DIAGNOSE DE FALHAS DE SISTEMAS A EVENTOS DISCRETOS COM TRANSIÇÕES PONDERADAS

Gustavo da Silva Viana

Dissertação de Mestrado 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 Mestre em Engenharia Elétrica.

Orientador: João Carlos dos Santos Basilio

Rio de Janeiro
Março de 2014

DIAGNOSE DE FALHAS DE SISTEMAS A EVENTOS DISCRETOS COM TRANSIÇÕES PONDERADAS

Gustavo da Silva Viana

DISSERTAÇÃO 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 MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

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

Prof. Marcos Vicente de Brito Moreira, D.Sc.

Prof. Rafael Santos Mendes, Dr.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2014

Viana, Gustavo da Silva

Diagnose de Falhas de Sistemas a Eventos Discretos com Transições Ponderadas/Gustavo da Silva Viana. – Rio de Janeiro: UFRJ/COPPE, 2014.

XIV, 81 p.: il.; 29, 7cm.

Orientador: João Carlos dos Santos Basilio

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2014.

Referências Bibliográficas: p. 78 – 81.

1. diagnose de falhas. 2. autômatos temporizados. 3. álgebra max plus. I. Basilio, João Carlos dos Santos. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

À minha mãe Claudia Marcia,
minha vó Julia Filomena e meu
avô José Cassimiro (in
memoriam).

Agradecimentos

Gostaria de agradecer primeiramente a Deus, porque é d'Ele, por Ele e para Ele são todas as coisas. Sem sua proteção e força durante toda a minha vida, nada seria possível.

À minha mãe Claudia Marcia da Silva, que não só me deu a vida, mas sempre supriu todas as minhas necessidades financeiras e de carinho. Sendo professora, ensinou-me o valor do conhecimento como melhor forma de crescimento pessoal, profissional e financeiro.

À minha vó Julia Filomena da Silva, que me criou desde pequeno juntamente com a minha mãe e me ensinou valores fundamentais como: honestidade, respeito e trabalho. Sempre confiou em mim e me incentivou a seguir meu sonho de fazer o mestrado.

Ao meu avô José Cassimiro da Silva (*in memoriam*), que me criou como um verdadeiro pai, sendo um grande exemplo de uma pessoa trabalhadora, honesta e simpática. Era tão bom, que Deus o levou para junto dele em 2002. Nunca esquecerei os momentos felizes que passamos juntos, estão gravados para sempre na minha mente e no meu coração.

Ao meu orientador João Carlos dos Santos Basilio, que confiou em mim e no meu trabalho, sempre me aconselhando, orientando com o carinho não de um chefe, mas de um pai. Seus ensinamentos serão levados por mim para a vida toda.

Aos professores Lilian Kawakami, Marcos Moreira e Leonardo Bermeo, pela paciência, amizade e apoio técnico.

A todos os colegas de laboratório e do mestrado, em especial a Marcos Vinicius, Rafael Pereira, Victor Alvarez e Felipe Radtke que estiveram comigo desde a época das matérias.

Aos meus amigos da graduação e que fazem mestrado em outras áreas: João Pedro Salvador, Renan Fernandes e Flávio Goulart pela amizade, ajuda e palavras de incentivo.

Por fim, gostaria de agradecer à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES, pelo suporte financeiro, sem o qual seria impossível concluir este trabalho.

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

DIAGNOSE DE FALHAS DE SISTEMAS A EVENTOS DISCRETOS COM TRANSIÇÕES PONDERADAS

Gustavo da Silva Viana

Março/2014

Orientador: João Carlos dos Santos Basilio

Programa: Engenharia Elétrica

Uma das maneiras de se abordar o problema da diagnose de falhas é por meio da construção de um modelo a eventos discretos do sistema cuja ocorrência de falhas deve ser diagnosticada. Nesse caso, a decisão sobre a ocorrência da falha (diagnose) é tomada considerando-se somente os eventos que tenham sido observados, isto é, registrados pelos sensores. Na prática, isso é feito utilizando-se diagnosticadores. Diagnosticadores são autômatos determinísticos cujos estados são conjuntos formados pelos estados do autômato do sistema (planta), juntamente com rótulos que indicam se a sequência de eventos ocorrida possui ou não o evento associado à falha e podem ser usados tanto na diagnose de falhas em tempo real quanto na verificação da diagnosticabilidade de uma linguagem. Neste trabalho é proposta uma nova abordagem para verificação da diagnosticabilidade de falhas baseada em um novo autômato chamado diagnosticador de teste, que permite verificar a diagnosticabilidade de uma linguagem por meio de busca por componentes fortemente conexas e não por ciclos, como usualmente feito na literatura. Essa abordagem é, em seguida, aplicada a autômatos ponderados para estimar o tempo máximo para diagnosticar uma falha após sua ocorrência. Uma outra contribuição desse trabalho é a proposição de um algoritmo para calcular a máxima sublinguagem harmoniosa de uma linguagem regular.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

FAILURE DIAGNOSIS OF WEIGHTED DISCRETE EVENT SYSTEMS

Gustavo da Silva Viana

March/2014

Advisor: João Carlos dos Santos Basilio

Department: Electrical Engineering

One approach to fault diagnosis is by constructing a discrete-event model of the system whose fault occurrence must be diagnosed. In this case, the decision on the fault occurrence (diagnosis) is taken based on the observed events only, i.e., those events whose occurrence can be recorded by sensors. In practice, this is carried out by using diagnosers. Diagnosers are deterministic automata whose states are sets formed with the states of the system automaton (plant) together with labels that indicate if the trace occurred so far possesses or not the fault event, and can be used for both online diagnosis and in the verification of language diagnosability. In this work we propose a new approach to fault diagnosability verification based on a new automaton called test diagnoser, that allows language diagnosticability verification by searching for strongly connected components instead of cycles, as usually done in the literature. This approach is then applied to weighted automata to estimate the maximum time to diagnose the failure after its occurence. Another contribution of this work is the proposition of an algorithm to calculate the maximum harmonious sublanguage of regular language.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

