CODIAGNOSTICABILIDADE ROBUSTA DE SISTEMAS A EVENTOS DISCRETOS SUJEITOS A PERDAS DE OBSERVAÇÃO

Jean Hilaire Adebai Tomola

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: Marcos Vicente de Brito Moreira

Rio de Janeiro
Março de 2016

CODIAGNOSTICABILIDADE ROBUSTA DE SISTEMAS A EVENTOS DISCRETOS SUJEITOS A PERDAS DE OBSERVAÇÃO

Jean Hilaire Adebai Tomola

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. Marcos Vicente de Brito Moreira, D.Sc.

Prof. Amit Bhaya, Ph.D.

Prof. Paulo Eigi Miyagi, Dr.Eng.

Prof. Antonio Eduardo Carrilho da Cunha, D.Sc.

Prof. Max Hering de Queiroz, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2016

Adebai Tomola, Jean Hilaire

Codiagnosticabilidade Robusta de Sistemas a Eventos Discretos Sujeitos a Perdas de Observação/Jean Hilaire Adebai Tomola. – Rio de Janeiro: UFRJ/COPPE, 2016.

XII, 133 p.: il.; 29, 7cm.

Orientador: Marcos Vicente de Brito Moreira

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2016.

Referências Bibliográficas: p. 127 – 133.

1. Codiagnosticabilidade. 2. Diagnose de falha. 3. Robustez. I. Moreira, Marcos Vicente de Brito. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Se Deus é por nós, quem será contra nós? Romanos 8:31b

Agradecimentos

Agradeço muito a Deus por ter chegado até aqui pois, sem Sua ajuda nada disso seria possível.

Agradeço a minha esposa Rubia Souza Siqueira Adebai Tomola e aos meus filhos Jean François Siqueira Adebai Tomola e Thierry Oyenowo Siqueira Adebai Tomola pelo apoio dado e, principalmente, por compreenderem os momentos da minha ausência entre eles para me dedicar a este trabalho.

Agradeço a meus pais Esther Oyenowo e Missia Oyenowo (In memoriam) pelas orações a meu favor.

Agradeço a minha sogra Cecy Mariano Souza Siqueira e Rubens da Silva Siqueira pela ajuda e apoio proporcionados.

Agradeço a meu padrinho Maurício Bravo pela sua torcida e confiança.

Agradeço ao Pr. Daniel dos Santos Silva pelas orações e apoio.

Agradeço o prof. Marcos Vicente de Brito Moreira pela sua competência e excelência como professor orientador. Sua dedicação com amor e alegria pelo o que faz, torna-o um exemplo a ser seguido e reproduzido.

Agradeço ao colega Felipe Gomes de Oliveira Cabral pelo apoio e ajuda proporcionados nas diversas vezes que foram, por mim, solicitados. Sua ajuda foi de grande importância nesta conquista.

Em fim, agradeço a todos que de uma forma ou de outra, me ajudaram a concretizaram este sonho.

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

CODIAGNOSTICABILIDADE ROBUSTA DE SISTEMAS A EVENTOS DISCRETOS SUJEITOS A PERDAS DE OBSERVAÇÃO

Jean Hilaire Adebai Tomola

Março/2016

Orientador: Marcos Vicente de Brito Moreira

Programa: Engenharia Elétrica

Diversas noções de diagnosticabilidade robusta de Sistemas a Eventos Discretos (SED) têm sido apresentadas na literatura. Em todos esses trabalhos, o objetivo é a detecção de eventos de falha não-observáveis em SEDs sujeitos a incertezas na observação dos eventos e/ou no modelo da planta. Recentemente, as noções de diagnosticabilidade robusta de SEDs sujeitos a perdas permanentes de observação (DRPPO) e codiagnosticabilidade robusta de SEDs sujeitos a perdas intermitentes de observação (CRPIO) foram introduzidas, em que a incerteza está no conjunto de eventos observáveis do sistema. Nesse sentido, a linguagem do sistema é dita ser robustamente diagnosticável se é possível detectar a ocorrência da falha, com um atraso limitado, mesmo que alguns sensores falhem permanentemente ou intermitentemente em comunicar a ocorrência dos eventos para o diagnosticador. Neste trabalho, a noção de DRPPO é estendida para o caso decentralizado, levando à definição de codiagnosticabilidade robusta a perdas permanentes de observação (CRPPO). Além disso, é apresentada uma nova definição de CRPIO que permite abordar uma classe maior de problemas. A tese também aborda o problema de implementação online de diagnosticadores e é apresentado um esquema eficiente para a codiagnose robusta. Outra contribuição da tese são algoritmos com complexidade polinomial no número de estados e eventos do sistema para a verificação de CRPPO e CRPIO, e para o cálculo do limite de atraso para a codiagnose robusta.

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

ROBUST CODIAGNOSABILITY OF DISCRETE EVENT SYSTEMS SUBJECT TO LOSS OF OBSERVATIONS

Jean Hilaire Adebai Tomola

March/2016

Advisor: Marcos Vicente de Brito Moreira

Department: Electrical Engineering

Different notions of robust diagnosability of discrete-event systems (DESs) have been introduced in the literature. In all these works, the objective is the detection of unobservable fault events in DESs subject to uncertainties in the observation of the events and/or in the plant model. Recently, the notions of robust diagnosability of DESs against permanent loss of observations (RDPLO) and robust codiagnosability against intermitent loss of observations (RCILO) have been introduced, where the uncertainty is in the observable event set of the system. In this regard, the language generated by the system is said to be robustly diagnosable if it is possible to detect the fault occurrence, within a bounded delay, even when some sensors permanently or intermittently fail to communicate the occurrence of the events to the diagnoser. In this work, we extend the definition of RDPLO to the decentralized case leading to the definition of robust codiagnosability against permanent loss of observations (RCPLO). Moreover, we present a new definition of RCILO, that allows a large class of problems to be addressed. The thesis also addresses the issue of online implementation of diagnosers, and we propose an efficient scheme to carry out online robust codiagnosis. Another contribution of the thesis is the development of polynomial time algorithms in the number of states and events of the system for the verification of RCPLO, RCILO, and for the computation of the delay bound for robust codiagnosis.

Sumário

Lista de Figuras

Capítulo 1

Introdução

Em um mundo em que a demanda por produtos manufaturados de qualidade, maior segurança e por melhor conforto está sempre em crescimento, faz-se necessário dispor de sistemas produtivos seguros, eficazes, confiáveis e ágeis, capazes de atender aos anseios da humanidade. Em diversas unidades produtivas, máquinas são empregadas no lugar de humanos reduzindo assim, em alguns processos, o controle e a supervisão humana. Esses sistemas precisam, dentre outras coisas, serem resistentes a falhas, garantindo um funcionamento contínuo ou uma parada rápida para manutenção, em situações de anormalidade. Assim, a identificação de uma falha, sua previsão, ou até mesmo sua localização e correção, torna-se um grande desafio para os engenheiros e demais profissionais das diversas áreas da ciência. A falha é entendida, aqui, como um evento indesejável (máquina quebrada, válvula emperrada, sensor defeituoso, etc) que causa alteração no comportamento do sistema levando-o a se comportar de forma anormal e, dependendo do grau de sua importância, pode ser tolerada ou não. O evento responsável por tal comportamento anormal do sistema é dito ser um evento de falha. O evento de falha pode ser observado, desde que se tenha um sensor para isso, ou não observável, como ocorre na grande maioria dos casos.

Nesse contexto, dispositivos de sensoriamento são distribuídos ao longo de um sistema produtivo com a finalidade de informar, para uma determinada unidade de controle ou de supervisão, os valores atuais das variáveis do sistema. Com base nessas informações as unidades de controle ou de supervisão tomam suas decisões

seguindo regras preestabelecidas. Neste trabalho, o interesse está na ocorrência de uma falha no sistema. O processo de identificar a ocorrência de falhas em um sistema é, geralmente, classificado de três formas complementares: (i) detecção da falha, (ii) isolamento da falha e (iii) identificação da falha [2, 3]. A detecção da falha é uma funcionalidade que decide se o sistema está funcionando nas condições normais ou se houve ocorrência da falha. Caso a falha tenha ocorrido, o procedimento de isolamento da falha consiste em localizar o(s) componente(s) do sistema causador(es) dessa falha. Na identificação da falha, procura-se identificar sua natureza quanto a abrangência, importância, etc. Neste trabalho, é considerado somente o problema de detectar e isolar falhas em sistemas, isto é, o problema a ser considerado consiste na diagnose de falhas.

Os sistemas dinâmicos, de modo geral, podem ser classificados como sistemas de variáveis contínuas (SVCs) ou sistemas a eventos discretos (SEDs). Nos estudos de diagnose de falhas realizados em SVCs são empregados métodos baseados no modelo matemático, em que o comportamento físico do sistema é modelado utilizando-se equações diferenciais ou equações a diferenças finitas [3–6]. Um segundo método encontrado na literatura para a diagnose de falhas é baseado em inteligência artificial [7], [8], e pode ser aplicado em SVCs ou em SEDs. O terceiro método encontrado na literatura é baseado SEDs. Nesse tipo de sistema, o comportamento lógico e sequencial é mapeado através de modelos baseados em eventos e estados discretos [9]. O foco deste trabalho consiste no estudo de diagnose de falhas em SEDs.

Para garantir o correto funcionamento de um SED, diversos estudos foram realizados no sentido de construir dispositivos capazes de diagnosticar (detectar e isolar) a ocorrência de um evento de falha no sistema de forma segura e confiável. A construção de tais dispositivos requer, primeiramente, a modelagem do SED que pode ser feita utilizando-se, por exemplo, redes de Petri [10] ou autômatos [11]. A partir do modelo do SED, espera-se poder diferenciar o comportamento normal do sistema do comportamento anormal após a ocorrência do evento de falha. Para isso, é necessário criar um conjunto de regras ou protocolos fundamentados derivados de conceitos

desenvolvidos ao longo dos anos. Os métodos de diagnose de falhas a serem abordados neste trabalho têm sua origem em abordagens sobre observabilidade parcial de SEDs feitas nas décadas de 80 e 90. As abordagens lidavam com problemas de estimação de estado atual ou inicial no contexto de controle supervisório [12–16]. No entanto, esses problemas não estão diretamente relacionadas com o problema de detecção de falhas.

Inicialmente, o problema de diagnose de falhas foi abordado no contexto de SEDs em [17] e [18], quando foi introduzido o conceito da capacidade de se diagnosticar a ocorrência de uma falha em um sistema, baseado na observação de seus eventos. As definições e os algoritmos propostos em [18] forneceram os conceitos básicos e os fundamentos sobre diagnose e diagnosticabilidade de falhas em SEDs modelados por autômatos.

Em [18] e [19], um autômato determinístico, denominado de diagnosticador, é proposto para a diagnose online de falhas e para a verificação da diagnosticabilidade da linguagem de um SED. Desde então, o problema de diagnose de falhas de SEDs tem sido abordado em diversos trabalhos na literatura para sistemas modelados por autômatos determinísticos [9, 20–22], autômatos temporizados [23–28], autômatos probabilísticos [29–32], e redes de Petri [33–51]. Neste trabalho é estudado apenas a diagnose de falhas em SEDs modelados por autômatos determinísticos.

A diagnose de falhas pode ser realizada utilizando-se arquitetura centralizada [18, 19, 52–54] ou descentralizada [20, 22, 55, 56]. Na arquitetura centralizada, as ocorrências de eventos são comunicadas para um diagnosticador central, que identifica a ocorrência do evento de falha baseado nessas informações e no modelo da planta. Alguns sistemas, no entanto, são fisicamente distribuídos, e a comunicação de todos os eventos observáveis para um diagnosticador central não é possível ou apropriado. Nesses casos, diagnosticadores locais podem ser implementados em diferentes locais, e somente parte dos eventos observáveis do sistema é comunicada para cada diagnosticador local. Se os diagnosticadores locais não trocam informação entre si, um coordenador recebe a informação dos diagnosticadores com relação à ocor-

rência do evento de falha, e o coordenador indica a ocorrência da falha seguindo um protocolo. O esquema de diagnose de falhas descentralizado, com diagnosticadores locais que não trocam informação entre si, é chamado de codiagnose [55, 56].

Em DEBOUK *et al.* [22], diversos protocolos para diagnose descentralizada são propostos. Esses protocolos diferem com relação às regras de comunicação entre os diagnosticadores locais e o coordenador e com relação à regra de decisão sobre a ocorrência de falha tomada pelo coordenador. Nos protocolos 1 e 2 apresentados em DEBOUK *et al.* [22], os diagnosticadores locais informam para o coordenador após a observação de um evento, o seu estado atual, uma estimativa de estado e um bit de status que indica se o evento observado pelo diagnosticador local é observável por outro diagnosticador local. Essas informações são utilizadas pelo coordenador para diagnosticar as falhas do sistema. No Protocolo 3 de DEBOUK *et al.* [22], que será utilizado neste trabalho, não existe comunicação entre os diagnosticadores locais e o coordenador. Cada diagnosticador local detecta a ocorrência do evento de falha baseado nas suas próprias observações, e o coordenador indica a ocorrência da falha quando pelo menos um dos diagnosticadores locais identifica a sua ocorrência. Essa noção de diagnosticabilidade descentralizada é chamada na literatura de codiagnosticabilidade disjuntiva [56].

É importante ressaltar que nas abordagens centralizada e descentralizada, a planta é suposta ser completamente conhecida, e os sensores não podem falhar na detecção e comunicação da ocorrência dos eventos para os diagnosticadores.

Recentemente, diversas noções de diagnose de falhas robusta de SEDs têm sido apresentadas na literatura [57–65]. Em todos esses trabalhos, o objetivo principal é a detecção da ocorrência dos eventos de falha em um SED com incertezas nas observações dos eventos e/ou no modelo da planta.

A definição de codiagnosticabilidade $\mathcal{R}_m$-robusta de SEDs supondo que diagnosticadores locais podem falhar na comunicação da ocorrência do evento de falha para o coordenador é introduzida em [57]. Nesse contexto, o sistema é dito ser $\mathcal{R}_m$-robustamente codiagnosticável se a ocorrência de um evento de falha pode ser

detectada, após um número limitado de ocorrências de eventos, mesmo que m diagnosticadores locais parem de operar.

O problema de diagnose de falhas em SED sujeitos a perdas intermitentes de sensores é abordado em [58] e [63]. Nesses trabalhos, a observação do evento é incerto no sentido de que o sensor pode não ser capaz de identificar e comunicar a ocorrência do evento para o diagnosticador. Condições necessárias e suficientes para a diagnosticabilidade robusta da linguagem gerada pelo sistema com relação a perdas intermitentes de observação (DRPIO) são apresentadas em [63]. A definição de DRPIO é estendida para o caso descentralizado em [63], levando à definição de codiagnosticabilidade robusta a perdas intermitentes de observação (CRPIO). O problema de CRPIO pode também ser resolvido utilizando o método proposto em [66].

Os autores em [59] e [64] consideram perdas permanentes de sensores e apresentam a definição de diagnosticabilidade robusta a perdas permanentes de observação (DRPPO), *i.e.*, alguns sensores do sistema podem perder a capacidade de detectar e/ou comunicar a ocorrência dos eventos para o diagnosticador, e não podem se recuperar desse defeito. Nesses trabalhos, é suposto que o defeito no sensor ou na comunicação acontece antes da primeira ocorrência do evento a ser detectado pelo sensor com mal funcionamento.

Incertezas de modelagem no contexto de diagnose de falhas em SEDs foi primeiramente abordado em [60, 62]. A definição de diagnosticabilidade robusta apresentada em [62] requer que o sistema seja descrito por um conjunto de possíveis modelos $\{G_i : i \in I_q\}$ com um conjunto comum de eventos Σ, em que $I_q := \{1, 2, \ldots, q\}$ e $q \in \mathbb{N}$ denota o número de modelos do sistema. Nessa definição, cada modelo tem sua própria especificação de falha e a linguagem é viva, isto é, em qualquer estado x_i de G_i, existe um evento ativo pertencente a Σ que pode promover a mudança de estado do sistema. Diferentemente de [64], em [62], todos os modelos G_i possuem o mesmo conjunto de eventos observáveis.

Neste trabalho, a definição de DRPPO, introduzida em [64], é estendida para

o caso descentralizado, levando à definição de codiagnosticabilidade robusta a perdas permanentes de observação (CRPPO). Para tanto, a definição de bases para a diagnose é estendida para a definição de bases para a codiagnose. Além disso, é apresentada neste trabalho uma definição de CRPIO diferente da apresentada em [63], que permite abordar uma classe maior de problemas práticos. O trabalho também aborda o problema de implementação online de diagnosticadores robustos, e é proposto um esquema eficiente para implementação da codiagnose robusta. Outra contribuição do trabalho é o desenvolvimento de algoritmos em tempo polinomial para a verificação da CRPPO e CRPIO, e para o cálculo do limite de atraso para a codiagnose robusta em ambos os casos de perdas permanentes e intermitentes.

Este trabalho está estruturado da seguinte forma. No Capítulo 2 são apresentados os fundamentos teóricos para o entendimento deste trabalho. No Capítulo 3, a definição de CRPPO é apresentada. Além disso, algoritmos em tempo polinomial no número de estados e transições do sistema são apresentados para a verificação da codiagnosticabilidade robusta e para o cálculo do limite de atraso para a codiagnose robusta a perdas permanentes. No Capítulo 4, o problema de CRPIO é revisto, e uma nova abordagem com relação à forma como as observações dos eventos podem ser perdidas e uma nova definição de CRPIO são apresentadas. Um algoritmo em tempo polinomial no número de estados e transições do sistema é proposto para a verificação da CRPIO. Finalmente, no Capítulo 5, são apresentadas as conclusões e sugestões de trabalhos futuros.

Capítulo 2

Fundamentos teóricos

Neste capítulo, são apresentados os conceitos fundamentais para o estudo da diagnose de falhas em SEDs. Inicialmente, uma revisão da teoria de SEDs será apresentada e, em seguida, será considerado o problema da diagnose de falhas em SEDs utilizando arquitetura centralizada e descentralizada. Por fim, algoritmos de busca em grafos e de ordenação topológica, utilizados futuramente neste trabalho para a busca de caminhos cíclicos em grafos e para o cálculo do limite de atraso para a diagnose são revistos.

Este capítulo está organizado da seguinte forma. Na seção 2.1 são apresentadas as definições e os conceitos sobre sistemas a eventos discretos. Na seção 2.2 são apresentadas as definições e os conceitos sobre a diagnose centralizada assim como o autômato diagnosticador. Na seção 2.3 são apresentadas as definições e os conceitos sobre a diagnose descentralizada assim como o autômato verificador, e na seção 2.4 são apresentados os algoritmos de busca em profundidade, de ordenação topológica e de busca por componentes fortemente conexos. Na seção 2.5 são revistas as notações θ, O e Ω para estudo de complexidade computacional de algoritmos. Finalmente, na seção 2.6, são apresentadas as conclusões deste capítulo.

2.1 Sistemas a eventos discretos

Sistemas a eventos discretos (SEDs) são sistemas cujos estados são discretos e a evolução não é dirigida pelo tempo mas sim, pela ocorrência, em geral, assíncrona de eventos. O conjunto de eventos é denotado por Σ. Os eventos de Σ podem ser entendidos como ações realizadas pelo homem (por exemplo, acionar uma botoeira), ou como ocorrências instantâneas que satisfazem determinadas condições pré-determinadas (por exemplo, a detecção da presença de uma pessoa nas imediações de uma porta automática), ou ainda, como ocorrências espontâneas indesejáveis ou inesperadas (uma máquina apresenta defeito). A ocorrência de vários eventos que levam o sistema de um estado a outro é chamada de sequência, e o conjunto de todas as sequências possíveis de serem executadas forma a linguagem do sistema.

2.1.1 Linguagens e operações com linguagens

A linguagem, possui como alfabeto eventos de Σ. As sequências geradas a partir desses eventos constituem as palavras de uma linguagem. O comprimento de uma sequência s, denotado por $|s|$, consiste no número de eventos nela contidos, sendo assim, quando $|s| = 0$, a sequência é dita ser vazia e denotada por ε. O conjunto de todas as possíveis sequências de comprimento finito geradas a partir do conjunto dos eventos, incluindo a sequência vazia ε, é denominada de fecho de Kleene de Σ, denotado por Σ^*.

Formalmente, pode-se definir as linguagens da forma a seguir [67].

Definição 2.1. *(Linguagem) Uma linguagem, definida sobre um conjunto de eventos Σ, é um conjunto de sequências de comprimento finito formadas por eventos de Σ.* □

As seguintes operações podem ser definidas para linguagens [67].

Definição 2.2. *(Concatenação) Sejam $L_1, L_2 \subseteq \Sigma^*$, então a concatenação de L_1 e L_2 é dada por:*

$$L_1 L_2 := \{s \in \Sigma^* : (s = s_1 s_2) \wedge (s_1 \in L_1) \wedge (s_2 \in L_2)\}.$$

□

Definição 2.3. *(Fecho de Kleene) Seja $L \subseteq \Sigma^*$, então o fecho de Kleene de L é definido como:*

$$L^* := \{\varepsilon\} \cup L \cup LL \cup LLL \cup \ldots$$

□

Definição 2.4. *(Fecho de prefixo). Seja $L \subseteq \Sigma^*$, então o fecho de prefixo de L é definido como:*

$$\overline{L} := \{s \in \Sigma^* : (\exists t \in \Sigma^*)[st \in L]\}.$$

□

Note que se $L = \emptyset$ então, $\overline{L} = \emptyset$ e se $L \neq \emptyset$, então, $\varepsilon \in \overline{L}$.

Definição 2.5. *(Pós-linguagem) Seja $L \subseteq \Sigma^*$, então, a pós-linguagem de L após s, é a linguagem $L/s := \{t \in \Sigma^* : st \in L\}$. Por definição, $L/s = \emptyset$ se $s \notin \overline{L}$.* □

A operação de projeção é uma operação que pode ser aplicada tanto sobre uma sequência como sobre uma linguagem. Ela permite mapear, uma sequência de eventos pertencente a $\Sigma^\star$ em uma outra sequência pertencente a $\Sigma_o^\star$, em que $\Sigma_o \subset \Sigma$. Assim sendo, a projeção $P_o : \Sigma^* \to \Sigma_o^*$ é definida como [68]:

$$P_o(\varepsilon) := \varepsilon,$$

$$P_o(\sigma) = \begin{cases} \sigma, & \text{se } \sigma \in \Sigma_o \\ \varepsilon, & \text{se } \sigma \in \Sigma \setminus \Sigma_o, \end{cases}$$

$$P_o(s\sigma) = P_o(s)P_o(\sigma),$$

e

$$\text{para todo } s \in \Sigma^\star, \sigma \in \Sigma.$$

Note que a operação de projeção de uma sequência s consiste em apagar de s os eventos de Σ que não pertencem a Σ_o e manter os eventos pertencentes a Σ_o na ordem em que aparecem em s.

A projeção inversa $P_o^{-1} : \Sigma_o^\star \to 2^{\Sigma^\star}$ é definida da seguinte forma:

$$P_o^{-1}(t) = \{s \in \Sigma^\star : P_o(s) = t\}.$$

Com base nas definições de projeção e projeção inversa, pode-se estender a operação de projeção de sequências para linguagens. Seja $L \subseteq \Sigma^*$ uma linguagem definida sobre Σ, então, a projeção de L é dada por:

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

A projeção inversa de uma linguagem $L_o \subseteq \Sigma_o^*$ é dada por:

$$P_o^{-1}(L_o) := \{s \in \Sigma^* : (\exists t \in L_o)[P_o(s) = t]\}.$$

2.1.2 Autômatos

Um autômato é um dispositivo capaz de representar uma linguagem de acordo com regras bem definidas [67], e pode ser representado graficamente através de diagramas de transição de estados. Formalmente, um autômato determinístico G é definido como uma sêxtupla [11]:

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

em que X é o espaço de estados, Σ é o conjunto finito de eventos, $f : X \times \Sigma \to X$ é a função de transição de estados, $\Gamma : X \to 2^\Sigma$ é a função dos eventos ativos (viáveis), x_0 é o estado inicial de G e $X_m \subseteq X$ é o conjunto de estados marcados de G.

Visto que linguagens são formadas por sequências, na sua formalização estende-se a função de transição f do autômato G. Tal extensão requer que o domínio da função f seja mudado de $X \times \Sigma$ para $X \times \Sigma^*$ de forma recursiva, como mostrado a seguir:

$$f(x, \varepsilon) := x,$$
$$f(x, s\sigma) := f[f(x, s), \sigma], \text{ para } s \in \Sigma^*, \sigma \in \Sigma.$$

Nos diagramas de transição de estados, cada estado de G é representado por um círculo. O estado inicial de G é identificado por uma seta sem estado de origem, e os estados marcados são representados por dois círculos concêntricos e, geralmente, representam o cumprimento de uma determinada tarefa do sistema. A interligação dos estados é feita por meio de arcos orientados e rotulados com eventos de Σ.

Exemplo 2.1. *Seja* $G = (X, \Sigma, f, \Gamma, x_0, X_m)$ *em que* $X = \{0, 1, 2\}$, $\Sigma = \{a, b, c\}$, $f(0, a) = 1, f(1, b) = 2, f(1, c) = 1, f(2, a) = 0$, $\Gamma(0) = \{a\}, \Gamma(1) = \{b, c\}, \Gamma(2) = \{a\}, x_o = 0, X_m = \{1\}$. *O diagrama de transição de estados de* G *é apresentado na figura* 2.1.

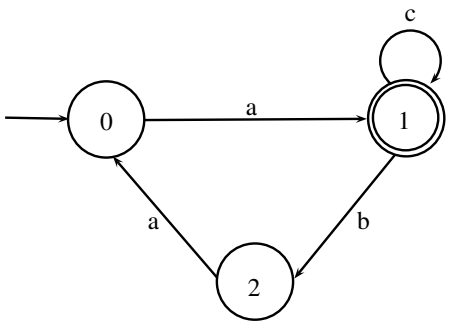


Figura 2.1: Diagrama de transição de estados do autômato G.

O funcionamento do autômato da figura 2.1 *se dá da forma descrita a seguir. Inicialmente, o sistema está no estado* 0*, cujo único evento viável ou ativo é o evento* a*, isto é,* $\Gamma(0) = \{a\}$. *Ocorrendo o evento* a*, o sistema muda de estado passando do estado* 0 *para o estado* 1 *que é o único estado marcado deste autômato, sendo que neste novo estado, os eventos* b *e* c *são viáveis, ou seja,* $\Gamma(1) = \{b, c\}$. *Se o evento* c *ocorrer, o sistema permanecerá no estado* 1 *ao passo que, ocorrendo o evento* b*, o sistema mudará do estado* 1 *para o estado* 2*, em que* $\Gamma(2) = \{a\}$. *No estado* 2*, se o evento* a *ocorrer, o sistema volta para o estado inicial 0.* □

O conjunto de todas as sequências que podem ocorrer em um autômato G, no diagrama de transição de estados, desde o estado inicial, constitui a linguagem gerada por G, denotada neste trabalho por $L(G)$ ou, simplesmente, L. Deste conjunto, é possível extrair um subconjunto formado por todas as sequências de L que terminam em um estado marcado. A esse subconjunto dá-se o nome de linguagem marcada, denotada por L_m.

As linguagens gerada L e marcada L_m pelo autômato G podem ser definidas como se segue:

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

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

Note que L é prefixo fechada, isto é, $L = \overline{L}$ e que $\varepsilon \in L$. Por outro lado, em geral, L_m não é prefixo fechada, e $\varepsilon \in L_m$, se e somente se, $x_0 \in X_m$.

Operações com autômatos

A parte acessível de um autômato G é a operação unária que elimina todos os estados de G que não são alcançáveis a partir do estado inicial x_0. Formalmente, a parte acessível de um autômato é dada pela definição 2.6 [67].

Definição 2.6. *(Parte acessível) Seja $G = (X, \Sigma, f, \Gamma, x_0, X_m)$. A parte acessível de G, denotada por $Ac(G)$, é o subautômato*

$$Ac(G) = (X_{ac}, \Sigma, f_{ac}, x_0, X_{ac,m}),$$

sendo $X_{ac} = \{x \in X : (\exists s \in \Sigma^*)[f(x_o, s) = x]\}$; $X_{ac,m} = X_m \cap X_{ac}$; $f_{ac} : X_{ac} \times \Sigma^* \to X_{ac}$ é a função de transição obtida após a restrição do domínio de f para o domínio dos estados acessíveis X_{ac}, isto é, os estados alcançáveis a partir de x_o. □

A parte coacessível de um autômato G é a operação unária que elimina todos os estados de G a partir dos quais é impossível alcançar os estados marcados. Formalmente, a parte coacessível de um autômato é dada pela definição 2.7 [67].

Definição 2.7. *(Parte coacessível) Seja* $G = (X, \Sigma, f, \Gamma, x_0, X_m)$*. A parte coacessível de* G*, denotada por* $CoAc(G)$*, é o subautômato*

$$CoAc(G) = (X_{coac}, \Sigma, f_{coac}, x_{0,coac}, X_m),$$

$X_{coac} = \{x \in X : (\exists s \in \Sigma^\star)[f(x,s) \in X_m]\}$; $x_{0,coac} = x_0$*, se* $x_0 \in X_{coac}$ *e* $X_{0,coac}$ *é indefinido, se* $x_0 \notin X_{coac}$; $f_{coac} : X_{coac} \times \Sigma^* \to X_{coac}$*, denota a nova função de transição obtida após a restrição do domínio de* f *para o domínio dos estados coacessíveis* X_{coac}*, isto é, os estados a partir dos quais se alcança um estado marcado.* □

Duas outras operações com autômatos são definidas em [67], são elas a operação de composição produto, denotada por $\times$, e a composição paralela, denotada por $||$. A composição paralela é também conhecida como composição síncrona e o produto como composição completamente síncrona. A operação de composição paralela entre dois autômatos $G_1 = (X_1, \Sigma_1, f_1, \Gamma_1, x_{01}, X_{m_1})$ e $G_2 = (X_2, \Sigma_2, f_2, \Gamma_2, x_{02}, X_{m_2})$, procura mapear o comportamento síncrono entre os dois autômatos, isto é, um evento σ, comum aos dois autômatos ($\sigma \in \Sigma_1 \cap \Sigma_2$), só poderá ser executado se ocorrer simultaneamente nos dois autômatos, ao passo que os eventos particulares, isto é, $\sigma \in (\Sigma_1 \backslash \Sigma_2) \cup (\Sigma_2 \backslash \Sigma_1)$ podem ser executados livremente sempre que forem possíveis. Na composição produto, apenas os eventos pertencentes a ambos os autômatos G_1 e G_2 podem promover transições de estados e são executados somente quando ocorrem simultaneamente nos autômatos. A composição paralela e produto dos autômatos G_1 e G_2 são definidas a seguir.

Definição 2.8. *(Composição paralela) Sejam os autômatos* $G_1 = (X_1, \Sigma_1, f_1, \Gamma_1, x_{01}, X_{m_1})$ *e* $G_2 = (X_2, \Sigma_2, f_2, \Gamma_2, x_{02}, X_{m_2})$*. Então, a composição paralela, denotada por* $G_1||G_2$*, é dada por:*

$$G_1||G_2 := Ac(X_1 \times X_2, \Sigma_1 \cup \Sigma_2, f_{1||2}, \Gamma_{1||2}, (x_{01}, x_{02}), X_{m_1} \times X_{m_2})$$

sendo

$$f((x_1, x_2), e) = \begin{cases} (f_1(x_1, e), f_2(x_2, e)), & \textit{se } e \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ (f_1(x_1, e), x_2), & \textit{se } e \in \Gamma_1(x_1) \setminus \Sigma_2 \\ (x_1, f_2(x_2, e)), & \textit{se } e \in \Gamma_2(x_2) \setminus \Sigma_1 \\ \textit{indefinida}, & \textit{caso contrário} \end{cases}$$

além disso,

$$\Gamma_{1||2}(x_1, x_2) = [\Gamma_1(x_1) \cap \Gamma_2(x_2)] \cup [\Gamma_1(x_1) \setminus \Sigma_2] \cup [\Gamma_2(x_2) \setminus \Sigma_1]$$

□

A seguir, é apresentada a definição da operação de composição produto entre os autômatos G_1 e G_2.

Definição 2.9. *(Composição produto) Sejam os autômatos* $G_1 = (X_1, \Sigma_1, f_1, \Gamma_1, x_{01}, X_{m1})$ *e* $G_2 = (X_2, \Sigma_2, f_2, \Gamma_2, x_{02}, X_{m2})$*. Então, a composição produto, denotada por* $G_1 \times G_2$*, é dada por:*

$$G_1 \times G_2 := Ac(X_1 \times X_2, \Sigma_1 \cup \Sigma_2, f_{1\times2}, \Gamma_{1\times2}, (x_{01}, x_{02}), X_{m1} \times X_{m2})$$

sendo

$$f((x_1, x_2), e) = \begin{cases} (f_1(x_1, e), f_2(x_2, e)), & \textit{se } e \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ \textit{indefinida}, & \textit{caso contrário} \end{cases}$$

e

$$\Gamma_{1\times2}(x_1, x_2) = \Gamma_1(x_1) \cap \Gamma_2(x_2).$$

□

Para caracterizar as linguagens gerada e marcada pelo autômato $G_1||G_2$, denotadas por $L(G_1||G_2)$ e $L_m(G_1||G_2)$, respectivamente, em função das linguagens de G_1 e G_2, é necessário definir as seguintes projeções:

$$P_i : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_i^*, \text{ para } i = 1, 2.$$

Usando as projeções P_i e considerando que L_1 e L_2 são, respectivamente, as linguagens geradas por G_1 e G_2, e que L_{m_1} e L_{m_2} são, respectivamente, as linguagens marcadas por G_1 e G_2, tem-se que:

$$L(G_1||G_2) = P_1^{-1}(L_1) \cap P_2^{-1}(L_2),$$

$$L_m(G_1||G_2) = P_1^{-1}(L_{m1}) \cap P_2^{-1}(L_{m2}).$$

No caso particular em que os conjuntos de eventos Σ_1 e Σ_2 são iguais, a composição paralela se reduz à composição produto. A linguagem gerada e marcada pelo autômato $G_1 \times G_2$ são dadas, respectivamente, por:

$$L(G_1 \times G_2) = L_1 \cap L_2.$$
$$L_m(G_1 \times G_2) = L_{m1} \cap L_{m2}.$$

2.1.3 Sistemas a eventos discretos com observação parcial de eventos

Em sistemas reais modelados como SEDs, nem sempre todos os eventos são observáveis, ou seja, nem sempre possuem sensores capazes de detectar e comunicar a sua ocorrência para um agente observador. Neste sentido, o conjunto dos eventos Σ pode ser particionado da seguinte forma, $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$, sendo Σ_o o conjunto de eventos observáveis e Σ_{uo}, o conjunto dos eventos não observáveis.

Para obter um autômato determinístico que gere e marque as linguagens observadas do autômato com observação parcial, é necessário introduzir o conceito de observador que, por sua vez, deverá preceder um outro conceito, o do alcance não observável.

Definição 2.10. *(Alcance não observável) O alcance não observável de um estado $x \in X$, denotado por $UR(x, \Sigma_o)$, é definido como:*

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

O alcance não observável pode ser definido também para um conjunto $B \in 2^X$ *da seguinte forma:*

$$UR(B, \Sigma_o) = \cup_{x \in B} UR(x, \Sigma_o).$$

□

Note que $UR(x, \Sigma_o)$ retorna todos os estados alcançáveis a partir de x através de transições rotuladas por eventos não observáveis. A seguir, será apresentada a definição de autômato observador com base na definição 2.10 [69].

Definição 2.11. *O observador de um autômato* G *com relação a um conjunto de eventos observáveis* Σ_o*, denotado por* $Obs(G, \Sigma_o)$*, é dado por*

$$Obs(G, \Sigma_o) = (X_{Obs}, \Sigma_o, f_{Obs}, \Gamma_{Obs}, x_{0,Obs}, X_{mObs}),$$

sendo $X_{Obs} \subseteq 2^X$ *e* $X_{mObs} = \{B \in X_{Obs} : B \cap X_m \neq \emptyset\}$. f_{Obs}, Γ_{Obs} *e* $x_{0,Obs}$ *são definidos de acordo com o algoritmo* 2.1 *de construção do observador.* □

Algoritmo 2.1. *Observador*

Entrada: *Autômato* G *e o conjunto de eventos observáveis* Σ_o

Saída: *Autômato observador* $Obs(G, \Sigma_o)$

- *Passo 1: Defina* $x_{0,Obs} = UR(x_0, \Sigma_o)$. *Faça* $X_{Obs} = \{x_{0,Obs}\}$ *e* $\tilde{X}_{Obs} = X_{Obs}$.
- *Passo 2:* $\overline{X}_{Obs} = \tilde{X}_{Obs}$ *e* $\tilde{X}_{Obs} = \emptyset$.
- *Passo 3: Para cada* $B \in \overline{X}_{Obs}$:

  Passo 3.1: $\Gamma_{Obs}(B) = (\cup_{x \in B} \Gamma(x)) \cap \Sigma_o$

  Passo 3.2: Para cada $\sigma \in \Gamma_{Obs}(B)$: $f_{Obs}(B, \sigma) = UR(\{x \in X : (\exists y \in B)[x = f(y, \sigma)]\})$.

  Passo 3.3: $\tilde{X}_{Obs} \leftarrow \tilde{X}_{Obs} \cup \{f_{Obs}(B, \sigma)\}$

- *Passo 4:* $X_{Obs} \leftarrow X_{Obs} \cup \tilde{X}_{Obs}$.
- *Passo 5: Repetir os passos 2 a 4 até obter a parte acessível de* $Obs(G, \Sigma_o)$.
- *Passo 6:* $X_{mObs} = \{B \in X_{Obs} : B \cap X_m \neq \emptyset\}$.

Exemplo 2.2. *Considere o autômato G cujo diagrama de transição de estados está representado na figura 2.2 (a). Suponha que* $\Sigma_o = \{a, b, c\}$ *e* $\Sigma_{uo} = \{\sigma_f\}$ *são os conjuntos de eventos observáveis e não observáveis, respectivamente, e que* $\Sigma_f = \{\sigma_f\}$ *é o conjunto de eventos de falha do sistema. Aplicando o passo 1 do algoritmo 2.1, obtém-se* $x_{o,obs} = \{0, 2\}$, $X_{obs} = \{0, 2\}$ *e* $\tilde{X}_{obs} = \emptyset$. *No passo 2,* $\overline{X}_{Obs} = \tilde{X}_{Obs} = \{0, 2\}$ *e* $\tilde{X}_{Obs} = \emptyset$. *No passo 3,* $B = \{0, 2\}$, $\Gamma_{Obs}(B) = (\Gamma(0) \cup \Gamma(2)) \cap \Sigma_o = \{a, b\}$, $f_{obs}(0, a) = f_{obs}(2, b) = \{1\}$ *e* $\tilde{X}_{Obs} = \{1\}$. *No passo 4* $X_{obs} = \{\{0, 2\}, \{1\}\}$. *Repetindo o passo 2, tem-se que* $\overline{X}_{Obs} = \tilde{X}_{Obs} = \{1\}$ *e* $\tilde{X}_{Obs} = \emptyset$. *Repetindo o passo 3,* $B = \{1\}$, $\Gamma_{Obs}(B) = \Gamma(1) \cap \Sigma_o = \{c\}$, $f_{obs}(1, c) = \{1\}$ *e* $\tilde{X}_{Obs} = \{1\}$. *Repetindo o passo 4* $X_{obs} = \{\{0, 2\}, \{1\}\}$. *No passo 6,* $X_{mobs} = \{1\}$. *Assim, o autômato observador* $Obs(G, \Sigma_o)$ *obtido através do algoritmo 2.1 é mostrado na figura 2.2 (b).*

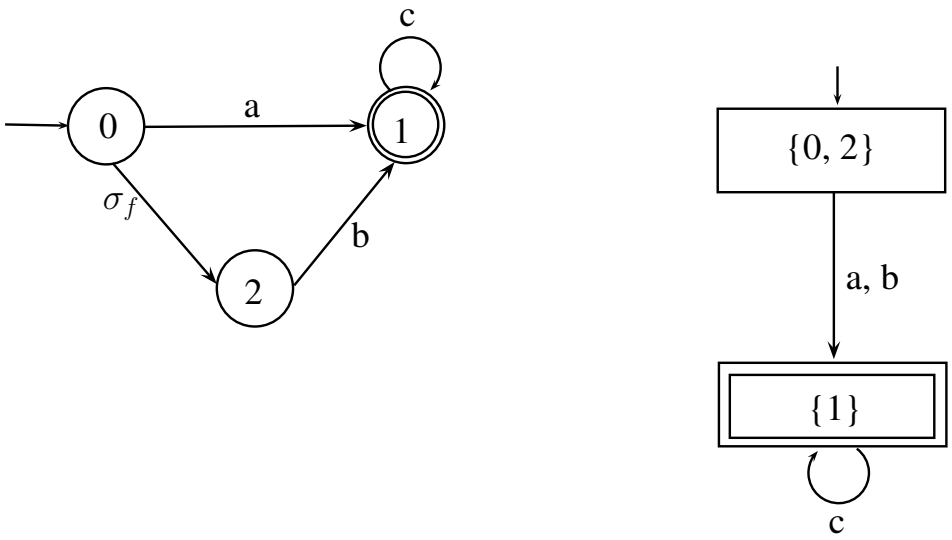


Figura 2.2: Autômato G (a), Autômato observador de G (b).

2.2 Diagnose centralizada de falhas em SEDs

De acordo com a seção 2.1, em SEDs com observação parcial de eventos, existem eventos cuja ocorrência não é registrada por nenhum dos sensores disponíveis no sistema. Dentre tais eventos, existem eventos de falha cuja ocorrência é indesejável e inesperada. Quando a ocorrência de um evento de falha pode ser observada por um sensor, a diagnose de falha se torna trivial. Entretanto, em muito casos, o evento de falha não pode ser observado e, por esta razão, a inferência da sua ocorrência deve ser feita com base na teoria de diagnose de falhas apresentada neste trabalho. Neste sentido, a ideia da diagnosticabilidade da linguagem de um SED está relacionada com a capacidade de se inferir a ocorrência do evento de falha, após um número finito de observações de eventos. Para tanto, usualmente, é feita a seguinte hipótese [57], [69]:

$\mathbf{H_1}$ - A linguagem gerada por L é viva, isto é, para todo $s \in L$, existe um evento $e \in \Sigma$, tal que $se \in L$.

Seja $\Sigma_f \subseteq \Sigma_{uo}$ o conjunto de eventos de falha e suponha, por simplicidade, que há um único evento de falha, isto é, $\Sigma_f = \{\sigma_f\}$[1]. Então, a seguinte definição pode ser feita.

Definição 2.12. *(Sequências de falha e normais) Uma sequência de falha é uma sequência de eventos s tal que σ_f é um dos eventos que formam s. Uma sequência normal, por outro lado, não contém o evento σ_f.* □

Seja G_N o subautômato de G que representa o comportamento normal (sem falha) do sistema com relação ao conjunto de eventos de falha Σ_f, e seja $L_N \subset L$ a linguagem gerada por G_N. Então, L_N é o conjunto formado por todas as sequências normais de L, e o conjunto formado por todas as sequências de falha é dado por $L \setminus L_N$.

A definição de diagnosticabilidade de linguagem de um sistema é dada a seguir [18].

[1]Se houver mais de um tipo de falha, então cada falha pode ser tratada separadamente, ou seja, para a verificação da diagnosticabilidade da linguagem do sistema com relação a um tipo específico de falha, todos os eventos de tipos diferentes não são considerados como eventos de falha.

