(G_{isf,d})$. Logo, existe $s_{ij} \in L(G)$ tal que $s' \in P'_o[D_{isf}(s_{ij})]$ e $x_{ij} = f(x_0, s_{ij})$, com $x_{ij}\ell_{ij} \in x_{isf,d}$, em que $\ell_{ij} = Y$, se $\Sigma_f \in s_{ij}$, ou $\ell_{ij} = N$, se $\Sigma_f \notin s_{ij}$. Portanto, existe $x_{d,i} = f_d(x_{0,d}, P_o(s_{ij}))$ tal que $x_{ij}\ell_{ij} \in x_{d,i}$. Como $P'_{oo}[D_{isf}[P_o(s_{ij})]] = s'$ e $\hat{G}_{isf,d}$ é uma autômato determinístico, então, de acordo com Basilio e Lafortune (2009), existe $\hat{x}_{isf,d} = \hat{f}_{isf,d}(\hat{x}_{0,isf,d}, s')$ tal que $x_{d_i} \in \hat{x}_{isf,d}$ e, portanto $x_{isf,d} \subseteq \cup_{i=1}^{q} x_{d,i}$.

Considere agora os ciclos escondidos em $G_{isf,d}$ e $\hat{G}_{isf,d}$. Note que em função da equação (4.8), todo ciclo escondido em $\hat{x}_{isf,d}$ será também um ciclo escondido em $x_{isf,d}$. Suponha que os estados $x_{i1}, x_{i2}, \ldots, x_{ip}$, $p \leq p_i$, formem um ciclo em G tal que $f(x_{ik}, \sigma_k) = x_{i,k+1}$, $k = 1, 2, \ldots, p-1$ e $f(x_{ip}, \sigma_p) = x_{i1}$, no qual $\sigma_j \in \Sigma_{uo}$, $j \in \{1, 2, \ldots, p\}$. Então, $x_{i1}, x_{i2}, \ldots, x_{ip}$ também forma um ciclo de estados de G_{isf} e existe $x_{isf,d} \in X_{isf,d}$ tal que $\{x_{i1}, x_{i2}, \ldots, x_{ip}\} \subseteq x_{isf,d}$, e, portanto $x_{i1}, x_{i2}, \ldots, x_{ip}$ é um ciclo escondido em $x_{isf,d}$. Da mesma forma, pela construção de G_d, existe $x_{d,i} \in X_d$ tal que $x_{d,i} = \{x_{i1}\ell_{i1}, x_{i2}\ell_{i2}, \ldots, x_{ip_i}\ell_{ip_i}\}$ tem um ciclo escondido. Como $\hat{G}_{isf,d}$ é obtido agrupando-se os estados de G_d conectados pelos eventos em $\Sigma_{uo} \cup \Sigma'_{isf}$ então existe $\hat{x}_{isf,d}$ tal que $x_{d,i} \in \hat{x}_{isf,d}$, e, portanto os estados $x_{i1}, x_{i2}, \ldots, x_{ip}$ também formam um ciclo escondido em $\hat{x}_{isf,d}$.

Considere agora que $x_{i1}, x_{i2}, \ldots, x_{ip'}$, $p' \leq p_i$ e $x_{j1}, x_{j2}, \ldots, x_{jp''}$, $i \neq j$, $p'' \leq p_j$ sejam estados de G tais que $f(x_{ik}, \sigma_k) = x_{i,k+1}$, $k = 1, 2, \ldots, p'-1$ e $f(x_{jk}, \sigma_k) = x_{j,k+1}$, $k = 1, 2, \ldots, p'' - 1$, em que $\sigma_k \in \Sigma_{uo}$, $k \in \{1, 2, \ldots, \max(p', p'')\}$. Então, $x_{i1}, x_{i2}, \ldots, x_{ip'}$ e $x_{j1}, x_{j2}, \ldots, x_{jp''}$ não formam ciclos em G. Contudo, pela construção do diagnosticador, existe $x_{d,i}, x_{d,j} \in X_d$ tal que $\{x_{i1}, x_{i2}, \ldots, x_{ip'}\} \in x_{d,i}$ e $\{x_{j1}, x_{j2}, \ldots, x_{jp''}\} \in x_{d,j}$. Suponha que os eventos $\sigma_{p'}, \sigma_{p''} \in \Sigma'_{isf}$ sejam tais $f(x_{ip'}, \sigma'_p) = x_{j1}$ e $f(x_{jp''}, \sigma''_p) = x_{i1}$. Então $x_{i1}, x_{i2}, \ldots, x_{ip'}$, $x_{j1}, x_{j2}, \ldots, x_{jp''}$ formam um ciclo em G_{isf}, e, portanto existe um estado $x_{isf,d} \in X_{isf,d}$ tal que $x_{d,ij} = \{x_{i1}\ell_{i1}, x_{i2}\ell_{i2}, \ldots, x_{ip'}\ell_{ip'}, x_{j1}\ell_{j1}, x_{j2}\ell_{j2}, \ldots, x_{jp''}\ell_{jp''}\} \subseteq x_{isf,d}$. Portanto,

$x_{i1}, x_{i2}, \ldots, x_{ip'}, x_{j1}, x_{j2}, \ldots, x_{jp''}$ define um ciclo escondido em $x_{isf,d}$. A hipótese que $\sigma_{p'}, \sigma_{p''} \in \Sigma'_{isf}$ implica que $x_{d,i}, x_{d,j}$ formam um ciclo em $G_{d,isf}$, e como $\hat{G}_{isf,d}$ é obtido pelo agrupamento dos estados de G_d conectados pelos eventos em $\Sigma_{uo} \cup \Sigma'_{isf}$ então existe $\hat{x}_{isf,d}$ tal que $\{x_{d,i}, x_{d,j}\} \subseteq \hat{x}_{isf,d}$, e, portanto $x_{i1}, x_{i2}, \ldots, x_{ip'}, x_{j1}, x_{j2}, \ldots, x_{jp''}$ também define um ciclo escondido em $\hat{x}_{isf,d}$. □

O exemplo a seguir ilustra o resultado do teorema 4.3.

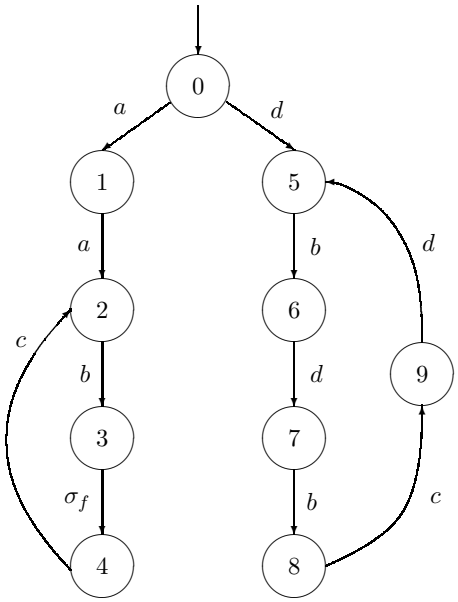


Figura 4.9: Autômato G do exemplo 4.4.

Exemplo 4.4 *Considere o autômato G cujo diagrama de transição de estados está representado na figura 4.9 e suponha que $\Sigma = \{a, b, c, d, \sigma_f\}$, $\Sigma_o = \{a, d\}$, $\Sigma_{uo} = \{b, c, \sigma_f\}$ e $\Sigma_f = \{\sigma_f\}$. Suponha também que o evento d esteja associado a um sensor que pode falhar intermitentemente,* i.e., *$\Sigma_{isf} = \{d\}$. Os diagnosticadores $G_{isf,d}$ e $\hat{G}_{isf,d}$ são mostrados nas figuras 4.10(a) e (b), respectivamente. Note que:* (i) *$G_{isf,d}$ e $\hat{G}_{isf,d}$ geram a mesma linguagem;* (ii) *os estados de $G_{isf,d}$ e $\hat{G}_{isf,d}$ satisfazem a relação dada na esquação (4.8);* (iii) *um ciclo escondido que aparece em um estado $x_{isf,d} \in X_{isf,d}$ também aparece como um ciclo escondido no estado correspondente $\hat{x}_{isf,d} \in \hat{X}_{isf,d}$.*

Observe que os estados 2, 3 e 4 formam um ciclo em G e como eles são conectados pelos eventos não-observáveis b, c e σ_f, então eles formam um ciclo escondido em $G_{isf,d}$ e G_d. Além disso, pela construção de $\hat{G}_{isf,d}$, este ciclo escondido também está

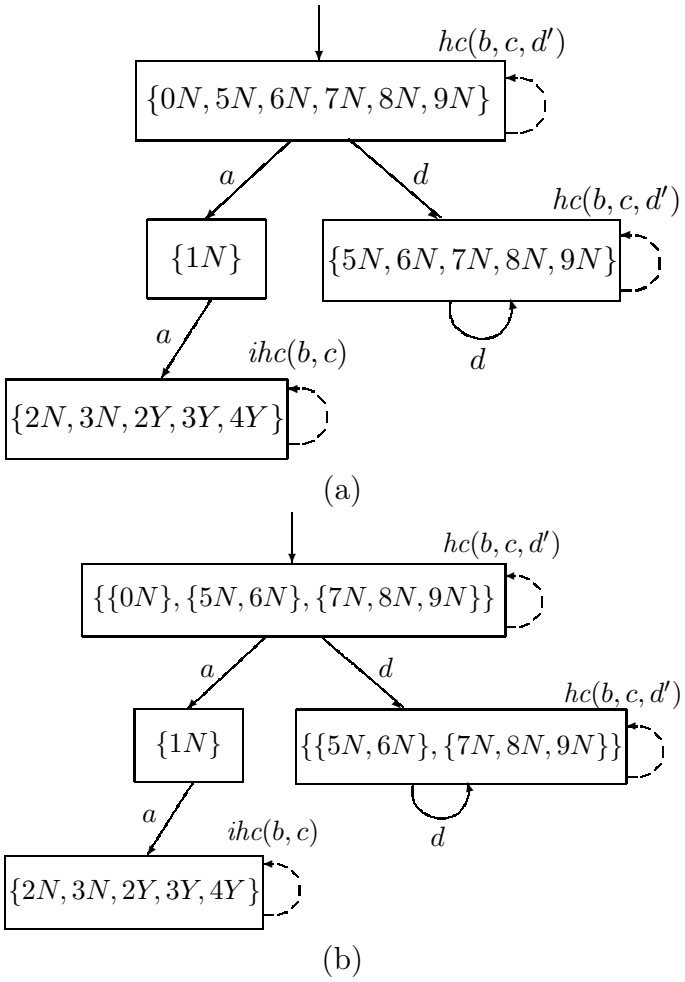


Figura 4.10: Diagnosticadores $G_{isf,d}$ (a) e $\hat{G}_{isf,d}$ (b) para ilustrar o teorema 4.3.

em $\hat{G}_{isf,d}$. Comparando os diagnosticadores $G_{isf,d}$ e $\hat{G}_{isf,d}$, pelas figuras 4.10(a) e (b), é possível verificar que ambos os autômatos têm um ciclo escondido no estado $\{2N,3N,2Y,3Y,4Y\}$ devido aos eventos b, c e σ_f.

Os estados 5, 6, 7, 8 e 9 não formam um ciclo escondido em G_d já que o evento d é observável. Entretanto, como o evento d pode perder observabilidade, esses estados formam um ciclo conectado com eventos não-observáveis em G_{isf}. Portanto, eles irão formar um ciclo escondido em $G_{isf,d}$. Note na figura 4.10(a) que os estados $\{0N,5N,6N,7N,8N,9N\}$ e $\{5N,6N,7N,8N,9N\}$ têm ciclos escondidos devido aos eventos (b,c,d') — o evento d' representa a não ocorrência do evento d. Quando $\hat{G}_{isf,d}$ for construído, os estados conectados pelo evento d' se fundem, e, portanto formam o mesmo ciclo nos estados $\{\{0N\},\{5N,6N\},\{7N,8N,9N\}\}$ e $\{\{5N,6\},\{7N,8N,9N\}\}$. □

4.4 Verificação de diagnosticabilidade robusta utilizando verificadores

Para verificação da diagnosticabilidade robusta de uma linguagem gerada por um autômato em relação a perdas intermitentes de sensores utilizando verificadores, o teorema 2.5 deve ser modificado como se segue.