G	Autômato determinístico de estados finitos
X	conjunto de estados de um autômato
$\xi(\cdot,\cdot)$	Função de transição de um autômato
$\Gamma(\cdot)$	Conjunto de eventos ativos de um estado de um autômato
x_o	Estado inicial de um autômato
X_m	Conjunto de estados marcados de um autômato
Σ	Conjunto de eventos
Σ_o	Conjunto de eventos observáveis, $\Sigma_o \subseteq \Sigma$
Σ_{uo}	Conjunto de eventos não-observáveis, $\Sigma_{uo} \subseteq \Sigma$
Σ_c	Conjunto de eventos controláveis, $\Sigma_c \subseteq \Sigma$
Σ_{uc}	Conjunto de eventos não-controláveis, $\Sigma_{uc} \subseteq \Sigma$
$\Sigma_f = \{\sigma_f\}$	Conjunto de eventos de falha, $\Sigma_f \subseteq \Sigma_{uo}$
ϵ	Sequência nula
$\emptyset$	Conjunto (linguagem) vazio
$\|s\|$	Comprimento de uma sequência s
$L(G)$	Linguagem gerada por um autômato G
L	$L = L(G)$
$L_m(G)$	Linguagem marcada por um autômato G
L_m	$L_m = L_m(G)$
$\bar{L}$	Fecho do prefixo da linguagem L
L^*	Fecho de Kleene da linguagem L
L_aL_b	Operação de concatenação entre as linguagens L_a e L_b
L/s	Continuação da linguagem L após uma sequência s
$\Psi(\Sigma_f)$	Conjunto de todas as sequências de L que terminam com um evento de Σ_f
$\Sigma_f \in s$	$\bar{s} \cap \Psi(\Sigma_f) = \emptyset$
s_f	Último evento de uma sequência s
2^X	Conjunto potência de X
$Ac(\cdot)$	Operação de acessibilidade
$CoAc(\cdot)$	Operação de coacessibilidade
$Trim(\cdot)$	Operação *trim*
$P(\cdot)$	Projeção natural

$P^{-1}(\cdot)$	Projeção inversa
P_o	Projeção do conjunto de eventos Σ para o conjunto Σ_o
$P_{o'}$	Projeção do conjunto de eventos Σ^* para o conjunto $\Sigma_o^{'*}$
$P_{oo'}$	Projeção do conjunto de eventos Σ_o^* para o conjunto $\Sigma_o^{'*}$ sendo $\Sigma_o^{'} \subset \Sigma_o$
Σ_{o_i}	Conjunto de eventos observáveis dos módulos M_i, $i = 1, 2, \ldots, N$ $\Sigma_o \subseteq \Sigma$
P_{o_i}	Projeção de Σ^* em $\Sigma_{o_i}^{'*}$, $i = 1, 2, \ldots, N$
$UR(\cdot)$	Alcance não observável de um estado
$Obs(G, \Sigma_o)$	Observador do autômato G em relação a Σ_o
$G_1 \times G_2$	Composição produto entre os autômatos G_1 e G_2
$G_1 \| \| G_2$	Composição paralela entre os autômatos G_1 e G_2
K	Linguagem especificada para um SED
$K_1 \cup K_2$	Operação de união sobre as linguagens K_1 e K_2
$K_1 \cap K_2$	Operação de interseção sobre as linguagens K_1 e K_2
$K_1 \setminus K_2$	Operação de subtração de conjuntos entre as linguagens K_1 e K_2
$K^{\uparrow C}$	Sublinguagem controlável suprema de K
$K^{\uparrow H}$	Sublinguagem harmoniosa suprema de K
$K^{\uparrow CH}$	Sublinguagem controlável e harmoniosa suprema de K
$\exists$	Operador de existência
$\wedge$	Operador *e* lógico
$\vee$	Operador *ou* lógico
$\oplus$	operação soma max-plus
$\otimes$	operação produto max-plus
ε	elemento neutro da operação $\oplus$
e	elemento neutro da operação $\otimes$
A	matriz max-plus
$\mathbb{R}_{max}$	conjunto dos números reais
$E(n, n)$	matriz identidade max-plus $n \times n$
$\varepsilon(n, m)$	matriz nula max-plus $n \times m$
f	função de peso parcial
$A^{\otimes k}$	matriz potência max-plus
A^+	matriz de peso máximo max-plus
h	conjunto de relação de exclusão mútua intrínseca
h_E	conjunto de relação de exclusão mútua imposta
G_d	Diagnosticador centralizado associado a um autômato G
A_l	Autômato rotulador de falhas
G_{d_i}	Diagnosticador parcial para o módulo M_i, $i = 1, 2, \ldots, N$
G_t	Diagnosticador de teste centralizado associado a um autômato G
G_{t_i}	Diagnosticador parcial de teste para o módulo M_i, $i = 1, 2, \ldots, N$
G_h	Autômato que marca a linguagem harmoniosa suprema

Capítulo 1

Introdução

Diagnose de falhas em sistemas a eventos discretos (SEDs) tem despertado grande interesse nos últimos anos [1–18]. Os modelos mais comumente utilizados são os autômatos [19] e as redes de Petri [20–22]. Uma das razões para esse interesse é o fato de modelos a eventos discretos poderem ser aplicados não só a sistemas em que esses modelos são os mais apropriados (sistemas de computação, redes de comunicação e de manufatura, por exemplo), como também a diversos sistemas dinâmicos de variáveis contínuas (SDVC), uma vez que SVDC podem também ser modelados como SEDs dependendo do grau de abstração[23, 24].