Definição 2.13. *(Diagnosticabilidade de uma linguagem) Seja L a linguagem prefixo-fechada e viva gerada por um SED e seja $L_N \subset L$, a linguagem prefixo-fechada que representa o comportamento normal do sistema. Considere o conjunto de eventos Σ, particionado da seguinte forma, $\Sigma = \Sigma_o \cup \Sigma_{uo}$, sendo Σ_o e Σ_{uo} os conjuntos de eventos observáveis e não observáveis, respectivamente. Seja Σ_f o conjunto de eventos de falha tal que $\Sigma_f \subseteq \Sigma_{uo}$. Então, a linguagem L é diagnosticável com relação à projeção $P_o : \Sigma^* \to \Sigma_o^*$ e Σ_f, se e somente se,*

$$(\exists z \in \mathbb{N})(\forall s \in L \setminus L_N)(\forall st \in L \setminus L_N, ||t|| \geq z) \Rightarrow$$
$$[P_o(st) \neq P_o(\omega), \forall \omega \in L_N].$$

□

Exemplo 2.3. *Para ilustrar a definição 2.13, considere o autômato G cujo diagrama de transição de estados está representado na figura 2.2. Suponha que $\Sigma_o = \{a, b\}$ e $\Sigma_{uo} = \{\sigma_f, \sigma_u\}$ são os conjuntos de eventos observáveis e não observáveis, respectivamente, e que $\Sigma_f = \{\sigma_f\}$ é o conjunto de eventos de falha do sistema. Considere as sequências $s_N = ab\sigma_u^n$ e $s_Y = \sigma_f ab\sigma_u^n$, em que $n \in \mathbb{N}$, tais que $s_N \in L_N$ e $s_Y \in L \setminus L_N$. Aplicando a operação de projeção $P_o : \Sigma^* \to \Sigma_o^*$ sobre ambas as sequências obtém-se: $P_o(s_N) = ab$ e $P_o(s_Y) = ab$. Como $P_o(s_N) = P_o(s_Y)$, a condição de diagnosticabilidade da definição 2.13 é violada. Logo, a linguagem L gerada por G não é diagnosticável em relação a P_o e Σ_f.*

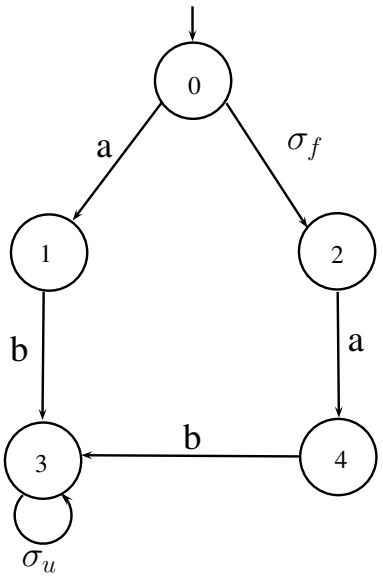


Figura 2.3: Autômato G referente ao exemplo 2.2.

2.2.1 Autômato diagnosticador

Para realizar a diagnose on-line de falhas em SEDs ou da verificação da diagnosticabilidade de uma linguagem, pode-se empregar um autômato determinístico chamado diagnosticador. O diagnosticador G_d tem o mesmo conjunto de eventos observáveis do autômato G e seus estados são rotulados com as letras Y(indicando que o evento de falha ocorreu) e N (indicando que o evento de falha não ocorreu). Formalmente, G_d é definido como

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

e o algoritmo que leva à construção do autômato G_d é apresentado a seguir.

Algoritmo 2.2. *Diagnosticador*

Entrada: *Autômato G e o conjunto de eventos observáveis Σ_o*

Saída: *Autômato diagnosticador $G_d = Obs(G, \Sigma_o)$*

- *Passo 1: Obtenha o autômato $G_l = G||A_l = (X_{G_l}, \Sigma, f_{G_l}, \Gamma_{G_l}, (x_0, N), \emptyset)$ em que A_l é o autômato rotulador: $A_l = (\{N, Y\}, \{\sigma_f\}, f_l, \Gamma_l, N, \emptyset)$ em que*

$f_l(N, \sigma_f) = f_l(Y, \sigma_f) = Y$ *e* $\Gamma_l(N) = \Gamma_l(Y) = \sigma_f$.

- *Passo 2: Calcule* $G_d = Obs(G_l, \Sigma_o)$.

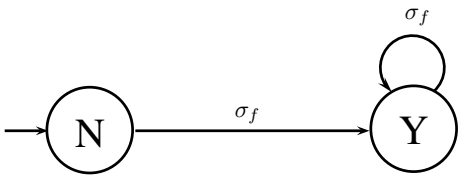


Figura 2.4: Autômato rotulador A_l.

Na figura 2.3 é apresentado o diagrama de transição de estados do autômato rotulador A_l. Note que pelo fato da composição paralela $G||A_l$ apenas rotular com as letras Y e N os estados de G, as linguagens dos autômatos G_l e G são iguais. Note ainda que os estados de G_l são da forma (x, Y) ou (x, N) ou, simplesmente, xY e xN, respectivamente.

Os estados do diagnosticador G_d podem ser classificados, em função dos rótulos N e Y, da seguinte forma [18].

Definição 2.14. *(Estados certos, normais e incertos) Um estado* $x_d \in X_d$ *é dito ser certo, se* $l = Y$ *para todo* $(x, l) \in x_d$*, é dito ser normal se* $l = N$ *para todo* $(x, l) \in x_d$ *e é dito ser incerto se existirem* (x, l), $(y, l^\star) \in x_d$ *, tais que* $l = Y$ *e* $l^\star = N$. □

As definições de caminho, caminho cíclico e ciclo são de relevância na solução dos problemas que são formulados e abordados nesta tese e são apresentadas a seguir.

Definição 2.15. *(Caminho) Em um autômato, um caminho* $(x_k, \sigma_1, x_{k+1}, \sigma_2, \ldots, \sigma_l, x_{k+l})$, $l > 0$, *é a sequência de estados e eventos tais que* $x_{k+i} = f(x_{k+i-1}, \sigma_i)$, $\forall i \in \{1, 2, 3, \ldots, l\}$. □

Definição 2.16. *(Caminho cíclico) um caminho* $(x_k, \sigma_1, x_{k+1}, \sigma_2, \ldots, \sigma_l, x_{k+l})$ *é dito ser cíclico se* $x_{k+l} = x_k$. □

Definição 2.17. *(Ciclo) Um conjunto de estados* $\{x_1, x_2, ..., x_n\} \subseteq X$ *forma um ciclo em um autômato* G *se existir uma sequência* $s = \sigma_1\sigma_2 \ldots \sigma_n$ *tal que* $(x_1, \sigma_1, x_2, \sigma_2, \ldots, x_n, \sigma_n, x_1)$ *forma um caminho cíclico em* G. □

A definição de ciclo indeterminado e ciclo escondido assim como a de ciclo escondido indeterminado é importante para o estudo da diagnosticabilidade de uma linguagem e são apresentadas a seguir.

Definição 2.18. *(Ciclo indeterminado) Um conjunto de estados incertos* $\{x_{d_1}, x_{d_2}, ..., x_{d_n}\} \subseteq X_d$ *forma um ciclo indeterminado se as seguintes condições forem satisfeitas:*

1. $\{x_{d_1}, x_{d_2}, ..., x_{d_p}\}$ *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* $\tilde{x_l}^{r_l}, l = 1, 2, ..., p$, $k_l = 1, 2, ..., m_l$, *e* $r_l = 1, 2, ..., \tilde{m}_l$ *tal que os estados* $\{\{x_l^{k_l}\}, l = 1, 2, ..., p\}, k_l = 1, 2, ..., m_l$ *e* $\{\tilde{x}_l^{r_l}\}$, $l = 1, 2, ..., p, r_l = 1, 2, ..., \tilde{m}_l$ *podem ser reagrupados para formar ciclos em* G. □

Definição 2.19. *(Ciclos escondidos) Suponha que* $x_d \in X_d$ *tenha sido obtido agrupando-se os estados* $x_{G_{l_1}}, x_{G_{l_2}}, \ldots, x_{G_{l_n}} \in X_{G_l}$. *Então, existe um ciclo escondido em* x_d *de* G_d *se para algum* $\{i_1, i_2, \ldots, i_k\} \subseteq \{1, 2, \ldots, n\}$, $\{x_{G_{l i_1}}, x_{G_{l i_2}}, \ldots, x_{G_{l i_k}}\}$ *forma um ciclo em* G_l. □

Definição 2.20. *(Ciclos escondidos indeterminados) Suponha que* $x_d \in X_d$ *seja um estado incerto e que os estados* $\{x_{G_{l i_1}}, x_{G_{l i_2}}, \ldots, x_{G_{l i_k}}\}$ *formam um ciclo escondido em* x_d. *Suponha ainda que os estados* $x_{G_{l i_1}}, \ldots, x_{G_{l i_k}}$ *são certos. Então, o ciclo escondido é dito ser indeterminado.* □

Nos diagramas de transição de estados dos diagnosticadores, os ciclos escondidos são representados por laços tracejados com as seguintes designações:(i) **hc** (do inglês *hidden cycle*) para os ciclos escondidos em G_d os quais não podem ser formados por estados certos de G_l. As designações com **hc** podem acontecer nos estados normais, certos, ou incertos de G_d; (ii) **ihc** (do inglês *indeterminate hidden cycle*) para ciclos escondidos indeterminados em G_d os quais devem ser formados por estados certos de G_l. Nesse caso, a designação com **ihc** só poderá ocorrer nos estados incertos de G_d.

Com base nas definições 2.13, 2.18, 2.20 e na hipótese $\mathbf{H_1}$, pode-se enunciar a seguinte condição necessária e suficiente para a diagnosticabilidade de uma linguagem [18].

Teorema 2.1. *Uma linguagem L, viva, gerada por um autômato G, é dita ser diagnosticável com relação à projeção P_o e Σ_f se, e somente se, seu diagnosticador G_d não contiver ciclos indeterminados inclusive os escondidos.*

Prova. Ver [18].

Observação 2.1. *A análise da diagnosticabilidade de uma linguagem por meio de diagnosticadores pode apresentar, no pior caso, complexidade computacional exponencial. A razão para tal se justifica pela possibilidade de crescimento exponencial do espaço de estados dos diagnosticadores em função da cardinalidade do espaço de estado do sistema original, uma vez que diagnosticadores são observadores de estado.*

Exemplo 2.4. *A figura 2.5 (a) mostra o diagrama de transição de estado de um autômato G, em que o conjunto de eventos observáveis é dado por $\Sigma_o = \{a, b, c\}$ enquanto, o conjunto dos eventos não observáveis é $\Sigma_{uo} = \{\sigma_f\}$, e o conjunto de eventos de falha é $\Sigma_f = \{\sigma_f\}$. Para a análise da diagnosticabilidade da linguagem L do autômato G, foram geradas as figuras 2.5 (b) e 2.5 (c) que apresentam os autômatos $G_l = G||A_l$ e G_d, respectivamente.*

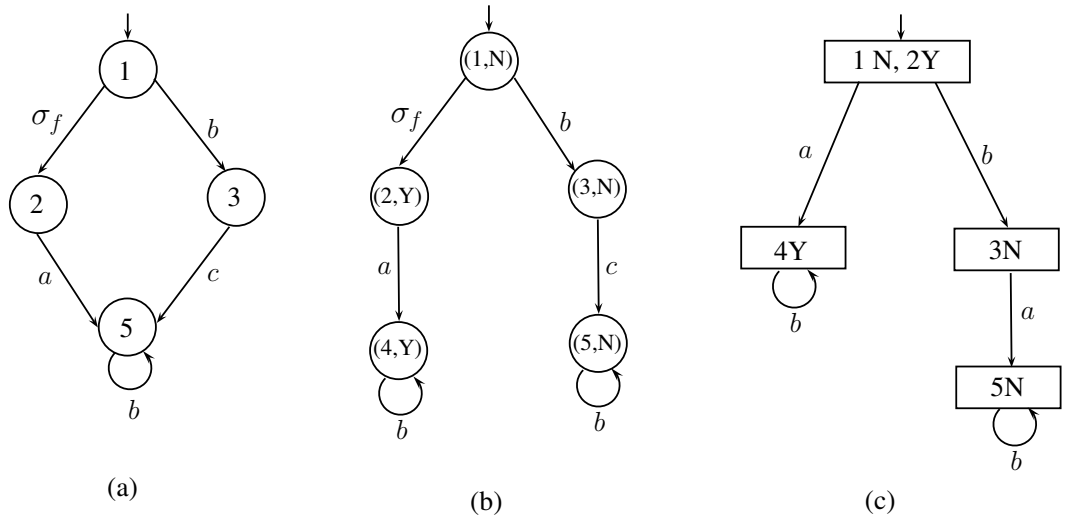


Figura 2.5: Autômato G (a), Autômato $G||A_l$ (b), Autômato G_d (c).

Como em G_d não existe nenhum ciclo formado por estados incertos (não há ciclo incerto) capaz de ser relacionado com dois ciclos distintos em G_l no qual um deles é formado por estados rotulados só com N e o outro só com Y, então a linguagem L é diagnosticável. □

Exemplo 2.5. *Seja o autômato G mostrado na figura 2.6 em que $\Sigma = \{a, b, c, d, e, g, \sigma_f\}$ e o conjunto de eventos de falha $\Sigma_f = \{\sigma_f\}$. Considere inicialmente que o conjunto de eventos observáveis seja $\Sigma_o = \{b, c, e\}$. A figura 2.7 (a) ilustra o diagnosticador obtido onde, é possível observar a existência de um estado incerto $\{7N, 11Y\}$. Como $f_d(\{7N, 11Y\}, b) = \{7N, 11Y\}$, então o estado incerto $\{7N, 11Y\}$ constitui um ciclo em G_d. Além disso, associados aos estados $\{7N\}$ e $\{11Y\}$ de $\{7N, 11Y\}$, existem dois ciclos em G_l formados pelos estados 7N e 11Y (ver figura 2.6). Logo, o estado $\{7N, 11Y\}$ de G_d possui um ciclo indeterminado e a linguagem L não é diagnosticável com relação a P_o e $\Sigma_f = \{\sigma_f\}$. Considere agora que o conjunto de eventos observáveis seja $\hat{\Sigma}_o = \{c, e\}$. Note, na figura 2.7(b), que o estado incerto $\{1N, 2N, 3N, 4N, 6N, 7N, 5Y, 9Y, 10Y, 11Y\}$ de $\hat{G}_d$ contém um estado certo 11Y de G_l. Além disso, como $f_{G_l}(11Y, b) = 11Y$, o que caracteriza um ciclo, então, o estado incerto $\{1N, 2N, 3N, 4N, 6N, 7N, 5Y, 9Y, 10Y, 11Y\}$ possui um ciclo escondido, o que, de acordo com o teorema 2.1 mostra que a linguagem L não é diagnosticável com relação a $P_o : \Sigma^\star \to \hat{\Sigma_o}^\star$ e $\Sigma_f = \{\sigma_f\}$.*

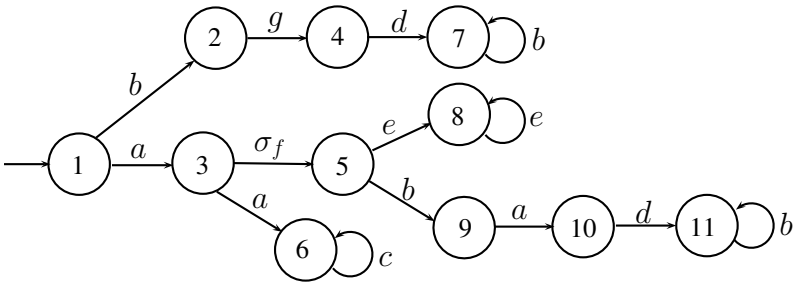


Figura 2.6: Autômato G.

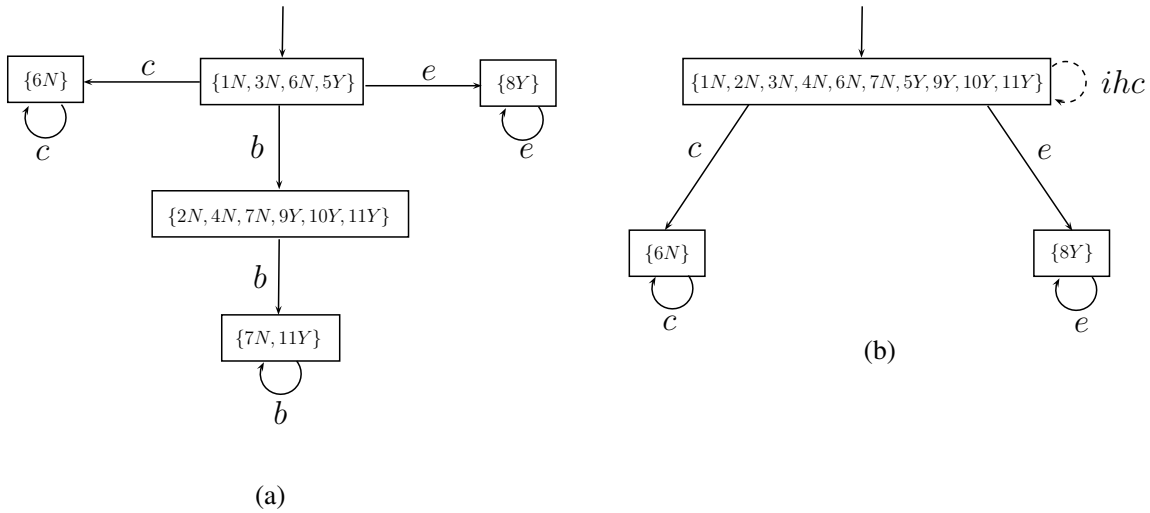


Figura 2.7: Diagnosticador G_d (a), Diagnosticador $\hat{G}_d$ (b).

Uma outra forma para verificar a diagnosticabilidade da linguagem de um sistema é utilizando autômatos verificadores [52] e [53]. A vantagem em se utilizar verificadores é que estes podem ser construídos em tempo polinomial. Na próxima seção, autômatos verificadores são apresentados para os casos de verificação da diagnosticabilidade descentralizada e centralizada.

2.3 Diagnose descentralizada de falhas

Alguns sistemas são fisicamente distribuídos e a comunicação de todos os eventos observáveis para um diagnosticador central não é apropriada ou é inviável. Nesses casos, diagnosticadores podem ser implementados em diferentes locais, e somente parte dos eventos observáveis são comunicados para cada diagnosticador local. Se não houver troca de informações entre os diagnosticadores locais, um coordenador recebe dos diagnosticadores informações relativas à ocorrência de evento de falha e o coordenador indica a ocorrência da falha de acordo com um protocolo específico. O esquema de diagnose descentralizada com diagnosticadores locais que não trocam informações entre si, é chamado na literatura de codiagnose [55], [56].

Em [22], diversos protocolos para a diagnose descentralizada são propostos. Em particular, no protocolo 3, cada diagnosticador local detecta a ocorrência do evento de falha baseado nas suas próprias observações, e o coordenador indica a ocorrência do evento de falha quando pelo menos um dos diagnosticadores locais identifica sua ocorrência. Essa noção de codiagnosticabilidade é chamada na literatura de codiagnosticabilidade disjuntiva [56]. Na figura 2.8, o esquema de codiagnose disjuntiva é apresentado supondo dois diagnosticadores locais G_{d_1} e G_{d_2}. Para apresentar a definição de codiagnosticabilidade disjuntiva de uma linguagem, considere que cada diagnosticador local observa um subconjunto do conjunto de eventos observáveis $\Sigma_{o_i} \subseteq \Sigma_o$, $i = 1, 2, \ldots, n$, em que n denota o número de diagnosticadores locais.

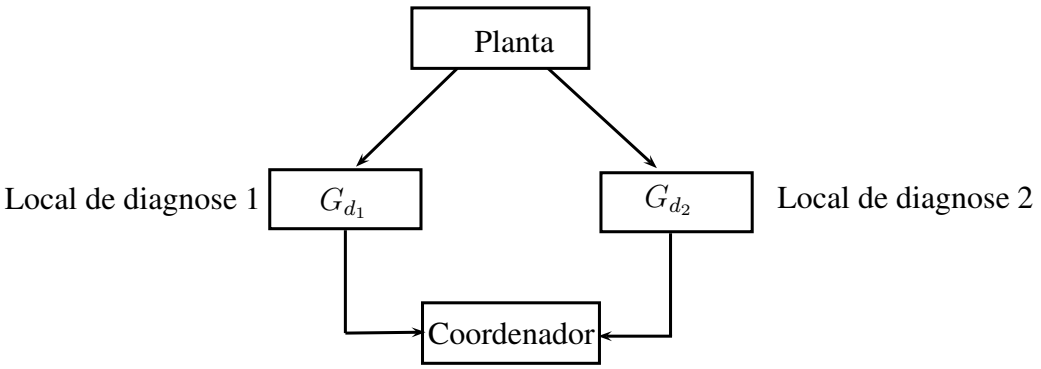


Figura 2.8: Esquema de codiagnose disjuntiva.

Definição 2.21. *(Codiagnosticabilidade de uma linguagem) Seja L a linguagem prefixo-fechada gerada por um sistema e seja $L_N \subset L$ a linguagem prefixo-fechada que representa o comportamento normal do sistema. Considere que existam n diagnosticadores locais com projeções $P_{o_i} : \Sigma^* \to \Sigma_{o_i}^* (i \in I_n = \{1, ..., n\})$, e seja $\Sigma_f = \{\sigma_f\}$ o conjunto de eventos de falha. Então, a linguagem L é codiagnosticável em relação a P_{o_i} e Σ_f, se e somente se*

$$(\exists z \in \mathbb{N})(\forall s \in L \setminus L_N)(\forall st \in L \setminus L_N, ||t|| \geq z) \Rightarrow$$
$$(\exists i \in I_n)[P_{o_i}(st) \neq P_{o_i}(\omega), \forall \omega \in L_N].$$

□

De acordo com a definição 2.21, L é dita ser codiagnosticável com relação a P_{o_i}, para $i = 1, 2, \ldots, n$ e Σ_f, se ao menos um dos diagnosticadores locais é capaz de identificar a ocorrência da falha baseada nas suas próprias observações, após um número limitado de ocorrência de eventos.

Naturalmente, a definição de diagnosticabilidade de L apresentada na definição 2.13, pode ser obtida a partir da definição 2.21 fazendo-se $n = 1$.

Para a verificação da codiagnosticabilidade da linguagem L, em [69] é construído um autômato diagnosticador. Neste trabalho, é usado no seu lugar, um segundo dispositivo chamado autômato verificador, pois este último apresenta uma complexidade computacional de ordem polinomial, melhor do que o anterior cuja complexidade é, no pior caso, exponencial.

2.3.1 Autômatos verificadores

Para contornar o problema de complexidade computacional dos diagnosticadores, conforme ressaltado na observação 2.1, foram desenvolvidos autômatos verificadores, com complexidade polinomial, para a análise da diagnosticabilidade de falhas em SEDs [53] , [56], [70]. Dentre esses, o algoritmo apresentado por MOREIRA et al. [70] possui menor complexidade computacional ao se considerar projeções para representar a observação dos diagnosticadores [71] e, por esta razão, será utilizado neste trabalho.

A seguir, o algoritmo de [70] será apresentado para a verificação da codiagnosticabilidade da linguagem L de um SED. Sem perda de generalidade, são considerados apenas dois diagnosticadores locais.

Algoritmo 2.3. *Verificação da codiagnosticabilidade.*

Entrada: G

Saída: *Decisão sobre a codiagnosticabilidade: Sim ou Não*

- *Passo 1: Construa o autômato G_N que modela o comportamento normal do sistema:*

  Passo 1.1: Defina $\Sigma_N = \Sigma \setminus \Sigma_f$.

  Passo 1.2: Construa o autômato A_N com apenas um estado N com um autolaço rotulado com os eventos de Σ_N.

  Passo 1.3: Construa o autômato $G_N = G \times A_N = (X_N, \Sigma, f_N, \Gamma_N, x_{0,N})$.

  Passo 1.4: Redefina o conjunto de eventos de G_N como Σ_N, isto é, $G_N = (X_N, \Sigma_N, f_N, \Gamma_N, x_{o,N})$.

- *Passo 2: Construa o autômato G_F para modelar o comportamento de falha do sistema:*

  Passo 2.1: Construa $G_l = G||A_l = (X_{G_l}, \Sigma, f_{G_l}, \Gamma_{G_l}, x_{0,G_l})$ e marque todos os estados de G_l que tem como segunda coordenada a letra Y.

  Passo 2.2: Construa o autômato de falha $G_F = CoAc(G_l) = (X_F, \Sigma, f_F, \Gamma_F, x_{0,F})$.

- *Passo 3: Defina a função de renomeação $R_i : \Sigma_N \to \Sigma_{R_i}$ como:*

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

  Em seguida, construa o autômato $G_{N,i} = (X_N, \Sigma_{R_i}, f_{N,i}, x_{0_N})$ 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 $G_V = G_{N,1}||G_{N,2}||G_F = (X_V, \Sigma_{R_1} \cup \Sigma_{R_2} \cup \Sigma, f_V, x_{0,V})$. Observe que cada estado de G_V é dado por $x_V = (x_{N,1}, x_{N,2}, x_F)$ em que $x_{N,1}, x_{N,2}$ e x_F são estados de $G_{N,1}, G_{N,2}$ e G_F, respectivamente, e $x_F = (x, x_l)$, sendo x e x_l estados de G e A_l, respectivamente.*

- *Passo 5: Procure por caminhos cíclicos* $cl = (x_V^k, \sigma_k, x_V^{k+1}, \sigma_{k+1}, \ldots, x_V^l, \sigma_l, x_V^k)$, *em que* $l \geq k \geq 0$, *tal que:*

$$\exists j \in \{k, k+1, \ldots, l\} \text{ tal que para algum } x_V^j, (x_l^j = Y) \wedge (\sigma_j \in \Sigma). \quad (2.1)$$

Caso exista um caminho cíclico que satisfaça a condição 2.1, então, L *não é codiagnosticável com relação a* P_{o_i}, $i = 1, 2$ *e* Σ_f. *Caso contrário,* L *é codiagnosticável.* □

De acordo com o algoritmo 2.3, é possível notar que a linguagem L_N de G_N contém todas as sequências de G livres do evento de falha. Por outro lado, G_F é o subautômato de G que modela o comportamento de falha do sistema, isto é, a linguagem gerada por G_F é $\overline{L \setminus L_N}$. Note ainda, que a função de renomeação R_i, apresentada no passo 3, efetua a alteração do rótulo dos eventos de $\Sigma_{uo_i} \setminus \Sigma_f$ de modo a garantir, durante a composição paralela no passo 4, que só os eventos observáveis sejam comuns a G_N e G_F. Assim, a evolução de estados em G_V com eventos observáveis garantirá a existência de sequências de falha em G_F que irão se confundir com sequências normais em G_N.

O teorema 2.2, a seguir, enuncia uma condição necessária e suficiente para a verificação da codiagnosticabilidade de uma linguagem L usando verificadores.

Teorema 2.2. *Sejam* L *e* L_N ($L_N \subset L$) *linguagens prefixo-fechadas geradas por* G *e* G_N, *respectivamente. Considere as projeções* $P_{o_i} : \Sigma^\star \to \Sigma_{o_i}^\star$, *para* $i = 1, 2, \ldots, n$ *e o conjunto de eventos de falha* Σ_f. *Então,* L *é codiagnosticável com relação a* P_{o_i} *e* Σ_f *se, e somente se, não existe um caminho cíclico em* G_V, $cl := (x_V^k, \sigma_k, x_V^{k+1}, \sigma_{k+1}, ..., x_V^l, \sigma_l, x_V^k)$ *em que* $l \geq k \geq 0$, *satisfazendo a seguinte condição:*

$$\exists j \in \{k, k+1, \ldots, l\} \text{ tal que para algum } x_V^j, (x_l^j = Y) \wedge (\sigma_j \in \Sigma). \quad (2.2)$$

□

Demonstração. Ver [72]. □

A seguir, é dado um exemplo que ilustra o uso de um verificador na análise da codiagnosticabilidade de uma linguagem.

Exemplo 2.6. *Considere o diagrama de transição de estados do autômato G da figura 2.5 do exemplo 2.4, em que $\Sigma = \{a, b, c, \sigma_f\}$, $\Sigma_{o_1} = \{a, b\}$, $\Sigma_{o_2} = \{a, c\}$ e $\Sigma_{uo} = \{\sigma_f\}$. Utilizando o algoritmo 2.3 obtém-se o verificador G_V apresentado na figura 2.9 (e). Como G_V não possui caminhos cíclicos que violam a condição para a codiagnosticabilidade, então a linguagem L é codiagnosticável com relação P_{o_i} e Σ_f.* □

Quando se tem apenas um agente local, $n = 1$, o passo 3 do algoritmo 2.3 pode ser modificado da seguinte forma:

Passo 3) Defina a função de renomeação $R : \Sigma_N \to \Sigma_R$ como:

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

Em seguida, construa o autômato $G_R = (X_N, \Sigma_R, f_R, x_{0_N})$ com $f_R(x_N, R(\sigma)) = f_N(x_N, \sigma)$ para todo $\sigma \in \Sigma_N$.

Nesse caso, o seguinte teorema pode ser enunciado para o caso quando se tem apenas um único agente local.

Teorema 2.3. *Seja $V = G_R \| G_F$ o verificador de G calculado considerando Σ_o como conjunto de eventos observáveis. Então, o estado (x_N, x_F) de V é alcançado se, e somente se, existe uma sequência $s_N \in L_N$ e uma sequência $s_Y \in \overline{L \setminus L_N}$ tais que $P_o(s_Y) = P_o(s_N)$, e x_N e x_F são os estados de G_R e G_F, alcançados após a ocorrência de $s_{N_R} = R(s_N)$ e s_Y, respectivamente.*

Demonstração. Ver [72].

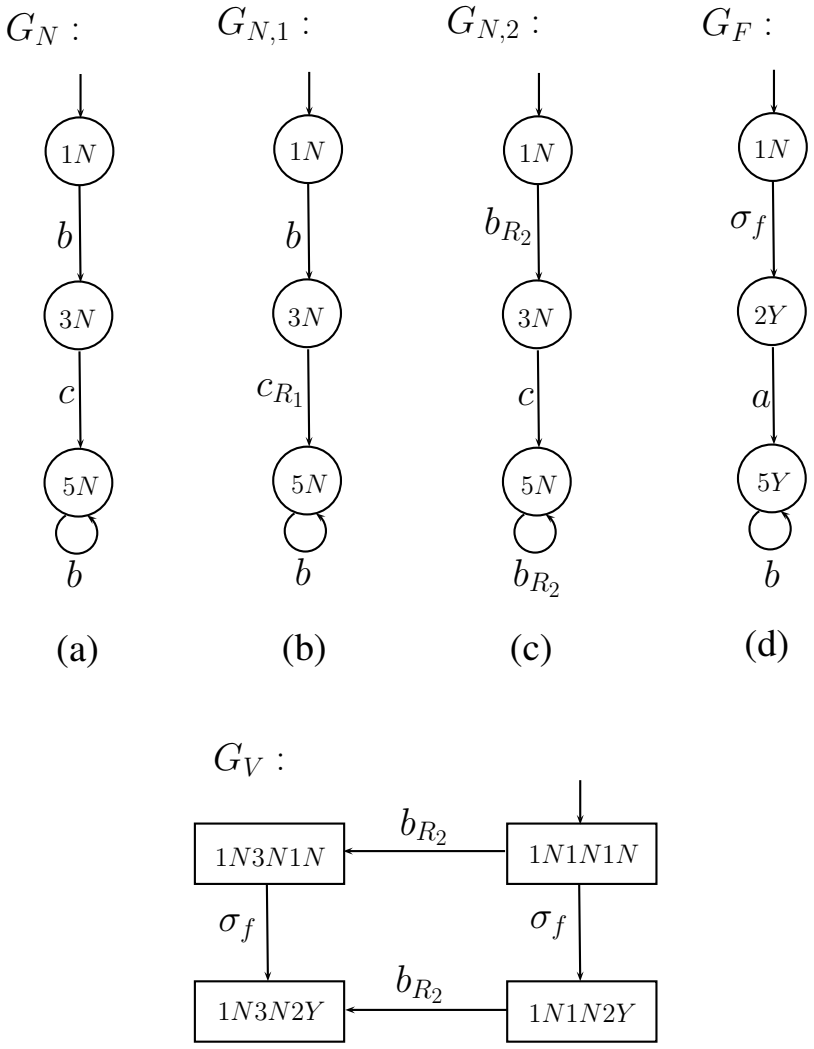


Figura 2.9: Autômato G_N(a), Autômato $G_{N,1}$(b), Autômato $G_{N,2}$(c), Autômato de falha G_F(d), Autômato verificador G_V(e).

O exemplo 2.7 a seguir, ilustra o caso em que se tem uma arquitetura centralizada.

Exemplo 2.7. *Considere, novamente, o diagrama de transição de estado do autômato G da figura 2.4 (a), em que $\Sigma = \{a, b, c, \sigma_f\}$ e $\Sigma_f = \{\sigma_f\}$. Considere ainda que exista um único diagnosticador com $\Sigma_o = \{a, b\}$. Utilizando o algoritmo 2.3 obtém-se o verificador G_V apresentado na figura 2.10.*

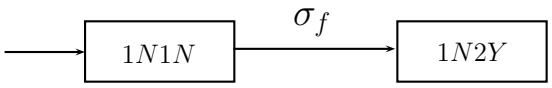


Figura 2.10: Autômato verificador centralizado, G_V.

Como G_V não possui caminhos cíclicos que violam a condição para a diagnosticabilidade, então a linguagem L é diagnosticável em relação a P_o e Σ_f. □

Note que as condições necessárias e suficientes para a verificação da diagnosticabilidade da linguagem L utilizando diagnosticadores e verificadores requerem a busca por caminhos cíclicos com determinadas características. Para tanto, algoritmos de busca em grafos devem ser utilizados. Na seção a seguir esses algoritmos de busca são apresentados.

2.4 Algoritmos de busca em grafos

Algoritmos de busca são procedimentos computacionais usados para gerar um valor, ou um conjunto de valores de saída, a partir de um valor ou de um conjunto de valores de entrada. As buscas podem ser realizadas em estruturas do tipo árvore ou grafo, sendo esta última a de interesse neste trabalho. A figura 2.11 ilustra as duas estruturas [73].

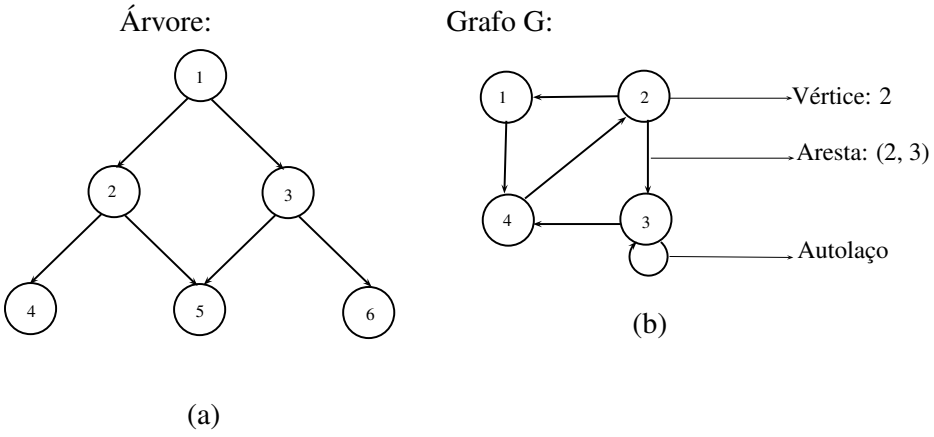


Figura 2.11: Árvore (a), Grafo orientado G (b).

A representação do grafo pode se feita de duas formas [73]: (i) por lista de adjacências; ou (ii) por matriz de adjacências conforme ilustrado na figura 2.12. A lista de adjacências é usualmente preferida em relação à matriz de adjacências quando o grafo é esparso, isto é, quando a cardinalidade do conjunto de arestas $|E|$ é muito menor que $|V|^2$, em que $|V|$ denota a cardinalidade do conjunto de vértices. A lista de adjacência de um grafo $G = (V, E)$ consiste em um vetor adj de |V| listas, sendo uma para cada vértice em V. Para cada $u \in V$, a lista de adjacência $adj[u]$ contém (ponteiros para) todos os vértices v tal que exista uma arresta $(u, v) \in E$, isto é, $Adj[u]$ consiste de todos os vértices adjacentes a u em G. Os vértices em cada lista de adjacência são armazenadas em ordem arbitrária. A preferência pela matriz de adjacências ocorre nos casos em que os grafos são densos, isto é, quando $|E|$ é muito próximo de $|V|^2$ ou quando se deseja saber rapidamente se existe uma aresta ligando dois vértices específicos. Para a obtenção da matriz de adjacências, considera-se que os vértices são numerados $1, 2, \ldots, |V|$ de forma arbitrária. A representação por matriz de adjacência de um grafo G consiste de uma matriz $A = (a_{ij})$ de dimensão $|V| \times |V|$, tal que

$$a_{ij} = \begin{cases} 1, & \text{se } (i, j) \in E \\ 0, & \text{caso contrário,} \end{cases}$$

Neste trabalho, será considerada a representação de autômatos por lista de adja-

cências uma vez que na maioria das vezes os grafos que representam sistemas reais são esparsos.

O grafo $G = (V, E)$ é dito ser orientado quando a aresta (u, v) sai do vértice u e entra no vértice v e neste caso, o vértice v é adjacente ao vértice u. Quando uma aresta nasce e morre num mesmo vértice, esta aresta é denominada de autolaço. A figura 2.11 ilustra esses casos.

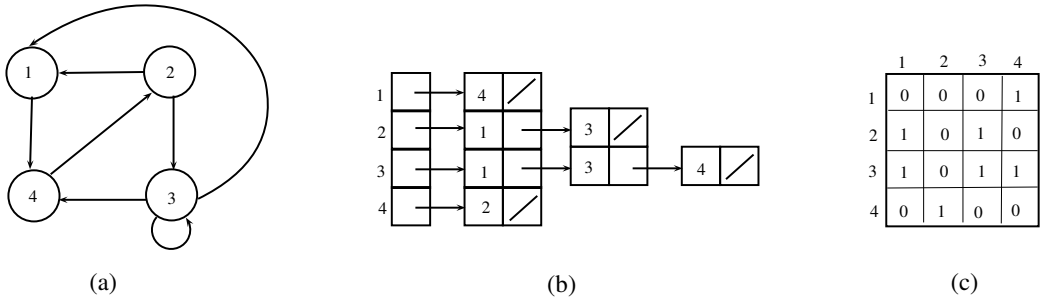


Figura 2.12: Grafo orientado G (a), Lista de adjacências de G (b), Matriz de adjacências de G (c).

Quando em um caminho $(v_0, v_1, ..., v_n)$, passa-se do vértice v_0 ao vértice v_1 através da aresta (v_0, v_1), diz-se que a aresta foi explorada ou visitada e que o vértice v_1 foi descoberto.

A busca em um grafo $G = (V, E)$, em que V e E são os conjuntos de vértices e de arestas do grafo, respectivamente, consiste em percorrer, a partir de um vértice inicial, as arestas do grafo para visitar os demais vértices até encontrar ou não, a informação desejada. Nesse sentido, vários algoritmos podem ser implementados com propósitos e objetivos específicos. Neste trabalho, os seguintes algoritmos são apresentados [73]: (i) Busca em profundidade. (ii) busca por componentes fortemente conexos. (iii) ordenação topológica e; (iv) algoritmo de busca pelo caminho mais longo em um grafo.

2.4.1 Busca em profundidade

A busca em profundidade, do inglês, $Depth - First\ Search(DFS)$, constitui a base para muitos outros algoritmos como o de ordenação topológica e de busca por componentes fortemente conexos. Para melhor acompanhar a evolução dos passos do algoritmo, os vértices são coloridos com as cores branco, cinza ou preto nos seguintes casos: ao iniciar o processo de busca, todos os vértices devem ser coloridos de branco. Quando um vértice é visitado pela primeira vez, ele é colorido de cinza e depois que toda sua lista de adjacência é explorada, ele é colorido de preto. Em [73], $\pi[u]$ é o vetor que armazena o vértice antecessor do vértice u. Quando o vértice u não possui um antecessor ou não for descoberto ainda, o vetor $\pi[u]$ recebe o valor NIL. O tempo em que o vértice u é descoberto é denotado de $d[u]$ enquanto que o tempo $f[u]$, denota o tempo em que a busca na lista de adjacência de u é finalizada e nesse tempo o vértice u é colorido de preto. Para todo vértice $u \in V$, $d[u] < f[u]$. As cores são armazenadas no vetor $cor[u]$. Em [73], [74] o algoritmo de busca em profundidade (DFS) é apresentado como mostrado a seguir.

Algoritmo 2.4. *Algoritmo de busca em profundidade - DFS*

DFS(G)

- *Passo 1: Para cada vértice* $u \in V[G]$

  $cor[u] \leftarrow branco$

  $\pi[u] \leftarrow NIL$

  $time \leftarrow 0$

- *Passo 2: Para cada vértice* $u \in V[G]$

  se cor[u] = branco

  então DFS-Visit(u)

DFS-Visit(u)

- *Passo 3: cor[u] ← cinza* ▷ *vértice visitado*
- *Passo 4: d[u] ← time ← time + 1*
- *Passo 5: Para cada v ∈ Adj[u]* ▷ *Explora a aresta (u, v)*

  se cor[v] = branca

  então $\pi[v] \leftarrow u$

  DFS-Visit(v)
- *Passo 6: cor[u] ← preto* ▷ *Colorindo o vértice u de preto; fim*
- *Passo 7: f[u] ← time ← time + 1*

A análise da complexidade computacional do algoritmo DFS é feita da seguinte forma. Após a inicialização, cada vértice é colorido de cinza ou de preto apenas uma única vez. Portanto, as operações de colorir um vértice de cinza e preto são feitas em $O(1)$ assim, o tempo total para colorir todos os vértice do grafo $G = (V, E)$ de preto e cinza é $O(|V|)$. A lista de adjacências de cada vértice é totalmente varrida, apenas uma única vez, quando o vértice é colorido de preto. Como a soma dos comprimentos de todas as listas de adjacências é $O(|E|)$ então, o tempo necessário para varrer todas as listas de adjacências será $O(|E|)$. Portanto, a complexidade computacional do algoritmo de busca em profundidade (DFS) é $O(|V| + |E|)$.

Exemplo 2.8. *No grafo da figura 2.13, será aplicado o algoritmo de busca em profundidade (DFS) e as etapas da evolução desse algoritmo são mostradas na figura 2.14.*

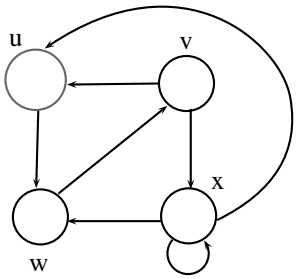


Figura 2.13: Grafo do exemplo 2.8.