Teorema 4.4 *L não é diagnosticável com relação a D_{isf}, P'_o e Σ_f se e somente se existir um ciclo $cl := (x_V^k, \sigma_k, x_V^{k+1}, ..., x_V^l, \sigma_l, x_V^k)$, sendo $l \geq k > 0$, em*

$$G_{isf,V} = G_{isf,N_1} \| G_{isf,F}, \tag{4.9}$$

que satisfaça à seguinte condição:

$$\exists j \in \{k, k+1, \ldots, l\} \text{ tal que para algum } x_V^j, (x_l^j = Y) \wedge (\sigma_j \in \Sigma'). \tag{4.10}$$

Demonstração: A prova desse teorema segue os mesmos passos da prova do teorema 2.5 apresentada em Moreira *et al.* (2011), substituindo-se L e Σ por L_{isf} e Σ', respectivamente. Portanto, a prova será omitida aqui. □

De acordo com o teorema 4.4, a diagnosticabilidade robusta é verificada utilizando-se os autômatos verificadores construídos de acordo com o algoritmo 2.2 para G_{isf}, denotado de verificador de robustez $G_{isf,V}$, e substituindo-se a função de renomeação dada no passo 3 na equação (2.14) por:

$$\begin{aligned} R_1 : \Sigma_N &\rightarrow \Sigma_{R_1} \\ \sigma &\mapsto R_1(\sigma) = \begin{cases} \sigma, \text{ se } \sigma \in \Sigma_o \\ \sigma_{R_1}, \text{ se } \sigma \in \Sigma'_{uo} \setminus \Sigma_f \end{cases}, \end{aligned} \tag{4.11}$$

em que $\Sigma'_{uo} = \Sigma_{uo} \cup \Sigma'_{isf}$.

Vale a pena ressaltar que diferentemente dos diagnosticadores, a construção dos verificadores não requer o uso de observadores (ver equação (4.7)). Portanto, é de se

esperar que os verificadores de robustez não possam, em geral, ser obtidos a partir do verificador para G como proposto na seção 4.3.2. Este ponto será ilustrado no exemplo a seguir.

Exemplo 4.5 *Considere novamente o autômato G da figura 4.1(a), em que $\Sigma = \{a, b, c, d, e, \sigma_f\}$, e $\Sigma_f = \{\sigma_f\}$. Suponha como no exemplo 4.2(b) que $\Sigma_{uo} = \{a, \sigma_f\}$ e $\Sigma_{isf} = \{b\}$. A figura 4.11 mostra os diagramas de transição de estados dos autômatos G_{isf} (a), G_{isf_N} (b) que modelam, respectivamente, o comportamento do sistema quando sujeito a perdas intermitentes de sensores e comportamento normal de G_{isf}. Após a renomeação dos eventos $\Sigma'_{uo} \setminus \Sigma_f$ em G_{isf_N} obtém se o autômato $G_{isf_{N,1}}$ representado na figura 4.11(a) e também é obtido o autômato G_{isf_F} que modela o comportamento de falha de G_{isf} mostrado na figura 4.11(b). O verificador de ro-*

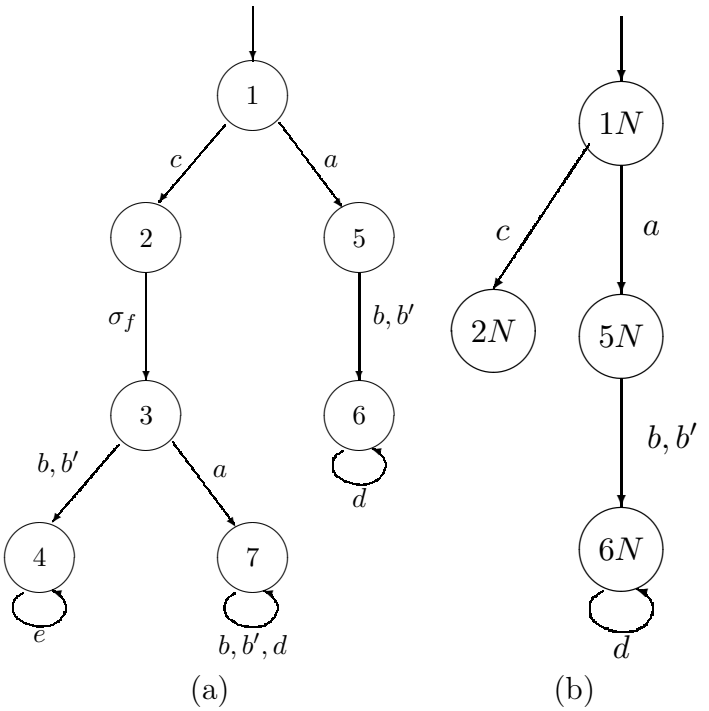


Figura 4.11: Autômato G_{isf} (a); G_{isf_N} que modela o comportamento normal de G_{isf}.

bustez $G_{isf_V} = G_{isf_{N,1}} \| G_{isf_F}$ está representado na figura 4.13. Note que o verificador G_{isf_V} tem um ciclo $\{2N, 7Y; b'; 2N, 7Y\}$ e como $\Sigma' = \Sigma \cup \Sigma'_{isf} = \{a, b, c, d, e, \sigma_f, b'\}$, pode-se concluir que a linguagem gerada por G_{isf} não é robustamente diagnosticável em relação a D_{isf}, P'_o e Σ_f, conforme foi também concluído no exemplo 4.2(a) utilizando o diagnosticador de robustez.

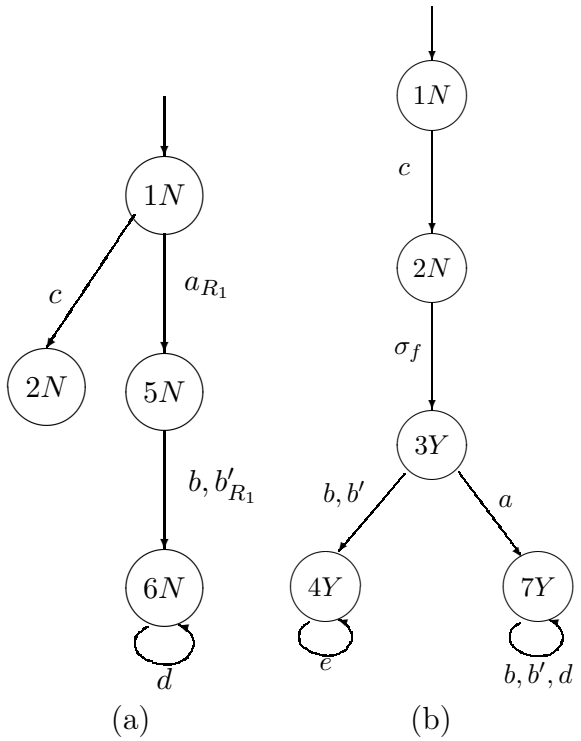


Figura 4.12: Autômatos $G_{isf_{N,1}}$ que é o autômato G_{isf_N} com os eventos renomeados (a), G_{isf_F} que modela o comportamento de falha de G_{isf} (b).

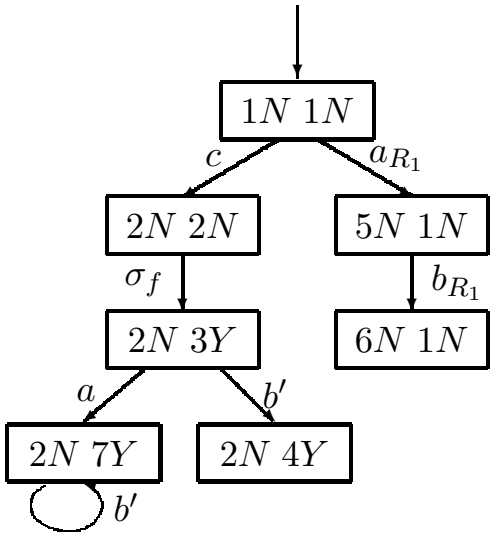


Figura 4.13: Autômato verificador robusto $G_{isf_V} = G_{isf_{N,1}} \| G_{isf_F}$, supondo $\Sigma_{uo} = \{a, \sigma_f\}$ e $\Sigma_{isf} = \{b\}$.

Considere agora o verificador G_V mostrado na figura 4.14(c) que foi obtido diretamente de G a partir de $G_{N,1}$ (figura 4.14(a)) e G_F (figura 4.14(b)). Note que a ideia da seção 4.3.2 de se encontrar um verificador para G_{isf} diretamente a partir de G_V não é, de fato, válida para esse exemplo uma vez que não é possível obter G_{isf_V} dilatando-se a linguagem de G_V. Isso decorre do fato de que $\Sigma_{isf} = \{b\}$ e

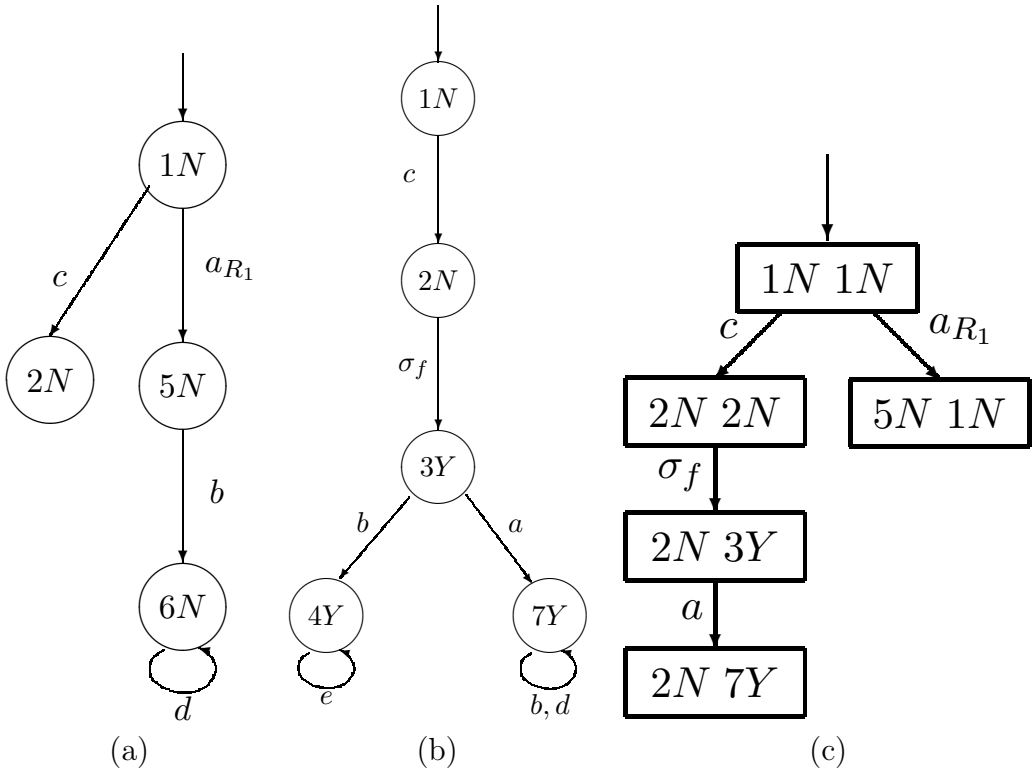


Figura 4.14: Autômato $G_{N,1}$ que modela o comportamento normal de G com os eventos renomeados (a); G_F que modela o comportamento de falha de G (b); autômato verificador G_V de G, supondo $\Sigma_{uo} = \{a, \sigma_f\}$ (c).

como não existe no verificador G_V mostrado na figura 4.14(c) nenhuma transição rotulada pelo evento b, então, $D_{isf}(L(G_V)) = L(G_V)$. □

4.5 Extensão para codiagnosticabilidade robusta

Como mencionado na seção 2.3.3, na prática, devido à natureza distribuída de alguns sistemas, diagnosticadores centralizado nem sempre podem ser empregados. Para contornar esse problema, uma arquitetura descentralizada foi proposta por Debouk *et al.* (2000), em que as informações dos sensores não são mais centralizadas, mas distribuídas por diferentes módulos, cada módulo observando o seu próprio conjunto de eventos observáveis Σ_{o_i}, $i = 1, 2, \ldots, N$ — supondo que existam N módulos independentes. Como mencionado na definição 2.8, a codiagnose considera que quando todos os módulos operam autonomamente, cada falha deve ser diagnosticada por pelo menos um módulo a partir de suas observações. Nesse contexto, a definição 2.8

pode ser estendida para a codiagnosticabilidade robusta de SEDs sujeitos a perdas intermitentes de sensores de forma semelhante à definição 4.2 , como se segue.