Dois paradigmas norteiam a diagnose de falhas em SEDs: (i) as falhas a serem diagnosticadas são eventos não observáveis, isto é, eventos cujas ocorrências não podem ser registradas por sensores; (ii) a ocorrência de falhas altera o comportamento do sistema, porém não necessariamente, leva o sistema a uma parada (em sistemas de manufatura, por exemplo, a ocorrência de uma falha não diagnosticada pode levar a uma degradação da produção e, consequentemente, à perda de qualidade, sem contudo, interromper o processo de fabricação).

Assim, o objetivo de um sistema de diagnose de falhas é inferir e informar a ocorrência de falhas tendo como base somente os eventos que tenham sido observados, isto é, registrados pelos sensores. O projeto de um sistema que permite diagnosticar falhas em SEDs pode ser dividido em duas etapas principais:

1. A construção de um modelo a eventos discretos do sistema que se deseja realizar a diagnose de falhas;

2. O desenvolvimento de um conjunto de regras (protocolo) a serem seguidas para a identificação e a diagnose de falhas.

A primeira parte consiste no desenvolvimento de um modelo que capture tanto o comportamento normal quanto o comportamento do sistema levando-se em consideração a ocorrência da falha. A segunda parte é calcada em um arcabouço teórico

introduzido por SAMPATH et al. [2] e desenvolvido nas duas últimas décadas. Grande parte dessa base teórica será revista neste trabalho, por ser fundamental para os resultados aqui apresentados.

O problema da diagnose de falhas foi trazido para o contexto de SED por LIN [1], que introduziu o conceito da capacidade de se diagnosticar a ocorrência de uma falha em um sistema. Logo a seguir, SAMPATH et al.[2] apresentaram condições necessárias e suficientes para a diagnosticabilidade de falhas de SEDs e propuseram a construção de um autômato diagnosticador que permite tanto inferir sobre a capacidade de diagnosticar as falhas presentes no sistema (diagnosticabilidade) quanto ser usado para realizar a diagnose de falhas em tempo real. O diagnosticador é um autômato cujos estados são construídos de forma a representar uma estimativa do estado atual do sistema, após a ocorrência de um evento observável, e possui rótulos que indicam se a sequência de eventos ocorrida possui ou não a falha. Logo, o diagnosticador infere, a partir do registro da ocorrência de eventos observáveis somente qual sequência de eventos (incluindo eventos não-observáveis) do sistema ocorreu.

A verificação da diagnosticabilidade por meio do diagnosticador proposto por SAMPATH et al.[2] requer a busca por ciclos indeterminados. Embora esse diagnosticador seja intuitivo e com aplicabilidade para diagnose em tempo real, a busca por ciclos indeterminados tem complexidade exponencial [25, 26]. Este trabalho propõe uma nova maneira de se verificar a diagnosticabilidade utilizando um novo autômato que substitui a busca por ciclos indeterminados pela busca por componentes fortemente conexas. A busca por componentes fortemente conexas possui a vantagem de ter complexidade linear [27].

Os trabalhos citados anteriormente não envolvem o contexto de diagnose de falhas levando em conta informações temporais. ALUR e DILL [28] introduziram uma complexa teoria dos autômatos temporizados, definindo regras e composições a partir de uma modelagem que considera que as transições disparam não somente pela ocorrência dos eventos, mas também em função do tempo que eles levam para ocorrer. A partir dessa modelagem, o problema da diagnose de falhas em autômatos que consideram a informação temporal, é abordado por TRIPAKIS [29]. Considerando que as falhas são não-observáveis, a ideia básica do trabalho de TRIPAKIS [29] é diagnosticar a falha pelo tempo que o sistema leva para mudar de um estado a outro, dependendo do valor de *clock* associado a determinadas transições e estados.

Mais recentemente autômatos ponderados foram trazidos para o contexto de SEDs [30, 31]. Essa modelagem é mais simples que a proposta por ALUR e DILL [28], sendo composta por um autômato de estados finitos [19] juntamente com uma função que associa a cada transição um peso. Esse peso pode ser interpretado como um custo ou o tempo estimado em que uma determinada transição leva para disparar. Este trabalho utiliza a modelagem proposta em [30] para estimar o tempo

máximo que o sistema leva para diagnosticar a falha. A motivação para esse estudo é a seguinte. Em um sistema real, uma falha pode comprometer toda uma linha de produção. Contudo, a certeza de que ela ocorreu pode não ser suficiente. Componentes podem queimar, peças se desalinharem antes da falha ter sido detectada. Assim, por razões de segurança, é importante estimar um tempo máximo para a diagnose da falha.

SU [30, 31] propõe a utilização de autômatos ponderados para o contexto de controle supervisório. Por meio de conceitos de álgebra maxplus [32, 33] e heap-of-pieces [34], encontra a sequência que realiza em tempo mínimo uma determinada tarefa. Este trabalho utiliza também os resultados da álgebra max-plus, porém para encontrar o tempo máximo para a diagnose de falhas. Especificamente, utiliza-se a técnica de representação de grafos por meio de matrizes para efetuar operações de forma a obter a matriz de ponderação máxima, isto é, aquela cujos elementos são os pesos máximos dos caminhos que ligam um estado a outro.