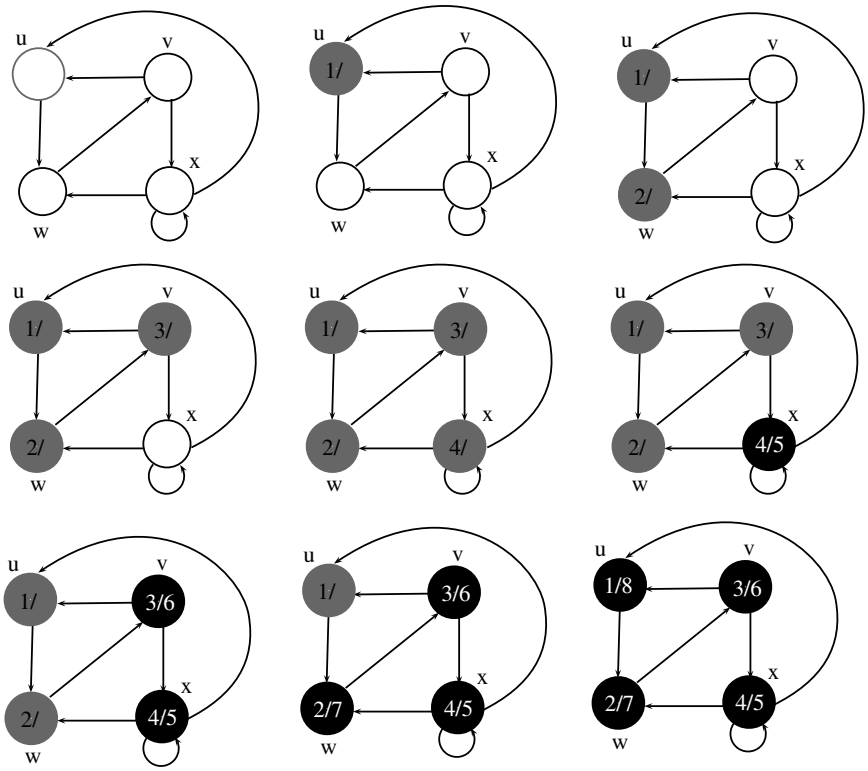


Figura 2.14: Evolução do algoritmo de busca em profundidade DFS em um grafo orientado.

No início do processo, todos os vértices são coloridos de branco, e quando o vértice u é visitado pela primeira vez, ele é colorido de cinza em um tempo $d[u] = 1$. No tempo de descoberta $d[w] = 2$, o vértice w, ao ser visitado pela primeira vez é colorido de cinza e na sequência o vértice v, em $d[v] = 3$. Em $d[x] = 4$, o

vértice x é visitado pela primeira vez recebendo a coloração cinza e como seu único vértice adjacente, w, já tinha sido visitado, então, em $f[x] = 5$, ele é colorido de preto. Evoluindo no caminho inverso, constata-se que todos os vértices adjacentes ao vértice v já foram visitados e portanto, $f[v] = 6$, e com isso o vértice v é colorido de preto. Na sequência, como todos os vértices adjacentes a w e u já foram visitados, então eles são coloridos de preto em $f[w] = 7$ e $f[u] = 8$, respectivamente.

2.4.2 Ordenação topológica

O algoritmo de ordenação topológica consiste na ordenação linear dos vértices de um grafo acíclico $G = (V, E)$ tal que se $(u, v) \in E$, então u vem antes de v na ordenação. Em outras palavras, a ordenação topológica pode ser entendida como um arranjo dos vértices na horizontal, formando uma lista encadeada, de modo que as arestas são orientadas da esquerda para a direita. Para grafos cíclicos, não é possível fazer uma ordenação linear dos vértices.

Algoritmo 2.5. *Algoritmo de ordenação topológica - OT.*

Entrada: $G = (V, E)$

Saída: *Lista encadeada dos vértices de G.*

- *Passo 1: Execute o algoritmo DFS (G) para calcular o tempo $f[u]$ de cada vértice $u \in V$.*
- *Passo 2: A medida que o tempo $f[u]$ vai sendo calculado, para cada vértice $u \in V$, insira o vértice no início de uma lista encadeada.*

Note que o resultado do algoritmo 2.5 é uma lista encadeada dos vértices de V, organizados em ordem decrescente de tempo de finalização.

A ordenação topológica pode ser realizada em tempo $O(|V| + |E|)$ visto que o algoritmo DFS(G) é executado em $O(|V| + |E|)$ e em $O(1)$ cada vértice é inserido na lista encadeada.

Exemplo 2.9. *Como o algoritmo de ordenação topológica não pode ser aplicado em grafos que contêm caminhos cíclicos, o arco* (w, v) *do grafo da figura 2.13 será invertido, e o autolaço no vértice* x *removido, de modo a tornar o grafo acíclico, conforme mostrado na figura 2.15.*

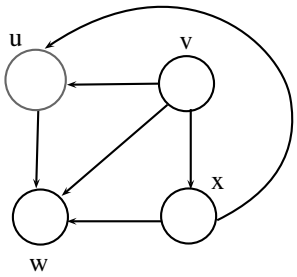


Figura 2.15: Grafo do exemplo 2.9.

Após aplicar o algoritmo de busca em profundidade (DFS) no grafo da figura 2.15 para encontrar os tempos de descoberta $d[u]$ *e de finalização* $f[u]$ *de cada vértice, obteve-se a figura 2.16.*

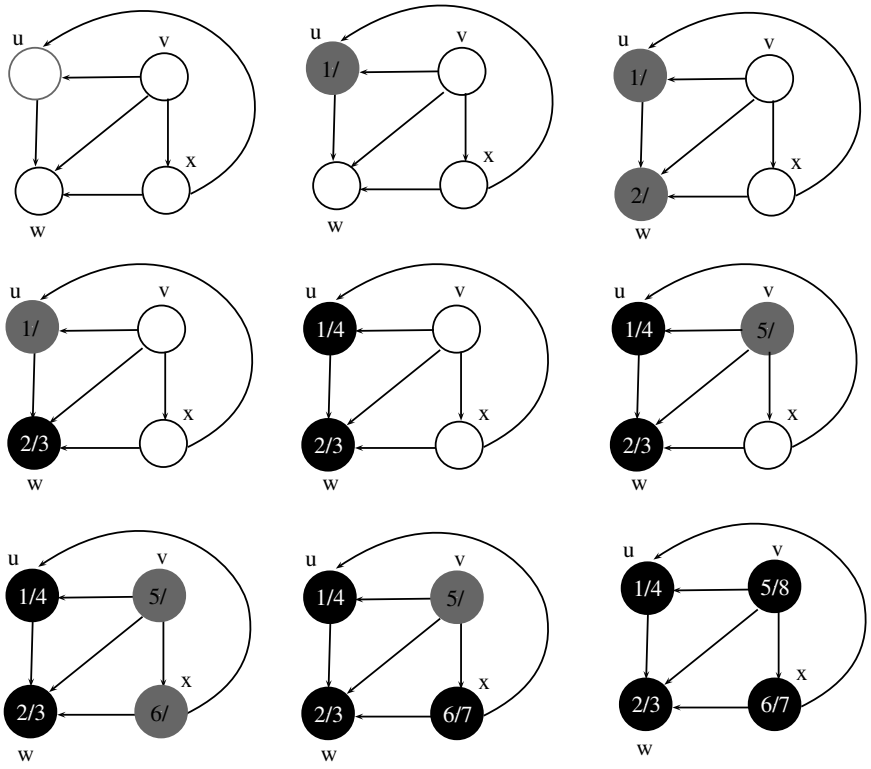


Figura 2.16: Tempos de descoberta $d[u]$ e de finalização $f[u]$ ($d[u]/f[u]$) de cada vértice do grafo G.

Após executar o algoritmo DFS no grafo da figura 2.15, é feita uma ordenação linear dos vértices do grafo G em ordem decrescente de $f[u]$. Segundo [73], a ligação dos vértices por arestas só pode ser feita da esquerda para a direita, isto é, dos vértices com maiores tempos de finalização $f[u]$ para os de menor $f[u]$, seguindo as orientações das arestas do grafo da figura 2.15. O resultado final dessa ordenação topológica pode ser visto na figura 2.17.

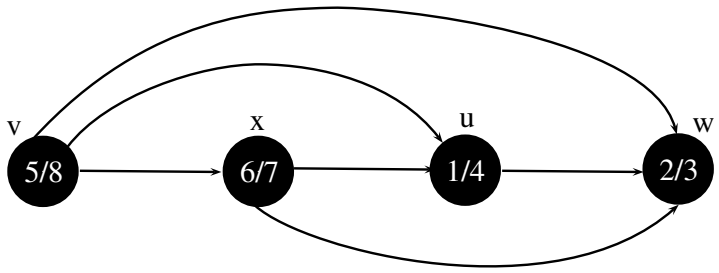


Figura 2.17: Ordenação topológica dos vértices do grafo G da figura 2.15.

Como dito anteriormente, o algoritmo de ordenação topológica não pode ser aplicado a um grafo que tem caminhos cíclicos. Uma das formas mais fáceis para encontrar os caminhos cíclicos existentes em um grafo, é procurar por componentes fortemente conexos no grafo. O algoritmo para essa procura é apresentado na subseção 2.4.3.

2.4.3 Componentes fortementes conexos - CFC

Um grafo orientado $G = (V, E)$ é dito ser fortemente conexo se para qualquer par de vértices (u, v) de G, o vértice v pode ser alcançado partindo do vértice u, e vice-versa. Os componentes fortementes conexos de um grafo orientado G são subconjuntos maximais de vértices de V que são mutuamente alcançáveis. Assim, um grafo orientado fortemente conexo tem apenas um componente. A figura 2.18 ilustra esses dois casos, nos quais os componentes fortemente conexos dos grafos G_1 e G_2 são delimitados por linhas tracejadas.

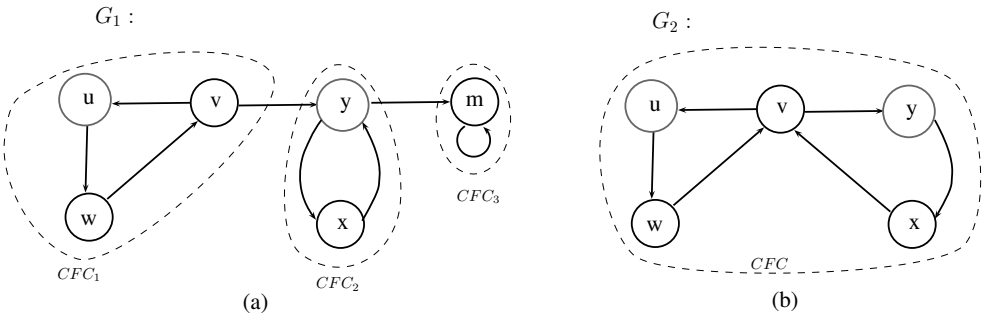


Figura 2.18: Grafo G_1 com três componentes fortemente conexos (a), Grafo fortemente conexo G_2 com apenas um único componente fortemente conexo (b).

Observe que o par (y, m) do grafo G_1 da figura 2.18 não constitui um componente fortemente conexo pois, não é possível alcançar o vértice y a partir do vértice m.

Antes de apresentar o algoritmo de busca por componentes fortemente conexos, alguns conceitos sobre grafos transpostos e árvores serão relembrados a seguir [73–75].

- **Grafo transposto**

  Seja $G = (V, E)$ um grafo orientado. O grafo orientado transposto de G é o grafo $G^T = (V^T, E^T)$ tal que $V = V^T$ e $E^T = \{(u, v) : (v, u) \in E\}$, ou seja, o grafo G^T é obtido invertendo-se o sentido das arestas do grafo G.

- **Árvore**

  A árvore é uma estrutura de dados ideal para representar informações dispostas de forma hierárquica. Os elementos de uma árvore são denominados de nós ou vértices, com uma hierarquia entre eles. O vértice principal de uma árvore é chamado raiz ou pai, pois dele nascem ligações para outros vértices chamados de ramos ou filhos. Destes, também, nascem ligações que levam a outros vértices, e assim sucessivamente. Todos os vértices gerados a partir de outros vértices são classificados como descendentes. Quando um vértice descente não tiver filho, ele é chamado de folha conforme mostra a figura 2.19.

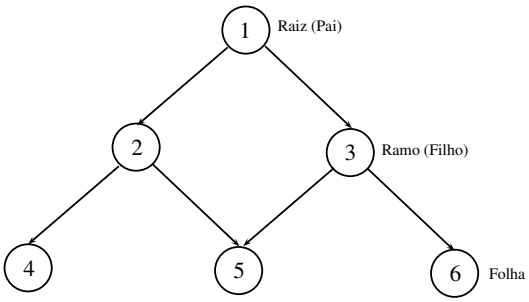


Figura 2.19: Estrutura de dados em árvore.

O vértice 1 é o pai dos vértices 2 e 3, e ancestral de todos eles. Os vértices 2 e 3 são filhos do vértice 1 e pais dos vértices $\{4, 5\}$ e $\{5, 6\}$, respectivamente. Os vértices 4, 5 e 6, por não possuírem descendentes (filhos) são chamados de folhas.

Um grafo acíclico não conexo é chamado de floresta.

Algoritmo 2.6. *Algoritmo de busca dos componentes fortementes conexos - CFC.*

Entrada: $G = (V, E)$

Saída: *Componentes fortemente conexos de* G.

- *Passo 1: Execute o algoritmo DFS (G) para calcular o tempo final* $f[u]$ *de cada vértice* $u \in V$.
- *Passo 2: Obtenha o grafo* G^T.
- *Passo 3: Execute o algoritmo* $DFS(G^T)$*, obedecendo uma ordem decrescente de tempo de finalização* $f[u]$ *obtido no passo 1.*
- *Passo 4: Retorne os vértices de cada árvore da floresta obtida no passo 3 com um componente fortemente conexo separado.*

Exemplo 2.10. *Para o grafo da figura 2.20, deseja-se encontrar seus componentes fortemente conexos e para tal, será aplicado o algoritmo 2.6.*

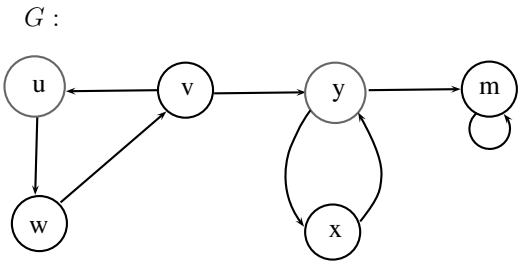


Figura 2.20: Grafo do exemplo 2.9.

No primeiro passo é feita a busca em profundidade de G e os tempos de descoberta e finalizações da exploração dos vértices são mostrados na figura 2.21.

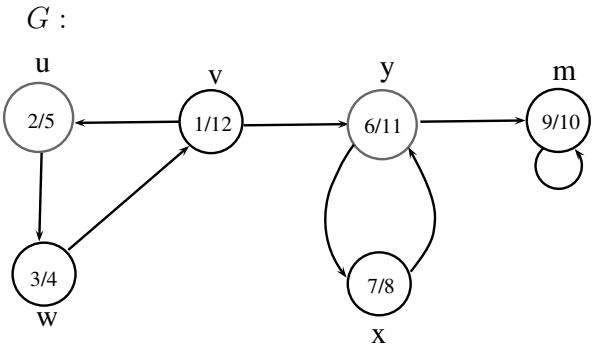


Figura 2.21: Os tempos finais $f[u]$ de cada vértice do grafo G.

No segundo passo é gerado o autômato G^T invertendo o sentido das arestas do grafo G, conforme ilustra a figura 2.20.

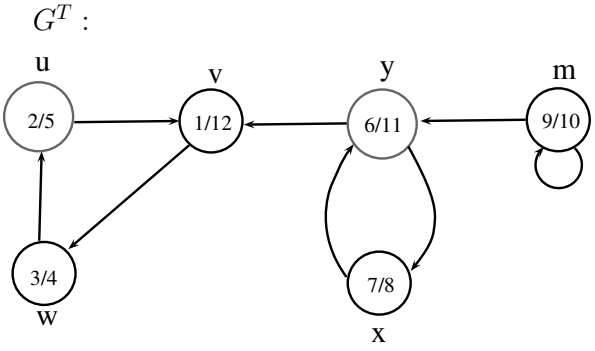


Figura 2.22: Grafo transposto de G.

No terceiro passo, são encontrados as árvores que constituem, cada uma, um componente fortemente conexo de G, ou seja, árvores cujos vértices formam um caminho cíclico. A floresta obtida é mostrada na figura 2.23.

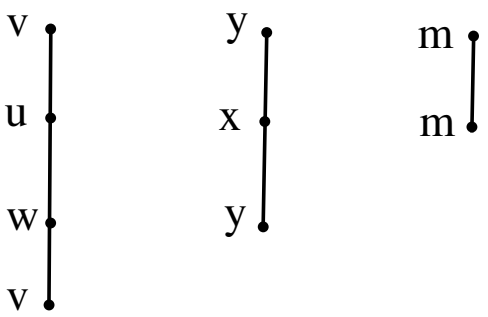


Figura 2.23: Árvores dos componentes fortemente conexos de G.

2.5 Complexidade computacional

A eficiência de um algoritmo é função da ordem de crescimento do seu tempo de execução. Tempo, aqui, refere-se à quantidade de rotinas a ser executada pelo algoritmo até a conclusão de uma determinada tarefa. A eficiência de um algoritmo permite estabelecer comparações entre diferentes algoritmos empregados para a resolução de determinado problema [73]. O desempenho de um algoritmo em relação a seu tempo de execução depende muito do tamanho do problema a ser solucionado, isto é, do tamanho da sua entrada. Nesse sentido, para entradas suficientemente grandes,

tornando a ordem de crescimento do tempo de execução relevante, leva ao estudo da eficiência assintótica de algoritmos [73]. Assim, um algoritmo assintoticamente mais eficiente é melhor para todas as entradas exceto para entradas relativamente pequenas [73]. O estudo de complexidade de um algoritmo pode ser de melhor caso, isto é, as rotinas são executadas no menor tempo possível para qualquer entrada de tamanho $n \in \mathbb{N}$. Quando a complexidade é de pior caso, as rotinas são executadas no maior tempo possível para qualquer entrada de tamanho $n \in \mathbb{N}$ e este, é o caso a ser estudado neste trabalho. Na complexidade de caso médio, as rotinas são executadas no tempo médio e nesse caso, se considera uma distribuição de probabilidade em relação ao conjunto de entradas.

A seguir, são apresentados as diferentes notações assintóticas para estudo da complexidade computacional de um algoritmo.

2.5.1 Notação assintótica

No estudo da complexidade computacional de um algoritmo, com relação à análise assintótica de funções, três notações principais são adotadas, a saber, a notação Θ, a notação O e a notação Ω [73]. Neste trabalho, será adotada a notação O.

- Notação Θ

  Para definir a notação Θ, considere a função $g(n)$. Denota-se por $\Theta(g(n))$ o conjunto de funções $f(n)$ tal que [73]

  $$(\exists c_1, c_2 \in \mathbb{R}_+)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)[0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)]$$

  A função $f(n)$ pertence ao conjunto $\Theta(g(n))$ se existirem as constantes positivas c_1, c_2 e n_0 tais que $f(n)$ possa estar entre $c_1 g(n)$ e $c_2 g(n)$, para $n \geq n_0$. Em outras palavras, a notação Θ limita assintoticamente uma função por valores superiores e inferiores a $f(n)$. As funções $f(n)$ e $g(n)$ devem ser não negativas [73]. Por exemplo, para mostrar que $\frac{1}{2}n^2 - 3n = \Theta(n^2)$, deve-se determinar os

valores das constantes positivas c_1, c_2 e n_0 tais que

$$c_1 n^2 \leq \frac{1}{2}n^2 - 3n \leq c_2 n^2 \tag{2.3}$$

para todo $n \geq n_0$. Dividindo a expressão (2.3) por n^2, obtém-se $c_1 \leq \frac{1}{2} - \frac{3}{n} \leq c_2$. A inequação da direita pode ser mantida verdadeira para qualquer valor de $n \geq 1$ escolhendo $c_2 \geq \frac{1}{2}$. Por outro lado, a inequação da esquerda pode ser mantida verdadeira para qualquer valor de $n \geq 7$ escolhendo $c_1 \leq \frac{1}{14}$. Portanto, para $c_1 = \frac{1}{14}, c_2 = \frac{1}{2}$ e $n_0 = 7$ pode-se verificar que $\frac{1}{2}n^2 - 3n = \Theta(n^2)$.

- Notação O

  A notação O, é definida considerando a função $g(n)$. Denota-se por $O(g(n))$ o conjunto de funções $f(n)$ tal que [73]

  $$(\exists c \in \mathbb{R}_+)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)[0 \leq f(n) \leq cg(n)]$$

  A função $f(n)$ pertence ao conjunto $O(g(n))$ se existir a constante positiva c e n_0 tais que $f(n)$ possa ser limitada superiormente por $cg(n)$, para $n \geq n_0$. Como exemplo, quer-se encontrar os valores de c e n_0 tais que $3n^3 + 2n^2 + n$ seja $O(n^3)$. Para tal, considere $n = 1$. Neste caso, $c = 3 + 2 + 1 = 6$, e $3n^3 + 2n^2 + n \leq 6n^3$. Essa inequação é mantida verdadeira para $n \geq 1$ e, portanto, para as constantes positivas $c = 6$ e $n_0 = 1$, $3n^3 + 2n^2 + n$ é $O(n^3)$, ou seja, a função $f(n) = 3n^3 + 2n^2 + n$ é assintoticamente e superiormente limitada pela função $g(n) = n^3$, isto é, $f(n) = O(g(n))$.

  Os algoritmos de complexidade constante são do tipo $O(k)$, em que k é uma constante. Os de complexidade linear são do tipo $O(n)$. Os algoritmos de complexidade polinomial são do tipo $O(p(n))$, em que $p(n)$ é um polinômio de grau n, ao passo que, os de complexidade exponencial são do tipo $O(k^n)$, com $k > 1$.

- Notação Ω

  A notação Ω, é definida considerando a função $g(n)$. Denota-se por $\Omega(g(n))$ o

conjunto de funções $f(n)$ tal que [73]

$$(\exists c \in \mathbb{R}_+)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)[0 \leq cg(n) \leq f(n)]$$

A função $f(n)$ pertence ao conjunto $\Omega(g(n))$ se existirem constantes positivas c e n_0 tais que $f(n)$ possa ser assintoticamente limitada inferiormente por $cg(n)$, para $n \geq n_0$. Ainda, segundo [73], para quaisquer duas funções $f(n)$ e $g(n)$, $f(n) = \Theta(g(n))$, se e somente se, $f(n) = O(g(n))$ e $f(n) = \Omega(g(n))$.

2.6 Comentários finais

Neste capítulo foi feita uma revisão dos fundamentos teóricos sobre SEDs modelados por autômatos. Aplicadas a esses autômatos, foram revistas as operações de composição paralela e produto, assim como as de projeção, necessárias para a solução dos problemas propostos neste trabalho. Também, foram revisados os conceitos teóricos de SEDs com observação parcial. Nesse caso, foi revista a definição de alcance não observável e de autômatos observadores, que geram e marcam, respectivamente, a projeção das linguagens geradas e marcadas de um autômato com observação parcial.

No contexto de diagnose de falhas, o conceito de diagnose centralizada foi abordado, em que as definições de sequências normais e de falha, assim como a definição de diagnosticabilidade de uma linguagem, foram apresentadas. Com base nessas definições, foi possível construir o autômato diagnosticador, o qual permite a realização da diagnose "on-line" além de possibilitar a verificação da diagnosticabilidade de uma linguagem. Assim, foram apresentadas as definições de estados certos, normais e incertos, de caminho e caminho cíclico, de ciclo e ciclo indeterminado, de ciclos escondidos e ciclos escondidos indeterminados que, levaram à formulação do teorema de diagnosticabilidade de uma linguagem usando o diagnosticador.

Considerando o caso da diagnose descentralizada, dentre os possíveis protocolos apresentados em [22], foi revisto o protocolo 3, o qual considera a existência de um coordenador responsável pela indicação da falha no sistema quando, pelo menos, um dos diagnosticadores locais indicar a ocorrência da falha. Com a definição de

codiagnosticabilidade de uma linguagem, foi construído o autômato verificador para a verificação da codiagnosticabilidade e para tal, é enunciada uma condição necessária e suficiente para a codiagnosticabilidade, que consiste na busca por um caminho cíclico que viola a condição de codiagnosticabilidade da linguagem do sistema.

A análise de complexidade computacional dos algoritmos apresentados neste trabalho é feita usando a notação O.

Capítulo 3

Codiagnosticabilidade robusta a perdas permanentes de observação de eventos

Neste capítulo, a definição de diagnosticabilidade robusta a perdas permanentes de observação, introduzida em [64], é estendida para o caso descentralizado, levando à definição da CRPPO. Para tanto, a definição de bases para a diagnose é estendida para a definição de bases para a codiagnose. O capítulo aborda também a questão da implementação de um esquema de codiagnose "on-line" robusta a perdas permanentes de observação. Outras contribuições apresentadas neste capítulo são o desenvolvimento de um algoritmo com complexidade polinomial para a verificação da CRPPO, com base no algoritmo de verificação proposto em [70], e o cálculo do atraso para a codiagnose robusta.

Este capítulo está organizado da seguinte forma. Na seção 3.1, é feita uma revisão sobre a diagnosticabilidade robusta a perdas permanentes de observação. Na seção 3.2, é feita a formulação do problema de CRPPO. Além disso, são apresentados um esquema para a codiagnose robusta, um algoritmo de verificação da CRPPO assim como sua complexidade computacional, na seção 3.3, é calculado o atraso para a codiagnose. Finalmente, na seção 3.4, são apresentadas conclusões sobre este

capítulo.

3.1 Diagnosticabilidade robusta a perdas permanentes de observação

Nos problemas tradicionais de diagnose de falhas, [18], é suposto que existe uma perfeita operação dos sensores e comunicação entre sensores e diagnosticador, isto é, não há perda de observação por inoperância dos sensores ou perda de comunicação entre sensores e diagnosticador. Contudo, na prática, os sensores podem parar de funcionar, ou ainda, pode haver uma perda de comunicação entre sensores e diagnosticador. Nesses casos, o diagnosticador obtido considerando-se que todos os eventos de Σ_o são observáveis, pode travar em alguns determinados estados do sistema ou pode, ainda, fazer um diagnóstico errado da falha no sistema [64]. Para melhor compreender esse fato, é apresentado a seguir um exemplo prático de uma planta mecatrônica montadora de cubos, mostrada na figura 3.1, que pertence ao Laboratório de Controle e Automação (LCA) da COPPE/UFRJ.

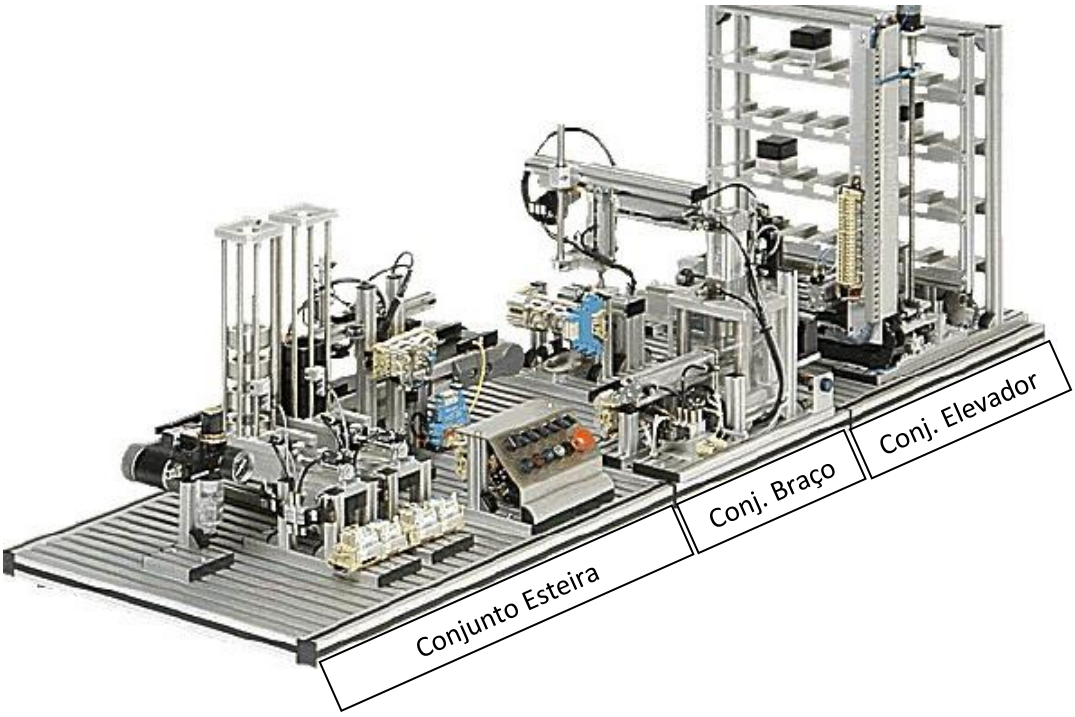


Figura 3.1: Planta mecatrônica para montagem de cubos (reproduzida de [1]).

O funcionamento da planta mecatrônica se dá através de um processo cíclico que consiste na entrega, transporte, armazenamento e descarte de duas peças por ciclo. Os cubos são empilhados em duas colunas denominadas por MAG 1 e MAG 2 e localizadas no início da esteira. Os cubos empilhados na coluna MAG 1 são metálicos enquanto que os da coluna MAG 2 são pretos e plásticos. Os cubos são entregues para a esteira no ponto P1, conforme ilustrado na figura 3.2. No início do processo, é feita a calibragem do braço girando-o no sentido horário até o acionamento de um sensor indutivo e depois desse acionamento, ele gira no sentido anti-horário até parar alinhado com a esteira. O primeiro cubo a ser entregue para a esteira rolante é o cubo metálico (cubo 1) e sua entrega se dá por meio de um atuador pneumático. Em seguida, a esteira é ligada realizando o transporte do cubo metálico até o fim da esteira (P2) onde um sensor óptico detecta a presença da peça e envia um sinal digital de nível lógico 1 para o controlador lógico programável da planta para que a esteira seja desligada e o braço acionado para pegar a peça.

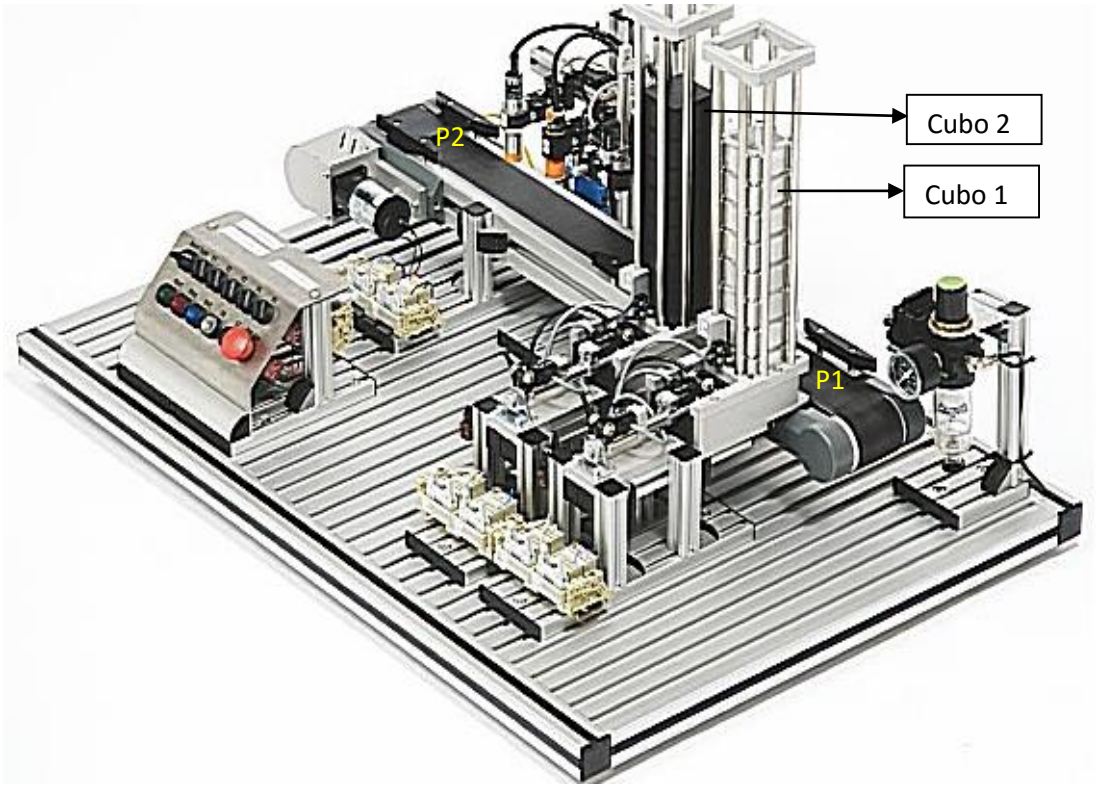


Figura 3.2: Planta mecatrônica: Conjunto Esteira (reproduzida de [1]).

O braço giratório transporta a peça para o elevador. No mesmo instante em que a peça é retirada da esteira, o sensor óptico envia o nível lógico 0 para o CLP e a segunda peça é entregue para a esteira no ponto P1. A esteira é novamente ligada para que a peça seja transportada até o ponto P2, enquanto o braço entrega a peça no elevador e retorna para a posição inicial alinhado com a esteira. É importante ressaltar que toda vez que o braço retorna à posição inicial o sensor indutivo é acionado indicando a calibração do braço.

Após a entrega da segunda peça para o elevador, o braço é comandado para levar duas peças do elevador, uma de cada vez, para um local de descarte de peças. Toda vez que o braço termina de realizar o movimento de descarte de uma peça, o sensor indutivo é acionado. Após o descarte das duas peças, o braço retorna para a posição inicial alinhado com a esteira para reiniciar o ciclo.

Os comportamentos da esteira e do braço robótico da planta mecatrônica podem ser modelados utilizando-se autômatos, como mostrado nas figuras 3.3 e 3.4, respec-

tivamente. O conjunto de eventos do autômato que modela o comportamento da esteira, G_E, é dado por $\Sigma_E = \{N_P, l, FC_{on}, FC_{off}, FC = 0, FC = 1, d, \sigma_u, \sigma_f\}$, em que σ_f é o evento de falha. A falha neste caso é modelada como sendo da bomba de vácuo pneumática, a qual leva o braço a não ser mais capaz de remover as peças (cubos) da esteira. O conjunto de eventos do autômato do braço, G_B, é dado por $\Sigma_B = \{L_B, SI, FC_{off}, [FC = 1], \sigma_u\}$. O significado dos eventos e estados dos autômatos G_E e G_B é apresentado nas tabelas 3.1, 3.2, e 3.3.

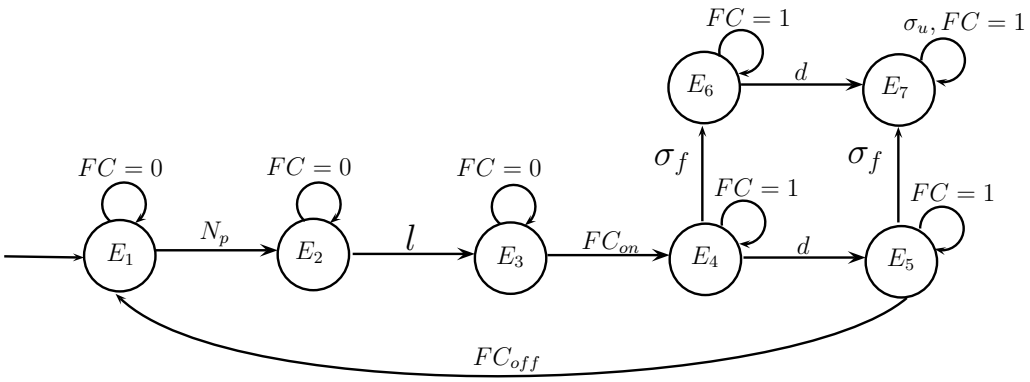


Figura 3.3: Autômato que modela a esteira, G_E.

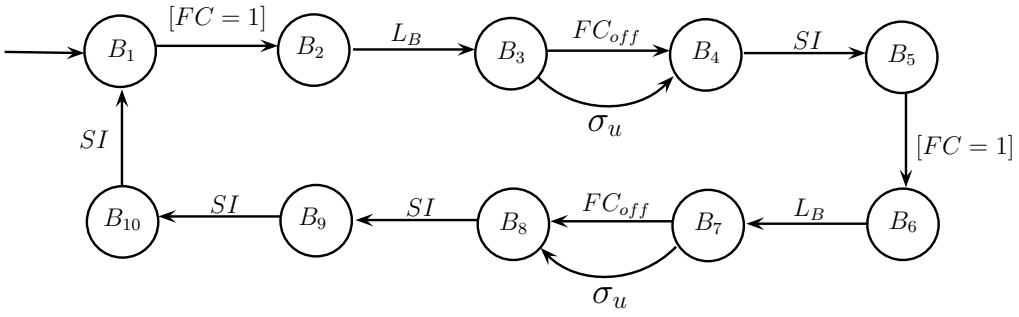


Figura 3.4: Autômato que modela o braço, G_B.

Para construir o autômato G_{EB} que modela o comportamento do conjunto esteira-braço, é feita a composição paralela dos autômatos G_E e G_B, isto é, $G_{EB} = G_E||G_B$ e o autômato obtido dessa operação é mostrado na figura 3.5

Para a análise de diagnosticabilidade da linguagem do sistema composto pelo conjunto esteira-braço e modelado pelo autômato 3.5, vamos supor que o conjunto

Tabela 3.1: Eventos do sistema.

Eventos	Significado
N_p	Chegada de nova peça
l	Ligar esteira
FC_{on}	Borda de subida do sensor de fim de curso da esteira
d	Desligar esteira
FC_{off}	Borda de descida do sensor de fim de curso da esteira
σ_f	Evento de falha
[FC = 1]	Sensor fim de curso com nível lógico 1
L_B	Braço acionado para pegar peça da esteira
SI	Sensor indutivo ativado pelo braço
σ_u	Evento não observável após a ocorrência da falha

Tabela 3.2: Estados da esteira.

Estados	Significado
E_1	Esteira desligada e sem peça
E_2	Esteira desligada com peça no ponto P1
E_3	Esteira ligada transportando peça
E_4	Esteira ligada com peça no ponto P2
E_5	Esteira desligada com peça no ponto P2
E_6	Esteira ligada com peça no ponto P2 após falha
E_7	Esteira desligada com peça no ponto P2 após falha

Tabela 3.3: Estados do braço.

Estados	Significado
B_1	Braço aguardando chegada da primeira peça na esteira
B_2	Braço pronto para pegar a primeira peça
B_3	Braço acionado para pegar a primeira peça na esteira
B_4	Braço transportando a primeira peça para o elevador
B_5	Braço aguardando chegada da segunda peça na esteira
B_6	Braço pronto para pegar a segunda peça
B_7	Braço acionado para pegar a segunda peça na esteira
B_8	Braço transportando a segunda peça para o elevador
B_9	Braço pronto para iniciar o descarte de duas peças
B_{10}	Braço finaliza o primeiro descarte de peça

de eventos observáveis desse sistema seja dado por $\Sigma_o = \{N_P, l, SI\}$. Analisando o autômato construído, é possível observar que a linguagem do sistema é diagnosticável com relação à projeção P_o e ao evento de falha σ_f pois, não é possível encontrar nenhuma sequência normal ω e de falha s_Y tais que $P_o(\omega) = P_o(s_Y)$. Por outro lado, se for considerada a possibilidade de se perder de forma permanente a observação associada ao evento SI de modo que o novo conjunto de evento observáveis seja $\Sigma_o = \{N_P, l\}$. Então, é possível encontrar uma sequência normal $\omega = N_P.l$ e uma sequência de falha arbitrariamente longa $s_Y = N_P.l.FC_{on}.d.\sigma_f.(FC_1.L_B.\sigma_u.SI.FC_1.L_B.\sigma_u.SI.SI.SI)^n$, em que n é um número inteiro positivo, tal que $P_o(\omega) = P_o(s_Y) = N_P l$. Portanto, diante de uma perda permanente da observação associada ao evento SI, isto é, a perda permanente do sensor indutivo, a linguagem do sistema que, antes, era diagnosticável, passou a ser não diagnosticável. Isso mostra a importância de se estudar a diagnosticabilidade de sistemas a eventos discretos sujeitos a perdas de observação de eventos.

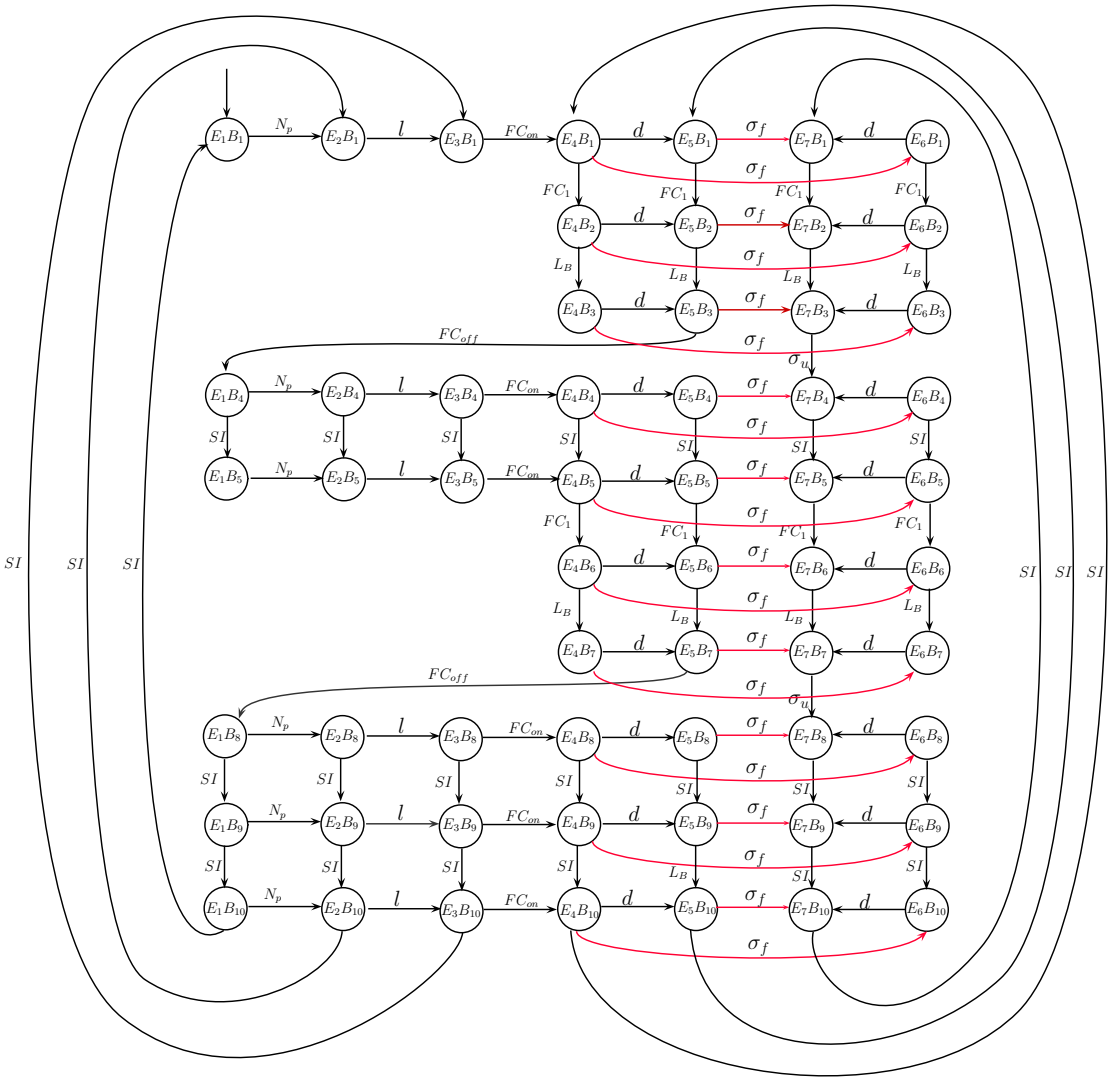


Figura 3.5: Autômato que modela o conjunto esteira-braço, G_{EB}.