Definição 4.4 *(Codiagnosticabilidade robusta de SEDs sujeito a perdas intermitentes de sensores) Uma linguagem L viva e prefixo-fechada não é robustamente codiagnosticável em relação ao protocolo 3, à dilatação D_{isf}, às projeções $P'_{o_i} : \Sigma'^* \to \Sigma^*_{o_i}$, $i = 1, \ldots, N$, e $\Sigma_f = \{\sigma_f\}$, se a seguinte condição for verificada:*

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

sendo a condição R_C expressa por

$$(\exists w_i \in L, i = 1, 2, \ldots, N)(\Sigma_f \notin w_i)(w_k \textit{ não necessariamente distinto de } w_l, k \neq l)$$
$$[P_{o_i}(D_{isf}(st)) = P_{o_i}(D_{isf}(w_i)), i = 1, 2, \ldots, N].$$

□

Uma extensão das condições necessárias e suficientes para codiagnosticabilidade apresentadas nos teoremas 2.4 e 2.5 pode ser obtida para a codiagnosticabilidade robusta a perdas intermitentes de sensores comparando-se as definições 2.8 e 4.4. Note que na definição 4.4, st e ω_i são substituídos por $D_{isf}(st)$ e $D_{isf}(\omega_i)$, respectivamente. Como $L_{isf} = D_{isf}(L)$, pode-se dizer que codiagnosticabilidade robusta com relação à dilatação D_{isf}, às projeções $P'_{o_i} : \Sigma'^* \to \Sigma^*_{o_i}$, $i = 1, \ldots, N$, e $\Sigma_f = \{\sigma_f\}$ é equivalente à codiagnosticabilidade de G_{isf} com relação às projeções $P'_{o_i} : \Sigma'^* \to \Sigma^*_{o_i}$, $i = 1, \ldots, N$, e $\Sigma_f = \{\sigma_f\}$.

Teorema 4.5 *Uma linguagem é robustamente codiagnosticável com relação a D_{isf}, 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,*

$$G_{isf,t} = (\|_{i=1}^{N} G_{isf,d_i}) \| G_{isf,d} \tag{4.12}$$

não tiver ciclos indeterminados, em que um ciclo de estados incertos de $G_{isf,t}$ será indeterminado se todos os correspondentes ciclos em G_{isf,d_i}, $i = 1, 2, \ldots, N$ forem

indeterminados como na definição 2.11. □

Teorema 4.6 *Uma linguagem L não é robustamente codiagnosticável com relação a D_{isf}, P'_{o_i}, $i = 1, \ldots, N$, e Σ_f se, e somente se, existir um ciclo $cl := (x_V^k, \sigma_k, x_V^{k+1}, ..., x_V^l, \sigma_l, x_V^k)$, sendo $l \geq k > 0$, em*

$$G_{isf,V} = (\|_{i=1}^{n} G_{isf,N_i}) \| G_{isf,F}, \tag{4.13}$$

que satisfaça à seguinte condição:

$$\exists j \in \{k, k+1, \ldots, l\} \text{ tal que para algum } x_V^j, (x_l^j = Y) \wedge (\sigma_j \in \Sigma'). \tag{4.14}$$

□

Como na construção do diagnosticador de robustez, a construção do diagnosticador teste de robustez $G_{isf,t}$ pode ser realizada de dois modos como apresentado na figura 4.15. O ramo da esquerda é obtido pela equação (4.12) enquanto o ramo da direita utiliza o resultado do teorema 4.3. A prova da equivalência entre $G_{isf,t}$ e $\hat{G}_{isf,t}$ é semelhante à do teorema 4.3 e será portanto, omitida.

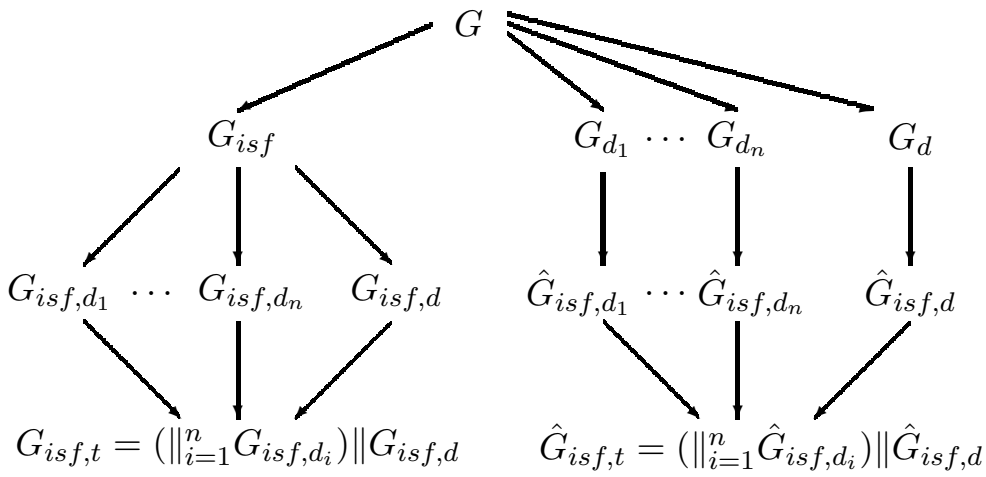


Figura 4.15: Dois modos de se obter diagnosticador teste de robustez para verificação da codiagnosticabilidade robusta.

Exemplo 4.6 *Para ilustrar a construção dos autômatos diagnosticador teste de robustez e verificador de robustez, considere o autômato G cujo diagrama de transição*

de estados está representado na figura 4.16. Suponha que o conjunto de eventos observáveis e não-observáveis de G sejam $\Sigma_o = \{a, b, c, d\}$ *e* $\Sigma_{uo} = \Sigma_f = \{\sigma_f\}$*, respectivamente. O diagnosticador centralizado* G_d *para G está representado na figura 4.17(a), que mostra que a linguagem L gerada por G pode ser diagnosticada em relação a* $P_o : \Sigma^* \rightarrow \Sigma_o^*$ *e* Σ_f*.*

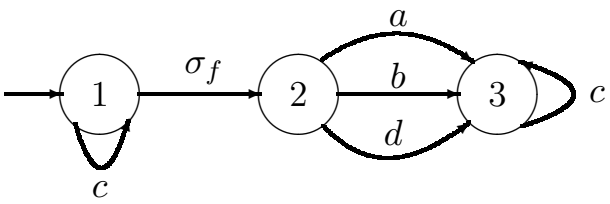


Figura 4.16: Autômato G.

Suponha agora que nem todos os eventos observáveis estejam disponíveis simultaneamente em um mesmo lugar e suponha que para contornar esse problema, um diagnosticador descentralizado com coordenação composto por dois módulos será construído para detectar a ocorrência de σ_f*, sendo os conjuntos de eventos observáveis* $\Sigma_{o_1} = \{a, c\}$ *e* $\Sigma_{o_2} = \{b, c, d\}$*. Os correspondentes diagnosticadores parciais* G_{d_1} *e* G_{d_2} *estão representados nas figuras 4.17(b) e (c), e o diagnosticador teste* $G_{test_3} = G_{d_1} \| G_{d_2} \| G_d$ *pode ser visto nas figura 4.18. Uma vez que* G_{test_3} *não possui ciclos indeterminados como pode ser visto na figura 4.18, então, conforme o teorema 2.4, L é diagnosticável em relação a* $P_{o_i} : \Sigma^* \rightarrow \Sigma_{o_i}$*,* $i = 1, 2$*, e* Σ_f

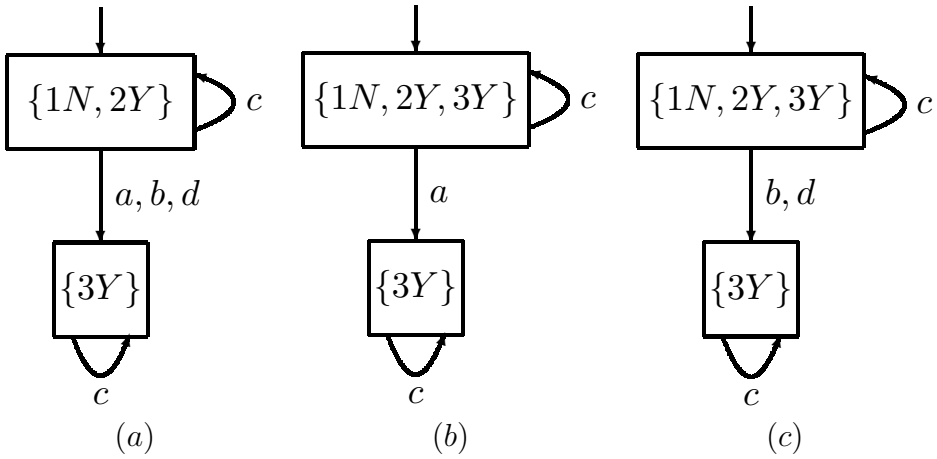


Figura 4.17: Diagnosticadores centralizado G_d (a) e parciais G_{d_1} (b) e G_{d_2} (c).

Suponha, agora, que o sistema esteja sujeito à perda intermitente do sensor as-

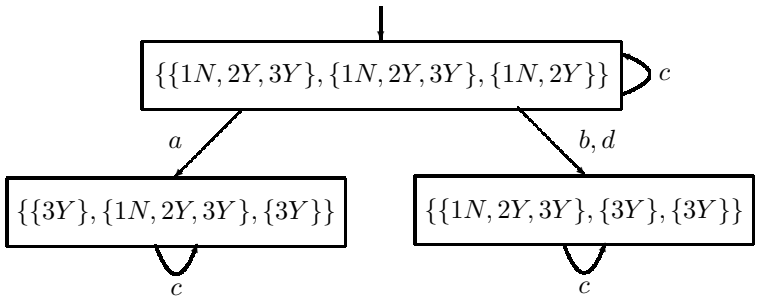


Figura 4.18: Diagnosticador teste $G_{\text{test}_3} = G_{d_1} \| G_{d_2} \| G_d$.

sociado ao evento c, ou seja, $\Sigma_{isf} = \{c\}$. *Considere o autômato* G_{isf} *e seu diagnosticador de robustez* $G_{isf,d}$ *representados na figura 4.19 e na figura 4.20(a), respectivamente. Pode-se facilmente verificar que a linguagem L gerada por G pode ser diagnosticada em relação a* D_{isf}, $P'_o : \Sigma'^* \to \Sigma_o^*$ *e* Σ_f.

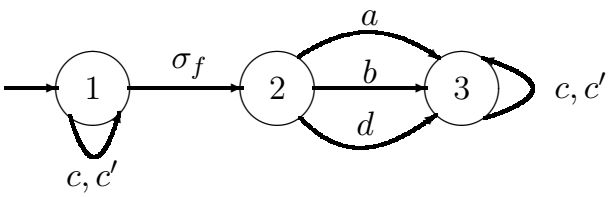


Figura 4.19: Autômato G_{isf}.

Suponha que os conjuntos de eventos observáveis continuem sendo $\Sigma_{o_1} = \{a, c\}$ *e* $\Sigma_{o_2} = \{b, c, d\}$. *Os correspondentes diagnosticadores parciais* G_{isf,d_1} *e* G_{isf,d_2} *estão representados nas figuras 4.20(b) e (c), e o diagnosticador teste de robustez* $G_{isf,t} = G_{isf,d_1} \| G_{isf,d_2} \| G_{isf,d}$ *pode ser visto na figura 4.21. Deve ser ressaltado que foram utilizadas a definição 2.11 e a observação 2.3 para definir os ciclos escondidos em* $G_{isf,t}$. *Como* $G_{isf,t}$ *não possui ciclos indeterminados, então L é diagnosticável em relação a* D_{isf}, $P'_{o_i} : \Sigma'^* \to \Sigma_{o_i}$, $i = 1, 2$, *e* Σ_f.

Considere agora a construção do verificador descentralizado de robustez para o autômato G_{isf} *da figura 4.19 e suponha novamente que* $\Sigma_{o_1} = \{a, c\}$, $\Sigma_{o_2} = \{b, c, d\}$ *e* $\Sigma_{isf} = \{c\}$. *De acordo com os passos 1 e 2 do algoritmo 2.2, o autômato* $G_{isf,F}$ *é o próprio autômato* G_{isf} *e* $G_{isf,N}$ *é um autômato que tem um único estado com um autolaço rotulado por c e c′. De acordo com o algoritmo 2.3 e lembrando que existem dois módulos (*$N = 2$*), obtém-se os autômatos* G_{isf,N_1} *e* G_{isf,N_2} *a partir de* $G_{isf,N}$ *pela renomeação dos eventos não observáveis nos conjuntos* $\Sigma'_{uo_1} \backslash \Sigma_f = \{b, c', d\}$

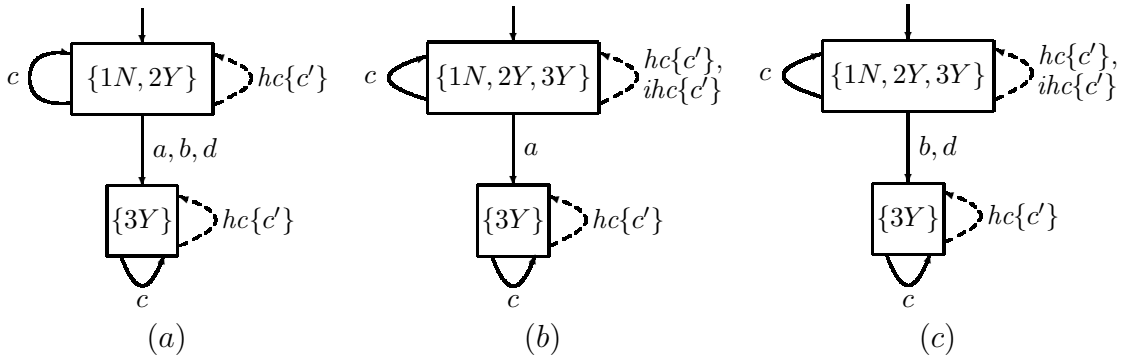


Figura 4.20: Diagnosticadores de robustez $G_{isf,d}$ (a) e parciais de robustez G_{isf,d_1} (b) e G_{isf,d_2} (c).