No contexto de controle supervisório, SU et al. [35] introduz o conceito de linguagem harmoniosa. Uma linguagem harmoniosa é aquela em que não há sobreposição de eventos que estão incluídos numa relação de exclusão mútua imposta, isto é, eventos que não podem ocorrer ao mesmo tempo por razões de especificação ou segurança. Uma contribuição deste trabalho é um algoritmo para o cálculo da sublinguagem harmoniosa suprema, isto é, a maior sublinguagem harmoniosa possível.

Em resumo, os objetivos deste trabalho são os seguintes:

1. Propor uma nova abordagem para a verificação da diagnosticabilidade utilizando um novo autômato, chamado de diagnosticador de teste, que substitui a busca por ciclos indeterminados pela busca de componentes fortemente conexas.

2. Propor um algoritmo baseado no diagnosticador de teste e em resultados da álgebra max-plus, que calcula o tempo máximo para a diagnose de falhas.

3. Propor um algoritmo que calcula a sublinguagem harmoniosa suprema.

Para atingir os objetivos traçados, este trabalho está estruturado da seguinte forma. No capítulo 2, são revistos os principais conceitos de SEDs necessários ao entendimento dos resultados sobre diagnose de falha. No capítulo 3 são apresentados os conceitos de álgebra max-plus, autômatos ponderados e autômatos ponderados pelo tempo. Ainda nesse capítulo é proposto um algoritmo que calcula a sublinguagem harmoniosa suprema. A diagnose de falhas centralizada é apresentada no capítulo 4, sendo proposto um algoritmo que calcula o tempo máximo para a diagnose de falhas. Finalmente, no capítulo 5, são apresentadas as conclusões e algumas propostas de trabalhos futuros sobre o tema.

Capítulo 2

Sistemas a eventos discretos

Neste capítulo são apresentadas as bases da teoria de sistemas a eventos discretos necessárias para a elaboração desta dissertação. Para tanto, este capítulo está estruturado da seguinte forma. Na seção 2.1, é apresentada a definição de sistemas a eventos discretos. Na seção 2.2 é introduzido o conceito de linguagens. Na seção 2.3, considera-se a teoria de autômatos.

2.1 Definição de SED

Os sistemas a serem considerados neste trabalho são denominados sistemas a eventos discretos [19]. Esses sistemas percebem as ocorrências no mundo externo a partir da recepção de estímulos, denominados eventos. São exemplos de eventos o início e o término de uma tarefa (mas não sua execução), a chegada de um cliente a uma fila ou a recepção de uma mensagem em um sistema de comunicação. A ocorrência de um evento causa, em geral, uma mudança interna no sistema, a qual pode ou não se manifestar a um observador externo. Além disso, uma mudança pode ser causada pela ocorrência de um evento interno ao próprio sistema, tal como o término de uma atividade ou de uma temporização. Em qualquer caso, essas mudanças se caracterizam por serem abruptas e instantâneas; ao perceber um evento, o sistema reage imediatamente, acomodando-se em tempo nulo a uma nova situação onde permanece até que ocorra um novo evento. Dessa forma, a simples passagem do tempo não é suficiente para garantir que o sistema evolua; para tanto, é necessário que ocorram eventos, sejam esses internos ou externos. Note ainda que a ocorrência desses eventos pode depender de fatores alheios ao sistema, que não podem, em geral, ser previstos pelo sistema. Assim, pode-se definir Sistemas a Eventos Discretos (SEDs) como um sistema dinâmico de estados discretos que evolui de acordo com a ocorrência abrupta de eventos físicos, em intervalos de tempo em geral irregulares e desconhecidos.

O conjunto de eventos de um SED pode ser considerado como o alfabeto do

sistema. Nesse contexto, cada sequência possível de eventos de comprimento finito é chamada de palavra, e o conjunto de todas as sequências possíveis de comprimento finito é chamada de linguagem. As linguagens são importantes para acompanhar a evolução de um SED. Uma vez que SEDs não podem ser modelados por equações diferenciais, como os sistemas dinâmicos de variáveis contínuas, então, ao se considerar a evolução dos estados de um SED, a maior preocupação é com a sequência de estados visitados e com os eventos que causaram as correspondentes transições de estado. Como consequência, tem-se que o modelo de um SED é composto basicamente de dois elementos: estados e eventos.

2.2 Linguagens

Uma linguagem é um conjunto de sequências de eventos finitas, ou palavras. O comprimento de uma palavra s, denotado por $|s|$, é contado como o número de eventos na palavra. Uma palavra vazia, ou seja, sem eventos, será expressada por ϵ. O processo básico de formação de uma linguagem é a concatenação. A concatenação de duas sequências de eventos resulta nos eventos da primeira imediatamente seguidos pelos eventos da segunda. Outra operação com eventos que é importante para o estudo sobre linguagens é o fecho de Kleene de um conjunto, denotado por *, e que representa o conjunto formado por todas as sequências de comprimento finito que podem ser formadas com os elementos (eventos) desse conjunto, mais o evento ϵ. Note, portanto, que o fecho de Kleene de um conjunto é um conjunto infinito, porém enumerável.