O problema de diagnosticar a ocorrência do evento de falha em um SED diante de falhas permanentes de sensores ou de perdas permanentes de comunicação entre sensores e diagnosticador é apresentado em [64]. Nesse caso, o atual conjunto de eventos do sistema é desconhecido, isto é, Σ_o denota o conjunto de eventos potencialmente observáveis do sistema, e o objetivo é construir um esquema de diagnose robusta a incertezas na observação dos eventos. Para tanto, a seguinte hipótese é

feita em [64].

H2. Toda perda de observação deve ocorrer antes da primeira ocorrência dos eventos registrados pelos sensores defeituosos e a perda deve ser permanente.

Para ilustrar esse problema, considere o autômato da figura 3.6(a) cujo diagnosticador, considerando $\Sigma_o = \{a, b, c\}$, e $\Sigma_{uo} = \{\sigma_f\}$ é mostrado na figura 3.6(b). Suponha que a observação do evento b tenha sido perdida de forma permanente, antes da sua primeira observação, devido a perda do sensor que registra a ocorrência desse evento ou devido a perda no canal de comunicação entre o sensor e o diagnosticador. Suponha, também, que a sequência $s_Y = \sigma_f abc^n$, $n \in \mathbb{N}$ tenha sido executada pelo sistema. Observe que o primeiro evento a ser observado pelo diagnosticador G_d é o evento a levando o diagnosticador G_d do seu estado inicial $\{1N, 3Y\}$ para o estado incerto $\{2N, 5Y\}$. O segundo evento a ser observado deveria ser o evento b, mas como sua observação foi perdida permanentemente, no lugar dele o diagnosticador observará o evento c levando, portanto, o diagnosticador para o estado certo $\{4N\}$. Como nesse estado o próximo evento a ser executado pelo sistema indefinidamente, não está ativo, o diagnosticador permanecerá, indefinidamente, no estado $\{4N\}$ com a certeza de que a falha não ocorreu. Esta decisão do diagnosticador de que a falha não ocorreu está errada uma vez que a sequência s_Y, de comprimento arbitrariamente longa, executada pelo sistema contém o evento de falha σ_f. Essa incapacidade do diagnosticador de diagnosticar corretamente a ocorrência da falha diante de uma perda permanente de observação sugere que o diagnosticador deve ser modificado de forma a tolerar perdas de observação de eventos em SEDs.

Em [64], todas as possibilidades de perdas de observação são previamente conhecidas, isto é, o atual conjunto de eventos observáveis do sistema é um dos possíveis conjuntos de eventos observáveis previamente definidos. Seja m o número dos possíveis conjuntos de eventos observáveis do sistema, e seja $\Sigma_{uo}^j \subset \Sigma_o$, para $j = 1, \ldots, m$, os conjuntos de eventos sujeitos a perdas permanentes de observação. Então, $\Sigma_o^j = \Sigma_o \setminus \Sigma_{uo}^j$ denota um possível conjunto de eventos observáveis do sistema. Segundo a hipótese **H2**, o problema de diagnose robusta a perdas permanentes de

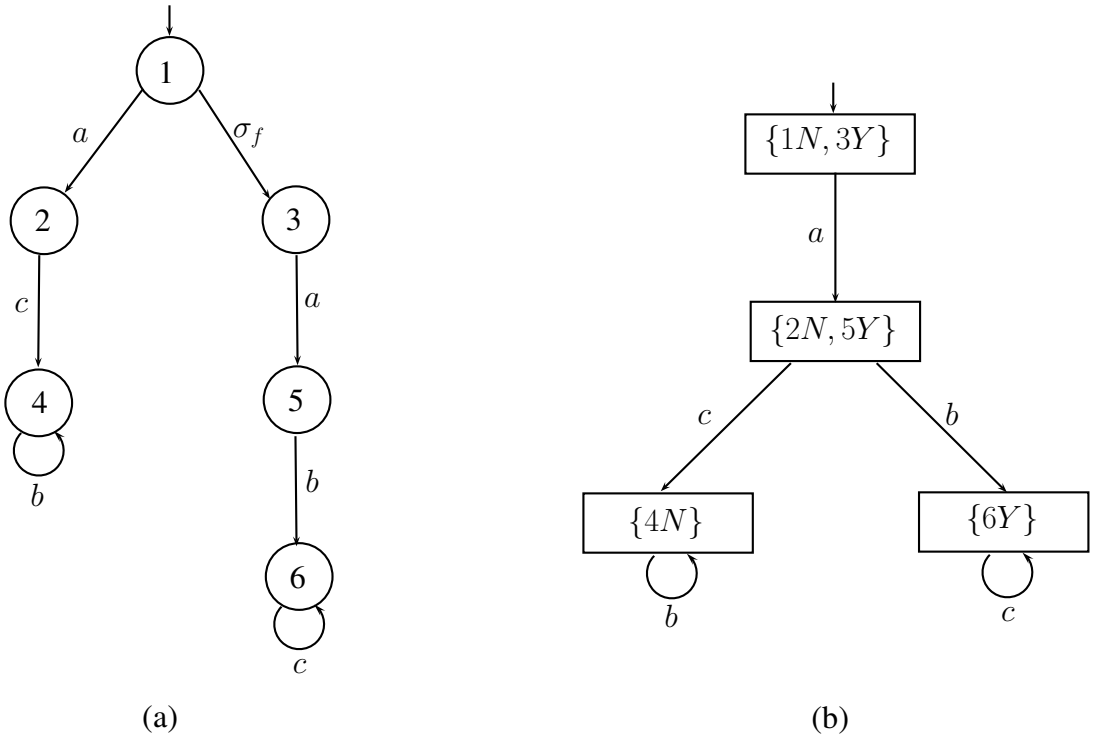


Figura 3.6: Autômato (G)(a), Diagnosticador G_d(b).

observação pode ser formulado como o problema de identificar a ocorrência da falha sem saber qual dos conjuntos Σ_o^j, $j = 1, \ldots, m$, é o real conjunto de eventos observáveis do sistema. É importante notar que se a linguagem L do sistema não é diagnosticável com relação a um possível conjunto de eventos observáveis Σ_o^j e Σ_f, então, trivialmente, L não é robustamente diagnosticável com relação a todos os possíveis conjuntos de eventos observáveis e Σ_f. Portanto, é considerado em [64] que L é diagnosticável com relação a Σ_o^j e Σ_f, para todo $j \in I_m = 1, 2, \ldots, m$. Isso leva a seguinte definição [21].

Definição 3.1. *Um conjunto $\Sigma_o^j \subseteq \Sigma_o$ é dito ser base para a diagnose de L, se L é diagnosticável com relação a $P_o^j : \Sigma^\star \to \Sigma_o^{j\star}$ e Σ_f.* □

A definição de diagnosticabilidade robusta a perdas permanentes de observação de eventos é apresentada a seguir [64].

Definição 3.2. *(Diagnosticabilidade robusta a perdas permanentes de observação) Seja $\Sigma_{db} = \{\Sigma_o^1, \Sigma_o^2, \ldots, \Sigma_o^m\}$, em que Σ_o^j, $j = 1, \ldots, m$, são bases para a diagnose*

de L. Defina o conjunto

$$\Sigma_{rob,d} = \{\Sigma_{uo}^1, \Sigma_{uo}^2, \ldots, \Sigma_{uo}^m\},$$

em que

$$\Sigma_{uo}^j = \Sigma_o \setminus \Sigma_o^j, j = 1, 2, \ldots, m.$$

Então, L é robustamente diagnosticável com relação às projeções P_o^j, $j = 1, 2, \ldots, m$, em que $P_o^j : \Sigma^* \to \Sigma_o^{j*}$, e Σ_f, ou equivalentemente, com relação a perdas permanentes de observação dos eventos de todos os conjuntos Σ_{uo}^j, $j = 1, 2, \ldots, m$, e Σ_f, se a condição a seguir for verdadeira:

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

em que a condição de diagnosticabilidade robusta R_D é dada por:

$$(\forall k, l \in I_m, k \neq l)[P_o^k(st) \neq P_o^l(\omega), \forall \omega \in L_N].$$

De acordo com a definição 3.2, L não é robustamente diagnosticável com relação a perdas permanentes de observação, se existir uma sequência de falha $st \in L \setminus L_N$, com comprimento arbitrariamente longo após a ocorrência do evento de falha, e uma sequência normal $\omega \in L_N$, tal que $P_o^k(st) = P_o^l(\omega)$, em que $k, l \in I_m$, são associados a duas bases distintas do sistema para a diagnose. Em palavras, L não é robustamente diagnosticável se existir incerteza com relação à sequência executada pelo sistema e à verdadeira observação do sistema, isto é, não se sabe se o sistema executou a sequência de falha st e o conjunto de eventos observáveis é Σ_o^k, ou se o sistema executou ω e o atual conjunto de eventos observáveis é Σ_o^l.

Quando L é robustamente diagnosticável, o esquema de diagnose robusta proposto em [64] pode ser implementado para identificar a ocorrência da falha em situação de perdas permanentes de observação. O esquema de diagnose robusta, mostrado na figura 3.7 para o caso com três bases para a diagnose, consiste em

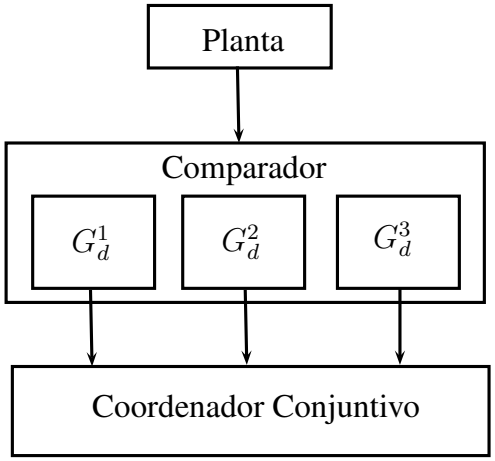


Figura 3.7: Esquema de diagnose robusta.

construir um diagnosticador G_d^j, para cada base para a diagnose Σ_o^j, $j = 1, 2, \ldots, m$, e então, realizar a diagnose robusta executando os diagnosticadores G_d^j, $j = 1, \ldots, m$, em paralelo. As decisões de todos os diagnosticadores G_d^j são enviadas para um coordenador, referido daqui em diante, como coordenador conjuntivo, e cuja função é comunicar a ocorrência da falha para o operador do sistema conforme as informações recebidas dos diagnosticadores G_d^j. Se um evento for observado e ele não pertencer a base para a diagnose Σ_o^j, então Σ_o^j não corresponde à base correta do sistema para a diagnose robusta, e a informação fornecida pelo diagnosticador G_d^j deve ser descartada pelo coordenador conjuntivo. A mensagem para descartar um diagnosticador é enviada ao coordenador conjuntivo por um comparador, localizado no mesmo local em que os diagnosticadores são implementados, o qual compara o evento observado com as bases para a diagnose. A falha é diagnosticada quando todos os diagnosticadores remanescentes a identificam e comunicam sua ocorrência ao coordenador conjuntivo.

3.2 Codiagnosticabilidade robusta a perdas permanentes de observação (CRPPO)

3.2.1 Formulação do problema

Em SEDs nos quais o processo de diagnose de falhas é feito com arquitetura descentralizada, como mostrado na figura 3.8, diagnosticadores são implementados nos diversos locais de diagnose existentes ao longo da planta.

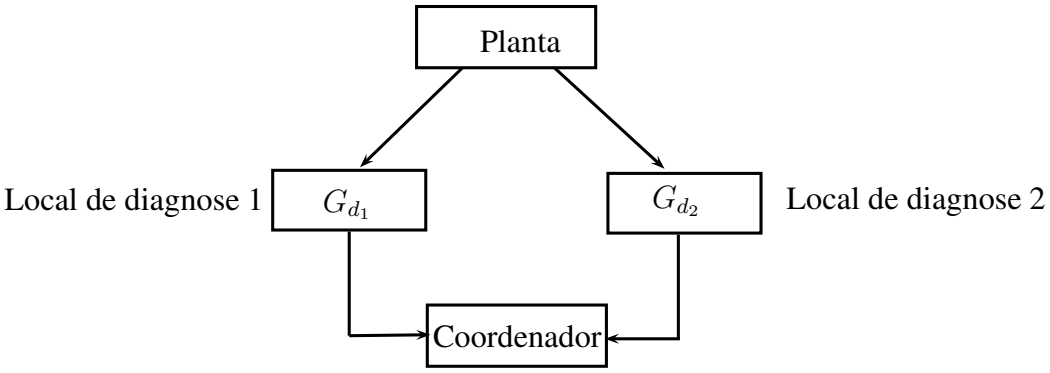


Figura 3.8: Esquema de codiagnose.

Seja n o número de locais de diagnose e Σ_{o_i}, $i = 1, \ldots, n$, os conjuntos de eventos potencialmente observáveis do sistema para cada local de diagnose [1]. Assim como em [64], é considerado, neste trabalho que a hipótese **H2** é verdadeira para o caso da codiagnose robusta, isto é, todas as possibilidades de perdas de observação de eventos em todos os locais de diagnose são previamente conhecidas, e todas as perdas de sensores ou de comunicação são permanentes.

Seja $\Sigma^j_{uo_i} \subseteq \Sigma_{o_i}$ as possíveis perdas permanentes de observação de eventos no i-ésimo local de diagnose. Então, o conjunto $\Sigma^j_{o_i} = \Sigma_{o_i} \setminus \Sigma^j_{uo_i}$, corresponde a uma possível observação de eventos pelo i-ésimo local de diagnose. Seja m o número de possíveis observações de eventos para todos os diagnosticadores locais, isto é, cada conjunto $\{\Sigma^j_{o_1}, \Sigma^j_{o_2}, \ldots, \Sigma^j_{o_n}\}$, para $j = 1, \ldots, m$, corresponde a uma possível obser-

[1] Note a diferença entre as notações de base para a diagnose Σ^j_o e o conjunto dos eventos potencialmente observáveis Σ_{o_i} na arquitetura descentralizada.

vação de eventos em todos os locais de diagnose. Então, o problema de codiagnose robusta pode ser formulado como o problema de identificar a ocorrência do evento de falha σ_f, baseado nas decisões dos diagnosticadores locais, sem saber quais dos conjuntos $\{\Sigma^j_{o_1}, \Sigma^j_{o_2}, \ldots, \Sigma^j_{o_n}\}$, $j = 1, \ldots, m$, correspondem aos verdadeiros conjuntos de eventos observáveis do sistema.

É importante observar que se a linguagem L não for codiagnosticável com relação ao conjunto $\{\Sigma^j_{o_1}, \Sigma^j_{o_2}, \ldots, \Sigma^j_{o_n}\}$ e Σ_f, então trivialmente, L não é robustamente codiagnosticável com relação a todos os conjuntos $\{\Sigma^j_{o_1}, \Sigma^j_{o_2}, \ldots, \Sigma^j_{o_n}\}$, para $j = 1, \ldots, m$, e Σ_f. Isso leva à seguinte extensão da definição de base para a diagnose de falhas.

Definição 3.3. *(Base para a codiagnose) O conjunto* $\hat{\Sigma}_{cb} = \{\hat{\Sigma}_{o_1}, \hat{\Sigma}_{o_2}, \ldots, \hat{\Sigma}_{o_n}\}$*, em que* $\hat{\Sigma}_{o_i} \subseteq \Sigma_{o_i}$*,* $i = 1, \ldots, n$*, é uma base para a codiagnose de* L *se* L *é codiagnosticável com relação às projeções* $\hat{P}_{o_i} : \Sigma^* \to \hat{\Sigma}^*_{o_i}$*,* $i = 1, \ldots, n$*, e* Σ_f*.* □

Observação 3.1. *Note que um conjunto* $\hat{\Sigma}_{o_i}$ *de uma base para a codiagnose* $\hat{\Sigma}_{cb}$ *pode ser igual ao conjunto vazio se a observação do* i*-ésimo diagnosticador local for redundante, isto é, todas as sequências de falha de* L *podem ser distinguidas das sequências normais, depois de um número limitado de observações de eventos, pelos diagnosticadores locais remanescentes.* □

3.2.2 Esquema de codiagnose robusta

Nesta seção, é proposto um esquema de codiagnose robusta baseado nos esquemas de diagnose robusta e de codiagnose apresentados nas figuras 3.7 e 3.8, respectivamente. O esquema de codiagnose proposto neste trabalho é mostrado na figura 3.9 para o caso de três bases para a codiagnose e dois locais de diagnose.

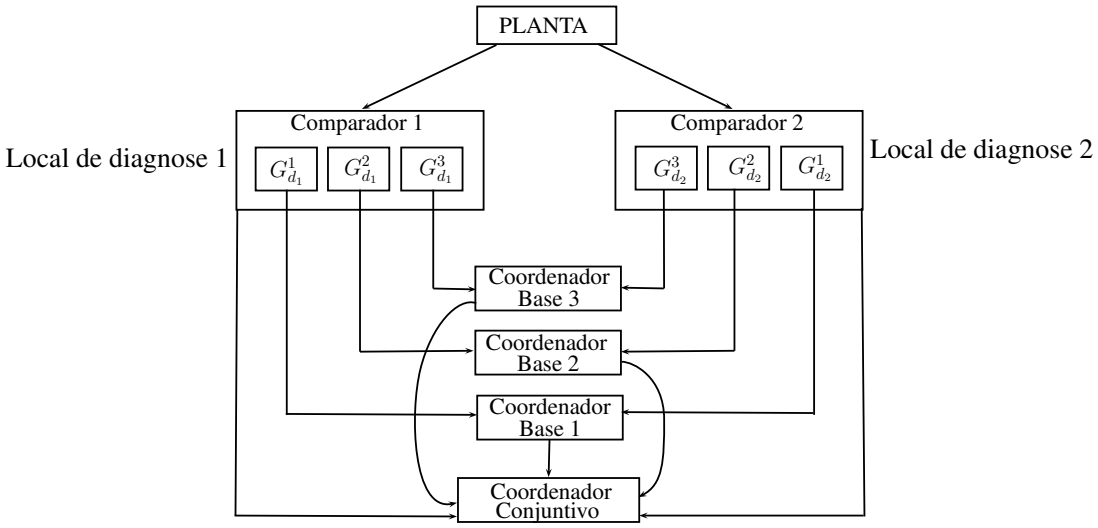


Figura 3.9: Esquema de codiagnose robusta.

Sejam $\Sigma^j_{cb} = \{\Sigma^j_{o_1}, \Sigma^j_{o_2}, \ldots, \Sigma^j_{o_n}\}$, $j = 1, \ldots, m$, bases para a codiagnose correspondente a todas as possíveis observações dos eventos do sistema por todos os locais de diagnose, e sejam $G^j_{d_i}$, para $j = 1, 2, \ldots, m$, diagnosticadores locais associados ao i-ésimo local de diagnose, construídos considerando $\Sigma^j_{o_i} \in \Sigma^j_{cb}$, como o conjunto de eventos observáveis. No esquema de codiagnose robusta proposto neste trabalho, os diagnosticadores locais $G^j_{d_i}$, para $j = 1, 2, \ldots, m$, são implementados funcionando em paralelo no i-ésimo local de diagnose, como mostrado na figura 3.9. Associado a cada base para a codiagnose Σ^j_{cb}, existe um coordenador de base que reporta a ocorrência da falha para um coordenador de maior nível, chamado de coordenador conjuntivo, quando pelo menos um dos diagnosticadores locais $G^j_{d_i}$, $i = 1, \ldots, n$, reportar a ocorrência da falha baseado nas suas próprias observações dos eventos do sistema. O coordenador conjuntivo usa a informação fornecida pelos coordenadores de base para diagnosticar o evento de falha. Se o i-ésimo local de diagnose observar um evento que não é viável no estado atual de um diagnosticador local $G^j_{d_i}$, então, toda a base para a codiagnose Σ^j_{cb} deve ser descartada, visto que, não é a base correta do sistema. Assim, uma comparação entre o evento observado e os eventos viáveis no estado atual de cada diagnosticador local $G^j_{d_i}$, para $j = 1, 2, \ldots, m$, é feita no i-ésimo local de diagnose. Caso o evento observado não seja viável no estado atual

do diagnosticador $G^j_{d_i}$, o comparador de evento do i-ésimo local de diagnose envia uma mensagem ao coordenador conjuntivo para descartar o j-ésimo coordenador de base. A falha é identificada quando todos os coordenadores de base remanescentes reportarem a ocorrência do evento de falha ao coordenador conjuntivo.

É importante observar que o esquema de codiagnose robusta online proposto neste trabalho é uma extensão do esquema de diagnose robusta online apresentado em [64](apresentado na figura 3.7) para o caso decentralizado, isto é, o esquema de diagnose robusta proposto em [64] é igual ao esquema de codiagnose proposto aqui para $n = 1$. Nesse caso, não existem os coordenadores de base e os diagnosticadores reportam a ocorrência da falha diretamente ao coordenador conjuntivo.

3.2.3 Definição de codiagnosticabilidade robusta a perdas permanentes de observação (CRPPO)

A seguir é apresentada a definição da CRPPO sob a hipótese **H2**.

Definição 3.4. *(CRPPO) Seja $\Sigma_{cb} = \{\Sigma^1_{cb}, \Sigma^2_{cb}, \ldots, \Sigma^m_{cb}\}$ o conjunto de bases para a codiagnose de L, em que $\Sigma^j_{cb} = \{\Sigma^j_{o_1}, \Sigma^j_{o_2}, \ldots, \Sigma^j_{o_n}\}$, $j = 1, \ldots, m$, é uma base para a codiagnose de L, e $\Sigma^j_{o_i}$ denota o conjunto de eventos observáveis associado ao i-ésimo local de diagnose e a j-ésima base para a codiagnose. Defina o conjunto*

$$\Sigma^j_{rob,c} = \{\Sigma^j_{uo_1}, \Sigma^j_{uo_2}, \ldots, \Sigma^j_{uo_n}\}, \tag{3.1}$$

em que

$$\Sigma^j_{uo_i} = \Sigma_{o_i} \setminus \Sigma^j_{o_i}, i = 1, 2, \ldots, n. \tag{3.2}$$

Então, L é robustamente codiagnosticável em relação às projeções $P^j_{o_i}$, $i = 1, 2, \ldots, n$ e $j = 1, 2, \ldots, m$, em que $P^j_{o_i} : \Sigma^* \to \Sigma^{j*}_{o_i}$, e Σ_f, ou equivalentemente, em relação à perdas permanentes de observações de eventos dos conjuntos $\Sigma^j_{uo_i}$, $i = 1, 2, \ldots, n$, $(i \in I_n = \{1, 2, \ldots, n\})$ para todas bases para a codiagnose

$j = 1, 2, \ldots, m$, *se a condição a seguir for verdadeira:*

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

em que a condição de codiagnosticabilidade robusta R_C *é dada como*

$$(\forall k, l \in I_m, k \neq l)(\exists i \in I_n)[P^k_{o_i}(st) \neq P^l_{o_i}(\omega), \forall \omega \in L_N].$$

Observação 3.2. *Note que a definição 3.2 pode ser obtida da definição 3.4 se o número de locais de diagnose for restrito a apenas um só, isto é, se houver apenas um diagnosticador central.* □

Observação 3.3. *O valor mínimo de* z *que satisfaz a condição de codiagnosticabilidade robusta será denotada, neste trabalho, por* z^**, e será referido como o tempo de atraso para a codiagnose robusta. Um algoritmo para o cálculo de* z^* *é apresentado na seção 3.3.* □

De acordo com a definição 3.4, L não é robustamente codiagnosticável com relação a perdas permanentes de observação se existir: (i) uma sequência de falha $st \in L \setminus L_N$, de comprimento arbitrariamente longo após a ocorrência do evento de falha; (ii) sequências normais $\omega_i \in L_N$ para $i = 1, 2, \ldots, n$, não necessariamente distintas entre elas, e; (iii) duas bases distintas para a codiagnose Σ^k_{cb} e Σ^l_{cb}, tais que $P^k_{o_i}(st) = P^l_{o_i}(\omega_i)$, para todo $i \in I_n$. Nesse caso, se Σ^k_{cb} corresponde à verdadeira observação dos locais de diagnose, e a sequência st é executada pelo sistema, então o conjunto dos diagnosticadores locais associado à base Σ^k_{cb} indica a ocorrência do evento de falha, pois Σ^k_{cb} é base para a codiagnose. Contudo, os diagnosticadores locais, associados à base para a codiagnose Σ^l_{cb}, não indicam a ocorrência do evento de falha, uma vez que para cada diagnosticador local $\Sigma^l_{o_i}$, existe uma sequência normal ω_i com a mesma observação sob a projeção $P^l_{o_i}$ que st sob a projeção $P^k_{o_i}$. Então, nesse caso, há incerteza quanto a ocorrência de uma sequência normal ou de uma sequência de falha no sistema após falha de sensor/comunicação. Portanto, L não é robustamente codiagnosticável.

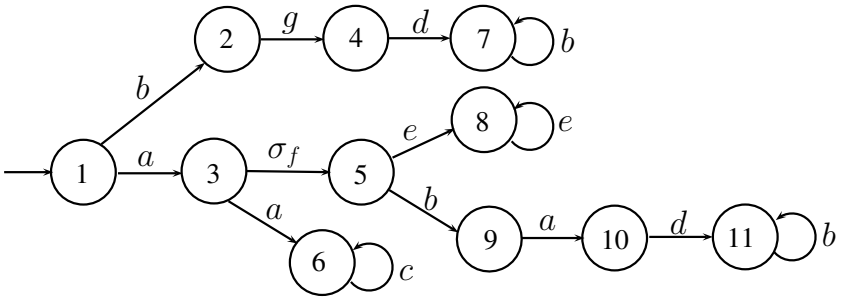
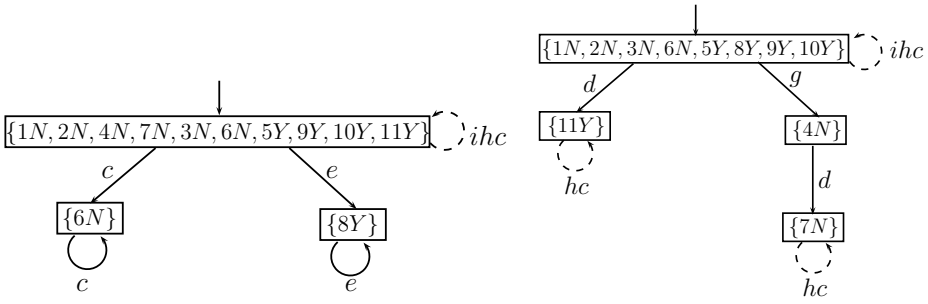


Figura 3.10: Autômato G.



(a) (b)

Figura 3.11: Diagnosticadores locais $G^1_{d_1}$ (a) e $G^1_{d_2}$ (b).

Em seguida, é apresentado um exemplo para ilustrar a definição de codiagnosticabilidade robusta apresentada neste trabalho.

Exemplo 3.1. *Seja G o autômato mostrado na figura 3.10, em que o conjunto de eventos é $\Sigma = \{a, b, c, d, e, g, \sigma_f\}$, e o conjunto de eventos de falha é $\Sigma_f = \{\sigma_f\}$. Considere que existem dois locais de diagnose cujos conjuntos eventos potencialmente observáveis são $\Sigma_{o_1} = \{b, c, e\}$ e $\Sigma_{o_2} = \{a, d, g\}$. Seja $\Sigma^1_{cb} = \{\Sigma^1_{o_1}, \Sigma^1_{o_2}\}$ e $\Sigma^2_{cb} = \{\Sigma^2_{o_1}, \Sigma^2_{o_2}\}$ duas bases para a codiagnose de L em que $\Sigma^1_{o_1} = \{c, e\}$, $\Sigma^1_{o_2} = \{d, g\}$, $\Sigma^2_{o_1} = \{b, e\}$, e $\Sigma^2_{o_2} = \{a, d\}$.*

De acordo com o esquema de codiagnose robusta proposto nesta seção, os diagnosticadores locais $G^1_{d_1}$ e $G^1_{d_2}$, mostrados nas figuras 3.11(a) e 3.12(a), respectivamente, são executados simultaneamente no local de diagnose 1, e os diagnosticadores locais $G^1_{d_2}$ e $G^2_{d_2}$, apresentados nas figuras 3.11(b) e 3.12(b), respectivamente, são

executados simultaneamente no local de diagnose 2. Suponha que os sensores que efetivamente foram perdidos na comunicação com os locais de diagnose são aqueles associados ao evento b no local de diagnose 1, e o evento a no local de diagnose 2. Portanto, as observações corretas do sistema são dadas pela base para a codiagnose Σ^1_{cb}. Considere, agora, que a sequência de falha $st = a\sigma_f badb^z$, $z \in \mathbb{N}$, foi executada pelo sistema. Então, as sequências observadas nos locais de diagnose 1 e 2 são, respectivamente, $P^1_{o_1}(st) = \varepsilon$ e $P^1_{o_2}(st) = d$.

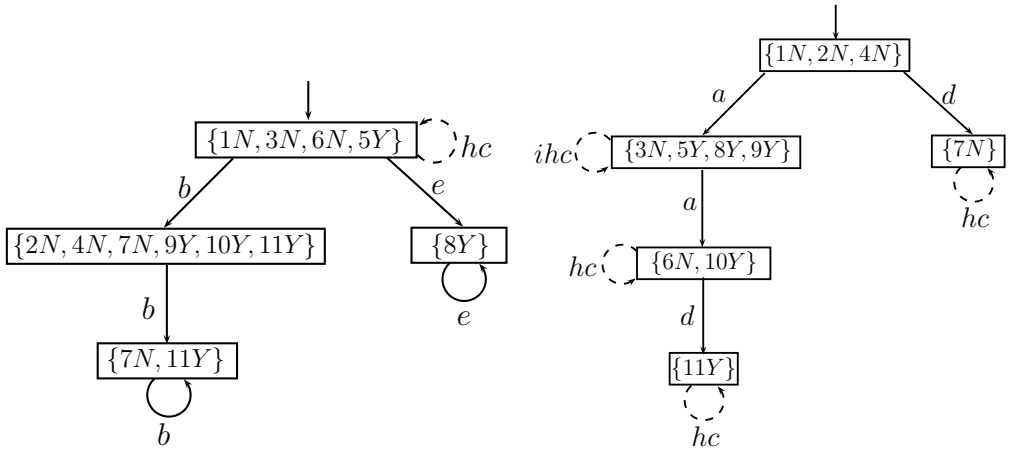


Figura 3.12: Diagnosticadores locais $G^2_{d_1}$ (a) e $G^2_{d_2}$ (b).

Na figura 3.11, foram apresentados os diagnosticadores locais $G^1_{d_1}$ e $G^1_{d_2}$, em que é possível observar que o evento de falha é detectado pelo diagnosticador local $G^1_{d_2}$. Contudo, a partir dos diagnosticadores locais $G^2_{d_1}$ e $G^2_{d_2}$, mostrados na figura 3.12, existe dúvida em relação a ocorrência do evento de falha, visto que, os estados alcançados depois da ocorrência das sequências ε no local de diagnose 1, e d no local de diagnose 2, são, respectivamente, $(1N, 3N, 6N, 5Y)$ e $(7N)$[2]. Isso é devido ao fato de existirem sequências normais $\omega_1 = \varepsilon$ e $\omega_2 = bgd$, tais que $P^1_{o_1}(st) = P^2_{o_1}(\omega_1) = \varepsilon$ e $P^1_{o_2}(st) = P^2_{o_2}(\omega_2) = d$. Portanto, a linguagem L não é robustamente codiagnosticável com relação à projeção $P^j_{o_i}$, $i = 1, 2$ e $j = 1, 2$, e Σ_f, ou equivalentemente, L

[2]Note que todos os diagnosticadores locais das figuras 3.11 e 3.12 têm ciclos escondidos ou ciclos escondidos indeterminados, visto que, existe em G caminhos cíclicos com eventos não observáveis. [21].

não é robustamente codiagnosticável com relação a perdas permanentes de observações de eventos $\Sigma^1_{uo_1} = \{b\}$ e $\Sigma^1_{uo_2} = \{a\}$, associados aos locais de diagnose 1 e 2, respectivamente, e $\Sigma^2_{uo_1} = \{c\}$ e $\Sigma^2_{uo_2} = \{g\}$, associados aos locais de diagnose 1 e 2, respectivamente. □

3.2.4 Algoritmo de verificação

Em alguns casos, a CRPPO de L pode ser imediatamente verificada. Esse caso é mostrado no teorema a seguir.

Teorema 3.1. *Seja* $\Sigma^1_{cb} = \{\Sigma^1_{o_1}, \Sigma^1_{o_2}, \ldots, \Sigma^1_{o_n}\}$ *e* $\Sigma^2_{cb} = \{\Sigma^2_{o_1}, \Sigma^2_{o_2}, \ldots, \Sigma^2_{o_n}\}$ *duas bases distintas para a codiagnose, tal que* $\Sigma^1_{o_i} \subseteq \Sigma^2_{o_i}$*, para* $i = 1, 2, \ldots, n$*. Então* L *é robustamente codiagnosticável com relação às projeções* $P^j_{o_i}$*,* $i = 1, 2, \ldots, n$ *e* $j = 1, 2$*, e* Σ_f*.*

Demonstração. Considere que L não é robustamente codiagnosticável com relação às projeções $P^j_{o_i}$, $i = 1, 2, \ldots, n$ e $j = 1, 2$, e Σ_f. Então, de acordo com a definição 3.4, existe uma sequência de falha arbitrariamente longa st e sequências normais ω_i tais que $P^1_{o_i}(st) = P^2_{o_i}(\omega_i)$ ou $P^2_{o_i}(st) = P^1_{o_i}(\omega_i)$, para todo $i \in I_n$. Considere, primeiro, o caso em que $P^1_{o_i}(st) = P^2_{o_i}(\omega_i)$, para todo $i \in I_n$. Como $\Sigma^1_{o_i} \subseteq \Sigma^2_{o_i}$,para todo $i \in I_n$, então, ω_i não contém nenhum evento de $\Sigma^2_{o_i} \setminus \Sigma^1_{o_i}$, o que implica que

$$P^1_{o_i}(\omega_i) = P^2_{o_i}(\omega_i), \forall i \in \{1, 2, \ldots, n\}. \tag{3.3}$$

Note, agora, que como Σ^1_{cb} é base para a codiagnose, então, existe $i \in \{1, 2, \ldots, n\}$ tal que

$$P^1_{o_i}(st) \neq P^1_{o_i}(\omega_i), \forall \omega_i \in L_N. \tag{3.4}$$

Portanto, segundo as equações (3.3) e (3.4), pode ser visto que existe $i \in I_n$ tal que $P^1_{o_i}(st) \neq P^1_{o_i}(\omega_i) = P^2_{o_i}(\omega_i)$, para todo $\omega_i \in L_N$, o que contradiz a hipótese de que $P^1_{o_i}(st) = P^2_{o_i}(\omega_i)$, para todo $i \in I_n$.

Considere, agora, o caso em que $P^2_{o_i}(st) = P^1_{o_i}(\omega_i)$, para todo $i \in I_n$. O mesmo raciocínio pode ser aplicado para verificar que como $\Sigma^1_{o_i} \subseteq \Sigma^2_{o_i}$, para todo $i \in I_n$, e

Σ^1_{cb} é base para a codiagnose, existe $i \in I_n$ tal que $P^2_{o_i}(st) = P^1_{o_i}(st) \neq P^1_{o_i}(\omega_i)$, o que contradiz a hipótese de que $P^2_{o_i}(st) = P^1_{o_i}(\omega_i)$, para todo $i \in I_n$. □

De acordo com o teorema 3.1, L é robustamente codiagnosticável com relação a duas bases distintas para a codiagnose, Σ^k_{cb} e Σ^l_{cb}, se os conjuntos de observação local satisfazem $\Sigma^k_{o_i} \subseteq \Sigma^l_{o_i}$ para $i = 1, \ldots, n$. Esse resultado mostra que o número de cálculos necessários para verificar a codiagnosticabilidade robusta da linguagem do sistema é significativamente reduzido, dependendo das bases para a codiagnose do sistema. No entanto, quando essa condição não é verdadeira, é necessário obter um método para verificar se L é robustamente codiagnosticável. Apresentamos, agora, um algoritmo para a verificação da codiagnosticabilidade robusta de SED baseado no algoritmo apresentado em [70]. A seguir, é apresentado o algoritmo para a verificação da CRPPO [76], [77]. Note que os passos 1 e 2 do algoritmo 3.1 são iguais aos passos 1 e 2 do algoritmo de verificação proposto em [70].

Algoritmo 3.1. *Verificação da codiagnosticabilidade robusta*

Entrada: *G.*

Saída: *Decisão da codiagnosticabilidade robusta: Sim ou Não.*

- *Passo 1: Calcule o autômato G_N como apresentado em [70].*
- *Passo 2: Calcule o autômato de falha G_F como apresentado em [70].*
- *Passo 3: Calcule o autômato $\bar{G}^{kl}_{N_i}$ eliminando todas as transições de G_N rotuladas com eventos de $\Sigma^l_{o_i} \setminus \Sigma^k_{o_i}$ para $i = 1, \ldots, n$, $k = 1, \ldots, m$, $l = 1, \ldots, m$, $k \neq l$. Calcule $G^{kl}_{N_i} = Ac(\bar{G}^{kl}_{N_i}) = (X^{kl}_{N_i}, \Sigma_N, f^{kl}_{N_i}, x_{0,N})$, e redefina o conjunto de eventos de $G^{kl}_{N_i}$ como $\Sigma^{kl}_{N_i} = \Sigma_N \setminus (\Sigma^l_{o_i} \setminus \Sigma^k_{o_i})$.*
- *Passo 4: Calcule o autômato $\bar{G}^{kl}_F$ eliminando todas as transições de G_F rotuladas com evento de $\bigcup^n_{i=1} \Sigma^k_{o_i} \setminus \Sigma^l_{o_i}$, para $k = 1, \ldots, m$, $l = 1, \ldots, m$, $k \neq l$. calcule $G^{kl}_F = Ac(\bar{G}^{kl}_F)$ e redefina o conjunto de eventos de G^{kl}_F como $\Sigma^{kl}_F = \Sigma \setminus (\bigcup^n_{i=1} \Sigma^k_{o_i} \setminus \Sigma^l_{o_i})$.*

- *Passo 5: Defina a função* $R_i^{kl} : \Sigma_{N_i}^{kl} \to \Sigma_{R_i}^{kl}$ *como*[3]*:*

$$R_i^{kl}(\sigma) = \begin{cases} \sigma, \ se\ \sigma \in \Sigma_{o_i}^k \cap \Sigma_{o_i}^l \\ \sigma_{R_i}, \ se\ \sigma \in \Sigma_{N_i}^{kl} \setminus (\Sigma_{o_i}^k \cap \Sigma_{o_i}^l) \end{cases}. \tag{3.5}$$

  Construa o autômato $G_{R_i}^{kl} = (X_{N_i}^{kl}, \Sigma_{R_i}^{kl}, f_{R_i}^{kl}, x_{0,N})$*, para* $i = 1, \ldots, n$ *e* $k = 1, \ldots, m$*,* $l = 1, \ldots, m$*,* $k \neq l$*, em que* $f_{R_i}^{kl}(x_{N_i}, R_i^{kl}(\sigma)) = f_{N_i}^{kl}(x_{N_i}, \sigma)$ *para todo* $\sigma \in \Sigma_{N_i}^{kl}$ *e* $x_{N_i} \in X_{N_i}^{kl}$*.*

- *Passo 6: Para cada par* (k, l)*,* $k, l = 1, \ldots, m$*,* $k \neq l$*, calcule o autômato verificador* $G_V^{kl} = G_{R_1}^{kl} \| \ldots \| G_{R_n}^{kl} \| G_F^{kl}$*, cujos estados são da forma* $x_{V_{kl}} = (x_{N_1}, x_{N_2}, \ldots, x_{N_n}, x_F)$*, em que* x_{N_i} *e* x_F *são estados de* $G_{R_i}^{kl}$ *e* G_F^{kl}*, respectivamente, e* $x_F = (x, x_\ell)$ *em que* $x \in X$ *e* $x_\ell \in \{N, Y\}$*.*

- *Passo 7: Verifique a existência de um caminho cíclico* $cl := (x_{V_{kl}}^q, \sigma_q, x_{V_{kl}}^{q+1}, ..., x_{V_{kl}}^p, \sigma_p, x_{V_{kl}}^q)$*, em que* $p \geq q > 0$*, em pelo menos um verificador* G_V^{kl} *para* $k, l = 1, \ldots, m$*,* $k \neq l$*, satisfazendo as seguintes condições:*

$$\exists \eta \in I_{qp} \text{ para algum } x_{V_{kl}}^\eta, (x_\ell^\eta = Y) \wedge (\sigma_\eta \in \Sigma), \tag{3.6}$$

  em que $I_{qp} = \{q, q+1, \ldots, p\}$*. Se a resposta for sim, então,* L *não é robustamente codiagnosticável. Caso contrário,* L *é robustamente codiagnosticável.*

Teorema 3.2. *L não é robustamente codiagnosticável com relação às projeções* $P_{o_i}^j$*,* $i = 1, 2, \ldots, n$ *e* $j = 1, 2, \ldots, m$*, e* Σ_f*, se, e somente se, existe um caminho cíclico*

$$cl := (x_{V_{kl}}^q, \sigma_q, x_{V_{kl}}^{q+1}, ..., x_{V_{kl}}^p, \sigma_p, x_{V_{kl}}^q),$$

em que $p \geq q > 0$*, em pelo menos um verificador* G_V^{kl} *para* $k, l = 1, \ldots, m$*,* $k \neq l$*,*

[3] $\Sigma_{R_i}^{kl}$ é o conjunto de eventos obtido a partir de $\Sigma_{N_i}^{kl}$ renomeando seus eventos não observáveis segundo (3.5).

satisfazendo a seguinte condição:

$$\exists \eta \in I_{qp} \text{ para algum } x^{\eta}_{V_{kl}}, (x^{\eta}_{\ell} = Y) \wedge (\sigma_{\eta} \in \Sigma). \tag{3.7}$$

Demonstração. De acordo com a definição 3.4, L é robustamente codiagnosticável se, e somente se, para todos os pares (k, l) de bases para a codiagnose existe $i \in I_n$, tal que $P^l_{o_i}(\omega) \neq P^k_{o_i}(st)$, para todo $\omega \in L_N$. Portanto, para verificar a codiagnosticabilidade robusta de L, é necessário procurar por sequências de falha st, com comprimento arbitrariamente longo após a ocorrência do evento de falha, e sequências normais ω_i, não necessariamente distintas, para $i = 1, \ldots, n$, tal que $P^k_{o_i}(st) = P^l_{o_i}(\omega_i)$ para todo $i \in I_n$. Note que se ω_i tiver um evento de $\Sigma^l_{o_i} \setminus \Sigma^k_{o_i}$, ou st tiver um evento de $\Sigma^k_{o_i} \setminus \Sigma^l_{o_i}$, então, $P^l_{o_i}(\omega_i) \neq P^k_{o_i}(st)$. Portanto, as sequências normais ω_i que possuem um evento de $\Sigma^l_{o_i} \setminus \Sigma^k_{o_i}$ e as sequências de falha st que têm um evento de $\cup^n_{i=1}\Sigma^k_{o_i} \setminus \Sigma^l_{o_i}$ são eliminadas nos passos 3 e 4 do algoritmo 3.1, respectivamente. Para a renomeação das sequências normais e de falha, pode ser visto que:

$$P^k_{o_i}(st) = P^{kl}_{o_i}(st), \tag{3.8}$$

e

$$P^l_{o_i}(\omega_i) = P^{kl}_{o_i}(\omega_i), \tag{3.9}$$

em que $P^{kl}_{o_i} : \Sigma^{\star} \to (\Sigma^k_{o_i} \cap \Sigma^l_{o_i})^{\star}$, o que implica que após a eliminação das sequências normais e de falha realizada nos passos 3 e 4, o problema de verificação da codiagnosticabilidade robusta de L com relação às projeções $P^k_{o_i}$ e $P^l_{o_i}$, para $i = 1, \ldots, n$, e Σ_f, é equivalente ao problema de verificação da codiagnosticabilidade de L, no caso não robusto, com relação a $P^{kl}_{o_i}$, para $i = 1, \ldots, n$, e Σ_f. Portanto, após a eliminação das sequências normais e de falha realizada nos passos 3 e 4, a prova de corretude do algoritmo 3.1 segue os mesmos passos que a do algoritmo apresentado em [70] para a verificação da codiagnosticabilidade no caso não robusto. □

Exemplo 3.2. *Considere, novamente, o autômato G da figura 3.10 e as bases para a codiagnose $\Sigma^1_{cb} = \{\Sigma^1_{o_1}, \Sigma^1_{o_2}\}$ e $\Sigma^2_{cb} = \{\Sigma^2_{o_1}, \Sigma^2_{o_2}\}$, em que $\Sigma^1_{o_1} = \{c, e\}$, $\Sigma^1_{o_2} = \{d, g\}$,*

$\Sigma^2_{o_1} = \{b, e\}$ *e* $\Sigma^2_{o_2} = \{a, d\}$. *De acordo com o algoritmo 3.1, para verificar a codiagnosticabilidade robusta da linguagem gerada por G, os autômatos* G^{12}_V *e* G^{21}_V *devem ser calculados. A seguir, será apresentada a construção do verificador* G^{12}_V.