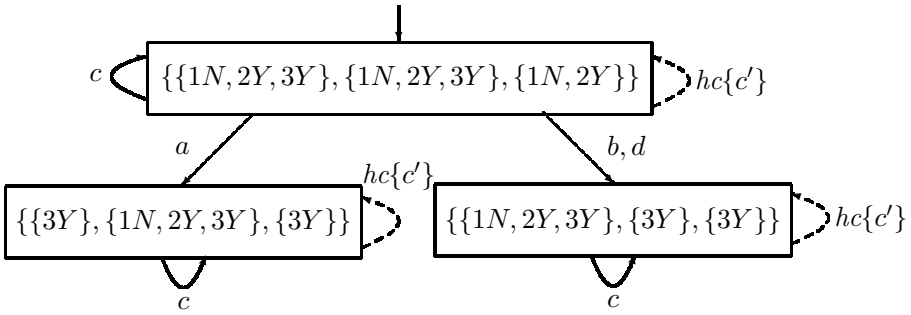


Figura 4.21: Diagnosticador teste de robustez $G_{isf,t} = G_{isf,d_1} \| G_{isf,d_2} \| G_{isf,d}$.

e $\Sigma'_{uo_2} \backslash \Sigma_f = \{a, c'\}$*, respectivamente. Assim, os conjuntos de eventos de* G_{isf,N_1} *e* G_{isf,N_2} *são* $\Sigma_{R_1} = \{a, b_{R_1}, c, c'_{R_1}, d_{R_1}\}$ *e* $\Sigma_{R_2} = \{a_{R_2}, b, c, c'_{R_2}, d\}$*, respectivamente. Portanto,* G_{isf,N_1} *é um autômato que tem um único estado com um autolaço rotulado por* c, c'_{R_1} *e* G_{isf,N_2} *é um outro autômato que tem um único estado com um autolaço rotulado por* c, c'_{R_2}*. Com os autômatos* G_{isf,N_1}*,* G_{isf,N_2} *e* $G_{isf,F}$*, pode-se obter o verificador descentralizado de robustez* $G_{isf,V} = G_{isf,N_1} \| G_{isf,N_2} \| G_{isf,F}$ *apresentado na figura 4.22.*

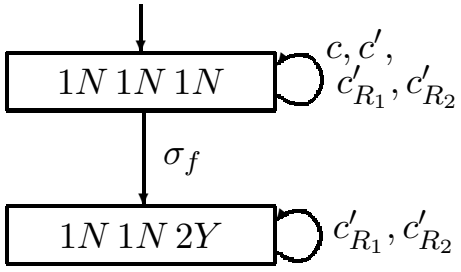


Figura 4.22: Verificador descentralizado de robustez $G_{isf,V}$.

Note que $G_{isf,V}$ *não tem nenhum ciclo que satisfaz a equação (4.14) e, portanto, pode-se concluir que* L *é diagnosticável em relação a* D_{isf}*,* P'_{o_i}*,* $i = 1, 2$ *e* Σ_f*. Observe*

que o resultado do diagnosticador teste de robustez é idêntico ao obtido utilizando-se o verificador descentralizado de robustez. □

4.6 Comentários finais

Nesse capítulo foi abordado o problema da diagnose de falhas de sistemas a eventos discretos sujeitos a perdas intermitente de sensores. Um novo operador de linguagem — dilatação — foi introduzido, o que permitiu obter um modelo que representa o comportamento do sistema quando sujeitos ou não a perdas intermitente de sensores. Além disso foi apresentada uma definição de diagnosticabilidade robusta e encontrou-se condições necessárias e suficientes para a diagnosticabilidade robusta a perdas intermitente de sensores. Finalmente, foram apresentados dois modos de se encontrar o diagnosticador robusto: *(i)* o diagnosticador foi obtido a partir do modelo da planta que leva em conta as perdas intermitentes de sensores; *(i)* o modelo que representa as perdas de sensores foi aplicado diretamente no diagnosticador e, na seqüência, obteve-se o observador do autômato resultante em relação ao conjunto de eventos observáveis. Além disso, foram apresentadas a verificação da diagnosticabilidade robusta a perdas intermitentes de sensores por verificadores e a codiagnose robusta utilizando-se verificadores e diagnosticadores.

Capítulo 5

Diagnose robusta generalizada

A definição de diagnosticabilidade robusta de sistema a eventos discretos sujeitos a perdas permanentes de sensores proposta por Lima *et al.* (2010) e revista no capítulo 3, considera as redundâncias que possam existir em um conjunto formado por bases para a diagnose visando garantir a diagnose da falha, mesmo quando da ocorrência de perda permanente de sensores. Para tanto é construído um diagnosticador robusto que utiliza vários diagnosticadores parciais construídos para a mesma planta G com um conjunto específico de eventos observáveis definidos pelos eventos da base para a diagnose. Dessa forma, o problema da diagnosticabilidade robusta proposto por Lima *et al.* (2010) pode ser considerado como um problema de identificação da ocorrência do evento de falha com incertezas no conjunto de eventos observáveis.

Uma outra definição de diagnosticabilidade (Takai, 2010) requer que o sistema seja descrito por um conjunto de possíveis modelos $\{G_i : i \in I_m\}$ em relação a um mesmo conjunto de eventos Σ, sendo $I_m := \{1, 2, \ldots, m\}$ e $m \in \mathbb{N}$ o número de modelos do sistema. Nessa definição, cada modelo gera uma linguagem viva e tem seu próprio comportamento de não falha. Diferente de Lima *et al.* (2010), todos os modelos G_i em Takai (2010) têm o mesmo conjunto de eventos observáveis. Deve ser ressaltado que a verificação da diagnosticabilidade robusta apresentada por Takai (2010) é baseada no algoritmo proposto por Qiu e Kumar (2006).

Nesse capítulo será apresentada uma nova formulação para o problema da diagnose robusta (Carvalho *et al.*, 2011). Assim como em Takai (2010), o modelo do

sistema pertence a uma classe formada pelos possíveis modelos e compartilham o mesmo conjunto de eventos Σ. Contudo, diferentemente de Takai (2010), cada modelo pode ter um conjunto de eventos observáveis distintos — como em Lima *et al.* (2010). Para a verificação da diagnosticabilidade robusta generalizada é proposto um algoritmo em tempo polinomial baseado em Moreira *et al.* (2011).

Esse capítulo está organizado da seguinte forma. Na seção 5.1 é apresentada a definição de diagnosticabilidade robusta generalizada. Na seção 5.2 é proposto um algoritmo em tempo polinomial para verificar a propriedade da diagnosticabilidade robusta generalizada, e, em seguida é determinada sua complexidade computacional. Na seção 5.3 é apresentado um exemplo para ilustrar o algoritmo proposto. Conclusões são apresentadas na seção 5.4.

5.1 Diagnosticabilidade robusta generalizada a perdas de sensores

A definição de diagnosticabilidade robusta de SEDs sujeitos a perdas permanentes de sensores apresentada na definição 3.3 é feita considerando as seguintes hipóteses: *(i)* a perda de um sensor se dá antes da primeira ocorrência do evento; *(ii)* a linguagem gerada pelo sistema não é alterada se ocorrerem perdas de observabilidade de eventos; *(iii)* a diagnosticabilidade robusta é determinada em termos de conjuntos de eventos observáveis distintos Σ_{o_i}, $i = 1, \ldots, m$, $m \in \mathbb{N}$, sendo Σ_{o_i} uma base para a diagnose (Basilio *et al.*, 2011; Lima, 2010), *i.e.*, L é diagnosticável com relação a projeções $P_{o_i} : \Sigma^* \rightarrow \Sigma_{o_i}^*$, $i = 1, \ldots, m$, e Σ_f. De acordo com a definição 3.3, a linguagem L não é robustamente diagnosticável se existir $i, j \in I_m$, $(i \neq j)$, e duas sequências $w, st \in L$ tal que $P_{o_i}(st) = P_{o_j}(w)$, sendo w uma sequência de não falha e st arbitrariamente longa após o evento de falha $\sigma_f \in \Sigma_f$.

Ao invés de se usar um único modelo, Takai (2010) propõe um conjunto de possíveis autômatos para representar o sistema, e introduz uma outra definição de diagnosticabilidade robusta, supondo que: *(i)* o modelo do sistema real pertence a

um conjunto de possíveis modelos $G_i = (X_i, \Sigma, f_i, x_{0_i})$, $i = 1, 2, \ldots, m$; *(ii)* cada autômato G_i gera uma linguagem L_i diferente e tem um comportamento de não falha distinto, descrito por uma sublinguagem fechada e não vazia $L_{N_i} \subseteq L_i$; *(iii)* todos os modelos têm o mesmo conjunto de eventos observáveis Σ_o.

No trabalho de Lima *et al.* (2010), o sistema é modelado por um único autômato G e as incertezas são devidas a perdas de eventos observáveis. Por outro lado, o problema de diagnosticabilidade robusta formulado em Takai (2010) só é adequado quando existem vários modelos e não pode ser utilizado diretamente quando as incertezas do modelo são devidas tanto a perdas permanentes (Lima *et al.*, 2010) ou intermitentes (Carvalho *et al.*, 2010) de sensores.

Afim de abranger tanto conjuntos de eventos observáveis distintos e diferentes modelos de autômatos e, portanto, generalizar as definições de diagnosticabilidade robusta apresentadas em Lima *et al.* (2010) e Takai (2010), a seguinte definição é apresentada.

Definição 5.1 *(Diagnosticabilidade robusta generalizada) Seja $L_i \subseteq \Sigma^*$ a linguagem gerada por G_i, $i = 1, \ldots, m$, e suponha que L_i seja viva. Além disso, suponha que cada modelo $i \in I_m$ tenha uma projeção $P_{o_i} : \Sigma^* \to \Sigma_{o_i}^*$, e que L_i seja diagnosticável com relação a P_{o_i} e Σ_f. Denote G_{N_i} como um subautômato de G_i que modela o comportamento de não falha do modelo, e $L_{N_i} \subset L_i$ a linguagem gerada por G_{N_i}. Então a classe,*

$$\mathbb{L} = \{L_i : i \in I_m\}$$

de todas as possíveis linguagens geradas pela classe de autômatos

$$\mathbb{G} = \{G_i : i \in I_m\},$$

é robustamente diagnosticável com relação às projeções P_{o_i}, $i = 1, \ldots, m$, e $\Sigma_f =$

$\{\sigma_f\}$, *se e somente se*

$$\begin{aligned}&(\forall i \in I_m)(\exists n_i \in \mathbb{N})(\forall s_i \in L_i \setminus L_{N_i})(\forall s_i t_i \in L_i \setminus L_{N_i}, |t_i| \geq n_i) \Rightarrow \\ &(\forall j \in I_m, j \neq i)(\forall w_j \in L_{N_j}, P_{o_i}(s_i t_i) \neq P_{o_j}(w_j)).\end{aligned} \tag{5.1}$$

□

Observação 5.1 *A modelagem de perdas intermitentes de sensores em uma linguagem L apresentada no capítulo 4 pode ser feita utilizando-se uma classe de linguagens em que cada linguagem representa as possíveis combinações de perdas de sensores. Considere, por exemplo, uma linguagem L na qual existe um evento* $\sigma \in L$, $\sigma \in \Sigma_{isf}$ *(*Σ_{isf} *o conjunto de eventos associado às perdas intermitentes de sensores) e o número de ocorrências de* σ *em L seja três. Portanto, as combinações possíveis de perdas de observabilidade do evento* σ *são oito, ou seja, o sensor associado a* σ *pode não falhar, pode falhar em uma das três ocorrências, pode falhar em duas das três ocorrências e pode falhar nas três ocorrências. Generalizando, se um evento* $\sigma \in \Sigma_{isf}$ *têm n ocorrências, então o número de linguagens que representam a linguagem L sujeita a perdas intermitentes de sensores é:*

$$\sum_{q=0}^{n} C_{n,q}, \tag{5.2}$$

em que $C_{n,q}$ *denota combinação de n elementos tomados p a p. Consequentemente, a diagnose robusta a perdas intermitentes de sensores pode também ser considerada no contexto da diagnose robusta generalizada.* □

5.2 Verificação da diagnosticabilidade robusta generalizada

De acordo com a definição 5.1, o problema da verificação da diagnosticabilidade robusta generalizada de um SED pode ser formulado como a busca de sequências $s_i t_i \in L_i \setminus L_{N_i}$ e $w_j \in L_{N_j}$, $i \neq j$, em que $s_i \in L_i \setminus L_{N_i}$ e $t_i \in L_i/s_i$ é uma sequência

de comprimento arbitrariamente longo, tal que $P_{o_i}(s_i t_i) = P_{o_j}(w_j)$. Se existirem sequências $s_i t_i$, e w_j que satisfaçam essas condições, então a classe de linguagens $\mathbb{L}$ não será robustamente diagnosticável com relação às projeções P_{o_i}, $i = 1, 2, \ldots, m$, e Σ_f.