Assim como no caso dos eventos, as linguagens também possuem operações relacionadas a concatenações e fechos de Kleene, com definições semelhantes. A concatenação de duas linguagens resulta em sequências formadas por sequências da primeira linguagem concatenadas a sequências da segunda linguagem. Dessa forma se $L_a, L_b \subseteq \Sigma^*$, então a concatenação de L_a e L_b é definida como:

$$L_a L_b := \{s = s_a s_b \in \Sigma^* : (s_a \in L_a) \wedge (s_b \in L_b)\}. \tag{2.1}$$

O fecho de Kleene de uma linguagem é o conjunto de todas as sequências possíveis formadas pela concatenação de todas as sequências dessa linguagem. Formalmente, se $L \subseteq \Sigma^*$, então o fecho de Kleene é definido como:

$$L^* := \{\epsilon\} \cup L \cup LL \cup LLL... \tag{2.2}$$

Outra operação com linguagens importante é o fecho do prefixo, que consiste no conjunto de todos os prefixos da linguagem. Uma palavra t será prefixo de uma

palavra s caso exista uma sequência v tal que $tv = s$. Dessa forma, tanto s quanto ϵ são prefixos de s. A seguinte definição formal é dada para o fecho em prefixo:

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

Pode-se considerar ainda a operação de projeção, denotada por P. Ela é realizada de um conjunto de eventos Σ para um subconjunto Σ_s de Σ. A projeção consiste na redução de uma sequência, por meio da eliminação de eventos que não constem em Σ_s. Formalmente, tem-se que a projeção de uma sequência, $P : \Sigma^* \to \Sigma_s^*$ é definida como:

$$P(\epsilon) := \epsilon, \tag{2.4}$$

$$P(e) = \begin{cases} e, & \text{se } e \in \Sigma_s, \\ \epsilon, & \text{caso contrário,} \end{cases} \tag{2.5}$$

$$P(se) := P(s)P(e) \text{ para } s \in \Sigma^*, e \in \Sigma. \tag{2.6}$$

A projeção inversa, $P^{-1} : \Sigma_s^* \to 2^{\Sigma^*}$, também é importante, sendo definida como:

$$P^{-1}(t) := \{s \in \Sigma^* : P(s) = t\}. \tag{2.7}$$

Linguagens representam uma maneira conveniente de se modelar SEDs. Contudo, na maioria dos casos de interesse prático, as linguagens serão complexas, não podendo ser simplesmente enumeradas. Portanto, para representá-las, duas estruturas são comumente utilizadas: o autômato e a rede de Petri. Este trabalho utilizará modelos por autômatos. Uma breve revisão da teoria de autômatos será apresentada a seguir.

2.3 Autômatos

2.3.1 Definição

O objetivo de utilizar autômatos nos modelos de SEDs é obter a representação gráfica das linguagens, por meio de relações entre estados e eventos. Em cada estado são retratados os eventos que podem ocorrer e para onde eles levam o sistema. Os estados que representam a conclusão de uma tarefa são denominados estados marcados.

Matematicamente, um autômato pode ser definido como uma sêxtupla:

$$G = (X, \Sigma, \xi, \Gamma, x_0, X_m), \tag{2.8}$$

em que X denota o conjunto de estados, Σ o conjunto de eventos, ξ a função de transição de estados, Γ o conjunto de eventos ativos, x_0 o estado inicial e X_m o conjunto de estados marcados. No geral, o conjunto de estados de um SED não necessita ser finito, porém esse será o caso deste trabalho.

Para se entender a evolução de um autômato, suponha que ele esteja no estado x_n e que, quando ocorre um evento σ, o sistema muda instantaneamente para o estado x_{n+1}. Essa evolução é caracterizada pela função de transição de estados da seguinte forma: $x_{n+1} = \xi(x_n, \sigma)$ tal que $\sigma \in \Gamma(x_n)$. Note, nessas condições, que uma forma mais clara de se caracterizar um autômato é por meio do diagrama de transição de estados. Nesses diagramas, estados são representados por circunferências e as transições entre estados por setas rotuladas pelos eventos que as causam. Os estados marcados são denotados por duas circunferências concêntricas, e o estado inicial possuirá uma seta apontando para ele.

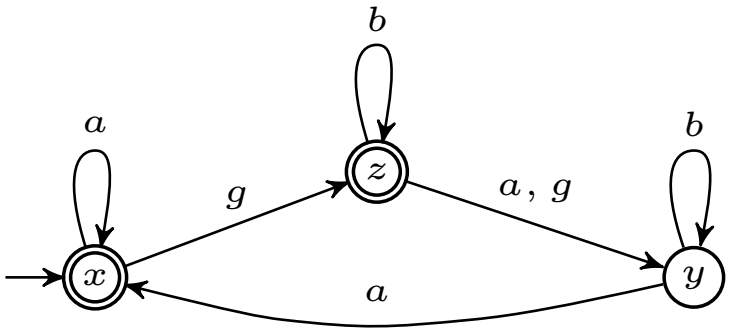

Figura 2.1: Exemplo de autômato G.

A figura 2.1 mostra um exemplo de um autômato $G = (X, \Sigma, \xi, \Gamma, x_0, X_m)$, cujo conjunto de eventos é $\Sigma = \{a, b, g\}$ e o conjunto de estados $X = \{x, y, z\}$. O estado inicial é x_0 e o conjunto de estados marcados é $X_m = \{x, z\}$. Note que $\Gamma(x) = \{a, g\}$, $\Gamma(z) = \{a, b, g\}$ e $\Gamma(y) = \{a, b\}$. A função de transição de estados de G é dada da seguinte forma: $\xi(x, a) = x$, $\xi(x, g) = z$, $\xi(z, b) = z$, $\xi(z, a) = y$, $\xi(z, g) = y$, $\xi(y, b) = y$ e $\xi(y, a) = x$.