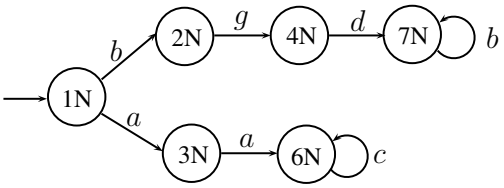


Figura 3.13: Autômato G_N.

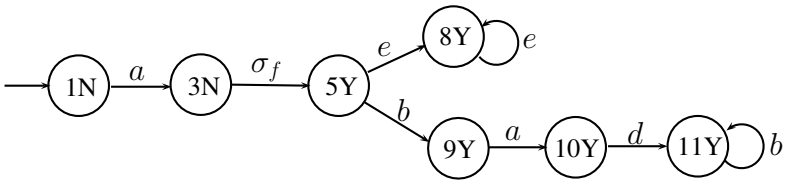


Figura 3.14: Autômato G_F. Neste exemplo $G^{12}_F = G_F$.

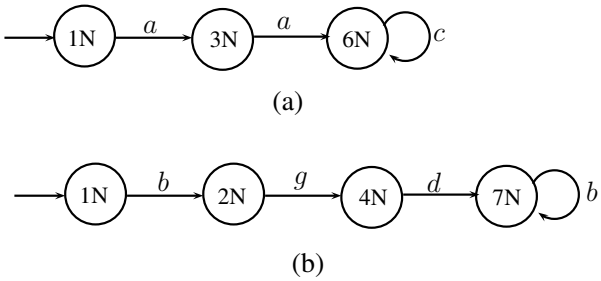


Figura 3.15: Autômatos $G^{12}_{N_1}$ (a) e $G^{12}_{N_2}$ (b).

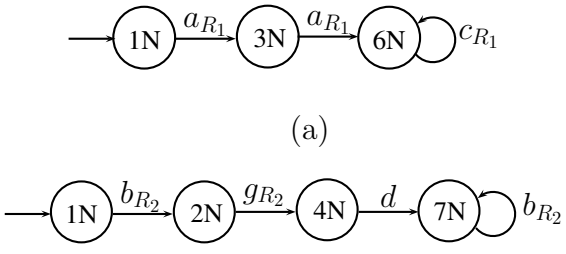


Figura 3.16: Autômatos $G^{12}_{R_1}$ (a) e $G^{12}_{R_2}$ (b).

No passo 1 do algoritmo 3.1, o autômato que modela o comportamento normal do sistema G_N, apresentado na figura 3.13, é calculado, e no passo 2, o autômato de falha G_F, mostrado na figura 3.14, é calculado. No passo 3, os autômatos $G^{12}_{N_1}$ e $G^{12}_{N_2}$ são calculados a partir de G_N eliminando as transições de G_N rotuladas com $\Sigma^2_{o_1} \setminus \Sigma^1_{o_1} = \{b\}$ e $\Sigma^2_{o_2} \setminus \Sigma^1_{o_2} = \{a\}$, respectivamente, e tomando suas partes acessíveis. O resultado obtido nesse passo é mostrado na figura 3.15. No passo 4, o autômato G^{12}_F é calculado a partir de G_F eliminando as transições rotuladas com $\bigcup^2_{i=1} \Sigma^1_{o_i} \setminus \Sigma^2_{o_i} = \{c, g\}$ e tomando sua parte accessível. Como G_F não possui transições rotuladas com c ou g, então, $G^{12}_F = G_F$. No passo 5, os autômatos $G^{12}_{R_1}$ e $G^{12}_{R_2}$ são mostrados na figura 3.16, obtidas após a renomeação dos eventos não observáveis de $G^{12}_{N_1}$ e $G^{12}_{N_2}$, respectivamente. Finalmente, no passo 6, o autômato verificador $G^{12}_V = G^{12}_{R_1} \| G^{12}_{R_2} \| G^{12}_F$ é calculado e a procura por caminhos cíclicos que violam a condição de codiagnosticabilidade robusta pode ser realizada. Se existir um caminho cíclico que satisfaz a condição (3.7) então, L não é robustamente codiagnosticável. Na figura 3.17, é apresentado o autômato verificador G^{12}_V, no qual pode-se observar a existência de três caminhos cíclicos, evidenciados na figura 3.18, $cl_1 = (\{1N, 7N, 11Y\}, b, \{1N, 7N, 11Y\})$, $cl_2 = (\{3N, 7N, 11Y\}, b, \{3N, 7N, 11Y\})$ e $cl_3 = (\{6N, 7N, 11Y\}, b, \{6N, 7N, 11Y\})$ que violam a condição (3.7) e, portanto, L não é robustamente codiagnosticável com relação às projeções $P^j_{o_i}$, $i = 1, 2$ e $j = 1, 2$, e Σ_f. Como G^{12}_V possui caminhos cíclicos que violam a codiagnosticabilidade robusta de L, então não é necessário calcular G^{21}_V. Contudo, caso G^{12}_V não apresentasse caminhos cíclicos que satisfazem a condição (3.7), seria necessária a busca por esses caminhos em G^{21}_V. □

3.2.5 Complexidade computacional

Uma vez que todos os cálculos computacionais requeridos no algoritmo 3.1 podem ser feitos em tempo linear no número de estados e de transições dos autômatos construídos no algoritmo, sua complexidade computacional foi analisada em termos do tamanho dos autômatos. Note que, nos passos 1 e 2 do algoritmo 3.1, G_N e G_F são

construídos com uma complexidade linear no número de estados e de transições de G. Nos passos 3, 4 e 5, para um dado par (k, l) de bases para a codiagnose, os autômatos $G_{N_i}^{kl}$, G_F^{kl} e $G_{R_i}^{kl}$, são, também, construídos com uma complexidade linear no número de estados e transições de G. No passo 6, o verificador G_V^{kl} deve ser calculado para cada par (k, l) das bases para a codiagnose. Pode ser observado que cada verificador G_V^{kl} pode ser obtido com a mesma complexidade do verificador G_V construído para o caso em que L não é robustamente codiagnosticável, isto é, o cálculo computacional de G_V^{kl} tem complexidade $O(n \times |X|^{n+1} \times |\Sigma|)$ [70], [71]. Finalmente, a busca por caminhos cíclicos que violam a condição (3.7) pode ser realizada procurando por componentes fortemente conexos em G_V^{kl}, cuja complexidade é linear no tamanho de G_V^{kl}. Assim, como, no pior caso, todas as bases para a codiagnose devem ser verificadas usando o algoritmo 3.1, $m(m-1)$ verificadores G_V^{kl} devem ser construídos e, portanto, a complexidade de todo o algoritmo 3.1 é $O(m^2 \times n \times |X|^{n+1} \times |\Sigma|)$, ou seja, polinomial no número de estados e transições, considerando o número de diagnosticadores locais n limitado, e exponencial em relação a n.

3.3 Cálculo do limite de atraso para a codiagnose robusta a perdas permanentes de observação

Nesta seção, é apresentado um algoritmo em tempo polinomial para o cálculo do atraso da codiagnose robusta z^*, isto é, o número máximo de eventos que podem ocorrer após a ocorrência do evento de falha σ_f, até que a falha possa ser detectada e nesse caso, L é considerada robustamente codiagnosticável a perdas permanentes de observação.

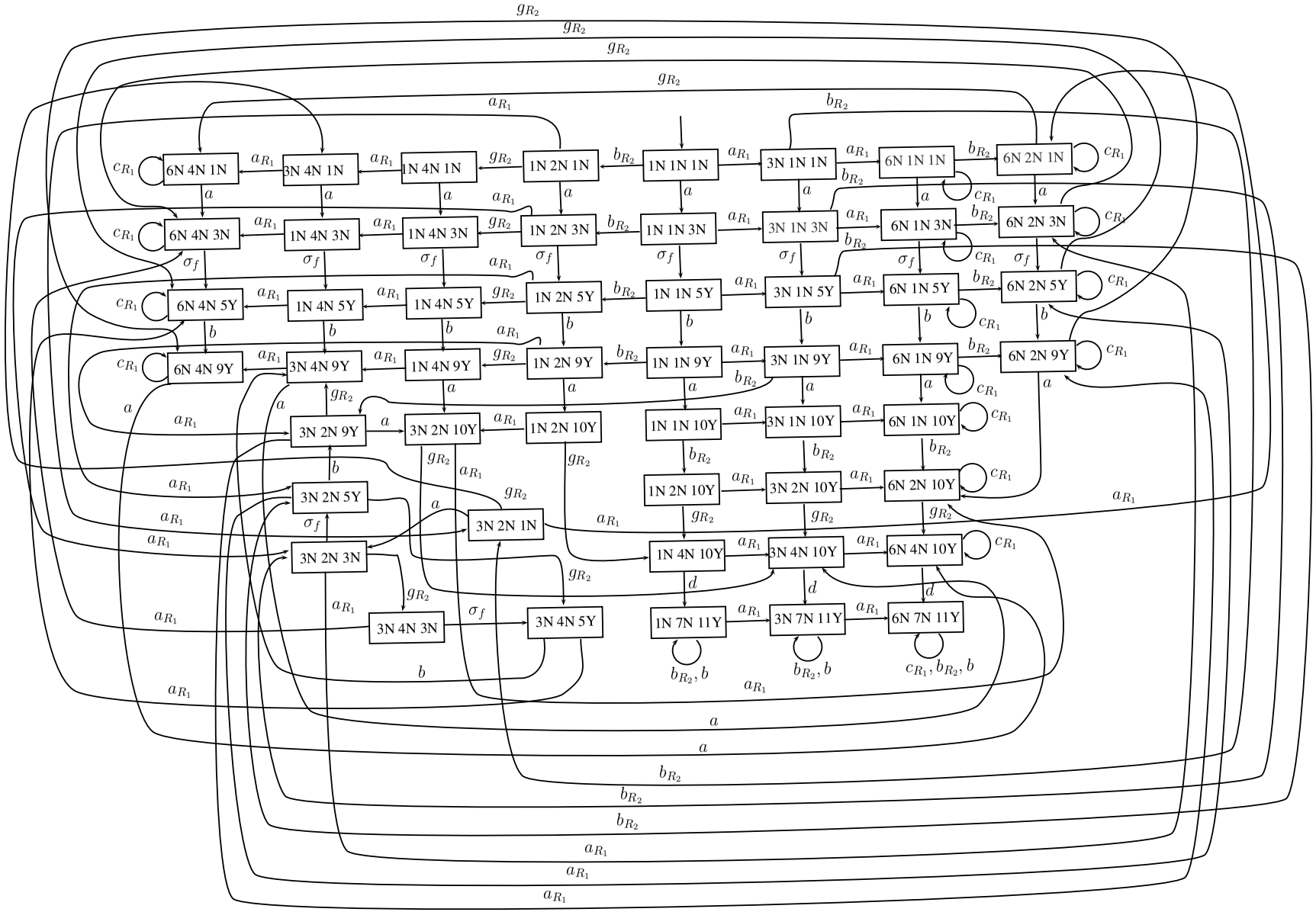


Figura 3.17: Autômato verificador $G_V^{12} = G_{R_1}^{12} \| G_{R_2}^{12} \| G_F^{12}$.

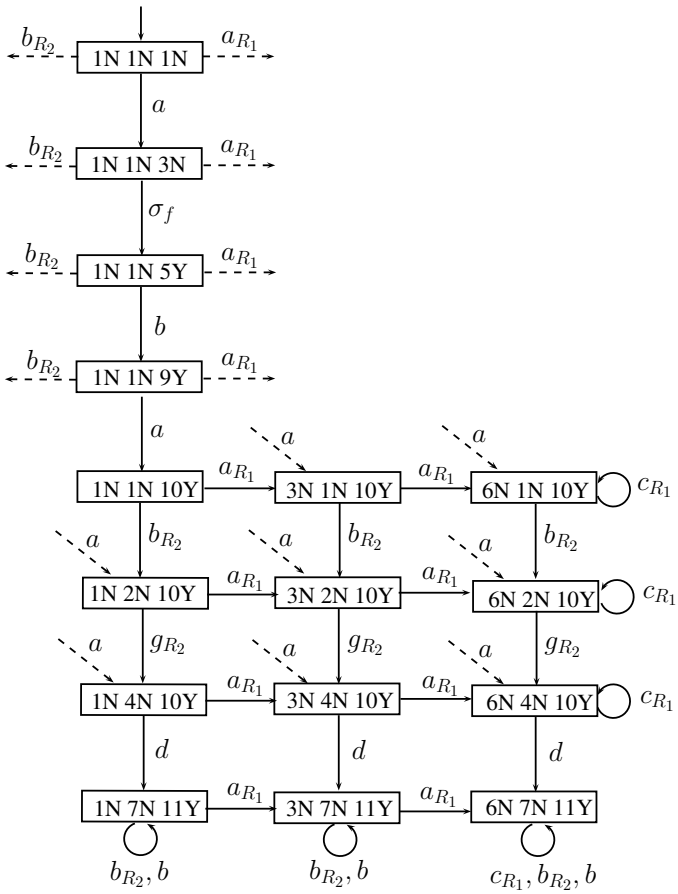


Figura 3.18: Parte do autômato G_V^{12}.

Para calcular o atraso z^*, é necessário calcular, primeiro, o número máximo de eventos associado a um par de bases (k, l) que o sistema pode executar após a ocorrência do evento de falha σ_f, para o qual existe uma sequência de falha st e sequências normais ω_i com as mesmas observações para todo $i \in I_n$:

$$\begin{aligned} d(k,l) &= \max\{\|t\| : (s \in L \setminus L_N)(st \in L \setminus L_N) \\ &\qquad (\forall i \in I_n, P_{o_i}^k(st) = P_{o_i}^l(\omega_i), \omega_i \in L_N)\}. \end{aligned}$$

Note que para calcular $d(k, l)$, é necessário procurar pela sequência $st \in L \setminus L_N$ e pelas sequências $\omega_i \in L_N$, para $i \in I_n$, tal que $P_{o_i}^k(st) = P_{o_i}^l(\omega_i)$, e que a sequência t tenha o maior comprimento. Como, de acordo com o teorema 2.3, G_V^{kl} representa somente as sequências de falha e normais que tenham as mesmas projeções $P_{o_i}^k$ e $P_{o_i}^l$,

para $i = 1, \ldots, n$, o cálculo computacional de $d(k, l)$ pode ser realizado procurando pelo caminho em G_V^{kl} associado à sequência em $\Sigma^\star$ com o maior comprimento após a ocorrência de σ_f.

Antes de apresentar um algoritmo para calcular $d(k, l)$, será apresentado a seguir o algoritmo 3.2 para calcular o comprimento do maior caminho em um grafo acíclico orientado (GAO) [78][4] É importante notar que no primeiro passo do algoritmo, uma ordenação topológica do GAO é realizada. O algoritmo 2.5 de ordenação topológica retorna a lista encadeada dos vértices de um GAO G, tal que se G tem uma aresta (u, v), então u aparece antes de v na lista [73, 78].

Algoritmo 3.2. *Cálculo do comprimento do maior caminho em um grafo acíclico orientado G.*

Entrada: *Grafo acíclico orientado $G = (V, E)$, em que V e E são os conjuntos de vértices e arestas de G.*

Saída: *Comprimento de maior caminho em G, l.*

Faça a ordenação topológica de G:

- *Passo 1: $(v_1, v_2, \ldots, v_\eta)$ = OT(G), em que $v_j \in V$, para $j \in I_\eta$, e $\eta = |V|$.*
- *Passo 2: Para $j = 1, \ldots, \eta$:*

$$l(v_j) = 1 + \max\{l(v_i) : (v_i, v_j) \in E\}.$$

- *Passo 3: $l = \max_{j \in I_\eta} l(v_j) - 1$.*

É importante observar que o algoritmo 3.2 pode ser usado somente se o grafo for acíclico. Note, também, que embora a linguagem do sistema seja considerada

[4]Neste trabalho, o comprimento de um caminho é considerado como o número de transições existentes nesse caminho.

robustamente codiagnosticável, e, de acordo com o teorema 3.2, todos os verificadores G_V^{kl}, para $k, l \in I_m$, não tem caminhos cíclicos com um dos eventos no caminho pertencente a Σ, um verificador G_V^{kl} pode ter caminhos cíclicos com todos os eventos renomeados. Assim, para usar o algoritmo 3.2 para calcular $d(k, l)$, é necessário, primeiro, eliminar os caminhos cíclicos de G_V^{kl}. A eliminação de todos os caminhos cíclicos de G_V^{kl} pode ser feita seguindo os passos do algoritmo 3.3. É importante observar que a saída do algoritmo 3.3 é um grafo acíclico não determinístico.

Algoritmo 3.3. *Eliminação de todos os caminhos cíclicos de* G_V^{kl}

Entrada: $G_V^{kl} = (X_V^{kl}, \Sigma_V^{kl}, f_V^{kl}, x_{0,V}^{kl})$, *em que* $\Sigma_V^{kl} = \Sigma_{R_1}^{kl} \cup \ldots \cup \Sigma_{R_n}^{kl} \cup \Sigma_F^{kl}$

Saída: $G_{nd}^{kl} = (X_{nd}^{kl}, \Sigma_F^{kl}, f_{nd}^{kl}, x_{0,nd}^{kl})$

- *Passo 1:* $x_{0,nd}^{kl} = UR(x_{0,V}^{kl}, \Sigma_F^{kl})$. $X_{nd}^{kl} = \{x_{0,nd}^{kl}\}$.
- *Passo 2: Para cada estado* x_V *de* G_V^{kl} *alcançado por um evento* $\sigma \in \Sigma_F^{kl}$*:*
  - *Passo 2.1: Calcule* $x_{nd} = UR(x_V, \Sigma_F^{kl})$.
  - *Passo 2.2:* $X_{nd}^{kl} = X_{nd}^{kl} \cup \{x_{nd}\}$.
- *Passo 3: Para cada* $x_{nd} \in X_{nd}^{kl}$ *e* $\sigma \in \Sigma_F^{kl}$*, defina a função não determinística* $f_{nd}^{kl}(x_{nd}, \sigma)$ *como a seguir:*
  - *Passo 3.1: Calcule*

$$X_V^e = \{x_V^e \in X_V^{kl} : (\exists x_V \in x_{nd})[f_V^{kl}(x_V, \sigma) = x_V^e]\}.$$

  - *Passo 3.2: Calcule* $x_{nd}^e = \{UR(x_V^e, \Sigma_F^{kl}) : x_V^e \in X_V^e\}$.
  - *Passo 3.3: Defina* $f_{nd}^{kl}(x_{nd}, \sigma) = X_{nd}^e$

Observe que os estados de G_{nd}^{kl} são formados pelos estados de G_V^{kl}, e os eventos de G_{nd}^{kl} são eventos não renomeados da parte de falha G_F^{kl}. Assim, σ_f pertence ao conjunto de eventos de G_{nd}^{kl}. É importante notar também que, como os eventos

renomeados pertencem aos autômatos $G^{kl}_{R_i}$, que modelam o comportamento normal de G, então, se um componente do estado $x_{nd} \in X^{kl}_{nd}$ tem o rótulo Y, então todos os componentes de x_{nd} também têm o rótulo Y.

O teorema a seguir mostra que a linguagem gerada por G^{kl}_{nd} é igual à projeção da linguagem gerada por G^{kl}_V, considerando Σ^{kl}_F como conjunto de eventos observáveis, e, portanto, a eliminação das sequências de eventos renomeados para obtenção de G^{kl}_{nd} não altera a linguagem de falha do sistema representada em G^{kl}_V.

Teorema 3.3. $L(G^{kl}_{nd}) = L(Obs(G^{kl}_V, \Sigma^{kl}_F))$.

Demonstração. Para provar que $L(G^{kl}_{nd}) = L(Obs(G^{kl}_V, \Sigma^{kl}_F))$, primeiramente, será provado que $L(G^{kl}_{nd}) \subseteq L(Obs(G^{kl}_V, \Sigma^{kl}_F))$. Seja $s = \sigma_1\sigma_2 \ldots \sigma_r \in L(G^{kl}_{nd})$, logo existe um caminho p tal que $p = (x^1_{nd}, \sigma_1, x^2_{nd}, \sigma_2, \ldots, \sigma_r, x^{r+1}_{nd})$, em que $x^1_{nd} = x_{0,nd}$. De acordo com o algoritmo 3.3, $f_{nd}(x^i_{nd}, \sigma_i) = x^{i+1}_{nd}$ quando existem $x_V \in x^i_{nd}$ e $x^e_V \in x^{i+1}_{nd}$ tais que $f_V(x_V, \sigma_i) = x^e_V$. Além disso, como $\sigma_i \in \Sigma^{kl}_F$, então, de acordo com o passo 3, $x^{i+1}_{nd} = UR(x^e_V, \Sigma^{kl}_F)$. Portanto, associado a s, existe uma sequencia $s' \in L(G^{kl}_V)$, em que $s = P^{kl}_F(s')$, sendo $P^{kl}_F : \Sigma^{kl^\star}_V \to \Sigma^{kl^\star}_F$. Portanto, $L(G^{kl}_{nd}) \subseteq P^{kl}_F(L(G^{kl}_V)) = L(Obs(G^{kl}_V, \Sigma^{kl}_F))$.

Seja $s' \in L(Obs(G^{kl}_V, \Sigma^{kl}_F))$. Logo, existe uma sequência $s'' \in L(G^{kl}_V)$ tal que $P^{kl}_F(s'') = s'$. Associado a s'' tem-se o caminho $p'' = (x^1_V, \sigma_1, x^2_V, \sigma_2, \ldots, \sigma_r, x^{r+1}_V)$ em G^{kl}_V, em que $x^1_V = x^{kl}_{0,V}$. O primeiro passo para a construção de G^{kl}_{nd} é a formação do seu estado inicial $x^{kl}_{0,nd} = UR(x^{kl}_{0,V}, \Sigma^{kl}_F)$. Portanto, $x^1_V \in x^{kl}_{0,nd}$. Suponha, sem perda de generalidade, que σ_1 seja um evento renomeado. Então, $x^2_V \in x^{kl}_{0,nd}$. Suponha agora que $\sigma_2 \in \Sigma^{kl}_F$. Nesse caso, de acordo com o algoritmo 3.3, haverá um estado x^{kl}_{nd} tal que $x^3_V \in x^{kl}_{nd}$. Continuando dessa forma conclui-se que haverá um caminho p em G^{kl}_{nd}, associado ao caminho p'' em G^{kl}_V, tal que a sequência associada a p é s'. Portanto, $L(Obs(G^{kl}_V, \Sigma^{kl}_F)) \subseteq L(G^{kl}_{nd})$. □

Algoritmo 3.4. *Cálculo de* $d(k, l)$

Entrada: G_{nd}^{kl}

Saída: $d(k,l)$

- *Passo 1: Marque todos os estados de G_{nd}^{kl} cujos componentes são rotulados com Y.*
- *Passo 2: Gere o autômato $\bar{G}_{nd}^{kl}$ eliminando de G_{nd}^{kl} todos os estados que não são marcados assim como suas transições.*
- *Passo 3: Calcule o comprimento $d(k,l)$ do maior caminho de $\bar{G}_{nd}^{kl}$, usando o algoritmo 3.2.*

Nos passos 1 e 2 do algoritmo 3.4, o autômato $\bar{G}_{nd}^{kl}$ é calculado a partir do autômato G_{nd}^{kl}, eliminando todos os estados não alcançados antes da ocorrência do evento σ_f, assim como suas transições. Assim, apenas os estados de G_{nd}^{kl} que são alcançados após σ_f são estados de $\bar{G}_{nd}^{kl}$. Depois disso, o comprimento máximo de um caminho em $\bar{G}_{nd}^{kl}$ é calculado. Note que todas as sequências em $\bar{G}_{nd}^{kl}$ são sequências $t \in \Sigma^\star$ na pós-linguagem de L após uma sequência de falha $s \in L \setminus L_N$. O valor de $d(k,l)$ é, por tanto, o comprimento da maior sequência t para a qual existem sequências normais $\omega_i \in L_N$, $i = 1, \ldots, n$, em que $P_{o_i}^k(st) = P_{o_i}^l(\omega_i)$, para todo $i \in I_n$, violando, assim, a definição 3.4.

Após o cálculo de $d(k,l)$, o atraso para a codiagnose robusta pode ser calculado como $z^* = d + 1$, em que

$$d = \max\{d(k,l) : k,l \in I_m\}. \tag{3.10}$$

O algoritmo 3.5 resume os passos para o cálculo de z^*.

Algoritmo 3.5. *Cálculo do atraso para a codiagnose robusta.*

Entrada: G_V^{kl}, *para* $k,l \in I_m$.

Saída: z^*.

- *Passo 1: Calcule* G_{nd}^{kl} *a partir de* G_V^{kl} *seguindo os passos do algoritmo 3.3 para* $k, l \in I_m$.

- *Passo 2: Calcule* $d(k,l)$ *de acordo com o algoritmo 3.4 para* $k, l \in I_m$.

- *Passo 3: Calcule o atraso para a codiagnose robusta* $z^* = d + 1$, *em que*

$$d = \max\{d(k,l) : k, l \in I_m\}.$$

Exemplo 3.3. *Considere, novamente, a planta mostrada na figura 3.10, e suponha que existam dois locais de diagnose com conjuntos de eventos potencialmente observáveis* $\Sigma_{o_1} = \{b, c, e, g\}$ *e* $\Sigma_{o_2} = \{a, d, g\}$. *Sejam* $\Sigma_{cb}^1 = \{\Sigma_{o_1}^1, \Sigma_{o_2}^1\}$ *e* $\Sigma_{cb}^2 = \{\Sigma_{o_1}^2, \Sigma_{o_2}^2\}$ *duas bases para a codiagnose, em que* $\Sigma_{o_1}^1 = \{b, c, e\}$, $\Sigma_{o_2}^1 = \{d, g\}$, $\Sigma_{o_1}^2 = \{b, e, g\}$, *e* $\Sigma_{o_2}^2 = \{a, d\}$. *Assim, na primeira base para a codiagnose* Σ_{cb}^1, *considera-se que a comunicação do evento g ao primeiro local de diagnose, e a comunicação do evento a para o segundo local de diagnose foram perdidas, e, na segunda base para a codiagnose* Σ_{cb}^2, *é suposto que a comunicação do evento c para o primeiro local de diagnose, e a comunicação do evento g para o segundo local de diagnose foram perdidas.*

Seguindo os passos do algoritmo 3.1, pode ser mostrado que L é robustamente codiagnosticável com relação a $P_{o_i}^j$, $i = 1, 2$ *e* $j = 1, 2$, *e* Σ_f, *pois nos verificadores* G_V^{11}, G_V^{22}, G_V^{12} *e* G_V^{21} *construídos com base no algoritmo 3.1 e ilustrados nas figuras 3.19, 3.21, 3.22 e 3.23, respectivamente, não foi encontrado nenhum caminho cíclico que viole a condição (3.7). Assim, podemos calcular o atraso para a codiagnose robusta* z^* *usando o algoritmo 3.5, que resulta em* $z^* = 3$ *no verificador* G_V^{11}, *pois este possui a maior sequência após a ocorrência do evento de falha. A partir da figura 3.19 do verificador* G_V^{11} *pode-se observar que um estado de bloqueio é alcançado após a ocorrência de dois eventos em* Σ *após a ocorrência de* σ_f, *isto é, as sequências de falha são distinguidas das sequências normais após a ocorrência de três eventos após a ocorrência de* σ_f. *O caminho, evidenciado na figura 3.20, desde o estado inicial, que contém o maior atraso*

é $(\{1N,1N,1N\},a,\ \{1N,1N,3N\},\sigma_f,\ \{1N,1N,5Y\},a_{R_2},\ \{1N,3N,5Y\},a_{R_2},$ $\{1N,6N,5Y\},b,\ \{2N,6N,9Y\},a,\ \{2N,6N,10Y\},g_{R_1},\ \{4N,6N,10Y\},d_{R_1},$ $\{7N,6N,10Y\},c_{R_2},\ \{7N,6N,10Y\})$, *e a sequência de eventos nesse caminho é* $a\sigma_f a_{R_2}a_{R_2}bag_{R_1}d_{R_1}c_{R_2}^n$ *em que a sequência* t *de maior comprimento é* $t = ba$ *e portanto,* $d(1,1) = 2$. *Isso quer dizer que o atraso máximo para a diagnose de falha no sistema será* $z^* = 3$, *ou seja, três eventos deverão ser observados após* σ_f, *no pior caso, para diagnosticar a falha* σ_f. □

A complexidade computacional do algoritmo 3.5 é polinomial no tamanho da planta G *e no número de bases para a codiagnose. Para tanto, vamos calcular a complexidade de cada passo do algoritmo 3.5.*

No primeiro passo, os autômatos G_{nd}^{kl} *são calculados, de acordo com o algoritmo 3.3, eliminando-se todos os caminhos cíclicos dos autômatos* G_V^{kl}, *para* $k,l \in I_m$. *O número de estados de* G_{nd}^{kl} *é, no pior caso, igual ao número de estados de* G_V^{kl}, *e como* G_{nd}^{kl} *é um autômato não determinístico, o número de transições de* G_{nd}^{kl} *é, no pior caso, igual a* $|X_V^{kl}|^2 \times \Sigma_F^{kl}$. *Como o número de estados de* G_V^{kl} *está limitado por* $2|X|^{n+1}$ *e* $\Sigma_F^{kl} \subseteq \Sigma$, *então, o cálculo computacional de* G_{nd}^{kl}, *para* $k,l \in I_m$, *tem complexidade* $O(m^2 \times |X|^{2(n+1)} \times \Sigma)$.

No segundo passo do algoritmo 3.5, $d(k,l)$ *é calculado seguindo os passos do algoritmo 3.4, para* $k,l \in I_m$. *Para tanto, o autômato* $\bar{G}_{nd}^{kl}$ *é formado eliminando-se os estados de* G_{nd}^{kl} *que não são rotulados com* Y. *Essa tarefa pode ser realizada em tempo linear com o tamanho de* G_{nd}^{kl}. *No passo 3 do algoritmo 3.4,* $d(k,l)$ *é calculado usando o algoritmo 3.2. O cálculo do comprimento do maior caminho de um GAO pode ser feito, também, em tempo linear com o tamanho do GAO [78]. Assim, a complexidade de todo o algoritmo 3.5 é* $O(m^2 \times |X|^{2(n+1)} \times \Sigma)$, *ou seja, polinomial no número de estados e transições, considerando o número de diagnosticadores locais* n *limitado, e exponencial em relação a* n.

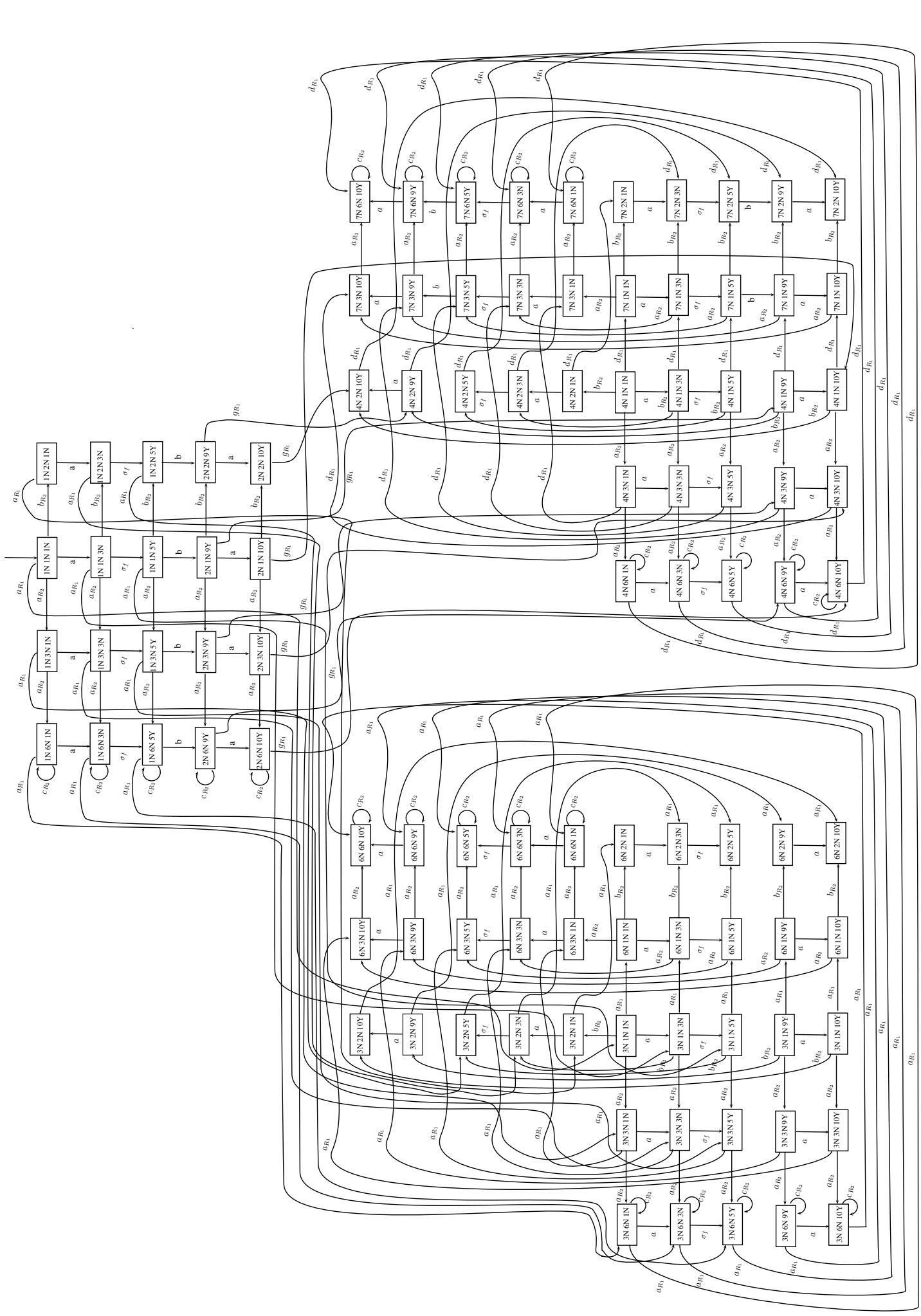


Figura 3.19: Autômato G_V^{11}.

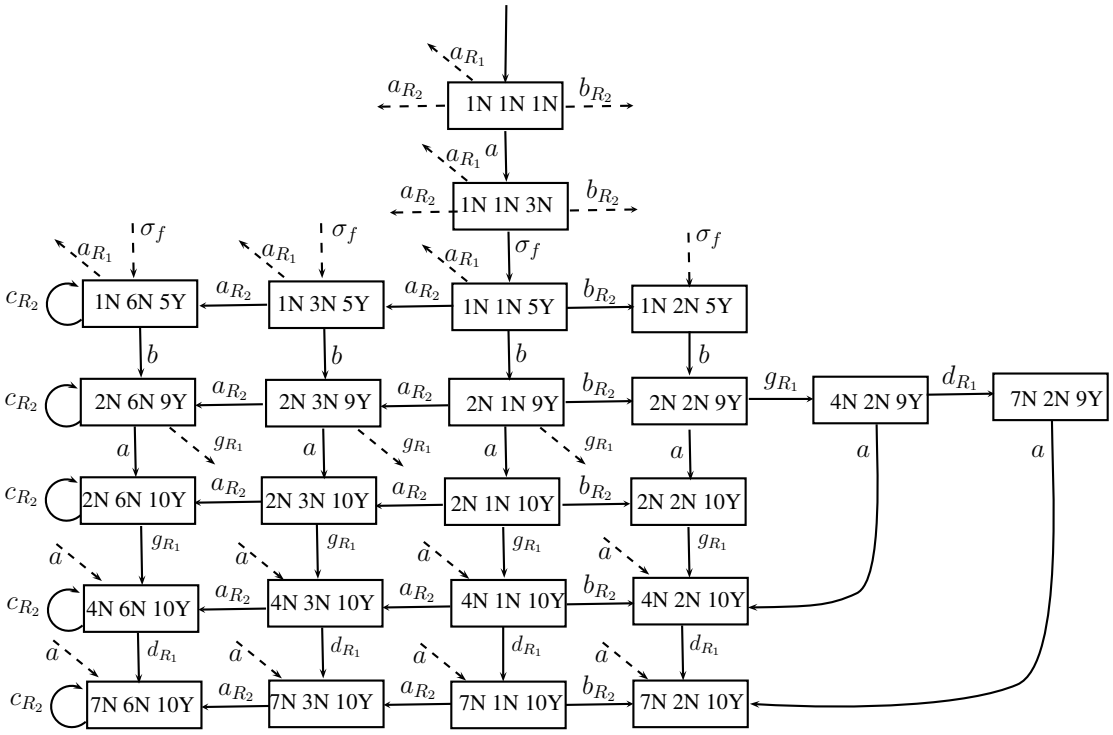


Figura 3.20: Parte do autômato G_V^{11}.

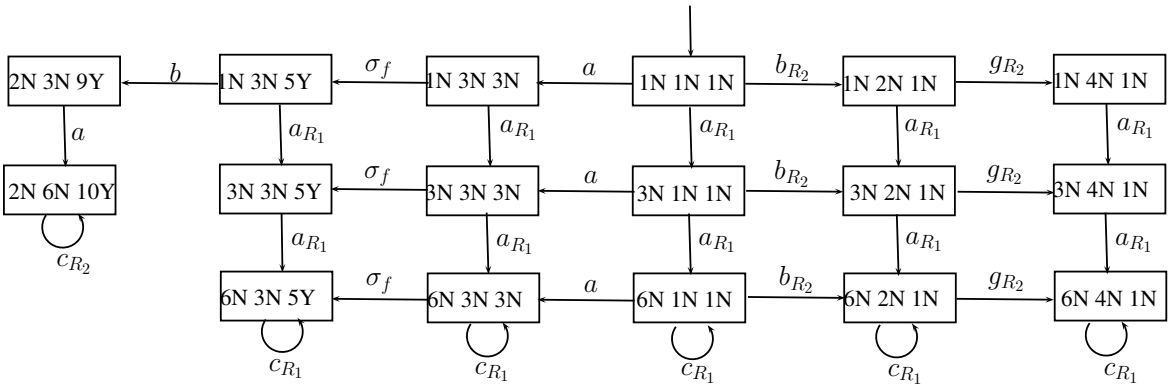


Figura 3.21: Autômato G_V^{22}.

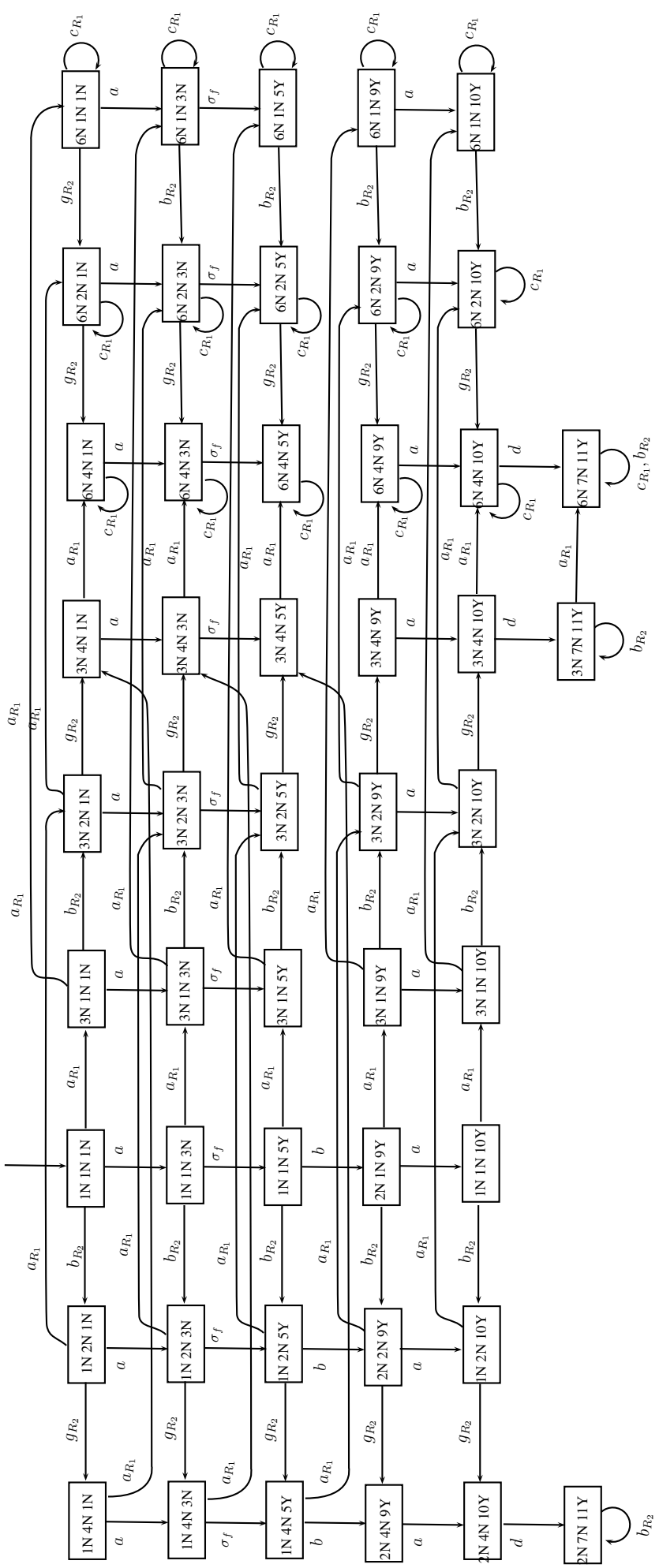


Figura 3.22: Autômato G_V^{12}.