Seja $G_i \in \mathbb{G}$ um possível modelo para o sistema com o conjunto de eventos Σ e suponha que G_i gere a linguagem $L_i \in \mathbb{L}$. Considere que Σ seja particionado como $\Sigma = \Sigma_{o_i} \cup \Sigma_{uo_i}$ sendo Σ_{o_i} e Σ_{uo_i}, respectivamente, conjuntos dos eventos observáveis e não-observáveis para o modelo G_i e defina $\Sigma_o = \bigcup_{i=1}^{m} \Sigma_{o_i}$.

Considere novamente a função de renomeação dada pela equação (3.3), para $i = 1, 2, \ldots, m$, como:

$$\begin{array}{rcl} R_i : \Sigma & \rightarrow & \Sigma_{R_i} \\ \sigma & \mapsto & R_i(\sigma) = \begin{cases} \sigma, \text{ se } \sigma \in \Sigma_{o_i} \cup \Sigma_f \\ \sigma_{R_i}, \text{ se } \sigma \in \Sigma_{uo_i} \setminus \Sigma_f \end{cases} . \end{array} \tag{5.3}$$

Para verificar a existência de sequências que levam à violação da condição de diagnosticabilidade robusta generalizada dada pela definição 5.1, será apresentado um algoritmo de tempo polinomial inspirado no algoritmo proposto por Moreira *et al.* (2011), apresentado no capítulo 2, e, na sequêcia, será apresentado um teorema que prova sua validade. Além disso, será apresentada a análise da complexidade computacional do algoritmo.

5.2.1 Construção do verificador de robustez generalizada

Será agora apresentado um algoritmo que verifica a diagnosticabilidade robusta generalizada.

Algoritmo 5.1 *(Verificador de robustez generalizada)*

Passo 1 *Para cada modelo* $G_i = (X_i, \Sigma, f_i, \Gamma_i, x_{i,0})$, $i = 1, 2, \ldots, m$, *construa o autômato* $G_{R_i} = (X_i, \Sigma_{R_i}, f_{R_i}, \Gamma_{R_i}, x_{i,0})$, *sendo* $\Sigma_{R_i} = R_i(\Sigma)$, $\Gamma_{R_i}(x) = R_i[\Gamma_i(x)]$, *e* $f_{R_i}(x, R_i(\sigma)) = f_i(x, \sigma)$ *para todo* $x \in X_i$ *e* $\sigma \in \Gamma_i(x)$.

PASSO 2 *Construa os autômatos de falha* G_{F_i}, $i = 1, 2, \ldots, m$, *como segue:*

- *Passo 2.1: Calcule* $G_{\ell_i} = G_{R_i} \| A_\ell$ *em que* A_ℓ *é o autômato rotulador de falhas mostrado na figura 2.5, e marque todos os estados de* G_{ℓ_i} *cujas segundas coordenadas são iguais a* Y.
- *Passo 2.2: Obtenha o autômato de falha* $G_{F_i} = CoAc(G_{\ell_i}) = (X_{F_i}, \Sigma_{R_i}, f_{F_i}, x_{0,F_i})$.
- *Passo 2.3: Refine o conjunto de eventos de* G_{F_i} *para* $\Sigma_{F_i} = \Sigma_{R_i} \cup \Sigma_o$.

PASSO 3 *Construa os autômatos de não falha* $G_{N_{R_i}}$, $i = 1, 2, \ldots, m$, *da seguinte forma:*

- *Passo 3.1: Define* $\Sigma_{N_i} = \Sigma_{R_i} \setminus \Sigma_f$, *e obtenha o autômato* $A_{N_i} = (\{N\}, \Sigma_{N_i}, f_{N_i}, N)$ *que tem um único estado com um autolaço rotulado com todos os eventos de* Σ_{N_i}.
- *Passo 3.2: Construa* $G_{N_{R_i}} = G_{R_i} \times A_{N_i} = (X_{N_{R_i}}, \Sigma_{R_i}, f_{N_{R_i}}, \Gamma_{N_{R_i}}, x_{0,N_{R_i}})$.
- *Passo 3.3: Refine o conjunto de eventos de cada* $G_{N_{R_i}}$ *para* $\Sigma_{N_{R_i}} = \Sigma_{R_i} \setminus \Sigma_f$.

PASSO 4 *Construa os autômatos aumentados* $G_{\mathcal{N}_i} = (X_{\mathcal{N}_i}, \Sigma_{\mathcal{N}_i}, f_{\mathcal{N}_i}, x_{0,\mathcal{N}_i})$, $i = 1, 2, \ldots, m$, *a partir de* $G_{N_{R_i}}$ *da seguinte forma:*

- *Passo 4.1: Defina* $\Sigma_{\mathcal{N}_i} = \Sigma_{N_{R_i}} \cup \Sigma_o$.
- *Passo 4.2: Defina* $x_{0,\mathcal{N}_i} = x_{0,N_{R_i}}$.
- *Passo 4.3: Adicione um novo estado* D_i *no espaço de estados de* $G_{N_{R_i}}$. *Então,* $X_{\mathcal{N}_i} = X_{N_{R_i}} \cup \{D_i\}$.
- *Passo 4.4: Para cada* $x_{\mathcal{N}_i} \in X_{N_{R_i}}$ *defina:*

$$f_{\mathcal{N}_i}(x_{\mathcal{N}_i}, \sigma) = \begin{cases} f_{N_{R_i}}(x_{\mathcal{N}_i}, \sigma), \text{ se } \sigma \in \Gamma_{N_{R_i}}(x_{\mathcal{N}_i}) \\ D_i, \text{ se } \sigma \in \Sigma_o \setminus \Gamma_{N_{R_i}}(x_{\mathcal{N}_i}) \\ \text{não definido, caso contrário} \end{cases}, \tag{5.4}$$

e para $x_{\mathcal{N}_i} = D_i$ *defina:*

$$f_{H_i}(x_{H_i}, \sigma) = \begin{cases} D_i, \text{ para todo } \sigma \in \Sigma_o \\ \text{não definido, caso contrário} \end{cases}. \tag{5.5}$$

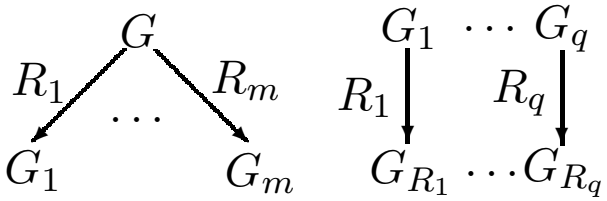
PASSO 5 *Para* $i = 1, 2, \ldots, m$*, calcule o autômato verificador* G_{V_i} *cujo* j*-ésimo estado* $x_{V_{ij}} \in X_{F_i} \times (\times_{q=1,q\neq i}^{m} X_{\mathcal{N}_q})$ *e o* j*-ésimo estado de* X_{F_i} *é* $x_{F_{ij}} \in X_i \times X_\ell$*, por uma composição de* G_{F_i},$G_{\mathcal{N}_1}$,...,$G_{\mathcal{N}_{i-1}}$,$G_{\mathcal{N}_{i+1}}$,...,$G_{\mathcal{N}_m}$*, seguindo o mesmo procedimento que a composição paralela* $G_{F_i} \| (\|_{j=1,j\neq i}^{m} G_{\mathcal{N}_j})$*, exceto quando o estado alcançado seja* $(x_{F_i}, D_1, \ldots, D_{i-1}, D_{i+1}, \ldots, D_m)$*, em que* $x_{F_i} \in X_{F_i}$*, caso em que o seu conjunto de eventos ativos é forçado a ser vazio, i.e.,*

$$\Gamma_{V_i}(x_{F_i}, D_1, \ldots, D_{i-1}, D_{i+1}, \ldots, D_m) = \emptyset.$$

□

Observação 5.2 *O verificador para perdas permanentes de sensores considera um único modelo* G *enquanto no caso generalizado considera-se um conjunto de autômatos* $\mathbb{G} = \{G_1, G_2, \ldots, G_q\}$ *como pode ser visto na figura 5.1. No caso da robustez em relação a perdas permanentes de sensores, a função* R_i *é utilizada para representar as possíveis formas de se observar o modelo* G*, conforme mostrado na figura 5.1(a), enquanto na abordagem generalizada a função renomeação diferencia os eventos não observáveis de cada modelo, conforme pode ser visto na figura 5.1(b), a fim de na composição paralela esses eventos serem eventos privados de cada modelo. Dessa forma, pode-se dizer que a abordagem que considera perdas permanentes está incluída na diagnose robusta generalizada uma vez que além de um único modelo observado com diferentes conjuntos de eventos observáveis* Σ_{o_i} *também inclui modelos que não necessariamente se originam do modelo nominal do sistema.*

Apesar da diferença entre a concepção do modelo, os algoritmos 3.3 e 5.1 definem da mesma forma os autômatos de falha, de não falha e aumentados, e a construção do verificador é semelhante. □



(a) (b)

Figura 5.1: Comparação entre a abordagem do verificador de robustez a perdas permanentes de sensores (a) e generalizada (b).

O teorema a seguir apresenta uma condição necessária e suficiente para verificar a condição de diagnosticabilidade robusta generalizada através do algoritmo 5.1.

Teorema 5.1 *A classe* $\mathbb{L}$ *não é robustamente diagnosticável com relação a* P_{o_i}, $i = 1, 2, \ldots, m$, *e* Σ_f *se e somente se existe um ciclo* $cl_i = (x_{V_{ik}}, \sigma_k, x_{V_{ik+1}}, \sigma_{k+1}, \ldots, \sigma_l, x_{V_{il}})$, *com* $l \geq k > 0$, *em ao menos um verificador* G_{V_i}, $i \in I_m$, *satisfaz a seguite condição:*

$$\exists j \in \{k, k+1, \ldots, l\} \text{ tal que } (\sigma_j \in \Sigma_{R_i}) \wedge (x_{F_{ij}} = \{x_{i_j}, Y\}). \tag{5.6}$$

Proof. A prova desse teorema é semelhante à prova do teorema 3.2. □

5.2.2 Complexidade computacional do algoritmo 5.1

A complexidade computacional do algoritmo 5.1 está representada na tabela 5.1, onde está mostrado o número máximo de estados e transições de todos os autômatos que devem ser computados para se obter o autômato verificador G_{V_i}. Suponha que existam m modelos possíveis em $\mathbb{G}$.

O primeiro passo do algoritmo 5.1 é a construção do autômato renomeado G_{R_i} de G_i. Como G_{R_i} é o autômato G_i com a renomeação dos eventos pertencentes ao conjunto Σ_{uo_i}, o número dos estados e transições de G_{R_i} são os mesmos de G_i.