2.3.2 Linguagens geradas e marcadas por autômatos

Para se modelar um SED, é necessário utilizar duas linguagens. A primeira é a linguagem gerada, ou seja, a linguagem que representa o comportamento geral do sistema, englobando todas as sequências de eventos que podem ser executadas a partir do estado inicial. A segunda é a linguagem marcada, que representa o comportamento desejado do sistema modelado, isto é, todas as sequências de eventos que levam o autômato do estado inicial aos estados marcados. Definindo formalmente, a linguagem gerada é:

$$L(G) := \{s \in \Sigma^* : \xi(x_0, s)!\}, \tag{2.9}$$

em que $\xi(x_0, s)!$ denota que $\xi(x_0, s)$ é definida, isto é, existe $x \in X$, $e \in \Sigma^*$ tais que $\xi(x_0, s) = x$. Assim, uma sequência de eventos s será parte da linguagem gerada de um autômato se a função de transição for definida para s a partir do estado inicial. De forma semelhante, é possível obter a linguagem marcada pelo autômato, que se trata das sequências de eventos que conduzem o sistema do estado inicial até algum estado marcado, isto é:

$$L_m(G) := \{s \in L(G) : \xi(x_0, s) \in X_m\} \tag{2.10}$$

Note que um autômato sem estados gera e marca o conjunto vazio e que o autômato que possui somente um estado gera a linguagem ϵ e marca ϵ ou $\emptyset$, dependendo se o estado inicial é ou não marcado, respectivamente. As definições de L e L_m levam a uma série de conclusões, como o fato de que a linguagem gerada é um conjunto prefixo fechado, já que para que uma certa sequência faça parte da linguagem, é necessário que seus prefixos também o façam. Além disso, é possível perceber que a linguagem marcada é um subconjunto da linguagem gerada.

É possível que mais de um autômato represente a mesma linguagem. Nesse contexto, dois autômatos são ditos iguais quando eles geram e marcam as mesmas linguagens. Formalmente, dois autômatos G_1 e G_2 são equivalentes se:

$$L(G_1) = L(G_2) \text{ e } L_m(G_1) = L_m(G_2). \tag{2.11}$$

Um autômato nem sempre consegue atingir um estado marcado, podendo ocorrer bloqueios ao longo do percurso, isto é, nem sempre é capaz de completar sua tarefa. Existem dois tipos de bloqueio. No primeiro, chamado *deadlock*, o sistema chega a um estado não marcado do qual ele não pode sair, ou seja, a partir do qual não existem eventos que causem transições de estado. O segundo é chamado de *livelock*, e consiste num grupo de estados que permitem que o sistema transite entre eles, porém não consegue sair desse ciclo e alcançar um estado marcado. Pela definição, um sistema terá bloqueios quando sua linguagem marcada for um subconjunto próprio da sua linguagem gerada, isto é, $L_m(G) \neq L(G)$.

2.3.3 Operações com Autômatos

Para analisar SEDs modelados por autômatos, é necessário ter um conjunto de operações em um autômato simples para modificar apropriadamente seu diagrama de transição de estados; por exemplo, por alguma operação de linguagem que se quer efetuar. Então, é necessário definir operações que permitam combinar, ou

compor, dois ou mais autômatos, para que modelos de sistemas completos possam ser construídos a partir de modelos de componentes individuais do sistema.

Dentre as operações unárias, ou seja, aquelas que alteram apenas um único autômato, destacam-se as operações em são obtidas a parte acessível, a coacessível, o complemento e a operação *trim*. Para construir um autômato a partir da composição de dois autômatos, pode-se utilizar duas operações distintas: o produto, denotado por $\times$, e a composição paralela, denotada por $||$. A diferença entre essas duas operações é a forma como elas lidam com os conjuntos de eventos. Enquanto o produto marca a interseção entre as linguagens dos autômatos, a composição paralela permite também que eventos particulares sejam mantidos.

Parte Acessível

Um estado é dito acessível se for alcançável a partir do estado inicial, ou seja, os estado acessíveis são todos os estados $x \in X$ tais que $\exists s \in \Sigma^*$, $\xi(x_0, s) = x$. A operação para obtenção da parte acessível consiste em se retirar de um autômatos $G = (X, \Sigma, \xi, \Gamma, x_0, X_m)$ todos esses estados. Como os estados a serem retirados nessa operação, por definição, não são atingidos por sequências pertencentes às linguagens geradas e marcadas, essas não serão alteradas. A operação de acessibilidade será referenciada por $Ac(G)$, e pode ser definida conforme a expressão:

$$Ac(G) := (X_{ac}, \Sigma, \xi_{ac}, \Gamma_{ac}, x_0, X_{ac,m}), \tag{2.12}$$

em que $X_{ac} = \{x \in X : (\exists s \in \Sigma)[\xi(x_0, s) = x]\}$, $X_{ac,m} = X_m \cap X_{ac}$ e $\xi_{ac} = \xi|_{X_{ac} \times \Sigma \to X_{ac}}$, isto é, ξ restrita aos estados de X_{ac}. Um autômato acessível é composto pelo conjunto de estados do autômato original que podem ser alcançados a partir do estado inicial, pelo mesmo conjunto de eventos do autômato original e pela função de transição de estados acessíveis.

Parte Coacessível

Um estado x é dito coacessível se há um caminho entre x e um estado marcado, isto é, x é coacessível se $\exists s \in \Sigma^*$, $\xi(x, s) \in X_m$. A operação para obtenção da parte coacessível pode causar alterações na linguagem gerada, visto que ela consiste na eliminação dos estados a partir dos quais não é possível alcançar um estado marcado. Dessa forma, é possível dizer que essa operação elimina eventuais bloqueios no sistema. Matematicamente, pode se dizer que $CoAc(G)$, é dado por:

$$CoAc(G) := (X_{coac}, \Sigma, \xi_{coac}, x_{0,coac}, X_m), \tag{2.13}$$

em que $X_{coac} = \{x \in X : (\exists s \in \Sigma)[\xi(x, s) \in X_m]\}$ e $\xi_{coac} = \xi|_{X_{coac} \times \Sigma \to X_{coac}}$. Pode-se observar que a coacessibilidade, por definição, não causa alterações na linguagem marcada.