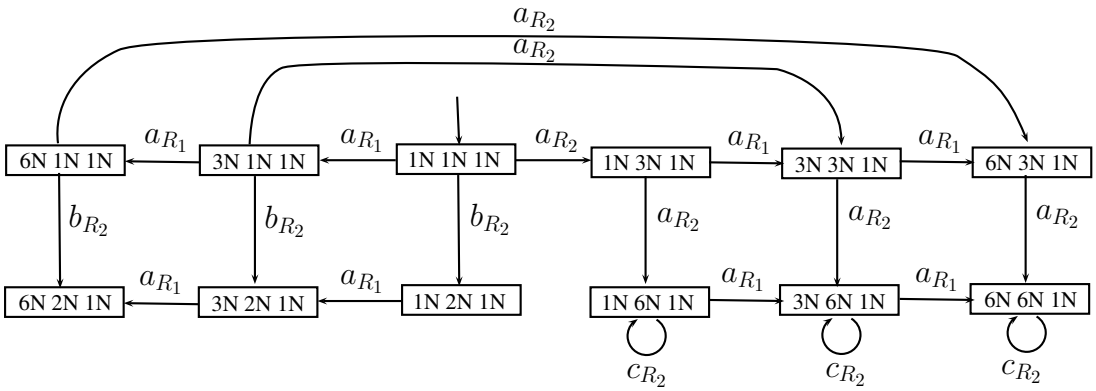


Figura 3.23: Autômato G_V^{21}.

3.4 Comentários finais

Neste capítulo foi resolvido o problema de CRPPO fazendo a extensão do problema de DRPPO. Para isso, foi feita a extensão de base para a diagnose para base para a codiagnose e um esquema de codiagnose robusta a perdas permanentes de observações foi apresentado. Com a definição de base para a codiagnose foi possível introduzir a definição de CRPPO a partir da qual foi apresentado o teorema de codiagnosticabilidade robusta baseado no verificador em [70]. Para a construção desse verificador, foi desenvolvido um algoritmo de verificação baseado no em [70] com algumas modificações que decorrem do fato de eliminar todas as transições de G_N rotuladas com eventos de $\Sigma_{o_i}^l \setminus \Sigma_{o_i}^k$ para $i = 1, \ldots, n$, $k = 1, \ldots, m$, $l = 1, \ldots, m$, $k \neq l$ e todas as transições de G_F rotuladas com evento de $\bigcup_{i=1}^n \Sigma_{o_i}^k \setminus \Sigma_{o_i}^l$, para $k = 1, \ldots, m$, $l = 1, \ldots, m$, $k \neq l$. A complexidade computacional do algoritmo permaneceu, como em [70], polinomial no número de estados e de transições do autômato G, mantendo os demais parâmetros constantes. Finalmente, foi apresentado um algoritmo em tempo polinomial, mantendo os demais parâmetros constantes, que calcula o limite de atraso para a CRPPO.

Capítulo 4

Codiagnosticabilidade robusta a perdas intermitentes de observação de eventos

Neste capítulo uma nova definição de CRPIO, diferente da apresentada em [63], é proposta. Essa nova definição permite considerar perdas de observabilidade devido a falhas na comunicação da ocorrência de um evento para um diagnosticador local, e permite considerar o caso em que há incerteza com relação a quais eventos podem sofrer perdas intermitentes de observação.

Este capítulo está organizado da seguinte forma. Na seção 4.1, são apresentadas as definições de diagnosticabilidade robusta a perdas intermitentes de observação e de CRPIO apresentadas em [63]. Na seção 4.2, a definição de CRPIO proposta neste trabalho é apresentada. Além disso, são apresentadas também as definições de bases e bases mínimas para a codiagnose robusta a perdas intermitentes de observação. Na seção 4.3, um algoritmo em tempo polinomial para a verificação da CRPIO é apresentad, na seção 4.4, são apresentados comentários sobre o cálculo do limite de atraso para a CRPIO. Finalmente, na seção 4.5, são apresentadas conclusões sobre este capítulo.

4.1 Definição de codiagnosticabilidade robusta a perdas intermitentes de observação apresentada em Carvalho et al. (2012).

Em [63] uma definição de diagnosticabilidade robusta a perdas intermitentes de observação de eventos (DRPI) é apresentada. De acordo com essa definição, a linguagem do sistema L é dita ser robustamente diagnosticável a perdas intermitentes de observação quando existe um subconjunto do conjunto de eventos observáveis formado por eventos passíveis de sofrerem perdas intermitentes de observação, isto é, o conjunto de eventos pode ser particionado como $\Sigma = \Sigma_{ilo} \dot{\cup} \Sigma_{nilo} \dot{\cup} \Sigma_{uo}$, em que Σ_{ilo} e Σ_{nilo} denotam, respectivamente, os conjuntos de eventos observáveis que podem perder intermitentemente a observação e que não perdem a observação dos eventos, e Σ_{uo} é o conjunto de eventos não-observáveis.

Visando resolver o problema de DRPI, em [63] é apresentada a operação de dilatação de uma sequência de eventos mostrada a seguir.

Definição 4.1. *Seja $\Sigma = \Sigma_{ilo} \dot{\cup} \Sigma_{nilo} \dot{\cup} \Sigma_{uo}$ uma partição de Σ, em que em que Σ_{ilo} e Σ_{nilo} denotam, respectivamente, os conjuntos de eventos observáveis que podem perder intermitentemente a observação e que não perdem a observação, e Σ_{uo} é o conjunto de eventos não-observáveis. Seja $\Sigma'_{ilo} = \{\sigma' : \sigma \in \Sigma_{ilo}\}$ e $\Sigma_{dil} = \Sigma \cup \Sigma'_{ilo}$. A dilatação D é o mapeamento*

$$D : \Sigma^\star \to 2^{\Sigma^\star_{dil}}$$

em que

$$\begin{aligned} &D(\epsilon) = \epsilon, \\ &D(\sigma) = \begin{cases} \sigma, \ \textit{se } \sigma \in \Sigma \setminus \Sigma_{ilo} \\ \{\sigma, \sigma'\}, \ \textit{se } \sigma \in \Sigma_{ilo} \end{cases}, \\ &D(s\sigma) = D(s)D(\sigma) \ \textit{para todo } s \in \Sigma^\star \ \textit{e } \sigma \in \Sigma. \end{aligned}$$

A operação de dilatação pode ser estendida para uma linguagem L como:

$$D(L) = \bigcup_{s \in L} D(s).$$

Note que a operação de dilatação associa a cada evento $\sigma \in \Sigma_{ilo}$ um evento $\sigma' \in \Sigma'_{ilo}$, cuja interpretação é a de que o evento $\sigma \in \Sigma$ associado a σ' ocorreu, mas a sua observação foi perdida, ou seja, σ' é um evento não-observável de Σ_{dil}.

A linguagem do sistema após dilatação é denotada por $L_{dil} = D(L)$, e em [63] é apresentado um algoritmo para obtenção de um autômato G_{dil} que gera a linguagem L_{dil} a partir do autômato do sistema G. Esse algoritmo é mostrado a seguir.

Algoritmo 4.1. *Obtenção de G_{dil}*

Entradas: $G = (X, \Sigma, f, \Gamma, x_0)$ *e* Σ_{ilo}.

Saída: $G_{dil} = (X, \Sigma_{dil}, f_{dil}, \Gamma_{dil}, x_0)$.

1: Defina $\Sigma_{dil} = \Sigma \cup \Sigma'_{ilo}$.

2: Faça $f_{dil}(x, \sigma) = f(x, \sigma)$, *para todo* $x \in X$ *e* $\sigma \in \Sigma$, *e* $f_{dil}(x, \sigma') = f(x, \sigma)$, *se* $\sigma \in \Sigma_{ilo} \cap \Gamma(x)$.

Exemplo 4.1. *Considere o autômato da figura 3.6(a) reapresentado na figura 4.1(a) em que* $\Sigma = \{a, b, c, \sigma_f\}$, $\Sigma_o = \{a, b, c\}$ *e* $\Sigma_{uo} = \Sigma_f = \{\sigma_f\}$. *A linguagem L gerada pelo autômato G pode ser expressa como* $L = \overline{\{a\}\{c\}\{b\}^\star \cup \{\sigma_f\}\{a\}\{b\}\{c\}^\star}$. *Admitindo a possibilidade de perdida intermitente do evento b, tem-se que* $\Sigma_{ilo} = \{b\}$, *e* $\Sigma_{nilo} = \{a, c\}$. *Devido à intermitência na observação do evento b, a linguagem L será dilatada usando a função de dilatação* $D : \Sigma^\star \to 2^{\Sigma^\star_{dil}}$. *Assim,* $D(b) = \{b, b'\}$, $\Sigma'_{ilo} = \{b'\}$, *e* $\Sigma_{dil} = \Sigma \cup \Sigma'_{ilo} = \{a, b, b', c\}$. *Portanto, a nova linguagem L_{dil} obtida a partir da dilatação $D(L)$, é expressa como* $L_{dil} = L(G) \cup \overline{\{\sigma_f\}\{a\}\{b'\}\{c\}^\star}$ *cujo autômato G_{dil} é mostrado na figura 4.1 (b).* □

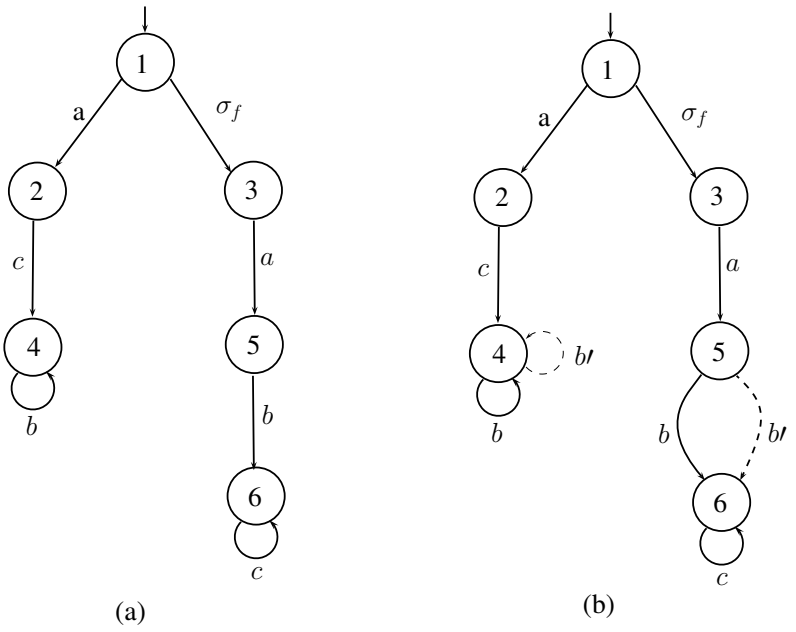


Figura 4.1: Autômato G (a), Autômato G_{dil} (b).

Uma vez definida a operação de dilatação, a definição de linguagem robustamente diagnosticável com relação a perdas intermitentes de observação de eventos pode ser apresentada [63].

Definição 4.2. *Uma linguagem prefixo-fechada e viva L é dita ser robustamente diagnosticável com relação à dilatação D, projeção $P_{dil,o} : \Sigma_{dil}^{\star} \rightarrow \Sigma_o^{\star}$ e Σ_f se*

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

sendo a condição de diagnosticabilidade robusta a perdas de observação intermitentes R_{ID} dada por:

$$[P_{dil,o}(D(st)) \cap P_{dil,o}(D(w)) = \emptyset, \forall w \in L_N]$$

□

Observação 4.1. *Note que se $\Sigma_{ilo} = \emptyset$ então $D(st) = \{st\}$ e $P_{dil,o}$ se reduz a P_o. Nesse caso, a definição 4.2 se reduz a definição usual de diagnosticabilidade de uma linguagem introduzida em [18].* □

De acordo com a definição 4.2, a linguagem L não é robustamente diagnosticável

se existe uma sequência de falha st de comprimento arbitrariamente longo após a falha e uma sequência normal w, tais que uma das possíveis observações da sequência st, considerando perdas intermitentes de observação dos eventos em Σ_{ilo}, é igual a uma das possíveis observações de w. Nesse caso, não é possível afirmar com certeza que o evento de falha ocorreu.

O exemplo a seguir ilustra a definição de linguagem robustamente diagnosticável a perdas intermitentes de observação apresentada em [63].

Exemplo 4.2. *Considere, novamente, o autômato G do exemplo 4.1, em que $\Sigma = \{a, b, c, \sigma_f\}$, $\Sigma_o = \{a, b, c\}$ e $\Sigma_{uo} = \Sigma_f = \{\sigma_f\}$. Admitindo que a observação do evento b possa ser perdida intermitentemente, quer-se verificar se a linguagem L é robustamente diagnosticável com relação à dilatação D, à projeção $P_{dil,o}$ e Σ_f.*

Considere a sequência de falha $s_Y = \sigma_f abc^n$ de comprimento arbitrariamente longo após a falha, obtida da linguagem do autômato G da figura 4.1(a). Verificando a condição R_{ID}, tem-se que $D(s_Y) = \{\sigma_f abc^n, \sigma_f ab'c^n\}$ e $P_{dil,o}(D(s_Y)) = \{abc^n, ac^n\}$. Note que a linguagem normal de G é dada por $L_N = \overline{\{a\}\{c\}\{b\}^\star}$. Logo, $P_{dil,o}[D(s_Y)] \cap P_{dil,o}[D(\omega)] = \emptyset$, para todo $\omega \in L_N$. Portanto, a linguagem L é robustamente diagnosticável com relação à dilatação D, projeção $P_{dil,o}$ e Σ_f.

□

O problema de diagnosticabilidade robusta a perdas intermitentes de observação é estendido em [63] para o caso de codiagnosticablidade robusta a perdas intermitentes de observação (CRPIO). Para tanto, a mesma dilatação D é aplicada à linguagem do sistema, formando a linguagem $L_{dil} = D(L)$, e são consideradas as observações dos eventos de cada local de diagnóstico, representadas pelas projeções $P_{dil,o_i} : \Sigma_{dil}^\star \to \Sigma_{o_i}^\star$. A definição de CRPIO é apresentada a seguir.

Definição 4.3. *Uma linguagem prefixo-fechada e viva L é dita ser robustamente codiagnosticável com relação à dilatação D, projeções P_{dil,o_i}, $i = 1, \ldots, n$, e Σ_f se*

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

sendo a condição de codiagnosticabilidade robusta R_{IC} dada por:

$$(\exists i \in I_n)[P_{dil,o_i}(D(st)) \cap P_{dil,o_i}(D(w)) = \emptyset, \forall w \in L_N].$$

□

De acordo com a definição 4.3, a linguagem L não é robustamente codiagnosticável com relação a D, P_{dil,o_i} e Σ_f, se existe uma sequência de falha st de comprimento arbitrariamente longo após a falha e sequências normais w_i, $i = 1, \ldots, n$, não necessariamente distintas, tais que uma das possíveis observações da sequência st, considerando perdas intermitentes de observação dos eventos em Σ_{ilo}, é igual a uma das possíveis observações de w_i, para todo $i \in I_n$. Nesse caso, não é possível afirmar com certeza que a sequência de falha st ocorreu.

Note que a definição 4.3 é equivalente à definição de codiagnosticabilidade apresentada em [22], aplicada à linguagem $L_{dil} = D(L)$. Portanto, a verificação da CRPIO pode ser feita utilizando o algoritmo apresentado em [70], substituindo apenas L por L_{dil} e substituindo a função de renomeação do passo 3 do algoritmo em [70] por $R_i : \Sigma_N \to \Sigma_{R_i}$ em que

$$R_i(\sigma) = \begin{cases} \sigma, & \text{se } \sigma \in \Sigma_{o_i} \\ \sigma_{R_i}, & \text{se } \sigma \in \Sigma'_{uo_i} \setminus \Sigma_f, \end{cases}$$

e $\Sigma'_{uo_i} = \Sigma_{uo_i} \cup \Sigma'_{ilo}$.

O exemplo a seguir ilustra a definição de CRPIO apresentada em [63].

Exemplo 4.3. *Considere o autômato G da figura 4.2 (a) em que $\Sigma = \{a, b, c, \sigma_f\}$, $\Sigma_{uo} = \{\sigma_f\}$ e dois locais de diagnose com os seguintes conjuntos de eventos observáveis $\Sigma_{o_1} = \{a, b\}$ e $\Sigma_{o_2} = \{b, c\}$. Suponha, também que o evento sujeito a perdida intermitente seja o evento b, ou seja, $\Sigma_{ilo} = \{b\}$. O autômato G_{dil} obtido após a dilatação da linguagem L é mostrado na figura 4.2 (b). Seguindo os passos do algoritmo 2.3 [70] com as devidas substituições, os autômatos que modelam o comportamento normal do sistema G_{N_1} e G_{N_2}, e o comportamento de falha do sistema nos locais de*

diagnose 1 e 2, respectivamente, foram substituídos por seus autômatos equivalentes em G_{dil} por G_{dil,N_1}, G_{dil,N_2} e $G_{dil,F}$, respectivamente, e mostrados na figura 4.3. O autômato verificador que deriva da composição paralela dos três autômatos da figura 4.3 é mostrado na figura 4.4 e como pode ser notado, o autômato verificador $G_{dil,V}$ não possui nenhum ciclo que atenda a condição (2.1). Portanto, a linguagem L é codiagnosticável com relação à dilatação D, P_{dil,o_i}, $i = 1, 2$, e Σ_f. □

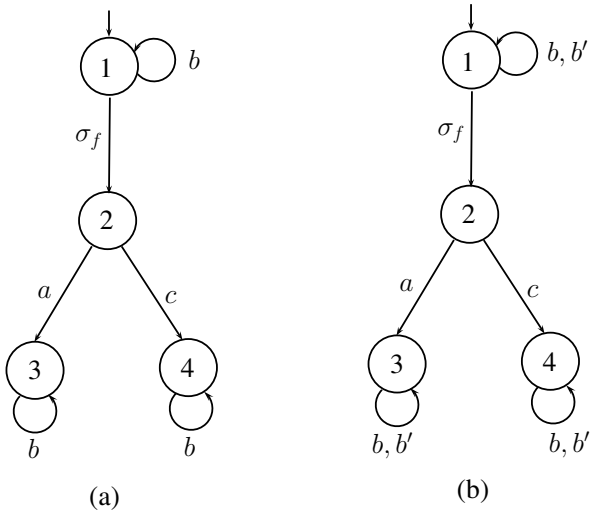


Figura 4.2: Autômato G (a), Autômato G_{dil} (b) do exemplo 4.3.

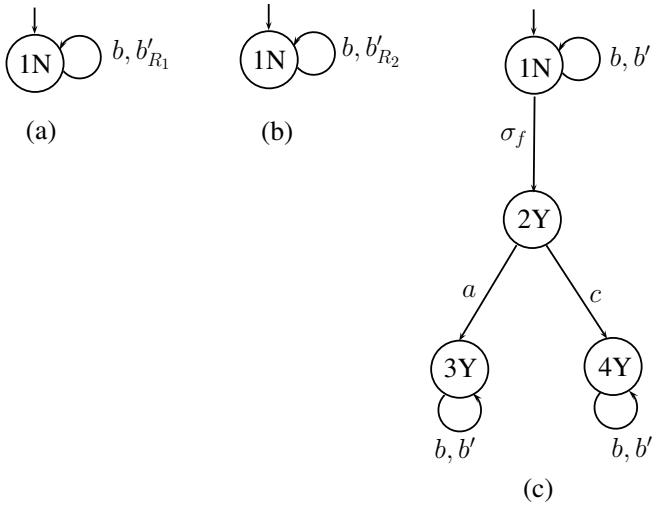


Figura 4.3: Autômato G_{dil,N_1} (a), Autômato G_{dil,N_2} (b), $G_{dil,F}$ (c).

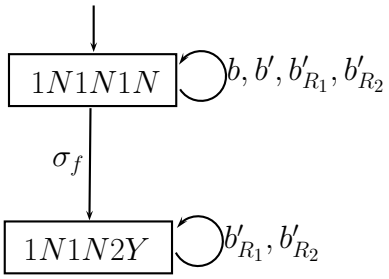


Figura 4.4: Autômato $G_{dil,V}$.

Note que na definição de CRPIO apresentada em [63], é suposto que se a observação de um evento $\sigma \in \Sigma_{ilo}$ é perdida para um determinado local de medição, então a observação de σ é perdida para todos os locais de diagnose que observam σ. Isso considera o caso em que a perda de observação se dá pelo mal funcionamento de um sensor do sistema. Contudo, supondo que a perda ocorra devido a problemas na comunicação da ocorrência do evento entre um sensor e um diagnosticador, então é possível que um mesmo evento σ sofra perdas intermitentes de observação para um determinado local de diagnose e não sofra para um outro local de diagnose. Logo, se a causa para a perda intermitente de observação está em um canal de comunicação, o problema não pode ser tratado com a abordagem apresentada em [63].

Um outro problema que não pode ser tratado pela abordagem apresentada em [63] é o caso de se ter incerteza no conjunto de eventos que podem sofrer perdas intermitentes de observação em cada local de diagnose, ou seja, no lugar de ter uma única partição do conjunto de eventos Σ, é possível ter mais do que uma partição $\Sigma = \Sigma^j_{ilo,i} \dot{\cup} \Sigma^j_{nilo,i} \dot{\cup} \Sigma_{uo_i}$, para $j = 1, \ldots, m$, em que m denota o número de partições possíveis de Σ. Para visualizar esse fato, considere que é sabido que dois eventos podem sofrer perdas intermitentes de observação por algum motivo como, por exemplo, tempo longo de operação do sistema sem manutenção dos sensores. Contudo, a probabilidade de que ambos comecem a perder a observação ao mesmo tempo é muito pequena e desprezível. Nesse caso, considerar que ambos podem ser perdidos intermitentemente e simultaneamente pode ser muito conservador e levar à conclusão de que o sistema não é robustamente codiagnosticável com relação a

perdas intermitentes de observação. Esse caso é ilustrado no exemplo a seguir.

Exemplo 4.4. *Considere o autômato da figura 4.5 em que* $\Sigma = \{a, b, c, \sigma_f\}$, $\Sigma_o = \{a, b, c\}$ *e* $\Sigma_f = \{\sigma_f\}$. *Se for considerada a perda intermitente de a e b, tem-se que a linguagem L não é robustamente codiagnosticável pois existe a sequência de falha* $s_Y = \sigma_f abc^n$, *cuja dilatação é dada por* $D(s_Y) = \sigma_f\{a, a'\}\{b, b'\}c^n$ *e projeção é* $P_{dil,o}(D(s_Y)) = \{a, \epsilon\}\{b, \epsilon\}c^n = \{c^n, ac^n, bc^n, abc^n\}$, *e existe a sequência normal* $w = c^n$, *cuja dilatação é* $D(w) = c^n$ *e projeção* $P_{dil,o}(D(w)) = c^n$. *Portanto,* $P_{dil,o}(D(s_Y)) \cap P_{dil,o}(D(w)) \neq \emptyset$.

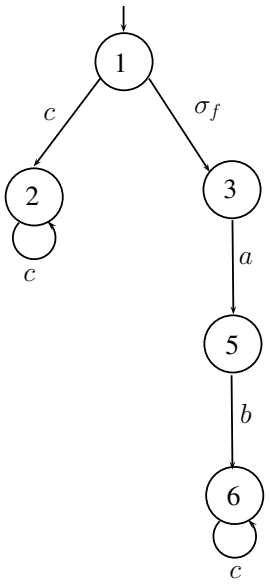


Figura 4.5: Autômato G do exemplo 4.4.

Por outro lado, considerando-se que os eventos não falham intermitentemente de modo simultâneo, então a ocorrência da sequência $s_Y = \sigma_f abc^n$ *passa a ter dilatação supondo a perda intermitente do evento a,* $\sigma_f\{a, a'\}bc^n$, *e dilatação supondo a perda intermitente do evento b,* $\sigma_f a\{b, b'\}c^n$. *Neste caso, as observações das sequências dilatadas serão* $P_{dil,o}(\sigma_f\{a, a'\}bc^n) = \sigma_f\{a, \epsilon\}bc^n$ *e* $P_{dil,o}(\sigma_f a\{b, b'\}c^n) = \sigma_f a\{b, \epsilon\}c^n$, *que não se confundem com a observação da sequência normal* $w = c^n$. *Portanto, é possível afirmar com certeza que a sequência de falha* s_Y *foi executada e a linguagem L é robustamente diagnosticável com relação a perda de observação intermitente do evento a ou do evento b.*

□

4.2 Nova definição de codiagnosticabilidade robusta a perdas intermitentes de observação (CRPIO)

Neste trabalho uma nova definição de CRPIO, que considera o caso de perdas de observação distintas para cada local de diagnose, é apresentada. Para tanto, é necessário primeiro definir a operação de dilatação D_i^j associada ao i-ésimo local de diagnose e j-ésima partição dos conjuntos de eventos observáveis $\Sigma_{o_i} = \Sigma_{ilo,i}^j \dot{\cup} \Sigma_{nilo,i}^j$, para $i = 1, \ldots, n$ e $j = 1, \ldots, m$. Assim como a dilatação D utilizada em [63], a operação de dilatação D_i^j associa a cada evento $\sigma \in \Sigma_{ilo,i}^j$ que pode ter sua observação perdida intermitentemente, o próprio evento observável σ, representando a sua correta observação, e um evento σ' não-observável, representando a perda de observação de σ. Seja $\Sigma_{ilo,i}^{j\prime} = \{\sigma' : \sigma \in \Sigma_{ilo,i}^j\}$, então a dilatação D_i^j pode ser definida como o mapeamento

$$D_i^j : \Sigma^\star \to 2^{\Sigma_{dil,i}^{j\star}},$$

em que $\Sigma_{dil,i}^j = \Sigma \cup \Sigma_{ilo,i}^{j\prime}$ e

$$\begin{aligned} &D_i^j(\epsilon) = \epsilon \\ &D_i^j(\sigma) = \begin{cases} \sigma, \text{ se } \sigma \in \Sigma \setminus \Sigma_{ilo,i}^j \\ \{\sigma, \sigma'\}, \text{ se } \sigma \in \Sigma_{ilo,i}^j \end{cases} \\ &D(s\sigma) = D(s)D(\sigma) \text{ para todo } s \in \Sigma^\star \text{ e } \sigma \in \Sigma. \end{aligned}$$

A operação D_i^j pode ser estendida para linguagens da mesma forma que D, ou seja,

$$D_i^j(L) = \bigcup_{s \in L} D_i^j(s).$$

Sejam $P_{dil,o_i}^j : \Sigma_{dil,i}^{j\star} \to \Sigma_{o_i}^\star$, para $i = 1, \ldots, n$ e $j = 1, \ldots, m$, operações de

projeção. Então, a seguinte definição de CRPIO pode ser enunciada.

Definição 4.4. *Uma linguagem prefixo-fechada e viva L é dita ser robustamente codiagnosticável com relação às dilatações D_i^j, projeções P_{dil,o_i}^j, $i = 1, \ldots, n$, $j = 1, \ldots, m$, e Σ_f se*

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

sendo a condição de codiagnosticabilidade robusta R_{DIC} dada por:

$$(\exists i \in I_n)(\forall k, l \in I_m)[P_{dil,o_i}^k(D_i^k(st)) \cap P_{dil,o_i}^l(D_i^l(w)) = \emptyset, \forall w \in L_N].$$

□

De acordo com a definição 4.4, L não é robustamente codiagnosticável com relação a D_i^j, $P_{dil,i}^j$ e Σ_f, se existe uma sequência de falha st, com comprimento arbitrariamente longo após a falha, e sequências normais w_i, para $i = 1, \ldots, n$, tais que uma observação da sequência st, com possível perda intermitente de observação de eventos em um conjunto $\Sigma_{ilo,i}^k$, é igual à observação de uma sequência normal w_i, com possível perda de observação de eventos em um conjunto $\Sigma_{ilo,i}^l$, para todo $i \in I_n$. Em outras palavras, se a observação correta da sequência de falha st é a dada pela k-ésima partição dos conjuntos de eventos observáveis, então, como em todos os locais de diagnose existem sequências normais w_i tais que $P_{dil,o_i}^k(D_i^k(st)) \cap P_{dil,o_i}^l(D_i^l(w_i)) \neq \emptyset$, nenhum dos diagnosticadores locais associados à l-ésima partição indicará a ocorrência do evento de falha. Portanto, a linguagem L não é robustamente codiagnosticável.

Assim como no capítulo anterior foram definidas bases para a codiagnose, em que essas bases são os conjuntos de eventos observáveis tais que a codiagnosticabilidade da linguagem do sistema é garantida, pode-se definir bases para a codiagnose robusta de L a perdas intermitentes de observação de eventos. Nesse caso, as bases são formadas pelos conjuntos de eventos observáveis de cada local de diagnose que não podem ter suas observações perdidas $\{\Sigma_{nilo,1}^j, \Sigma_{nilo,2}^j, \ldots, \Sigma_{nilo,n}^j\}$, em que j denota

uma base. A definição formal de bases para a codiagnose de L sujeita a perdas intermitentes de observação é apresentada a seguir.

Definição 4.5. *Suponha que L seja robustamente codiagnosticável com relação às dilatações D_i^j, projeções P_{dil,o_i}^j, para $i = 1, \ldots, n$ e $j \in I_m$, e Σ_f. Então o conjunto $\{\Sigma_{nilo,1}^j, \Sigma_{nilo,2}^j, \ldots, \Sigma_{nilo,n}^j\}$, em que $\Sigma_{o_i} = \Sigma_{ilo,i}^j \dot{\cup} \Sigma_{nilo,i}^j$, para $i = 1, \ldots, n$, é uma base para a codiagnose robusta de L a perdas intermitentes de observação.*

Assim como a base para a codiagnose de L sem perdas intermitentes de observação, a propriedade de monotonicidade também é observada para as bases para codiagnose robusta de L sujeita a perdas intermitentes de observação.

Teorema 4.1. *Seja $\Sigma_{rcb} = \{\Sigma_{nilo,1}^j, \Sigma_{nilo,2}^j, \ldots, \Sigma_{nilo,n}^j\}$ uma base para codiagnose robusta de L a perdas intermitentes de observação. Então, qualquer conjunto $\hat{\Sigma}_{rcb} = \{\hat{\Sigma}_{nilo,1}^j, \hat{\Sigma}_{nilo,2}^j, \ldots, \hat{\Sigma}_{nilo,n}^j\}$, tal que $\Sigma_{nilo,i}^j \subseteq \hat{\Sigma}_{nilo,i}^j$, para $i = 1, \ldots, n$, também é uma base para a codiagnose robusta de L a perdas intermitentes de observação.*

Demonstração. Seja Σ_{rcb} uma base para a codiagnose robusta de L a perdas intermitentes dos eventos $\Sigma_{ilo,i}^j$, para $i = 1, 2, \ldots, n$. Então, de acordo com a definição 4.4, tem-se que para todo $st \in L \setminus L_N$, de comprimento arbitrariamente longo após a falha, existe $i \in I_n$, tal que

$$P_{dil,o_i}^j(D_i^j(st)) \cap P_{dil,o_i}^j(D_i^j(w)) = \emptyset, \tag{4.1}$$

para todo $\omega \in L_N$. Defina a operação de dilatação $\hat{D}_i^j : \Sigma^\star \to 2^{\hat{\Sigma}_{dil,i}^j}$, em que $\hat{\Sigma}_{dil,i}^j = \Sigma \cup \hat{\Sigma}_{ilo,i}^{j\prime}$, e a operação de projeção $\hat{P}_{dil,o_i}^j : \hat{\Sigma}_{dil,i}^{j\star} \to \Sigma_{o_i}^\star$. Como $\hat{D}_i^j(s) \subseteq D_i^j(s)$ para todo $s \in L$ e $i \in I_n$, então, de acordo com (4.1), existe $i \in I_n$ tal que $\hat{P}_{dil,o_i}^j(\hat{D}_i^j(st)) \cap \hat{P}_{dil,o_i}^j(\hat{D}_i^j(w)) = \emptyset$ para todo $st \in L \setminus L_N$, com comprimento arbitrariamente longo após a falha, e $w \in L_N$.

□

É possível também definir bases mínimas para a codiagnose robusta de L como mostrado a seguir.

Definição 4.6. *O conjunto* $\Sigma_{rcb} = \{\Sigma^j_{nilo,1}, \Sigma^j_{nilo,2}, \ldots, \Sigma^j_{nilo,n}\}$ *é uma base mínima para a codiagnose robusta de* L *a perdas intermitentes de observação se* Σ_{rcb} *é uma base e não existe um conjunto* $\tilde{\Sigma}_{rcb} = \{\tilde{\Sigma}^j_{nilo,1}, \tilde{\Sigma}^j_{nilo,2}, \ldots, \tilde{\Sigma}^j_{nilo,n}\}$*, em que* $\tilde{\Sigma}^j_{nilo,i} \subseteq \Sigma^j_{nilo,i}$*, para todo* $i \in I_n$ *e* $\tilde{\Sigma}^j_{nilo,i} \subset \Sigma^j_{nilo,i}$ *para algum* $i \in I_n$*, que seja uma base para a codiagnose robusta de* L *a perdas intermitentes de observação.* □

É fácil ver que se existe $j \in I_m$ tal que $\{\Sigma^j_{nilo,1}, \Sigma^j_{nilo,2}, \ldots, \Sigma^j_{nilo,n}\}$ não é uma base para a codiagnose robusta de L, então, de acordo com a definição 4.4, L não é robustamente codiagnosticável com relação a D^j_i, P^j_{dil,o_i}, para $i = 1, \ldots, n$ e $j = 1, \ldots, m$, e Σ_f. Logo, uma condição necessária para a codiagnosticabilidade robusta de L a perdas intermitentes de observação é que $\{\Sigma^j_{nilo,1}, \Sigma^j_{nilo,2}, \ldots, \Sigma^j_{nilo,n}\}$ seja uma base para codiagnose robusta, para todo $j \in I_m$.

Em alguns casos, como mostrado a seguir, a verificação da nova definição de CRPIO proposta neste trabalho pode ser trivialmente verificada.

Teorema 4.2. *Sejam* $\{\Sigma^1_{nilo,1}, \ldots, \Sigma^1_{nilo,n}\}$ *e* $\{\Sigma^2_{nilo,1}, \ldots, \Sigma^2_{nilo,n}\}$ *duas bases para a codiagnose robusta de* L *a perdas intermitentes de observação, em que* $\Sigma^1_{nilo,i} \subseteq \Sigma^2_{nilo,i}$*, para* $i = 1, \ldots, n$*. Então,* L *é robustamente codiagnosticável com relação à dilatação* D^j_i*, projeções* P^j_{dil,o_i}*, para* $i = 1, \ldots, n$ *e* $j = 1, 2$*, e* Σ_f*.*

Demonstração. Como $\{\Sigma^1_{nilo,1}, \ldots, \Sigma^1_{nilo,n}\}$ é uma base para a codiagnose robusta de L, então, de acordo com a definição 4.4, tem-se que

$$P^1_{dil,o_i}(D^1_i(st)) \cap P^1_{dil,o_i}(D^1_i(w)) = \emptyset. \tag{4.2}$$

para algum $i \in I_n$. Como $P^2_{dil,o_i}(D^2_i(s)) \subseteq P^1_{dil,o_i}(D^1_i(s))$, para todo $s \in L$, então, de acordo com a equação (4.2), $P^1_{dil,o_i}(D^1_i(st)) \cap P^2_{dil,o_i}(D^2_i(w)) = \emptyset$. Da mesma forma, tem-se que $P^2_{dil,o_i}(D^2_i(st)) \cap P^1_{dil,o_i}(D^1_i(w)) = \emptyset$, o que conclui a prova. □

A seguir é apresentado um algoritmo para a verificação da CRPIO de eventos para os casos em que a condição do teorema 4.2 não é verificada. É importante ressaltar que o algoritmo 4.2 pode ser utilizado para a verificação de bases para a codiagnose robusta de L a perdas intermitentes de observação de eventos.

4.3 Algoritmo de verificação

Algoritmo 4.2. *Verificação da CRPIO.*

Entrada: $G = (X, \Sigma, f, \Gamma, x_0)$, $\Sigma^j_{ilo,i}$.

Saída: *Decisão: sim ou não.*

1: Obtenha o autômato $G_N = (X_N, \Sigma_N, f_N, (x_0, N))$, *em que* $\Sigma_N = \Sigma \setminus \Sigma_f$ *que modela o comportamento normal de* G *como mostrado no algoritmo 2.3 [70].*

2: Obtenha o autômato $G_F = (X_F, \Sigma, f_F, (x_0, N))$, *que modela o comportamento de falha de* G *como mostrado no algoritmo 2.3 [70].*

3: Para cada par $(k, l) \in I_m \times I_m$ *faça:*

3.1: Calcule os autômatos dilatados $G^l_{N,i} = (X, \Sigma^l_{N_{dil,i}}, f^{il}_N, (x_0, N))$, *utilizando o algoritmo 4.1 com entradas* G_N *e* $\Sigma^l_{ilo,i}$, *para* $i = 1, \ldots, n$.

3.2: Calcule os autômatos dilatados $G^k_{F,i} = (X, \Sigma^k_{dil,i}, f^{ik}_F, (x_o, N))$, *utilizando o algoritmo 4.1 com entradas* G_F *e* $\Sigma^k_{ilo,i}$, *para* $i = 1, \ldots, n$.

4: Defina o conjunto dos eventos não-observáveis de $\Sigma^l_{N_{dil,i}}$, $\Sigma^l_{uo,i} = \Sigma^{l\prime}_{ilo,i} \cup (\Sigma_{uo_i} \setminus \Sigma_f)$. *Para cada par* $(i, l) \in I_n \times I_m$ *defina o conjunto:*

$$\Sigma^l_{R,i} = \{\sigma^{il}_R : \sigma \in \Sigma^l_{uo,i}\},$$

e construa o autômato $G^l_{R,i} = (X, \Sigma_{o_i} \cup \Sigma^l_{R,i}, f^{il}_R, (x_0, N))$, *em que* $f^{il}_R(x, \sigma) = f^{il}_N(x, \sigma)$ *para todo* $x \in X$ *e* $\sigma \in \Sigma_{o_i}$, *e* $f^{il}_R(x, \sigma^{il}_R) = f^{il}_N(x, \sigma)$, *para todo* $x \in X$ *e* $\sigma^{il}_R \in \Sigma^l_{R,i}$.

5: Calcule para cada par $(k, l) \in I_m \times I_m$, $V^{kl}_i = G^l_{R,i} \| G^k_{F,i} = (Y^{kl}_i, \Sigma^{kl}_i, f^{kl}_i, y^{kl}_{i,0})$, *para* $i = 1, \ldots, n$.

6: Encontre todos os caminhos cíclicos $cl_i = (y_i^{kl^p}, \sigma_p, y_i^{kl^{p+1}}, \sigma_{p+1}, \ldots, \sigma_q, y_i^{kl^p})$, *em que* $q \geq p > 0$ *em* V_i^{kl} *que satisfazem à seguinte condição:*

$$\exists j \in \{p, p+1, \ldots, q\} \text{ tal que}$$
$$y_i^{kl^j} = (x, N, y, Y) \wedge \sigma_j \in \Sigma_{dil,i}^k. \tag{4.3}$$

7: Marque os componentes fortemente conexos que possuem caminhos cíclicos com as características apresentados no passo 6.

8: Para cada par $(k, l) \in I_m \times I_m$, *faça:*

8.1: Obtenha um novo autômato $\hat{V}_i^{kl}$ *a partir de* V_i^{kl} *renomeando cada transição de* V_i^{kl} *rotulada por um evento* $\sigma' \in \Sigma_{ilo,i}^{k\prime}$ *pelo evento correspondente* $\sigma \in \Sigma_{ilo,i}^k$.

8.2: Calcule o autômato verificador $V^{kl} = \hat{V}_1^{kl} \| \ldots \| \hat{V}_n^{kl} = (Y^{kl}, \Sigma^{kl}, f_{kl}, y_0^{kl}, Y_m^{kl})$.

9: Verifique a existência de um caminho cíclico $cl = (y^{kl^p}, \sigma_p, y^{kl^{p+1}}, \sigma_{p+1}, \ldots, \sigma_q, y^{kl^p})$ *em* V^{kl}, $q \geq p > 0$, *para algum* $(k, l) \in I_m \times I_m$, *que satisfaz à seguinte condição:*

$$y^{kl^j} \in Y_m^{kl}, \forall j \in \{p, p+1, \ldots, q\}, \text{ e para algum}$$
$$j \in \{p, p+1, \ldots, q\}, \sigma_j \in \Sigma.$$

Se a resposta é sim, então L não é robustamente codiagnosticável com relação às dilatações D_i^j, *projeções* P_{dil,o_i}^j *e* Σ_f. *Caso contrário, L é robustamente codiagnosticável.*

A ideia por trás do algoritmo 4.2 é apresentada a seguir. No primeiro e segundo passos são obtidos os autômatos G_N e G_F que modelam o comportamento normal e de falha de G, respectivamente. Em seguida, no passo 3, são gerados os autômatos

$G^l_{N_i}$ e $G^k_{F_i}$ cujas linguagens geradas são as dilatações $D^l_i(L_N)$ e $D^k_i(\overline{L \setminus L_N})$, respectivamente. No passo 4, os autômatos $G^l_{N_i}$ têm os seus eventos não-observáveis, incluindo os eventos pertencentes a $\Sigma^{l\prime}_{ilo,i}$, renomeados, gerando os autômatos $G^l_{R,i}$. Note que os eventos renomeados são eventos particulares de $G^l_{R,i}$. No passo 5, os autômatos V^{kl}_i são calculados. Note que somente as sequências de $G^l_{R,i}$ e $G^k_{F,i}$ que possuem a mesma projeção no conjunto dos eventos observáveis Σ_{o_i} são representadas em V^{kl}_i. Portanto, se existe uma sequência de falha st e uma sequência normal w_i tais que $P^k_{dil,o_i}(D^k_i(st)) \cap P^l_{dil,o_i}(D^l_i(w_i)) \neq \emptyset$, então uma sequência é gerada em V^{kl}_i associada a st e w_i. Nesse caso, se st possui comprimento arbitrariamente longo após a falha, então a sequência gerada em V^{kl}_i forma um ciclo com as características apresentadas no passo 6. No passo 8, o autômato V^{kl} é formado visando verificar se existe uma única sequência st associada a todos os locais de diagnose que satisfaz $P^k_{dil,o_i}(D^k_i(st)) \cap P^l_{dil,o_i}(D^l_i(w_i)) \neq \emptyset$, para $i = 1, \ldots, n$. Se essa sequência de falha st existe e possui comprimento arbitrariamente longo após a falha, então L não é robustamente codiagnosticável em relação a perdas intermitentes de observação por violar a condição da definição 4.4.

O teorema a seguir apresenta a condição necessária e suficiente para a codiagnosticabilidade robusta de L com relação a perdas intermitentes de observação de acordo com a definição 4.4.