O segundo passo do algoritmo 5.1 é a construção do autômato G_{F_i}. Para tanto, é necessário primeiro construir o autômato A_ℓ com dois estados, N e Y cujas transições são rotuladas somente por eventos de falha e, em seguida, obter $G_{\ell_i} = G_{R_i} \| A_\ell$. Note que $L(G_{\ell_i}) = L(G_i)$ e que os estados de G_{ℓ_i} são da forma (x, N) ou (x, Y), com

Tabela 5.1: Complexidade computacional do algoritmo 5.1

	No. de estados	No. de transições
G_i	$\|X_i\|$	$\|X_i\|\|\Sigma\|$
A_ℓ	2	2
G_{ℓ_i}	$2\|X_i\|$	$2\|X\|\|\Sigma\|$
G_{F_i}	$2\|X_i\|$	$2\|X_i\|\|\Sigma\|$
A_{N_i}	1	$\|\Sigma\| - \|\Sigma_f\|$
$G_{N_{R_i}}$	$\|X_i\|$	$\|X_i\|(\|\Sigma\| - \|\Sigma_f\|)$
$G_{\mathcal{N}_i}$	$\|X_i\| + 1$	$\|X_i\|(\|\Sigma_{uo_i}\| - \|\Sigma_f\| + \|\Sigma_o\|) + \|\Sigma_o\|$
G_{V_i}	N_{V_i}=$2\|X_i\|\prod_{j\neq i}(\|X_j\|+1)$	$N_{V_i}\,[m(\|\Sigma\| - \|\Sigma_f\|) + \|\Sigma_f\|]$
Complexidade		$O\Big(m\|X_i\|\prod_{j\neq i}\|X_j\|(\|\Sigma\| - \|\Sigma_f\|)\Big)$

$x \in X_i$. Portanto, o número máximo de estados de G_{ℓ_i} é $2|X_i|$. Finalmente, G_{F_i} é obtido tomando-se a parte coacessível de G_{ℓ_i} ($G_{F_i} = CoAc(G_{\ell_i})$), e então, o número máximo de estados e transições de G_{F_i} são $2|X_i|$ e $2|X|_i|\Sigma|$, respectivamente.

No passo 3, um autômato composto de um único estado A_{N_i} é utilizado para se encontrar o autômato de não falha $G_{N_{R_i}} = G_{R_i} \times A_{N_i}$. Portanto, como A_{N_i} é um autômato de um único estado com um autolaço rotulado por $\Sigma_{R_i} \setminus \Sigma_f$, o número máximo de estados e transições de $G_{N_{R_i}}$ são $|X_i|$ e $|X_i|(|\Sigma| - |\Sigma_f|)$, respectivamente.

No passo 4 o autômato aumentado $G_{\mathcal{N}_i}$ é obtido a partir de $G_{N_{R_i}}$ pela adição do estado D_i e as transições dos estados de $G_{N_{R_i}}$ para D_i rotulados com eventos de Σ_o que não pertencem ao conjunto dos eventos ativos de cada estado. Portanto, o número máximo de estados e transições de $G_{\mathcal{N}_i}$ são, respectivamente, $|X_i| + 1$ e $|X_i|(|\Sigma_{uo_i}| - |\Sigma_f| + |\Sigma_o|) + |\Sigma_o|$.

Finalmente, no passo 5, o verificador G_{V_i} é obtido a partir de uma nova composição entre G_{F_i} e $G_{\mathcal{N}_j}$, para todo $j \in I_{m_i}$, baseado na composição paralela com uma regra adicional que estabelece que alguns estados alcançados pela composição paralela são forçados a ser estados de bloqueio. Portanto, o número de estados e transições de G_{V_i} são no pior caso igual a $2|X_i|\prod_{j=1,j\neq i}^{m}(|X_j| + 1)$ e $[2|X_i|\prod_{j=1,j\neq i}^{m}(|X_j| + 1)][m(|\Sigma| - |\Sigma_f|) + |\Sigma_f|]$, respectivamente. Logo, a complexi-

dade do algoritmo 5.1 é $O(m|X_i|\prod_{j=1,j\neq i}^{m}|X_j|(|\Sigma|-|\Sigma_f|))$.

Observação 5.3 *Note que a complexidade computacional do algoritmo 5.1 é* $O(m|X_i|\prod_{j=1,j\neq i}^{m}|X_j|(|\Sigma|-|\Sigma_f|))$*, que é menor que a complexidade do algoritmo proposto por Takai (2010), considerando o número de estados de* $G_{N_{R_i}}$ *e* G_i *iguais, que é* $O(|X_i|^3\prod_{j=1,j\neq i}^{m}|X_j||\Sigma|^{m+1})$*. Vale a pena ressaltar que o tamanho do autômato verificador* G_{V_i} *é, em geral, menor que o pior caso apresentado na tabela 5.1 uma vez que o algoritmo busca somente sequências em* $L_i \setminus L_{N_i}$ *e* L_{N_j}*,* $i \neq j$*, que podem levar a uma violação da condição de diagnosticabilidade robusta generalizada.* □

5.3 Exemplo

Nessa seção, será apresentado um exemplo que ilustra a construção do verificador de robustez generalizada e sua utilização na análise da diagnose robusta generalizada de um SED. Seja $\mathbb{G} = \{G_1, G_2, G_3\}$ a classe de autômatos mostrada na figura 5.2 que descreve o comportamento de um SED, e suponha que $\Sigma = \{a, b, c, d, e, \sigma_u, \sigma_f\}$ seja o conjunto de todos os eventos utilizados na modelagem do sistema. Além disso, sejam $\Sigma_{o_1} = \{a, b, c\}$, $\Sigma_{o_2} = \{a, b, c, e\}$ e $\Sigma_{o_3} = \{a, b, d, e\}$ os conjuntos dos eventos observáveis de G_1, G_2 e G_3, respectivamente. O objetivo aqui é verificar se a classe de linguagens $\mathbb{L}$ gerada pela classe de autômatos $\mathbb{G}$ é robustamente diagnosticável com relação P_{o_i}, $i = 1, 2, \ldots, m$, e $\Sigma_f = \{\sigma_f\}$.

Inicialmente, note que $\Sigma_o = \bigcup_{i=1}^{3}\Sigma_{o_i} = \{a, b, c, d, e\}$. Agora, de acordo com o algoritmo 5.1, o primeiro passo é obter o autômato G_{R_i}, para $i = 1, 2, 3$, pela renomeação dos eventos em $\Sigma_{uo_i} \setminus \Sigma_f$. Portanto, os eventos "$d$", "$e$" e "$\sigma_u$" devem ser renomeados, respectivamente, para d_{R_1}, e_{R_1} e $\sigma_{u_{R_1}}$ no autômato G_1, e os eventos "c" para c_{R_3} em G_3. Observe que nenhum evento precisa ser renomeado no autômato G_2. Os diagramas de transição de estados dos autômatos G_{R_i}, $i = 1, 2, 3$, não serão mostrados uma vez que eles são idênticos aos digramas de G_i, $i = 1, 2, 3$, exceto pela renomeação.

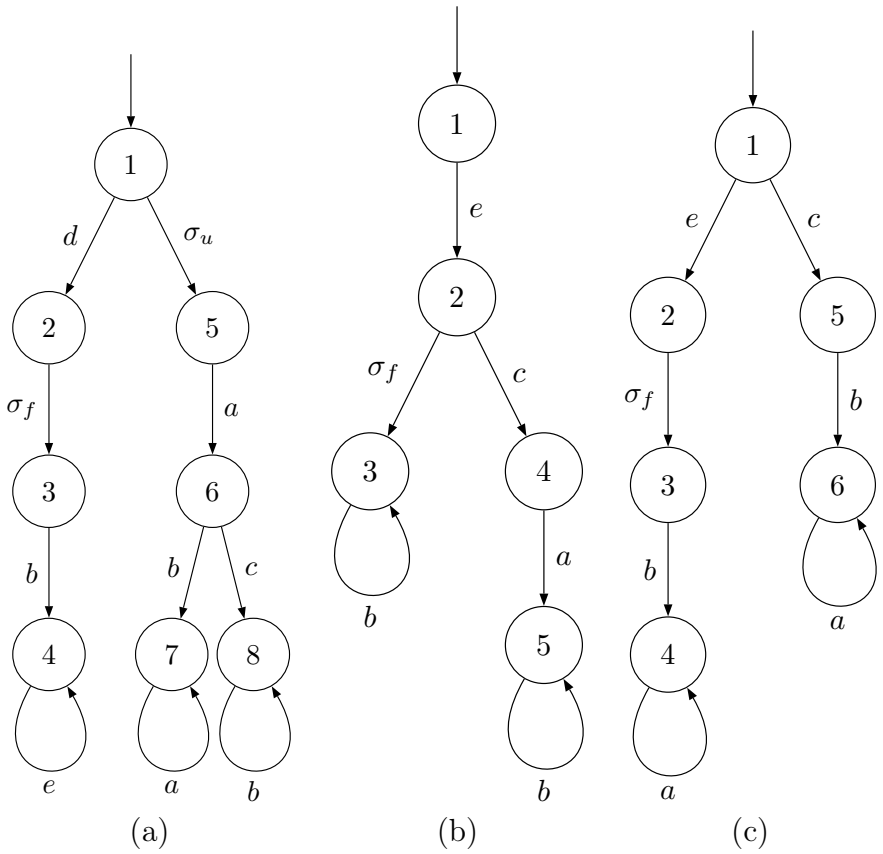


Figura 5.2: Classe de autômatos $\mathbb{G} = \{G_1, G_2, G_3\}$: (a) G_1 com $\Sigma_{o_1} = \{a, b, c\}$; (b) G_2 com $\Sigma_{o_2} = \{a, b, c, e\}$; (c) G_3 com $\Sigma_{o_3} = \{a, b, d, e\}$.

O próximo passo do algoritmo 5.1 é calcular os autômatos de falha G_{F_i}, $i = 1, 2, 3$. Seguindo os passos 2.1 a 2.3, são obtidos os autômatos G_{F_1}, G_{F_2} e G_{F_3} mostrados na figura 5.3. Note que, embora apenas os eventos b, d_{R_1}, e_{R_1} e σ_f aparecem no diagrama de transição de estados de G_{F_1}, seu conjunto de eventos é $\Sigma_{F_1} = \Sigma_{R_1} \cup \Sigma_o = \{a, b, c, d, e, d_{R_1}, e_{R_1}, \sigma_{u_{R_1}}, \sigma_f\}$. A mesma análise pode ser feita para G_{F_2} e G_{F_3} levando a $\Sigma_{F_2} = \{a, b, c, d, e, d_{R_2}, \sigma_{u_{R_2}}, \sigma_f\}$ e $\Sigma_{F_3} = \{a, b, c, d, e, c_{R_3}, \sigma_{u_{R_3}}, \sigma_f\}$. Após obter o comportamento de falha de G_{R_1}, G_{R_2} e G_{R_3}, o passo seguinte é obter os autômatos de não falha $G_{N_{R_1}}$, $G_{N_{R_2}}$ e $G_{N_{R_3}}$ que representam o comportamento de não falha de G_{R_1}, G_{R_2} e G_{R_3} e, em seguida, obter os autômatos aumentados $G_{\mathcal{N}_1}$, $G_{\mathcal{N}_2}$ e $G_{\mathcal{N}_3}$. Os diagramas de transição de estados de $G_{\mathcal{N}_1}$, $G_{\mathcal{N}_2}$ e $G_{\mathcal{N}_3}$ estão representados na figura 5.4. Note que os conjuntos de eventos dos autômatos aumentados são $\Sigma_{\mathcal{N}_1} = \Sigma_{F_1} \setminus \{\sigma_f\}$, $\Sigma_{\mathcal{N}_2} = \Sigma_{F_2} \setminus \{\sigma_f\}$ e $\Sigma_{\mathcal{N}_3} = \Sigma_{F_3} \setminus \{\sigma_f\}$.

O passo 5 do algoritmo 5.1 é calcular os autômatos verificadores G_{V_1}, G_{V_2} e G_{V_3},

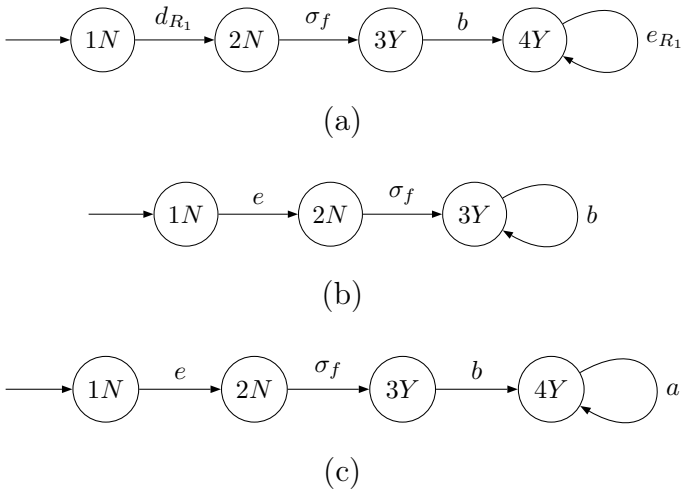
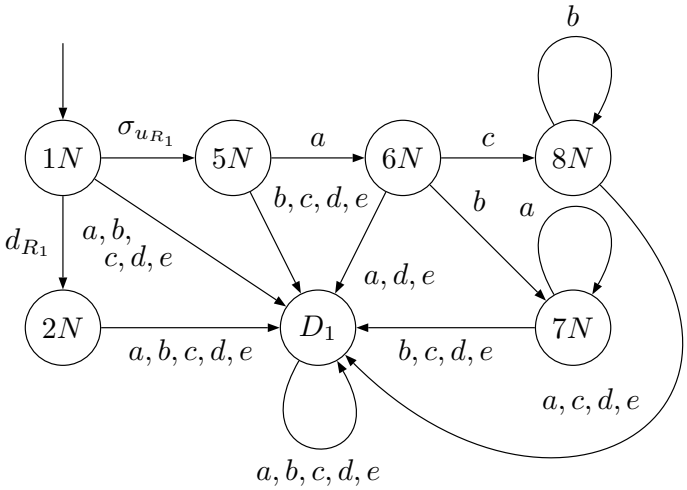


Figura 5.3: Autômatos de falha G_{F_1} (a), G_{F_2} (b) e G_{F_3} (c).