Operação Trim

Um autômato que é tanto acessível quanto coacessível é dito ser *trim*. A operação *trim* resulta em um autômato sem bloqueios e sem estados não acessíveis e é dada por:

$$Trim(G) := Ac[CoAc(G)] = CoAc[Ac(G)] \tag{2.14}$$

Produto

A operação produto pode ser compreendida como uma interseção entre as linguagens gerada e marcada pelos autômatos. Assim sendo, para que haja uma transição de estados, é necessário que um mesmo evento possa ser ativado nos estados atuais de ambos os autômatos. Caso não haja nenhum evento em comum entre os conjuntos dos eventos ativos dos estados iniciais dos autômatos, o produto resultará em um autômato cuja linguagem gerada é ϵ. O resultado do produto de dois autômatos G_1 e G_2 é dado por:

$$G_1 \times G_2 := Ac(X_1 \times X_2, \Sigma_1 \cup \Sigma_2, \xi_{1\times 2}, \Gamma_{1\times 2}, (x_{01}, x_{02}), X_{m1} \times X_{m2}), \tag{2.15}$$

sendo a função de transição de estados $\xi_{1\times 2}$ definida como:

$$\xi_{1\times 2}((x_1, x_2), e) = \begin{cases} (\xi_1(x_1, e), \xi_2(x_2, e)), & \text{se } e \in \Gamma_1(x_1) \cap \Gamma_2(x_2), \\ \text{não definida}, & \text{caso contrário.} \end{cases} \tag{2.16}$$

Seguindo a lógica apresentada, as linguagens gerada e marcada pelo autômato resultante são, respectivamente, as interseções das linguagens gerada e marcada dos autômatos originais. Um exemplo de produto entre autômatos é o produto dos autômatos G da figura 2.1 e G' da figura 2.2, mostrado na figura 2.3.

Composição Paralela

A composição paralela é mais abrangente que o produto, permitindo que eventos individuais de cada autômato possam ser executados. Essa característica é de extrema importância, já que permite a modelagem de sistemas complexos por meio

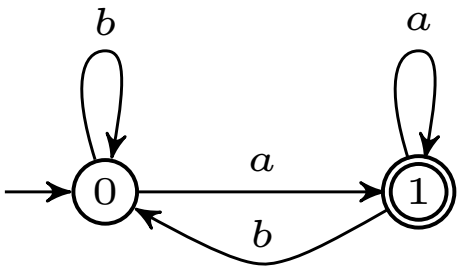

Figura 2.2: Exemplo de autômato $G^{'}$.

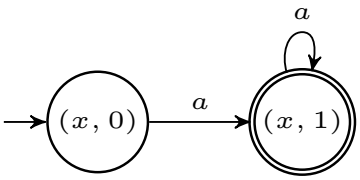

Figura 2.3: Produto entre os autômatos da figura 2.1 e 2.2.

de módulos individuais menores, com eventos responsáveis por seu próprio comportamento e eventos que garantem a interação entre os sistemas. No caso de eventos comuns aos dois autômatos, ou seja, $e \in \Sigma_1 \cap \Sigma_2$, a composição paralela se iguala ao produto. Porém, eventos que existam apenas em Σ_1 ou Σ_2, mas não em ambos, podem ser executados a qualquer momento, sem a participação do outro autômato. De modo formal, a composição paralela de dois autômatos G_1 e G_2 é definida por:

$$G_1||G_2 := Ac(X_1 \times X_2, \Sigma_1 \cup \Sigma_2, \xi_{1||2}, \Gamma_{1||2}, (x_{01}, x_{02}), X_{m1} \times X_{m2}), \tag{2.17}$$

sendo a função de transição de estados, $\xi_{1||2}$, definida da seguinte forma:

$$\xi((x_1, x_2), e) := \begin{cases} (\xi_1(x_1, \sigma), \xi_2(x_2, \sigma)), & \text{se } e \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ (\xi_1(x_1, e), x_2), & \text{se } e \in \Gamma_1(x_1) \setminus \Sigma_2 \\ (x_1, \xi_2(x_2, e)), & \text{se } e \in \Gamma_2(x_2) \setminus \Sigma_1 \\ \text{não definido}, & \text{caso contrário.} \end{cases}$$

Uma outra forma de se realizar a composição paralela entre autômatos é por meio do produto entre dois autômatos obtidos a partir dos autômatos originais adicionando-se em todos os estados auto-laços com eventos exclusivos do outro autômato. De fato, pode-se perceber que estes auto-laços são o equivalente ao autômato permitir a execução de todos os eventos que não pertençam ao seu conjunto de eventos. Um exemplo de composição paralela é ilustrado na figura 2.4. A composição é feita entre autômatos G da figura 2.1 e $G^{'}$ da figura 2.2.