Teorema 4.3. *A linguagem L, viva e prefixo-fechada, é robustamente codiagnosticável com relação a D^j_i, P^j_{dil,o_i}, $i = 1, \ldots, n$, $j = 1, \ldots, m$ e Σ_f, se e somente se não existe em V^{kl}, para todo $(k, l) \in I_m \times I_m$, um caminho cíclico $cl = (y^{klp}, \sigma_p, y^{klp+1}, \sigma_{p+1}, \ldots, \sigma_q, y^{klp})$ que satisfaz à seguinte condição:*

$$y^{klj} \in Y^{kl}_m, \forall j \in \{p, p+1, \ldots, q\}, \text{ e para algum}$$
$$j \in \{p, p+1, \ldots, q\}, \sigma_j \in \Sigma. \tag{4.4}$$

Demonstração. Suponha que L não é robustamente codiagnosticável com relação a D^j_i, P^j_{dil,o_i} e Σ_f. Nesse caso, de acordo com a definição 4.4, existe um par

$(k,l) \in I_m \times I_m$ tal que para todo $i \in I_n$ existe uma sequência de falha st e sequências normais w_i tais que $P^k_{dil,o_i}(D^k_i(st)) \cap P^l_{dil,o_i}(D^l_i(w_i)) \neq \emptyset$, isto é, existem sequências de falha $s_{Y_i} \in D^k_i(st)$ e sequências normais $s_{N_i} \in D^l_i(w_i)$ com a mesma projeção no conjunto $\Sigma^\star_{o_i}$. De acordo com o teorema 2.3, isso implica em existirem caminhos cíclicos com as características apresentadas na condição (4.3) em todos os autômatos V^{kl}_i para $i = 1, \ldots, n$. Note que todas as sequências $s_{Y_i} \in D^k_i(st)$, para $i = 1, \ldots, n$, estão associadas a uma única sequência st, e que ao executar o passo 8.1 para a obtenção de $\hat{V}^{kl}_i$, as sequências s_{Y_i} associadas às sequências de V^{kl}_i que levam aos caminhos cíclicos que satisfazem (4.3), são transformadas na própria sequência st. Note ainda que todos os demais eventos de $\hat{V}^{kl}_i$ são eventos renomeados e particulares. Assim, ao se calcular o autômato $V^{kl} = \hat{V}^{kl}_1 \| \ldots \| \hat{V}^{kl}_n$, existirá um caminho cíclico em V^{kl} associado à sequência st. Como todos os estados dos caminhos cíclicos de $\hat{V}^{kl}_i$ associados aos caminhos cíclicos de V^{kl}_i que satisfazem (4.3) são marcados, então existirá em V^{kl} um caminho cíclico cl formado por estados marcados. Além disso, como st possui comprimento arbitrariamente longo, pelo menos um dos eventos de cl pertence a Σ.

Suponha agora que L é robustamente codiagnosticável com relação a D^j_i, P^j_{dil,o_i} e Σ_f. Então, de acordo com a definição 4.4, para toda sequência de falha st tal que $\|t\| \geq z$, em que $z \in \mathbb{N}$, e para todo par $(k,l) \in I_m \times I_m$, existe $i \in I_n$ tal que $P^k_{dil,o_i}(D^k_i(st)) \cap P^l_{dil,o_i}(D^l_i(w)) = \emptyset$, para todo $w \in L_N$. Isso implica, de acordo com o teorema 2.3, que para todo $(k,l) \in I_m \times I_m$ existe um verificador V^{kl}_i que não possui caminhos cíclicos associados a st que satisfazem a condição (4.3). Portanto, V^{kl}_i não possuirá estados marcados associados a st e, consequentemente, V^{kl} não possuirá caminhos cíclicos que satisfazem a condição (4.4). □

Exemplo 4.5. *Considere o autômato da figura 4.6 em que* $\Sigma = \{a, b, c, d, \sigma_f\}$, $\Sigma_o = \{a, b, c, d\}$, $\Sigma_{uo} = \Sigma_f = \{\sigma_f\}$. *Suponha que existam dois diagnosticadores locais com os conjuntos de eventos observáveis* $\Sigma_{o_1} = \{a, b, c\}$ *e* $\Sigma_{o_2} = \{a, b, d\}$. *Suponha que nos locais de diagnose 1 e 2 a observação associada ao evento b esteja sujeita à perdas intermitentes de observação, ou seja,* $\Sigma^1_{ilo_1} = \{b\}$ *e* $\Sigma^1_{ilo_2} = \{b\}$, *assim como*

a observação associada ao evento a, ou seja, $\Sigma^2_{ilo_1} = \{a\}$ *e* $\Sigma^2_{ilo_2} = \{a\}$*. Nesse caso, os conjuntos de eventos não sujeitos à perdas intermitentes de observação e que constituem bases para a codiagnose robusta na perda intermitente da observação do evento b são* $\Sigma^1_{nilo_1} = \{a, c\}$ *e* $\Sigma^1_{nilo_2} = \{a, d\}$ *e na perda intermitente da observação do evento a são* $\Sigma^2_{nilo_1} = \{b, c\}$ *e* $\Sigma^2_{nilo_2} = \{b, d\}$*. Seguindo os passos 1 e 2 do algoritmo 4.2, foram calculados os autômatos* G_N *e* G_F *mostradas na figura 4.7. Nos passos 3.1 e 3.2 os autômatos* $G^2_{N,1}$ *e* $G^1_{F,1}$*, mostrados na figura 4.8 (a) e (b), respectivamente, foram calculados. No passo 4, o autômato* $G^2_{R,1}$ *foi calculado e mostrado na figura 4.8 (c).*

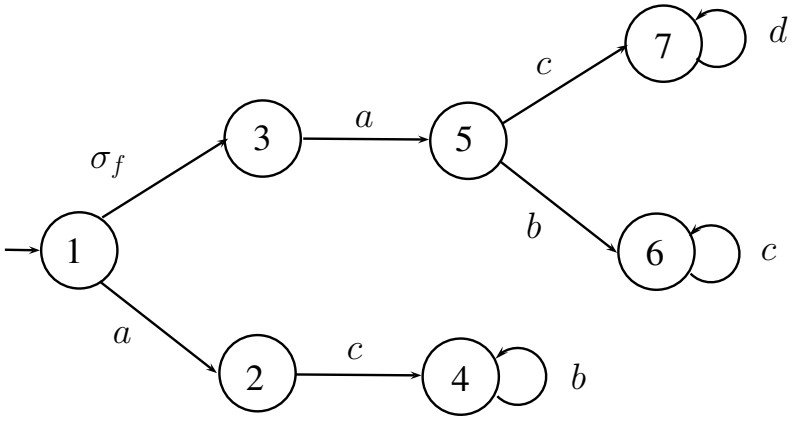


Figura 4.6: Autômato G.

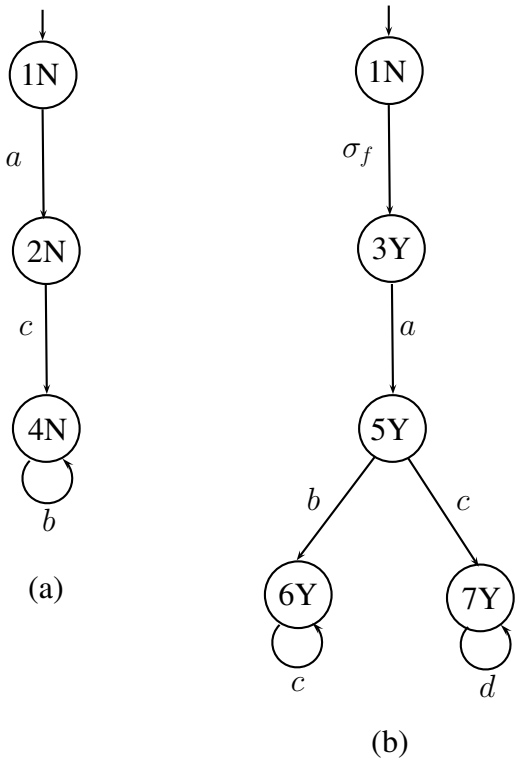


Figura 4.7: Autômato G_N (a), Autômato G_F (b).

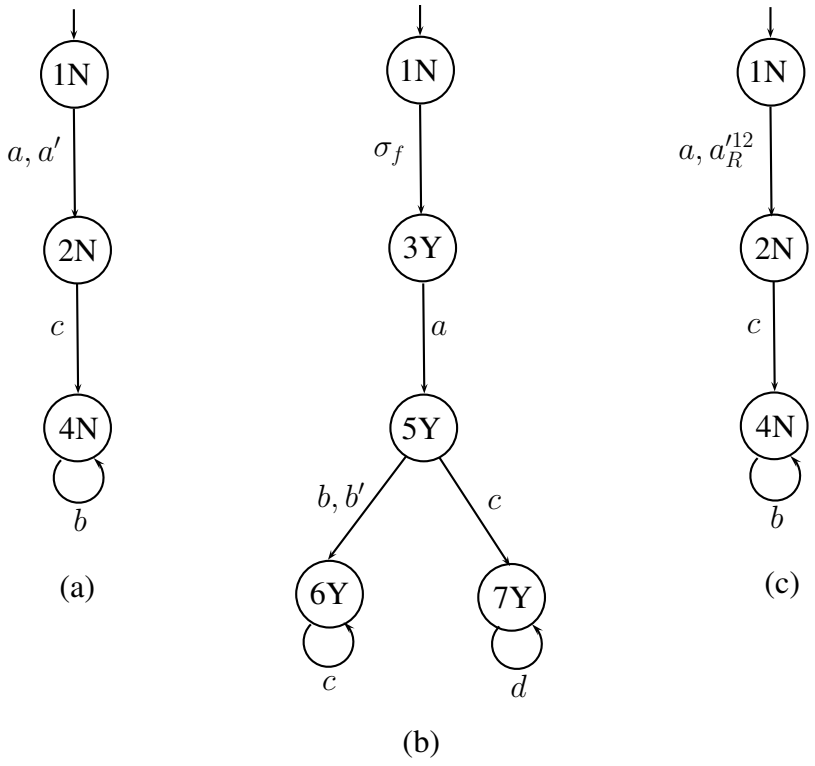


Figura 4.8: Autômato $G^2_{N_1}$ (a), Autômato $G^1_{F_1}$ (b) e Autômato $G^2_{R_1}$ (c).

No passo 5, foi construído o verificador V_1^{12}, ilustrado na figura 4.9. Observe que existe um caminho cíclico $\{\{4N, 7Y\}, d, \{4N, 7Y\}\}$ contendo estado com rótulo Y e portanto, esse estado é marcado. No passo 8.1, obtém-se o autômato $\hat{V}_1^{12}$ em que os eventos σ' são renomeados, conforme ilustra a figura 4.10. Para calcular o autômato $\hat{V}_2^{12}$, foram obtidos, primeiramente, os autômatos $G_{N_2}^2$, $G_{F_2}^1$ e $G_{R_2}^2$, os quais são apresentados na figura 4.11 (a), (b) e (c), respectivamente. Aplicando o passo 5 do algoritmo 4.2, foi construído o verificador V_2^{12} relativo ao local de diagnose 2 e mostrado na figura 4.12. Nesse autômato, podem ser observados dois caminhos cíclicos, $\{\{2N, 6Y\}, c, \{2N, 6Y\}\}$ e $\{\{4N, 6Y\}, c, \{4N, 6Y\}\}$ e ambos foram marcados, seguindo o passo 7 do algoritmo 4.2. Renomeando os eventos σ' desse autômato, obtém-se o autômato $\hat{V}_2^{12}$, mostrado na figura 4.13.

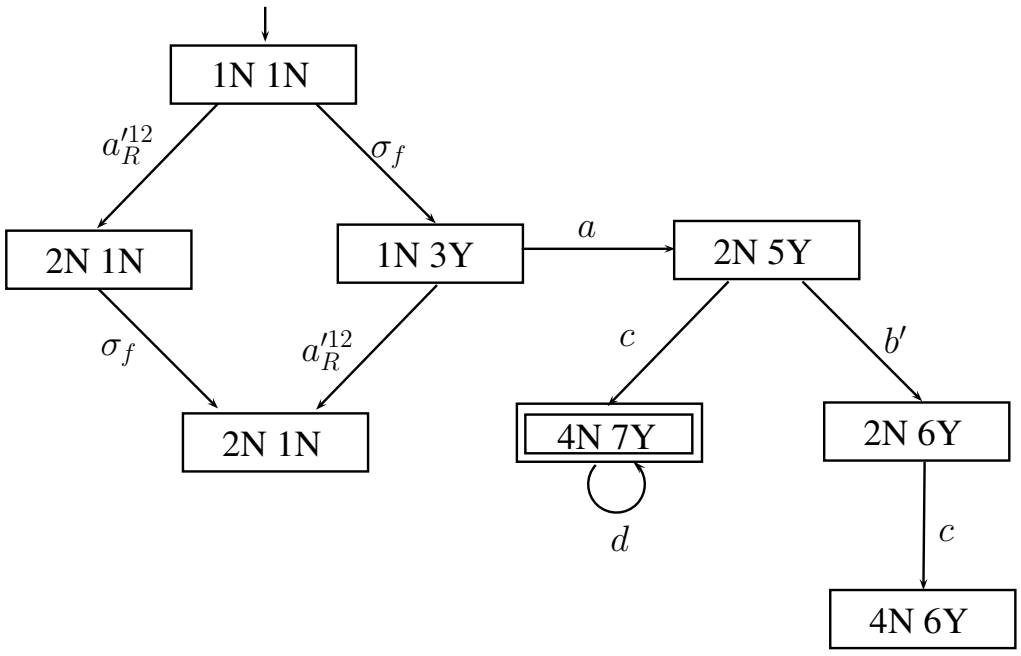


Figura 4.9: Autômato verificador V_1^{12}.

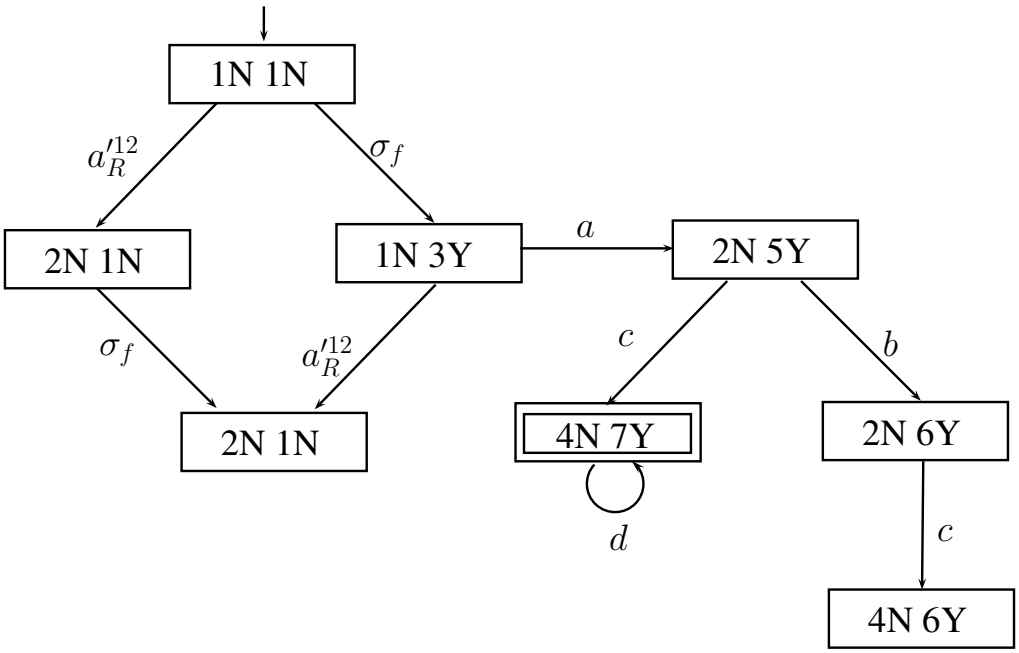


Figura 4.10: Autômato $\hat{V}_1^{12}$.

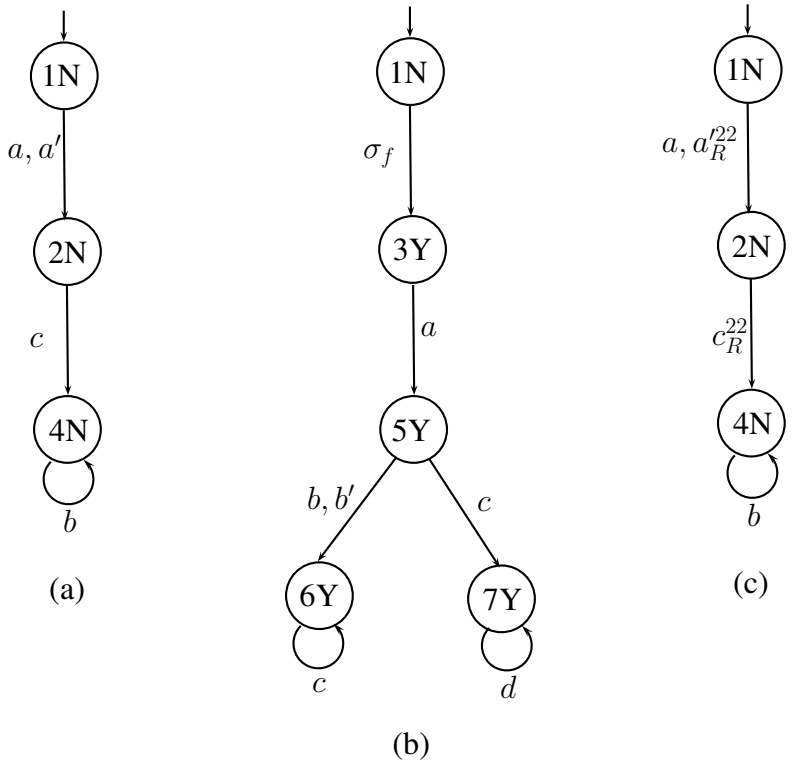


Figura 4.11: Autômato $G_{N_2}^2$ (a), Autômato $G_{F_2}^1$ (b) e Autômato $G_{R_2}^2$.

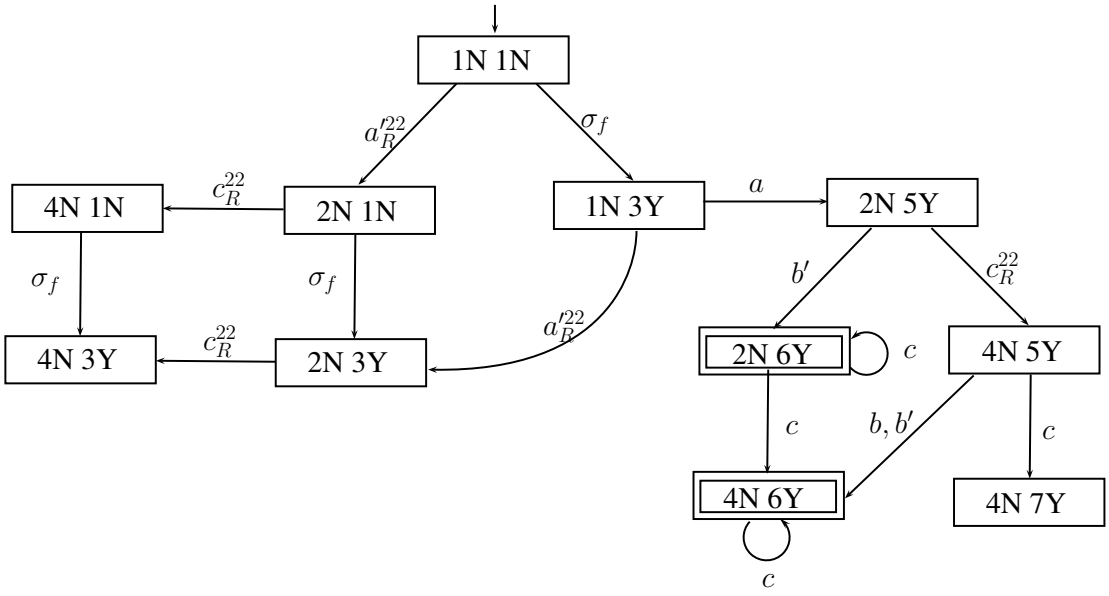


Figura 4.12: Autômato V_2^{12}.

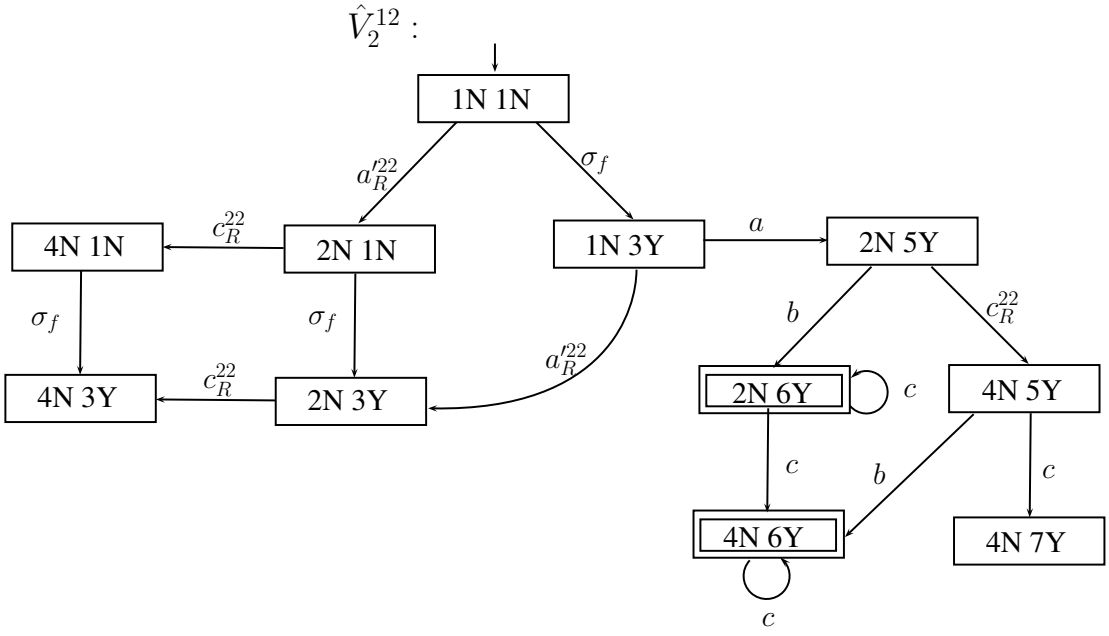


Figura 4.13: Autômato $\hat{V}_2^{12}$.

No passo 8.2 do 4.2 é construído o autômato V^{12} e como pode ser observado na figura 4.14 ele não possui nenhum estado marcado e portanto, nenhum caminho

cíclico que atenda a condição posta no passo 9 do 4.2. Para se certificar de que a linguagem L do sistema é robustamente codiagnosticável a perdas intermitentes de observação dos eventos a e b, além do verificador V^{12}, deve-se se construir os verificadores V^{21}, V^{11} e V^{22} seguindo os mesmos passos do algoritmo 4.2 para a construção de V^{12}. Nesse sentido, para calcular o verificador V^{21}, aplicando os

passos 3.1 e 3.2 os autômatos $G^1_{N,1}$ e $G^2_{F,1}$, mostrados na figura 4.15 (a) e (b), respectivamente, foram calculados. No passo 4, o autômato $G^1_{R,1}$ foi calculado e mostrado na figura 4.15 (c).

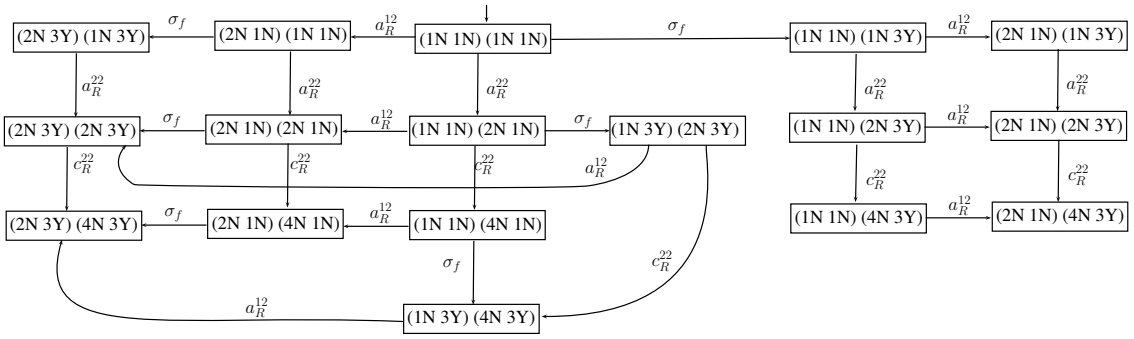


Figura 4.14: Autômato V^{12}.

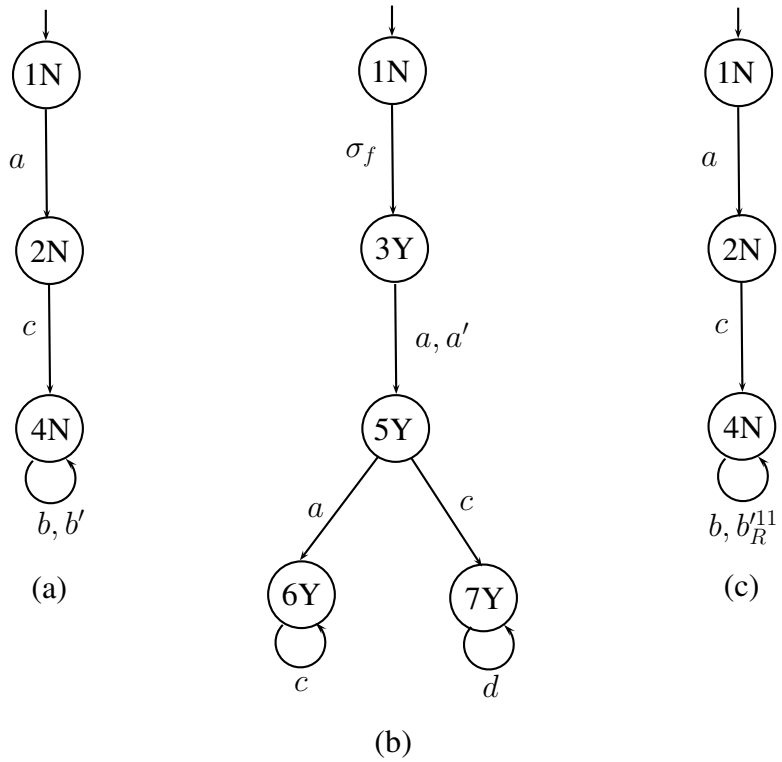


Figura 4.15: Autômato $G^1_{N_1}$ (a), Autômato $G^2_{F_1}$ (b) e Autômato $G^1_{R_1}$ (c).

No passo 5, foi construído o verificador V_1^{21}, ilustrado na figura 4.16. Observe que existe um caminho cíclico $\{\{4N, 7Y\}, d, \{4N, 7Y\}\}$ contendo estado com rótulo Y e portanto, esse estado é marcado. No passo 8.1, obtém-se o autômato $\hat{V}_1^{12}$ em que os eventos σ' são renomeados, conforme ilustra a figura 4.17. Para calcular o autômato $\hat{V}_2^{21}$, foram obtidos, primeiramente, os autômatos $G^1_{N_2}$, $G^2_{F_2}$ e $G^1_{R_2}$, os quais são apresentados na figura 4.18 (a), (b) e (c), respectivamente. Aplicando o passo 5 do algoritmo 4.2, foi construído o verificador V_2^{21} relativo ao local de diagnose 2 e mostrado na figura 4.19. Nesse autômato, seguindo o passo 7 do algoritmo 4.2, pode-se observar a existência do caminho cíclico, $\{\{4N, 6Y\}, c, \{4N, 6Y\}\}$ em que o estado $\{4N, 6Y\}$ é marcado. Renomeando os eventos σ' desse autômato, obtém-se o autômato $\hat{V}_2^{21}$, mostrado na figura 4.20.

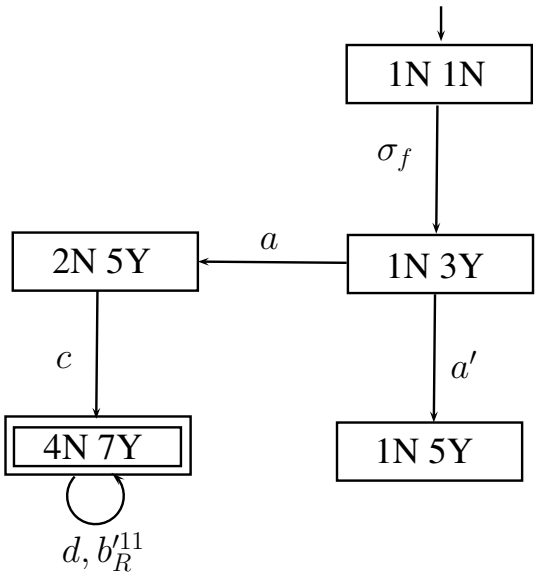


Figura 4.16: Autômato verificador V_1^{21}.

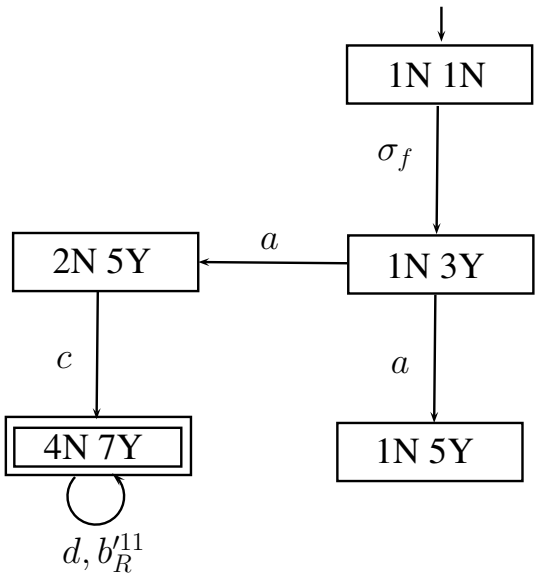


Figura 4.17: Autômato $\hat{V}_1^{21}$.

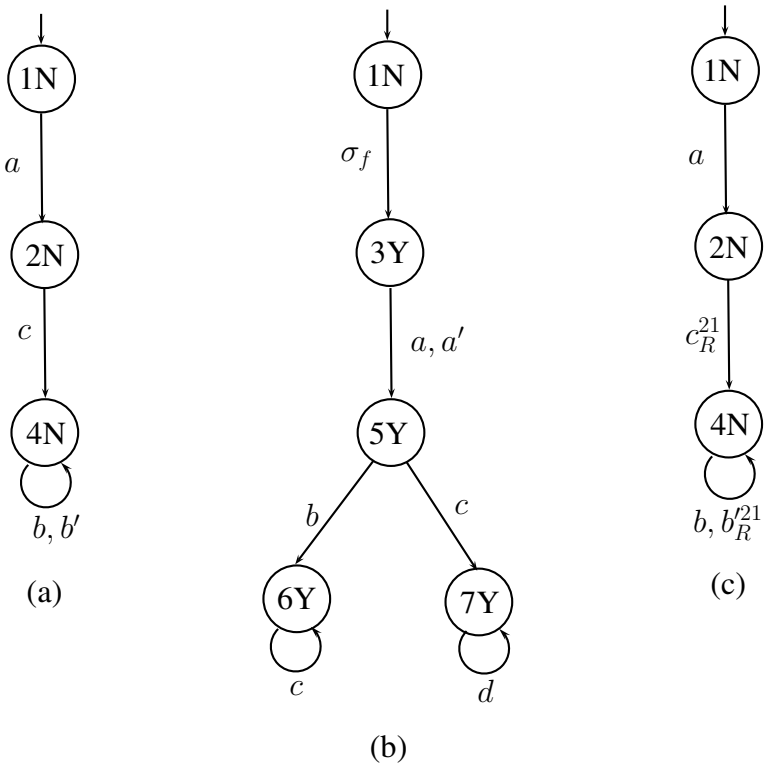


Figura 4.18: Autômato $G^1_{N_2}$ (a), Autômato $G^2_{F_2}$ (b) e Autômato $G^1_{R_2}$ (c).

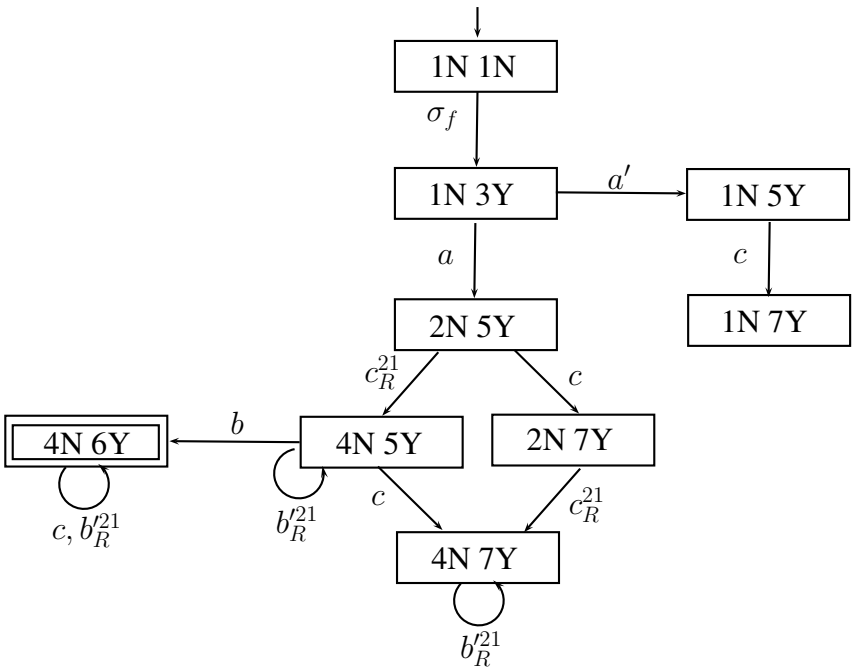


Figura 4.19: Autômato V_2^{21}.

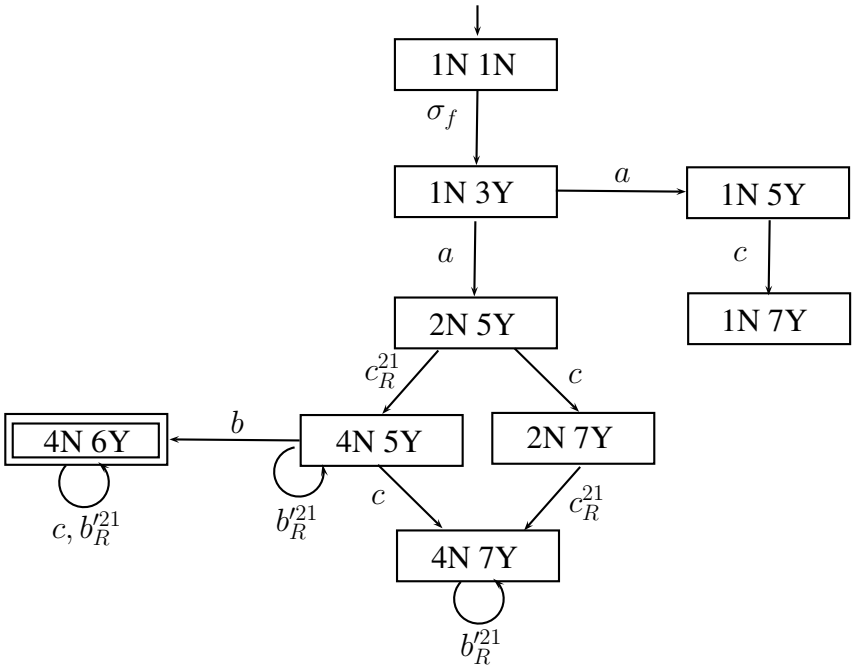


Figura 4.20: Autômato $\hat{V}_2^{21}$.

No passo 8.2 do 4.2 é construído o autômato V^{21} e como pode ser observado na figura 4.21 ele não possui nenhum estado marcado e portanto, nenhum caminho cíclico que atenda a condição posta no passo 9 do 4.2.

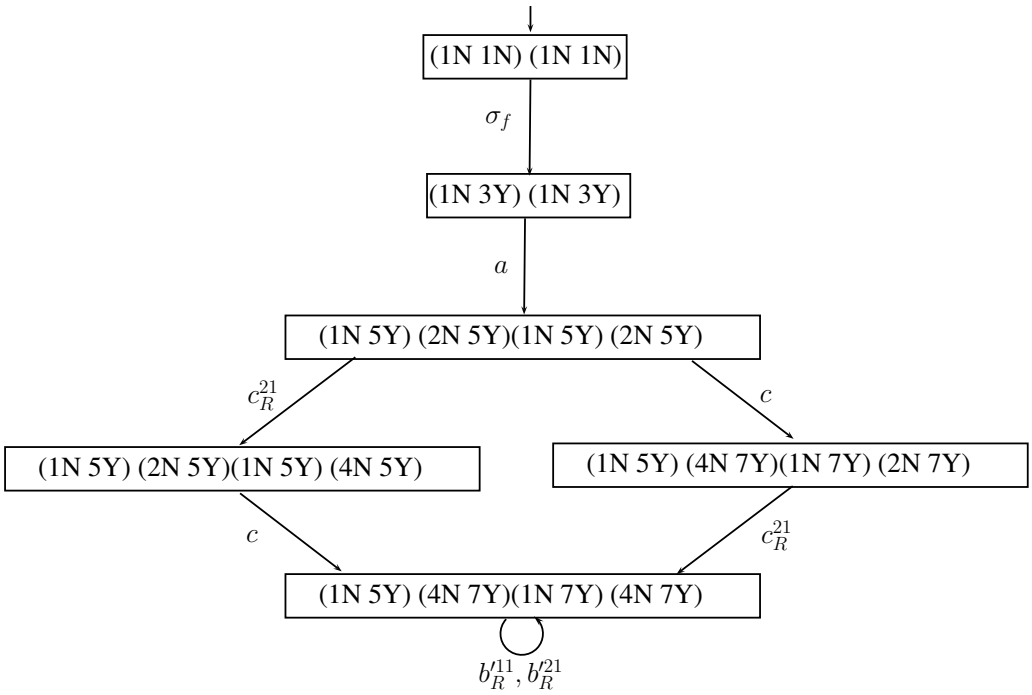


Figura 4.21: Autômato V^{21}.

Para verificar se os conjuntos $\Sigma^1_{nilo_1} = \{a, c\}$ *e* $\Sigma^1_{nilo_2} = \{a, d\}$*;* $\Sigma^2_{nilo_1} = \{b, c\}$ *e* $\Sigma^2_{nilo_2} = \{b, d\}$ *são bases para a codiagnose robusta a perdas intermitentes de observação associada aos eventos b e a, respectivamente, deve-se se construir os verificadores* V^{11} *e* V^{22} *seguindo os mesmos passos do algoritmo 4.2 para a construção dos verificadores* V^{12} *e* V^{21}*. Assim, para construir o verificador* V^{11} *são calculados, primeiramente, os verificadores* $G^1_{N_1}$*,* $G^1_{F_1}$ *e* $G^1_{R_1}$ *mostrados na figura 4.22 (a), (b) e (c), respectivamente.*

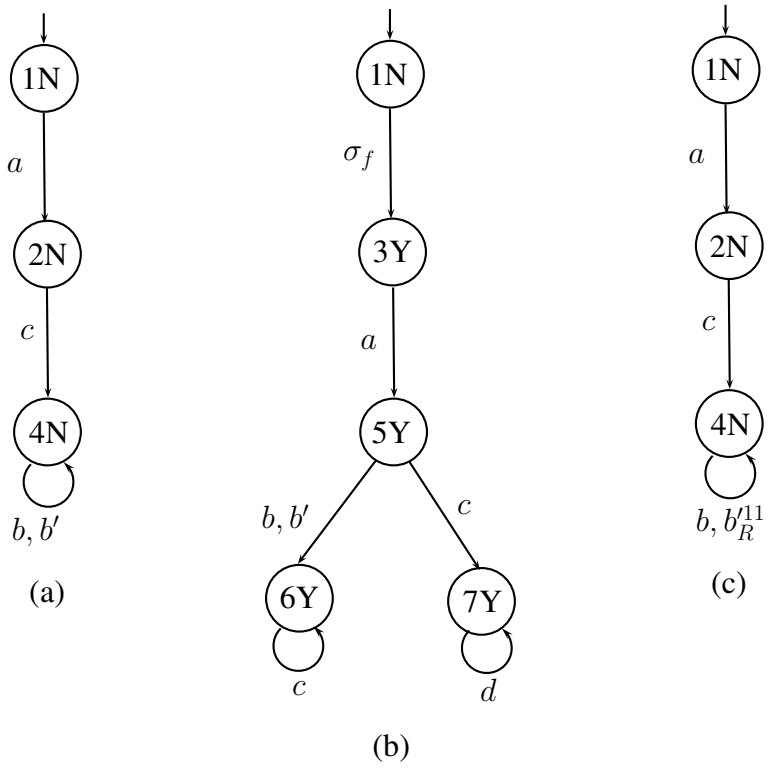


Figura 4.22: Autômato $G^1_{N_1}$ (a), Autômato $G^1_{F_1}$ (b) e Autômato $G^1_{R_1}$ (c).

Os verificadores V^{11}_1 *e* V^{11}_2 *são mostrados na figura 4.23 (a) e (b), respectivamente. Após a renomeação dos eventos* σ'*, são obtidos os verificadores* $\hat{V}^{11}_1$ *e* $\hat{V}^{11}_2$ *mostrados na figura 4.24 (a) e (b), respectivamente.*

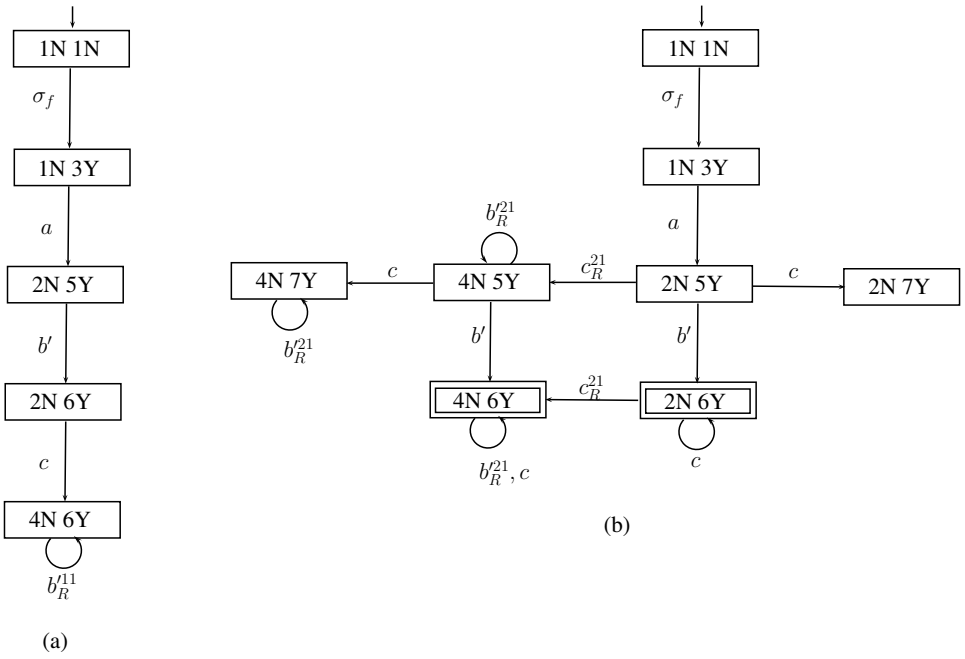


Figura 4.23: Autômato verificador V_1^{11} (a) e Autômato verificador V_2^{11} (b).