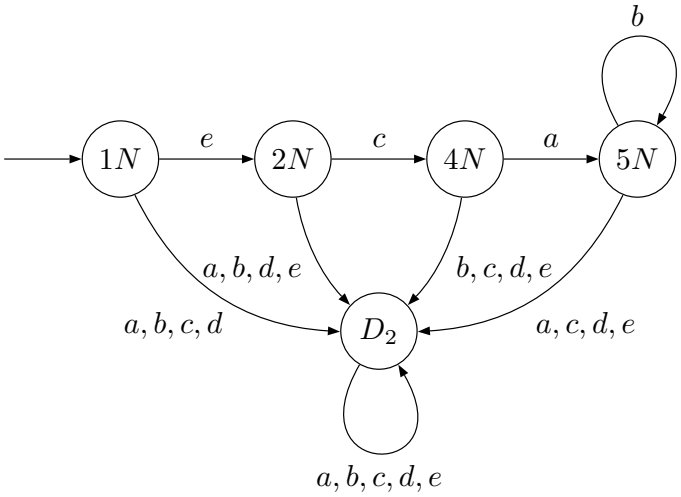
cujos diagramas de transição de estados estão representados nas figuras 5.5, 5.6 e 5.7, respectivamente.

Finalmente, para verificar se $\mathbb{L}$ é robustamente diagnosticável com relação a P_{o_i}, $i = 1, 2, \ldots, m$, e $\Sigma_f = \{\sigma_f\}$, deve-se utilizar o teorema 5.1. Para tanto, considere primeiro o verificador G_{V_1} representado na figura 5.5. Note que o ciclo $(4YD_23N, e_{R_1}, 4YD_23N)$ é formado por um evento de Σ_{R_1}. Portanto, embora os verificadores G_{V_2} (figura 5.6) e G_{V_3} (figura 5.7) não tenha ciclos com eventos em Σ_{R_2} e Σ_{R_3}, respectivamente, cujas as primeiras componentes dos seus estados têm rótulo de falha, pode-se concluir que $\mathbb{L}$ não é robustamente diagnosticável com relação a P_{o_i}, $i = 1, 2, \ldots, m$, e $\Sigma_f = \{\sigma_f\}$.

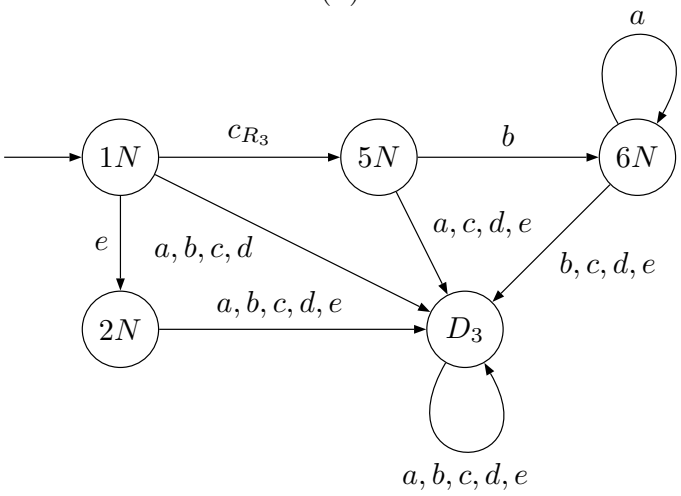
Observando G_{V_1}, note que a sequência responsável pela não diagnosticabilidade robusta é $s_{V_1} = d_{R_1}\sigma_f c_{R_3} b e_{R_1}^n$, com n arbitrariamente grande, levando G_{V_1} do seu estado inicial para o estado $4YD_23N$ e fazendo um ciclo nesse estado. Como $s_{F_1} = P_{F_1}(s_{V_1}) = d_{R_1}\sigma_f b e_{R_1}^n$ e $s_{\mathcal{N}_3} = P_{\mathcal{N}_3}(s_{V_1}) = c_{R_3} b$, então após utilizar a função renomeação inversa obtém-se $s_1 = d\sigma_f b e^n \in L_1$ e $s_3 = cb \in L_3$. Portanto, $P_{o_1}(s_1) = P_{o_3}(s_3) = b$, o que implica que, quando a sequência s_1 ocorrer, não será possível concluir se o sistema está no estado 4 de G_1, que é um estado alcançado após a ocorrência do evento de falha σ_f, ou no estado 6 de G_3, que é um estado alcançado por uma sequência normal.



(a)



(b)



(c)

Figura 5.4: Autômatos aumentados $G_{\mathcal{N}_1}$ (a), $G_{\mathcal{N}_2}$ (b), e $G_{\mathcal{N}_3}$ (c).

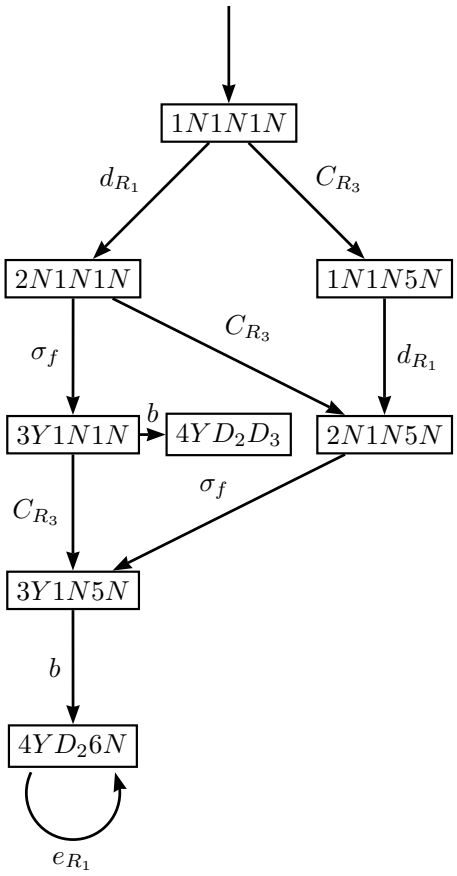


Figura 5.5: Autômatos verificadores G_{V_1}.

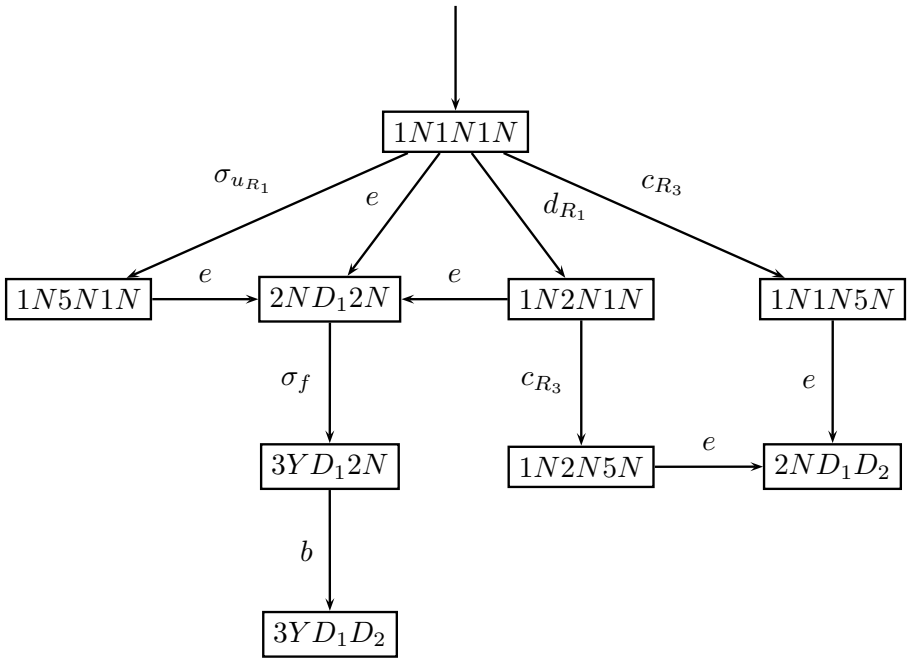


Figura 5.6: Autômatos verificadores G_{V_2}.

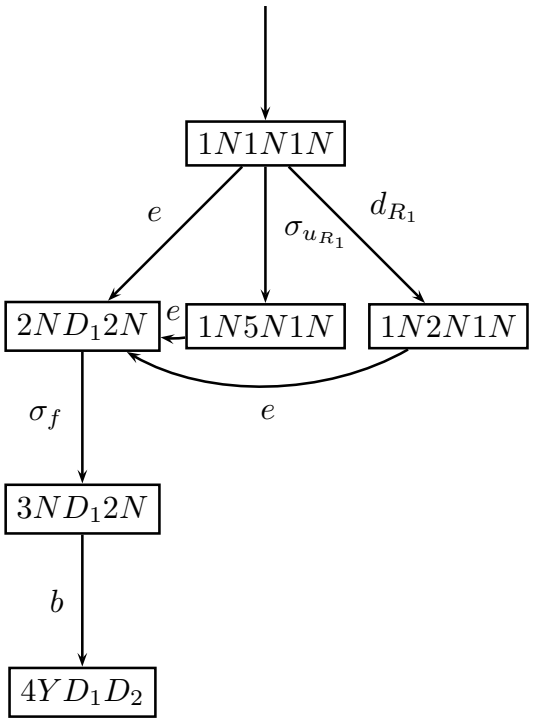


Figura 5.7: Autômato verificador G_{V_3}.

5.4 Comentários finais

Nesse capítulo foi proposta uma nova definição de diagnosticabilidade robusta que engloba as definições prévias de diagnosticabilidade robusta a perdas permanentes e intermitentes de sensores. Ao invés de uma única linguagem, considera-se agora uma classe de linguagens geradas por uma classe de autômatos que modelam o comportamento do sistema. Condições necessárias e suficientes para a diagnosticabilidade robusta são apresentadas. Foi também desenvolvido neste capítulo um algoritmo de complexidade polinomial para a verificação da diagnosticabilidade robusta de uma classe de linguagens.

Capítulo 6

Conclusão e trabalhos futuros

Esse trabalho considerou a diagnose robusta de sistemas a eventos discretos. Uma das contribuições da tese foi uma nova definição de ciclos escondidos, apresentada no capítulo 4, que engloba a definição existente na literatura (Basilio e Lafortune, 2009) e que foi revista no capítulo 2. A inclusão do conceito de ciclos escondidos permitiu remover uma hipótese usualmente feita nos trabalhos de diagnose de falhas de sistemas a eventos discretos que impede que sistemas que possuam ciclos de estados ligados por eventos não-observáveis possam ser considerados. Isso levou aos seguintes resultados: *(i)* uma nova condição necessária e suficiente para a diagnosticabilidade que depende tanto dos ciclos indeterminados observados quanto escondidos; *(ii)* o teorema 2.1 proposto por Sampath *et al.* (1995) pode ser visto como um caso particular do teorema 4.2, que leva em conta a existência de ciclos escondidos na análise da diagnosticabilidade robusta.

Uma outra contribuição da tese foi o desenvolvimento de um verificador de robustez a perdas permanentes de sensores no capítulo 3 que apresenta uma complexidade computacional menor que o diagnosticador união de perdas permanentes de sensores proposto em Lima *et al.* (2010). Isso completa o estudo de perdas permanentes de sensores uma vez que agora tem-se uma ferramenta para diagnose robusta *online* dada pelo diagnosticador proposto em Lima *et al.* (2010) e uma ferramenta para análise *offline* da diagnose robusta a perdas permanentes de sensores.

Dando continuidade ao trabalho de diagnose robusta a perda de sensores, no

capítulo 4 foi proposta uma abordagem totalmente nova para se modelar perdas intermitentes de sensores utilizando uma nova operação: a dilatação. Além disso, foi proposto um diagnosticador robusto, tendo sido também mostrado que esse diagnosticador pode ser obtido por dois caminhos: diretamente a partir do diagnosticador centralizado G_d associado ao sistema G e a partir do sistema G. Nesse capítulo foram também consideradas a análise da diagnose robusta a perdas intermitentes de sensores utilizando verificadores e a codiagnose robusta. Assim, todos os aspectos da diagnose foram abordados no contexto da diagnose robusta a perdas intermitentes de sensores.