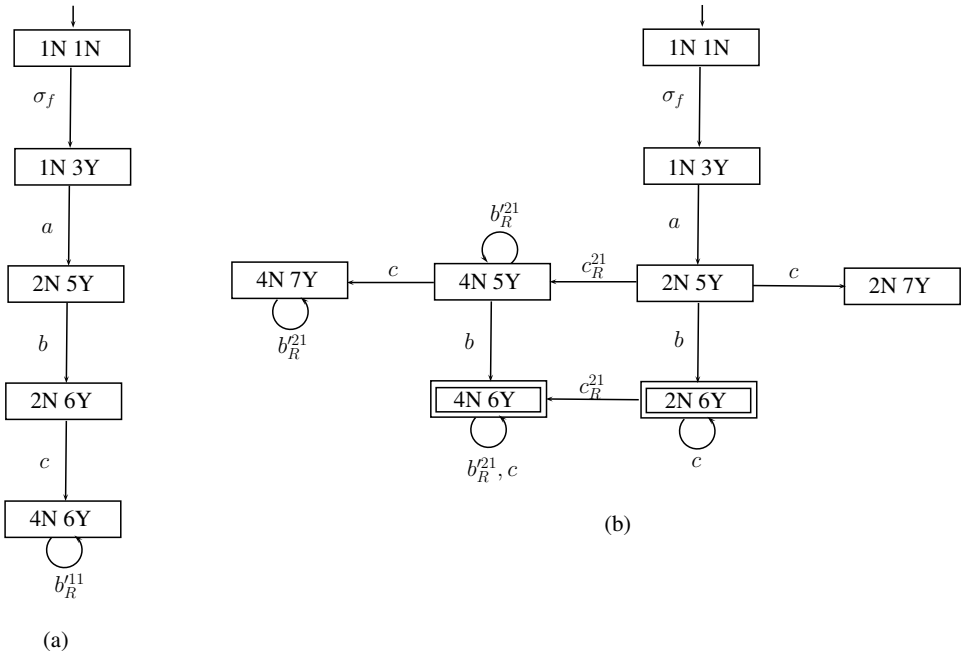


Figura 4.24: Autômato verificador $\hat{V}_1^{11}$ (a) e Autômato verificador $\hat{V}_2^{11}$ (b).

A figura 4.25 mostra o verificador $V^{11} = \hat{V}_1^{11}||\hat{V}_2^{11}$. *Observe que não existe nenhum caminho cíclico que atenda a condição posta no passo 9 do algoritmo 4.2 e portanto, o conjunto* $\Sigma^1_{nilo_1} = \{a, c\}$ *e* $\Sigma^1_{nilo_2} = \{a, d\}$ *é base para a codiagnose*

robusta a perdas intermitentes de observação associada ao evento b.

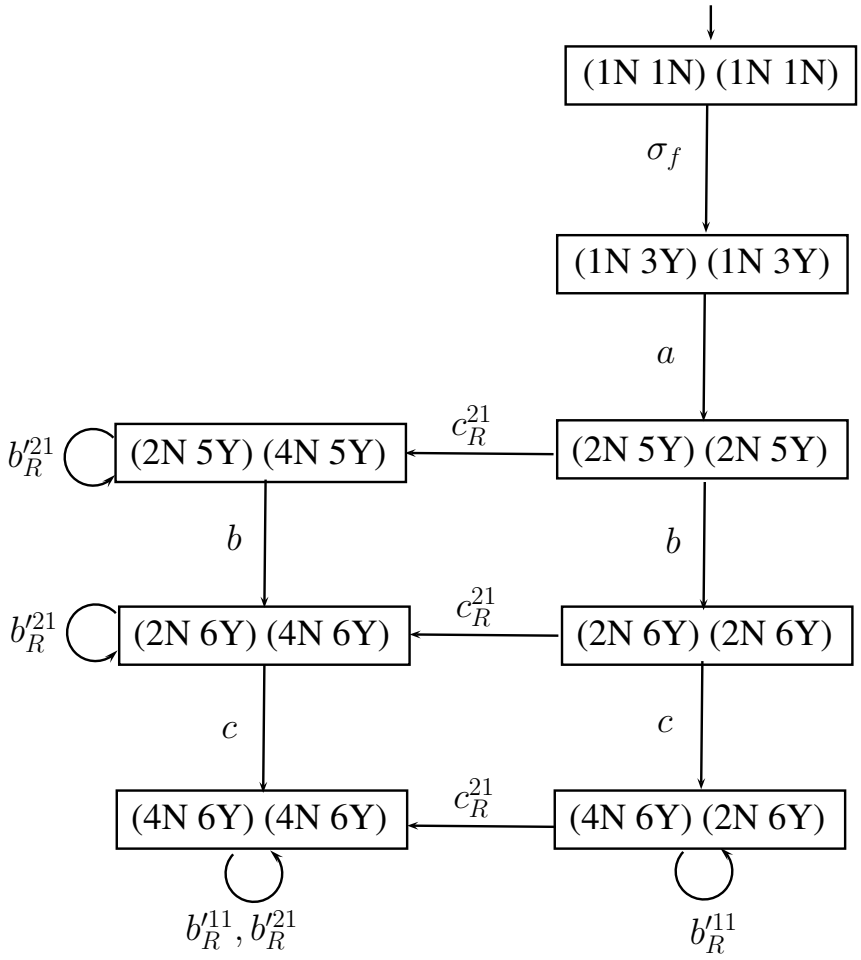


Figura 4.25: Autômato verificador V^{11}.

Para construir o verificador V^{22} são calculados, primeiramente, os verificadores $G^2_{N_1}$, $G^2_{F_1}$ e $G^2_{R_1}$ mostrados na figura 4.26 (a), (b) e (c), respectivamente.

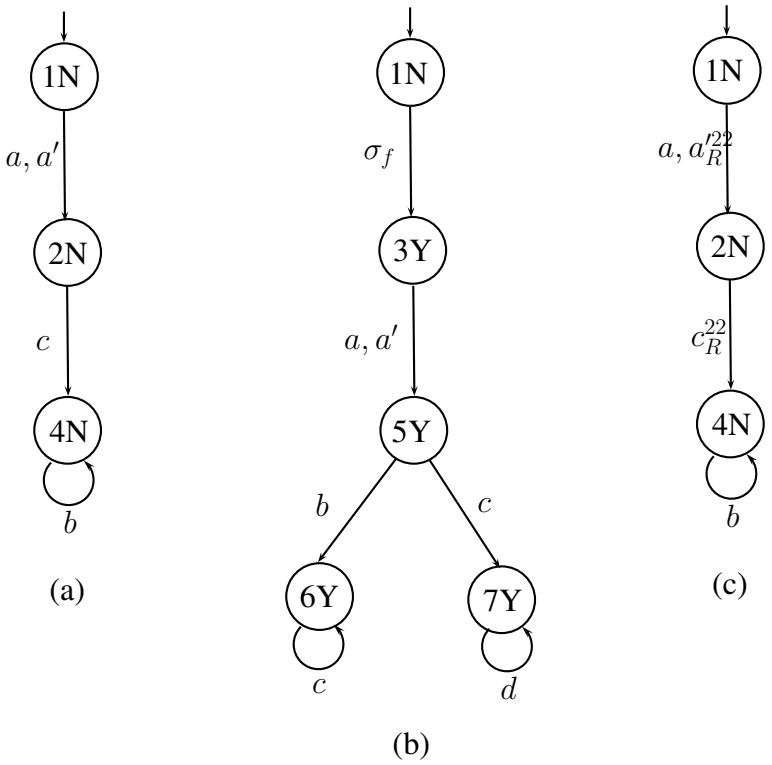


Figura 4.26: Autômato $G^2_{N_1}$ (a), Autômato $G^2_{F_1}$ (b) e Autômato $G^2_{R_1}$ (c).

Os verificadores V_1^{22} e V_2^{22} são mostrados na figura 4.27 e 4.28, respectivamente. Após a renomeação dos eventos σ', são obtidos os verificadores $\hat{V}_1^{22}$ e $\hat{V}_2^{22}$ mostrados na figura 4.29 e 4.30, respectivamente.

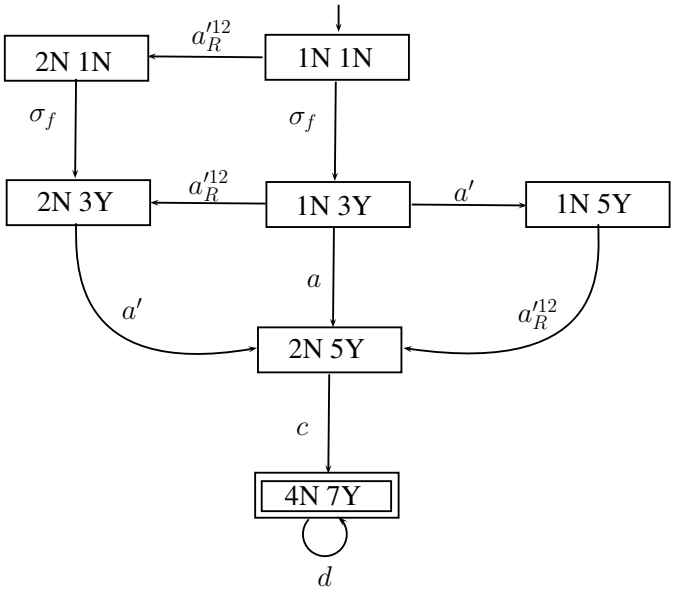


Figura 4.27: Autômato verificador V_1^{22}.

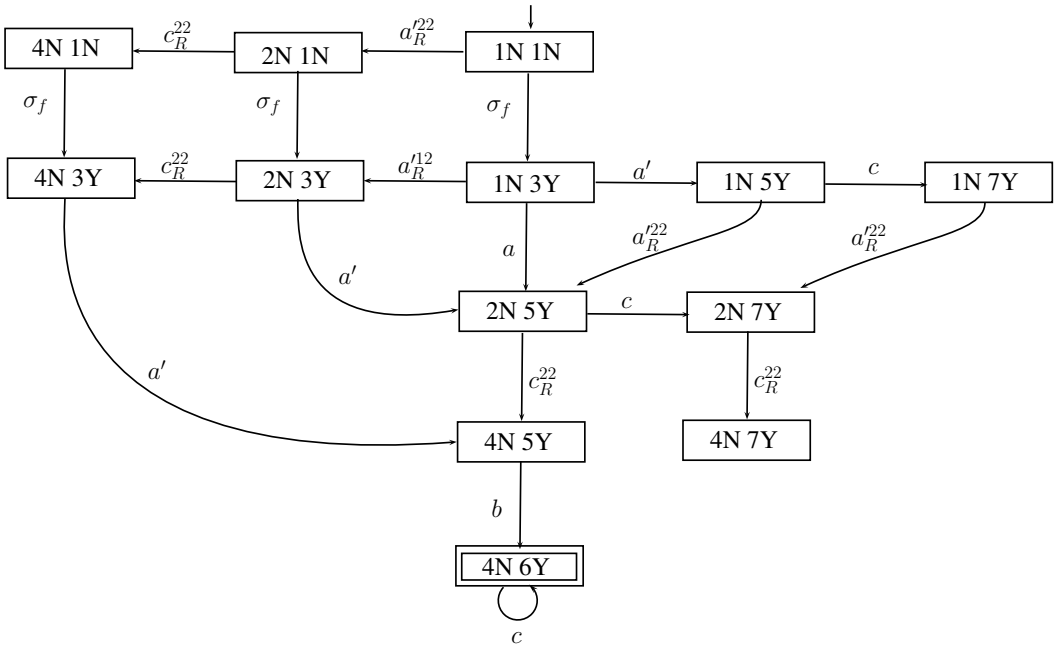


Figura 4.28: Autômato verificador V_2^{22}.

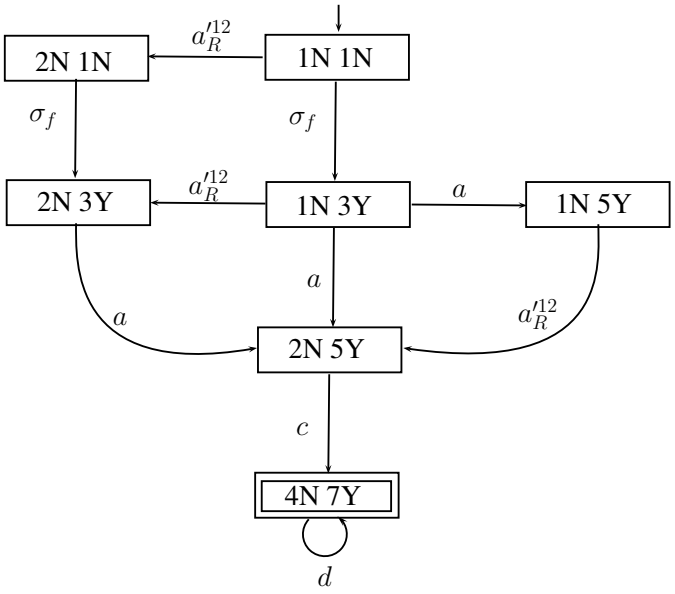


Figura 4.29: Autômato verificador $\hat{V}_1^{22}$.

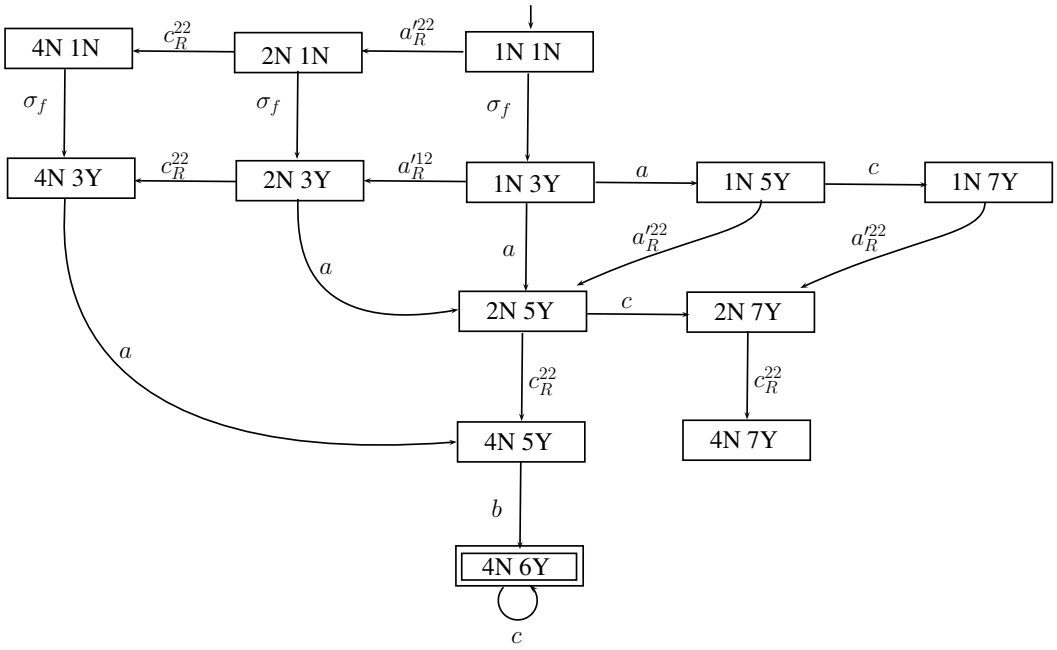


Figura 4.30: Autômato verificador $\hat{V}_2^{22}$.

A figura 4.31 mostra o verificador $V^{22} = \hat{V}_1^{22} || \hat{V}_2^{22}$. *Observe que não existe nenhum caminho cíclico que atenda a condição posta no passo 9 do algoritmo 4.2 e portanto, o conjunto* $\Sigma^2_{nilo_1} = \{b, c\}$ *e* $\Sigma^2_{nilo_2} = \{b, d\}$ *é base para a codiagnose robusta a perdas intermitentes de observação associada ao evento* a.

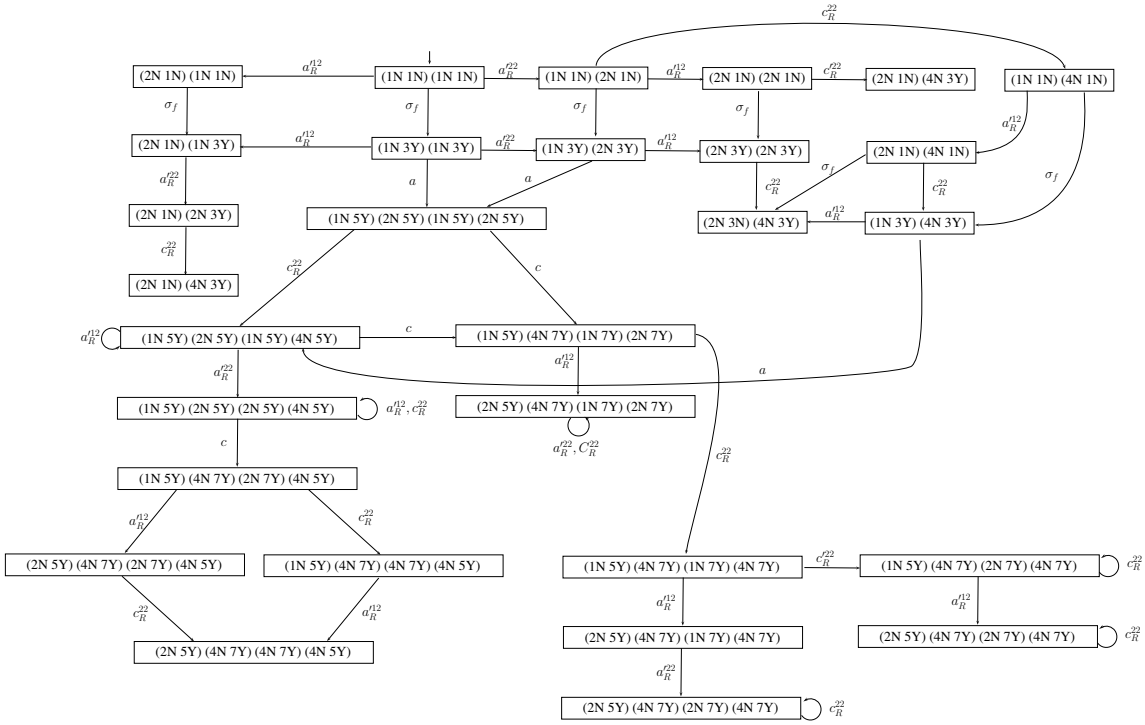


Figura 4.31: Autômato verificador V^{22}.

Como nos autômatos verificadores V^{12}, V^{21}, V^{11} e V^{22} não foi encontrado nenhum caminho cíclico que viola a condição de codiagnosticabilidade robusta a perdas intermitentes de observação associada aos eventos a e b, então a linguagem L do sistema é robustamente codiagnosticável a perdas intermitentes de observação associada aos eventos a e b.

□

Para calcular a complexidade computacional do algoritmo de verificação 4.2, vamos apresentar a complexidade para construção de cada autômato necessário para a verificação. Os autômatos G_N e G_F possuem, no pior caso, número de estados $|X|$ e $2|X|$, respectivamente. Além disso, como são determinísticos, então os números máximos de transições de G_N e G_F são $|X||\Sigma|$ e $2|X||\Sigma|$, respectivamente. Após a dilatação dos autômatos G_N e G_F, obtém-se os autômatos $G^l_{N_i}$ e $G^k_{F_i}$, que possuem o mesmo número de estados de G_N e G_F, respectivamente, mas que possuem número de transições, no pior caso, igual ao dobro do número de transições de G_N e G_F, isto

é, os números de transições de $G^l_{N_i}$ e $G^k_{F_i}$ são, respectivamente, $2|X||\Sigma|$ e $4|X||\Sigma|$. O autômato $G^l_{R,i}$ possui o mesmo número de estados e transições de $G^l_{N,i}$. Como o autômato V^{kl}_i é obtido pela composição paralela $G^l_{R,i} \| G^k_{F,i}$, então os números máximos de estados e transições de V^{kl}_i são, respectivamente, $2|X|^2$ e $8|X|^2|\Sigma|$. Por fim, o autômato V^{kl} é obtido fazendo a composição paralela $V^{kl} = \hat{V}^{kl}_1 \| \ldots \| \hat{V}^{kl}_n$. Como cada autômato $\hat{V}^{kl}_i$ possui o mesmo número de estados e transições que V^{kl}_i, então, no pior caso, os números de estados e transições de V^{kl} são, respectivamente, $2^n|X|^{2n}$ e $2 \times n \times 4^n|X|^{4n}|\Sigma|$. Como todos os pares $(k,l) \in I_m \times I_m$ têm que ser testados, então a complexidade do algoritmo de verificação é $O(m^2 \times n \times 4^n|X|^{4n}|\Sigma|)$, ou seja, considerando o número de diagnosticadores locais n e o número de bases para a CRPIO constantes, a complexidade computacional do algoritmo 4.2 é de tempo polinomial no número de estados e de transições, mas exponencial no número de diagnosticador locais.

4.4 Cálculo do limite de atraso para a CRPIO

O cálculo do limite de atraso para o caso de codiagnose robusta a perdas intermitentes de observação segue os mesmos passos do cálculo do limite de atraso para a codiagnose robusta a perdas permanentes de observação, visto na seção 3.3, em que os algoritmos de busca em profundidade, ordenação topológica, busca por componentes fortemente conexos e de cálculo do maior caminho devem ser aplicados com relação ao autômato verificador V^{kl}, calculado no passo 8.2 do algoritmo 4.2, quando a linguagem L de G for robustamente codiagnosticável a perdas intermitentes de observação.

4.5 Comentários finais

Neste capítulo foi resolvido o problema de CRPIO com uma abordagem diferente daquela considerada em [63]. Diferentemente do trabalho realizado em [63], as perdas de observação dos eventos em cada base podem ocorrer livremente entre os eventos

sujeitos a perdas intermitentes. Portanto, foi apresentada uma nova definição para a função de dilatação da linguagem, para base para a codiagnose assim como uma nova definição para o problema de CRPIO. Neste capítulo, foi construído um novo algoritmo com base no apresentado em [70], inserindo mudanças significativas que levaram à elaboração de um novo teorema de codiagnosticabilidade robusta considerando perdas intermitentes de observação, usando verificadores. A complexidade computacional do novo algoritmo é de tempo polinomial no número de estados e de transições, considerando os demais parâmetros constantes. Finalmente, foi apresentado um algoritmo em tempo polinomial para o cálculo do limite de atraso para a CRPIO.

Capítulo 5

Conclusão e trabalhos futuros

Nos trabalhos realizados até aqui com relação à diagnose de falhas em arquiteturas descentralizadas (codiagnose), a planta do sistema é suposta ser completamente conhecida, e os sensores responsáveis por registrar a ocorrências dos eventos observáveis do sistema não podem ser perdidos na sua comunicação com os diagnosticadores a eles associados. Neste trabalho, foi proposto e resolvido, pela primeira vez na literatura, o problema de codiagnosticabilidade robusta de sistemas a eventos discretos sujeitos a perdas permanentes e intermitentes de observação. Antes de definir o problema em questão, foi feita uma extensão de base para a diagnose robusta introduzida em [59] para o caso de base para a codiagnose robusta. As bases são sujeitas às incertezas devidas às possibilidades de perdas de observação do sistema. Após definir as bases, foi possível estender a definição de diagnosticabilidade robusta para a definição de CRPPO. Na abordagem deste trabalho, foi considerado o protocolo 3 de [22] em que não é permitida a comunicação entre os diagnosticadores locais. Com base no verificador em [70] foi apresentado o algoritmo 3.1, a tempo polinomial com relação ao número de estados e transições, para a construção de um autômato verificador para o problema de codiagnosticabilidade robusta de sistemas a eventos discretos sujeitos a perdas permanentes de observação [76], [77].

Outro avanço obtido neste trabalho foi com relação ao cálculo do limite de atraso para a CRPPO. Quando a linguagem do sistema é robustamente codiagnosticável com relação à perdas permanentes de observações, o valor $z^\star$ do limite de atraso

calculado no algoritmo 3.5 informa quantos eventos são necessários observar, após a ocorrência do evento de falha no sistema, para diagnosticar a falha no sistema.

Uma outra contribuição importante desta tese foi obtida com relação ao problema de CRPIO com uma abordagem diferente daquela considerada em [63]. Um exemplo considerando as duas abordagens foi apresentado. Uma nova definição de CRPIO, diferente da apresentada em [63], foi proposta, a qual permite considerar perdas de observação devido a falha de comunicação da ocorrência de um evento para um diagnosticador local, e permite considerar o caso em que há incerteza com relação à base correta para a codiagnose visto que as perdas de observação dos eventos em cada base podem ocorrer livremente entre os eventos sujeitos à perda intermitente de observação. Foi apresentada uma nova definição para a função de dilatação da linguagem, uma nova definição de base para a codiagnose e uma nova definição para o problema de CRPIO. Um novo algoritmo com complexidade a tempo polinomial no número de estados e de transições, considerando os demais parâmetros constantes, foi desenvolvido, usando verificadores. Diferentemente do trabalho em [63], foi inserido neste trabalho o cálculo do limite de atraso para a CRPIO usando um algoritmo de tempo polinomial no número de estados e de transições, considerando os demais parâmetros constantes.

Os objetivos alcançados neste trabalho são resumidos a seguir:

- Foi proposto e resolvido, pela primeira vez na literatura, o problema de CRPPO.
- Foi obtida a definição de base para a CRPPO.
- Foi apresentado um algoritmo, em tempo polinomial para a construção do verificador para o problema de CRPPO.
- Foi construído um algoritmo em tempo polinomial para o Cálculo do limite de atraso no problema de CRPPO.
- Foi resolvido, o problema de CRPIO com uma abordagem diferente daquele apresentada em [63] que do ponto de vista prático se tornou muito restritivo

com relação à possibilidade de falha das observações associadas aos eventos do sistema.

- Foram obtidas novas definições da função de dilatação da linguagem, de base para a codiagnose e para o problema de CRPIO.

- Foram desenvolvidos algoritmos em tempo polinomial para a construção do verificador e para calcular o limite de atraso para o problema de CRPIO.

Como sugestões de trabalhos futuros podemos citar: (*i*) o desenvolvimento de técnicas para a diagnose robusta descentralizada utilizando os protocolos 1 e 2 de [22]; (*ii*) estender o método proposto neste trabalho para a prognose de falhas em sistemas a eventos discretos, isto é, prever a possibilidade da ocorrência de um evento de falha, baseado na observação dos eventos executados pelo sistema, supondo perdas permanentes ou intermitentes de observação; (*iii*) relaxamento da hipótese **H2**, isto é, considerar que as perdidas de observação dos sensores/comunicação podem também ocorrer após a primeira ocorrência dos eventos registrados pelos sensores com mau funcionamento. Essa hipótese foi recentemente relaxada em [65] para o problema de diagnose robusta a perdas permanentes de observação de eventos utilizando-se autômatos de Mealy com máscaras de observação dependentes do estado. Porém, o tamanho do autômato de Mealy calculado em [65] pode crescer exponencialmente com o número de eventos cujas observações podem ser perdidas. Um possível trabalho futuro seria estender esse problema para o caso descentralizado, e também desenvolver algoritmos em tempo polinomial, ou métodos para mitigar a complexidade computacional, para a verificação da codiagnosticabilidade de sistemas a eventos discretos sujeitos a perdas permanentes de observação que possam ocorrer também após a primeira observação do evento.

Referências Bibliográficas

[1] "Cube Assembly Compact Pneumatic Fully assembled". Disponível em: <`http://www.christiani.eu/product_info.php/products_id/2972`>. Acessado em 24 de junho de 2016.

[2] ZAYTOON, J., LAFORTUNE, S. "Overview of fault diagnosis methods for Discrete Events Systems", *Annual Reviews in Control*, v. 37, pp. 308–320, 2013.

[3] GERTLER, J. *Fault detection and diagnosis in engineering systems*. New York, Marcel Dekker, 1998.

[4] KINNAERT, M., LUNZE, M., STAROSWIECKI, M. *Diagnosis and fault-tolerant control*. Germany, Springer, 2006.

[5] HAMSCHER, W., CONSOLE, L., KLEER, J. *Readings in model-based diagnosis*. Morgan Kaufmann, 1992.

[6] PATTON, R. J., FRANK, P. M., CLARK, R. *Issues of fault diagnosis for dynamic systems*. Springer, 2000.

[7] ANGELII, C. "Online expert systems for fault diagnosis in technical processes", *Expert Systems*, v. 25, n. 2, pp. 115–132, 2008.

[8] BISWAS, G., CORDIER, M. O., LUNZE, J., et al. "Diagnosis of complex systems: bridging the methologies of the FDI and DX communities", *IEEE Transactions on Systems, Man, and Cybernetics - Part b: Cybernetics*, v. 34, n. 5, pp. 2159–2162, 2004.

[9] CASSEZ, F., TRIPAKIS, S. "Fault diagnosis with static and dynamic observers", *Fundamenta Inform.*, v. 88, n. 4, pp. 497–540, 2008.

[10] DAVID, R., ALLA, H. *Discrete, Continuous and Hybrid Petri Nets*. Springer, 2005.

[11] HOPCROFT, J. E., MOTWANI, R., ULLMAN, J. D. *Introduction to automata theory, languages, and computation*. Addison Wesley, 2013.

[12] CAINES, P., GREINER, R., WANG, S. "Classical and logic based dynamic observers for finite automata", *IMA Journal of Mathematical Control and Information*, pp. 45–80, 1991.

[13] CIESLAK, R., DESCLAUX, D., FAWAZ, A., et al. "Supervisory control of discrete-event processes with partial observations", *IEEE Transactions Automatic Control*, v. 33, pp. 249–260, 1988.

[14] LIN, F., WONHAM, W. "On observability of discrete-event systems", *Information Sciences*, v. 44, pp. 173–198, 1988.

[15] OZVEREN, C. M., WILLSKY, A. S. "Observability of discrete event dynamic systems", *IEEE Transactions on Automatic Control*, v. 35, n. 7, pp. 797–806, 1990.

[16] RAMADGE, P. J. "Observability of discrete-event systems", *In Proc. 25th IEEE conference decision and control*, pp. 1108–1112, 1986.

[17] LIN, F. "Diagnosability of discrete event systems and its applications", *Journal of Discrete Event Dynamic Systems*, v. 4, n. 2, pp. 197–212, 1994.

[18] SAMPATH, M., SENGUPTA, R., LAFORTUNE, S., et al. "Diagnosability of discrete-event systems", *IEEE Trans. on Automatic Control*, v. 40, n. 9, pp. 1555–1575, 1995.

[19] SAMPATH, M., SENGUPTA, R., LAFORTUNE, S., et al. "Failure diagnosis using discrete-event models", *IEEE Trans. on Control Systems Technology*, v. 4, n. 2, pp. 105–124, 1996.

[20] QIU, W., KUMAR, R. "Distributed diagnosis under bounded-delay communication of immediately forwarded local observations", *IEEE Transactions on Automatic Control*, v. 38, pp. 628–643, 2008.

[21] BASILIO, J. C., LIMA, S. T. S., LAFORTUNE, S., et al. "Computation of minimal event bases that ensure diagnosability", *Discrete Event Dynamic Systems: Theory And Applications*, v. 22, pp. 249–292, 2012.

[22] DEBOUK, R., LAFORTUNE, S., TENEKETZIS, D. "Coordinated decentralized protocols for failure diagnosis of discrete event systems", *Discrete Event Dynamic Systems: Theory and Applications*, v. 10, n. 1, 2000.

[23] BOUYER, P., CHEVALIER, F., D'SOUZA, D. "Fault diagnosis using timed automata", *In Proc. 8th International Conference on Foundations of Software Science and Computation Structures*, , n. 3441, pp. 219–233, 2005.

[24] CASSEZ, F. "A note on fault diagnosis algorithms", *In Proc. 48th IEEE conference on decision and control and 28th Chinese control conference*, 2009.

[25] CHEN, Y. L., PROVAN, G. "Modelling and diagnosis of timed discrete event systems", *In Proc. American control conference*, pp. 31–36, 1997.

[26] JIANG, S., KUMAR, R. "Diagnosis of repeated failures for discrete event systems with linear-time temporal-logic specifications", *IEEE Transactions Automation Science and Engineering*, v. 3, n. 1, pp. 47– 59, 2006.

[27] TRIPAKIS, S. "Fault diagnosis for timed automata", *In Proc. int. conf. on formal techniques in real time and fault tolerant systems*, v. 24, pp. 205–224, 2002.

[28] ZAD, S., KWONG, R., WONHAM, W. "Fault diagnosis in discrete-event systems: Incorporating timing information", *IEEE Transactions Automatic Control*, v. 50, n. 7, pp. 1010–1015, 2005.

[29] ATHANASOPOULOU, E., HADJICOSTIS, C. N. "Probabilistic approaches to fault detection in networked discrete event systems", *IEEE Transactions Neural Networks*, v. 16, n. 5, pp. 1042–1052, 2005.

[30] FABRE, E., JEZEQUEL, L. "On the construction of probabilistic diagnosers", *In Proc. 10th international workshop on discrete event systems*, 2010.

[31] THORSLEY, D., TENEKETZIS, D. "Diagnosability of stochastic discrete-event systems", *IEEE Trans. on Automatic Control*, v. 50, pp. 476–492, 2005.

[32] THORSLEY, D., YOO, T. S., GARCIA, H. E. "Diagnosability of stochastic discreteevent systems under unreliable observations", *In Proc. American control conference*, pp. 1158–1165, 2008.

[33] BASILE, F., CHIACCHIO, P., DE TOMMASI, G. "Sufficient conditions for diagnosability of Petri nets", *In Proc. 9th international workshop on discrete event systems*, pp. 436–442, 2008.

[34] BASILE, F., CHIACCHIO, P., DE TOMMASI, G. "An efficient approach for online diagnosis of discrete event systems", *IEEE Transactions on Automatic Control*, v. 54, n. 4, pp. 748–759, 2009.

[35] CABASINO, M. P., GIUA, A. N., SEATZU, C. "Fault detection for discrete event systems using Petri nets with unobservable transitions", *Automatica*, v. 46, n. 9, pp. 1531–1539, 2010.

[36] DOTOLI, M., FANTI, M., MANGINI, A. "Fault detection of DES by Petri nets and integer linear programming", *Automatica*, v. 45, n. 11, pp. 2665–2672, 2009.

[37] FANTI, M., MANGINI, A. M., WALTER, U. "Fault detection by labelled Petri nets and time constraints", *In Proc. 3rd international workshop on dependable control of Discrete Systems*, pp. 170–175, 2011.

[38] GENC, S., LAFORTUNE, S. "Distributed diagnosis of place-bordered Petri nets", *IEEE Transactions Automation Science and Engineering*, v. 4, n. 2, pp. 206–219, 2007.

[39] RAMIREZ-TREVINO, A., RUIZ-BELTRAN, E., RIVERA-RANGEL, I., et al. "Online fault diagnosis of discrete event systems. A Petri net-based approach", *IEEE Transactions on Automation Science and Engineering*, v. 4, n. 1, pp. 31–39, 2007.

[40] CHUNG, S. L. "Diagnosing PN-based models with partial observable transitions", *International Journal of Computer Integrated Manufacturing*, v. 18, pp. 158–169, 2005.

[41] LEFEBVRE, D., DELHERM, C. "Diagnosis of DES with Petri net models", *IEEE Transactions Automation Science and Engineering*, v. 4, n. 1, pp. 114 –118, 2007.

[42] MIYAGI, P. E., RIASCOS, L. A. M. "Modeling and analysis of fault-tolerant systems for machining operations based on Petri nets", *Control Engineering Practice*, v. 14, n. 4, pp. 397–408, 2010.

[43] USHIO, T., ONISHI, L., OKUDA, K. "Fault detection based on Petri net models with faulty behaviors", *In Proceedings SMC'98 IEEE inteL. rnational conference on systems, man, and cybernetics*, pp. 113–118, 1998.

[44] WU, Y., HADJICOSTIS, C. N. "Algebraic approach for fault identification in discrete-event systems", *IEEE Transactions Robotics and Automation*, v. 50, pp. 2048–2053, 2005.

[45] CABASINO, M. P., GIUA, A., POSSI, M., et al. "Discrete event diagnosis using labeled Petri nets. An application to manufacturing systems." *Control Engineering Practice*, v. 19, n. 9, pp. 989–1001, 2011.

[46] BENVENISTE, A., FABRE, E., HAAR, S., et al. "Diagnosis of asynchronous discrete event systems: A net unfolding approach", *IEEE Transactions Automatic Control*, v. 48, n. 5, pp. 714–727, 2003.

[47] FABRE, E., BENVENISTE, A., HAAR, S., et al. "Distributed monitoring of concurrent and asynchronous systems", *Discrete Event Dynamic Systems*, v. 15, n. 1, pp. 33–84, 2005.

[48] JIROVEANU, G., BOEL, R. "On-line monitoring of large Petri net models under partial observation", *Discrete Event Dynamic Systems*, v. 18, pp. 323–354, 2008.

[49] JIROVEANU, G., BOEL, R. "The diagnosability of Petri net models using minimal explanations", *IEEE Transactions Automatic Control*, v. 55, n. 7, pp. 1663–1668, 2010.

[50] AGHASARYAN, A., FABRE, E., BENVENISTE, A., et al. "Fault detection and diagnosis in distributed systems: An approach by partially stochastic Petri nets", *Discrete Event Dynamic Systems*, v. 8, n. 2, pp. 203–231, 1998.

[51] MAHULEA, C., SEATZU, C., CABASINO, M. P., et al. "Fault diagnosis of discrete-event systems using continuous Petri nets", *IEEE Trans SMC: Part A*, v. 42, n. 4, pp. 970–984, 2012.

[52] JIANG, S., HUANG, Z., CHANDRA, V., et al. "A polynomial algorithm for testing diagnosability of discrete-event systems", *Automatic Control, IEEE Transactions on*, v. 46, n. 8, pp. 1318 – 1321, 2001.

[53] YOO, T. S., LAFORTUNE, S. "Polynomial- time verification of diagnosability of partially observed discrete-event systems", *IEEE Trans. Automat. Contr*, v. 47, n. 8, pp. 1491 – 1495, 2002.

[54] CABRAL, F. G., MOREIRA, M. V., DIENE, O., et al. "A Petri net diagnoser for discrete event systems modeled by finite state automata", *IEEE Transactions on Automatic Control*, v. 60, pp. 59–71, 2015.

[55] QIU, W., KUMAR, R. "Decentralized failure diagnosis of discrete event systems", *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans,*, v. 36, n. 2, pp. 384 – 395, 2006.

[56] WANG, Y., YOO, T.-S., LAFORTUNE, S. "Diagnosis of discrete event systems using decentralized architectures", *Discrete Event Dynamic Systems: Theory And Applications*, v. 17, pp. 233–263, 2007.

[57] BASILIO, J. C., LAFORTUNE, S. "Robust codiagnosability of discrete event", *in Proc 2009 American Control Conference*, pp. 2202 – 2209, 2009.

[58] CARVALHO, L. K., BASILIO, J. C., MOREIRA, M. V. "Robust diagnosability of discrete event systems subject to intermittent sensor failures". In: *Preprints of the 10th International Workshop on Discrete Event Systems*, pp. 94–99, Berlin, Germany, 2010.

[59] LIMA, S. T. S., BASILIO, J. C., LAFORTUNE, S., et al. "Robust diagnosis of discrete-event systems subject to permanent sensor failures". In: *Preprints of the 10th International Workshop on Discrete Event Systems*, pp. 100–107, Berlin, Germany, 2010.

[60] TAKAI, S., KUMAR, R. "Decentralized diagnosis for nonfailures of discrete event systems using inference-based ambiguity management", *IEEE Transactions on Systems, Man and Cybernetics Part A*, v. 40, n. 2, pp. 406–412, 2010.

[61] CARVALHO, L. K., MOREIRA, M. V., BASILIO, J. C. "Generalized robust diagnosability of discrete event systems". In: *18th IFAC World Congress*, pp. 8737–8742, Milano, Italy, 2011.

[62] TAKAI, S. "Verification of robust diagnosability for partially observed discrete event systems", *Automatica*, v. 48, pp. 1913–1919, 2012.

[63] CARVALHO, L. K., BASILIO, J. C., MOREIRA, M. V. "Robust diagnosis of discrete-event systems against intermittent loss of observations", *Automatica*, v. 48, n. 9, pp. 2068–2078, 2012.

[64] CARVALHO, L. K., MOREIRA, M. V., BASILIO, J. C., et al. "Robust diagnosis of discrete-event systems against permanent loss of observations", *Automatica*, v. 49, n. 1, pp. 223–231, 2013.

[65] KANAGAWA, N., TAKAI, S. "Diagnosability of discrete event systems subject to permanent sensor failures", *International Journal of Control*, 2015.

[66] TAKAI, S., USHIO, T. "Verification of Codiagnosability for Discrete-Event Systems Modeled by Mealy Automata with Nondeterministic Output Functions", *IEEE Transactions on Automatic Control*, v. 57, n. 3, pp. 798–803, 2012.

[67] CASSANDRAS, C., LAFORTUNE, S. *Introduction to Discrete Event System*. Secaucus, NJ, Springer-Verlag New York, Inc., 2008.

[68] RAMADGE, P. J., WONHAM, W. M. "The control of discrete event systems". In: *Proc. IEEE, Special Issue on Discrete Event Systems*, v. 77, pp. 81–98, 1989.

[69] BASILIO, J., CARVALHO, L. K., MOREIRA, M. V. "Diagnose de falhas em sistemas a eventos discretos modelados por autômatos finitos", *Revista Controle & Automação*, v. 21, n. 5, pp. 510–533, 2010.

[70] MOREIRA, M. V., JESUS, T. C., BASILIO, J. C. "Polynomial time verification of decentralized diagnosability of discrete event systems", *IEEE Transactions on Automatic Control*, pp. 1679–1684, 2011.

[71] MOREIRA, M. V., BASILIO, J. C., CABRAL, F. G. ""Polynomial Time Verification of Decentralized Diagnosability of Discrete Event Systems" versus "Decentralized Failure Diagnosis of Discrete Event Systems": a critical appraisal." *IEEE Transactions on Automatic Control*, v. 61, n. 1, pp. 178–181, 2016.

[72] WANG, W., GIRARD, A. R., LAFORTUNE, S., et al. "On codiagnosability and coobservability with dynamic observations", *Automatic Control, IEEE Transactions on*, v. 56, n. 7, pp. 1551–1566, 2011.

[73] CORMEN, T. H., LEISERSON, C. E., STEIN, R. L. R. *Introduction to algorithms.* Massachusetts, MIT Press, 2007.

[74] JOHNSON, D. B. "Finding all the elementary circuits of a directed graph", *In: SIAM J. Comput.*, pp. 77–84, 1975.

[75] SAFAR, K. A., ALBEHAIRY, S. "Counting Cycles in an Undirected Graph using DFS-XOR Algorithm", *IEEE Transactions on Automatic Systems, 2009*, pp. 132–139, 2009.

[76] TOMOLA, J. H. A., MOREIRA, M. V., BASÍLIO, J. C., et al. "Robust Codiagnosability of Discrete - Event Systems against Permanent Loss of Observations", *IEEE Conference on Science and Engineering, CASE*, pp. 813–818, 2015.

[77] TOMOLA, J. H. A., MOREIRA, M. V. B., CARVALHO, L. K. "Robust Disjunctive-Codiagnosability of Discrete-Event Systems Against Permanent Loss of Observations", *IEEE Transactions on Automatic Control, submetido para publicação*, 2016.

[78] DASGUPTA, S., PAPADIMITRIOU, C., VAZIRANI, U. *Algorithms.* McGraw-Hill, 2008.