Por fim, foi proposta no capítulo 5 uma generalização das abordagens apresentadas nos capítulos anteriores desse trabalho e daquelas propostas por Lima *et al.* (2010), Carvalho *et al.* (2010) e Takai (2010). Nessa formulação considera-se uma classe de modelos e supõe-se que cada modelo da classe tenha o seu próprio conjunto de eventos observáveis. Além disso, foi proposto um verificador cuja complexidade computacional é menor que a do algoritmo proposto por Takai (2010).

Possíveis tópicos de pesquisa que podem dar continuidade a esse trabalho são listados a seguir:

1. Diagnose de falhas intermitentes dos sensores.

   Lima *et al.* (2010) introduz o conceito de diagnose robusta de um evento de falhas supondo que o sistema seja sujeito a perdas permanentes de sensores. Como um subproduto da construção do diagnosticador união é possível que a ocorrência de falhas permanentes nos sensores possam ser detectadas (capítulo 3). Entretanto, uma nova abordagem para a diagnose de falhas intermitentes nos sensores pode ser considerada. A ideia é propor um novo autômato rotulador que tenha um estado que rotula quando o sensor volta a funcionar após a ocorrência de sua falha. Como consequência, será necessário redefinir as classificações dos estados e de ciclos indeterminados. Além disso, será preciso apresentar novas condições necessárias e suficientes para a diagnose robusta a falhas intermitentes nos sensores utilizando novos diagnosticadores

e verificadores.

2. Aplicação da teoria desenvolvida nesse trabalho a sistemas reais.

   O objetivo inicial era aplicar a teoria desenvolvida nessa tese ao sistema de processo de separação óleo-gás (Manyari-Rivera *et al.*, 2007) ou ao sistema HVAC ou AVAC (aquecimento, ventilação e ar condicionado) (Sampath *et al.*, 1996) com o objetivo de validar os resultados teóricos aqui apresentados. Contudo, o sistema de processo de separação óleo-gás de Manyari-Rivera *et al.* (2007) possui diversos sensores virtuais que dependem da medida da temperatura e que foram introduzidos para tornar o sistema diagnosticável. Portanto, ao considerar perda intermitentes de sensores, o sensor de temperatura pode parar de funcionar, tornando o sistema trivialmente não robusto no que se refere a perdas de sensores. No que se refere ao sistema AVAC, o sensor de pressão na bomba e o sensor de fluxo na válvula são adicionados ao sistema pelo mapa de sensores para torná-lo diagnosticável. Dessa forma, assim como no processo de separação óleo-gás de Manyari-Rivera *et al.* (2007), ao considerar perda intermitentes de sensores o sistema não é robustamente diagnosticável.

3. Diagnosticador robusto no caso da diagnosticabilidade generalizada.

   Na diagnose robusta generalizada proposta no capítulo 5 somente foi considerada a análise, tendo sido proposto um método baseado no verificador. Um trabalho futuro é desenvolver uma metodologia para diagnose robusta *online*. Para tanto, um caminho é utilizar diagnosticadores de acordo com a ideia proposta por Jesus *et al.* (2010).

4. Diagnose robusta, segura e ativa em sistemas com falhas intermitentes nos sensores.

   Paoli *et al.* (2008) utilizam a ideia de diagnose segura para controle supervisório e propõem o conceito de controlabilidade segura de modo a garantir que o sistema mantenha-se fora da zona proibida após a ocorrência de uma falha. Além disso, é definida a noção de sistema tolerante a falhas ativo, permitindo

que o sistema opere de forma segura mesmo após a ocorrência de uma falha. Baseado nessas ideias, pode ser considerada a aplicação da diagnose segura e ativa em sistemas sujeitos a falhas intermitentes.

Referências Bibliográficas

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

Angeli, C. (2008), "Online expert systems for fault diagnosis in technical processes," *Expert Systems*, 25(2), 115–132.

Athanasopoulou, E., Lingxi, L., e Hadjicostis, C. (2010), "Maximum likelihood failure diagnosis in finite state machines under unreliable observations," *IEEE Transactions on Automatic Control*, 55(3), 579 –593.

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., Carvalho, L.K., e Moreira, M.V. (2010), "Diagnose de falhas em sistemas a eventos discretos modelados por autômatos finitos," *Revista Controle & Automação*, 21(5), 510–533.

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

Basilio, J.C., Lafortune, S., Moreira, M.V., e Lima, S.T.S. (2011), "Computation of Minimal Event Bases that Ensure Diagnosability," *Discrete Event Dynamic Systems: Theory and Applications* (Submetido para publicação).

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.

Biswas, G., Cordier, M.O., Lunze, J., Trave-Massuyes, L., e Staroswiecki, M. (2004), "Diagnosis of complex systems: bridging the methodologies of the FDI and DX communities," *IEEE Transactions on Systems, Man, and Cybernetics — Part b: Cybernetics*, 34(5), 2159–2162.

Blanke, M., Kinnaert, M., Lunze, J., e Staroswiecki, M. (2006), *Diagnosis and fault-tolerant control*, 2nd ed., Germany, Springer.

Boel, R.K., e van Schuppen, J.H. (2002), "Decentralized failure diagnosis for discrete-event systems with costly communication between diagnosers," in *Proc. of the 2002 International Workshop on Discrete Event Systems – WODES'02*, Zaragoza, Spain.

Carvalho, L.K., Basilio, J.C., e Moreira, M.V. (2010), "Robust diagnosability of discrete event systems subject to intermittent sensor failures," in *Proceedings of 10th International Workshop on Discrete Event Systems*, Berlin, Germany, pp. 94–99.

Carvalho, L., Basilio, J., e Moreira, M. (2011), "Generalized robust diagnosability of discrete event systems," in *18th IFAC World Congress* (aceito para apresentação), August, Milan, Italy.

Cassandras, C.G., e Lafortune, S. (2007), *Introduction to discrete event systems*, 2nd ed., New York, Springer.

Chen, J., e Patton, R. (1999), *Robust model-based fault diagnosis for dynamic systems*, Norwell, MA, USA, Kluwer Academic Publishers.

Chen, Y.L., e Provan, G. (1997), "Modeling and diagnosis of timed discrete event systems - a factory automation example," in *Proceedings of the American Control Conference*, Vol. 1, 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," in *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*, New York, NY, Springer.

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.

Gertler, J.J. (1998), *Fault detection and diagnosis in engineering systems*, New York, Marcel Dekker.

Gissinger, G.L., Loung, M., e Reynaund, H.F. (2000), "Failure detection and isolation – optimal design of instrumentation system," in *Proc. IFAC 4th Symp. Fault Detection, Supervision, and Safety of Tech. Syst. SAFEPROCESS*, Budapest, Hungary, pp. 863–898.

Giua, A., e Seatzu, C. (2005), "Fault detection for discrete event systems using Petri nets with unobservable transitions," in *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.

Hamscher, W., Console, L., e Kleer, J. (1992), *Readings in model-based diagnosis*, Morgan Kaufmann.

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.

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

Jesus, T.C., Moreira, M.V., e Basilio, J.C. (2010), "Diagnóstico de falhas em tempo real de sistemas a eventos discretos descritos por autômatos finitos," in *Anais do 18º Congresso Brasileiro de Automática*, Bonito, Brasil, pp. 712–719.

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.

Kumar, R., e Takai, S. (2007), "Inference-Based ambiguity management in decentralized decision-making: decentralized control of discrete event systems," *IEEE Transactions on Automatic Control*, 52(10), 1783–1794.

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

Lefebvre, D., e Delherm, C. (2007), "Diagnosis of discrete-event systems 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., "Diagnose robusta de sistemas a eventos discretos sujeitos à perda permanente de sensores," Dissertação de mestrado, Universidade Federal do Rio de Janeiro, COPPE - Programa de engenharia elétrica (2010).

Lima, S.T.S., Basilio, J.C., Lafortune, S., e Moreira, M.V. (2010), "Robust diagnosis of discrete-event systems subject to permanent sensor failures," in *Preprints of the 10th International Workshop on Discrete Event Systems*, Berlin, Germany, pp. 100–107.

Lin, F. (1994), "Diagnosability of discrete event systems and its applications," *Journal of Discrete Event Dynamic Systems*, 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.

Luong, M., Maquin, D., e Ragot, J. (1997), "Sensor network design for failure detection and isolation," in *Proceedings of the 3rd IFAC Symposium SICICA*, Annecy, France.

Mangoubi, R. (1998), *Robust estimation and failure detection: A concise treatment*, Berlin, Springer -Verlag.

Manyari-Rivera, M., Basilio, J.C., e Bhaya, A. (2007), "Integrated fault diagnosis based on Petri net models," in *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," in *Proceedings of the 2010 American Control Conference*, Baltimore, MD, pp. 3353–3358.

Moreira, M.V., Jesus, T.C., e Basilio, J.C. (2011), "Polynomial time verification of decentralized diagnosability of discrete event systems," *IEEE Transactions on Automatic Control* (Aceito para publicação).

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," in *Proceedings of the 17th IFAC World Congress*, Vol. 17, South Korea.

Patton, R.J., Frank, P.M., e Clark, R. (2000), *Issues of fault diagnosis for dynamic systems*, Springer.

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

Pulido, B., e Alonso, C. (2004), "Possible conflicts: a compilation technique for consistency-based diagnosis," *IEEE Transactions on Systems, Mand, and Cybernetics – Part B: Cybernetics*, 34, 2192–2206.

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.

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.

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," in *Proceedings of IEEE International Conference on Robotics and Automation*, Vol. 2004, New Orleans, LA, pp. 541–546.

Rohloff, K., Yoo, T.S., e Lafortune, S. (2003), "Deciding co-observability is PSPACE-complete," *IEEE Transactions on Automatic Control*, 48(11), 1995–1999.

Rohloff, K.R. (2005), "Sensor failure tolerant supervisory control," in *Proc. and 2005 European Control Conference Decision and Control CDC-ECC '05. 44th IEEE Conference on*, 12–15 Dec., Seville, Spain, pp. 3493–3498.

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," in *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.

Sanchez, A.M., e Montoya, F.J. (2006), "Safe supervisory control under observability failure," *Discrete Event Dynamic Systems: Theory and Applications*, 16(4), 493–525.

Spanache, S., Escobert, T., e Travé-Massuyè, L. (2004), "Sensor optimization using genetic algorithms," in *Proc. 15th Int. Workshop on Principles of Diag. DX*, Carcassonne, France, pp. 179–183.

Supavatanakul, P., Lunze, J., Puig, V., e Quevedo, J. (2006), "Diagnosis of timed automata: Theory and application to the DAMADICS actuator benchmark problem," *Control Engineering Practice*, 14(6), 609–619.

Takai, S. (2010), "Robust failure diagnosis of partially observed discrete event systems," in *Proceedings of 10th International Workshop on Discrete Event Systems*, Berlin, Germany.

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

Thorsley, D., Yoo, T.S., e Garcia, H. (2008), "Diagnosability of stochastic discrete-event systems under unreliable observations," in *American Control Conference, 2008*, pp. 1158 –1165.

Travé-Massuyè, L., Escobet, T., e Milne, R. (2001), "Model based diagnosability and sensor placement – Application to a frame 6 gas turbine subsystem," in *Proc. Int. Joint Conf. Artif. Intell.*, Seattle, WA, pp. 205–212.

Travé-Massuyè, L., Escobet, T., e Olive, X. (2006), "Diagnosability analysis based on component-supported analytical redundancy relations," *IEEE Transactions on Systems, Mand, and Cybernetics – Part A: Systems and Humans*, 36, 1146–1160.

Tripakis, S. (2002), "Fault diagnosis for timed automata," in *Formal Techniques in Real Time and Fault Tolerant Systems (FTRTFT)*, ed. Springer-Verlag, Vol. 2469 of *Lecture notes in computer sciences*, pp. 205–221.

Ushio, T., Onishi, I., e Okuda, K. (1998), "Fault detection based on Petri net models with faulty behaviors," in *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 Garcia, H.E. (2008), "Diagnosis of behaviors of interest in partially-observed discrete-event systems," *Systems and Control Letters*, 57(12), 1023–1029.

Yoo, T.S., e Garcia, H.E. (2009), "Event counting of partially-observed discrete-event systems with uniformly and nonuniformly bounded diagnosis delays," *Discrete Event Dynamic Systems: Theory and Applications*, 19(2), 167–187.

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.H., e Wonham, W.M. (2003), "Fault diagnosis in discrete-event systems: framework and model reduction," *IEEE Transactions on Automatic Control*, 48(7), 1199–1212.

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

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