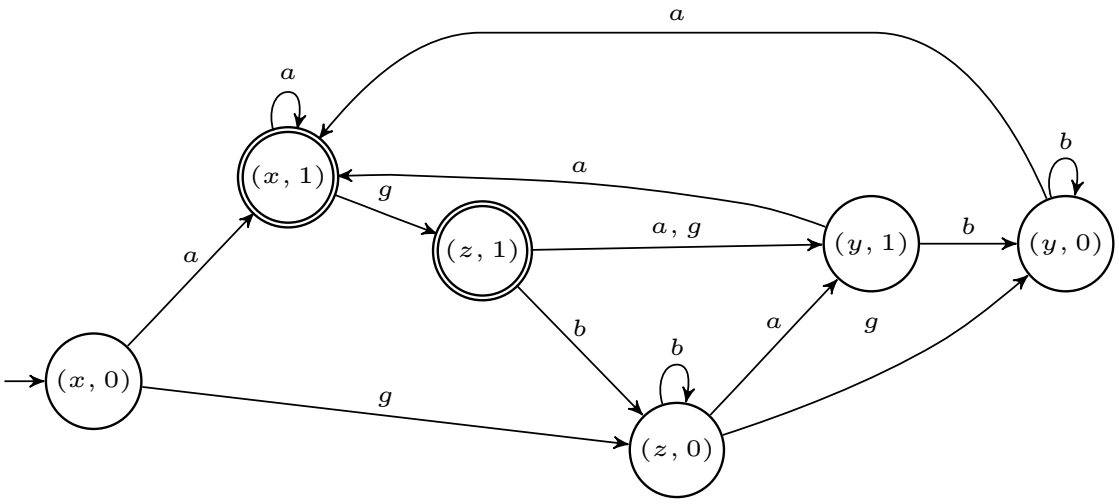

Figura 2.4: Autômato resultante da composição paralela entre os autômatos da figura 2.1 e 2.2.

Observador

Suponha, agora, que o conjunto de eventos Σ seja particionado como $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$, isto é, $\Sigma = \Sigma_o \cup \Sigma_{uo}$ e $\Sigma_o \cap \Sigma_{uo} = \emptyset$, sendo Σ_o o conjunto de eventos observáveis e Σ_{uo} o conjunto de eventos não observáveis. Um evento é observável quando sua ocorrência puder ser registrada por meio de sensores ou quando estiver associado a comandos. Os eventos não-observáveis, por sua vez, designam aqueles eventos do sistema cujas ocorrências não podem ser observadas por sensores (incluindo os eventos de falhas) ou então, embora haja sensores para registrá-los, esses eventos não podem ser vistos em função da natureza distribuída do sistema.

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

$$Obs(G) = (X_{obs}, \Sigma_o, \xi_{obs}, \Gamma_{obs}, x_{0_{obs}}, X_{m_{obs}}), \tag{2.18}$$

sendo $X_{obs} \subset 2^X$ e $X_{m_{obs}} = \{B \in X_{obs} : B \cap X_m \neq \emptyset\}$. Para a definição dos outros componentes, é necessário o conceito de alcance não-observável de um estado $x \in X$, aqui denotado por $UR(x)$, que é definido como:

$$UR(x) = \{y \in X : (\forall t \in \Sigma_{uo}^{\star})[\xi(x,t) = y]\}. \tag{2.19}$$

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

como:

$$UR(B) = \bigcup_{x \in B} UR(x) \tag{2.20}$$

A construção do autômato observador é realizada da seguinte forma.

Algoritmo 2.1 *Construção do Observador*

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

Passo 2: Faça $\hat{X}_{obs} \leftarrow \tilde{X}_{obs}$ *e* $\tilde{X}_{obs} \leftarrow \emptyset$

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

- $\Gamma_{obs}(B) = (\bigcup_{x \in B} \Gamma(x)) \cap \Sigma_o$;
- *Para cada* $e \in \Gamma_{obs}(B)$, $\xi_{obs}(B, e) = UR(\{x \in X : (\forall y \in B)[x = \xi(y, e)]\})$ *e faça* $\tilde{X}_{obs} \leftarrow \tilde{X}_{obs} \cup \xi_{obs}(B, e)$

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

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

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

O objetivo do algoritmo 2.1 é calcular o alcance não observável para cada estado de G da seguinte forma. No passo 1, calcula-se o alcance não-observável do estado inicial, formando o estado inicial do observador. No passo 3, calculam-se os conjuntos dos eventos ativos dos estados do observador obtidos no passo anterior ou na iteração anterior (o primeiro se refere ao alcance observável do estado inicial e o último aos estados de G alcançados por meio de eventos observáveis juntamente com os respectivos alcances não-observáveis). Além disso são calculados os próximos estados do observador, que correspondem aos alcances não-observáveis dos estados de G alcançados a partir do estado atual do observador por meio de eventos observáveis. Essa sequência é repetida até que todos os estados acessíveis do observador tenham sido encontrados.

A partir da construção do observador, pode-se concluir que a linguagem gerada por Obs(G) é a projeção da linguagem de G sobre o conjunto de eventos observáveis, isto é, $L(Obs(G)) = P_o[L(G)]$, sendo $P_o : \Sigma^* \rightarrow \Sigma_o^*$.

Exemplo 2.1 *Para ilustrar a construção de observadores, considere o autômato G da figura 2.5 e suponha que o evento a seja não-observável, isto é, tem-se que* $\Sigma = \{a, b, c\}$, $\Sigma_o = \{b, c\}$ *e* $\Sigma_{uo} = \{a\}$.

Quando o autômato inicia sua operação, não é possível dizer se ele se encontra no estado inicial $x_0 = 0$ *ou no estado* $x = 1$, *já que a ocorrência do evento a*

não pode ser registrada. Sendo assim, o estado inicial do autômato observador mostrado na figura 2.6 é $\{0, 1\}$*. Caso o próximo evento observável a ocorrer seja o evento b, pode-se, então, afirmar que o autômato estará, inicialmente no estado* $x = 3$*, porém, como o evento a é não-observável, o autômato pode mudar para o estado* $x = 1$*, sem que essa mudança seja percebida; por conseguinte, a transição rotulada pelo evento b no observador leva do estado inicial para o estado* $\{1, 3\}$*. Há ainda transições rotuladas pelo evento c que levam ao estado* $\{1, 2, 3\}$*, do observador, partindo tanto do estado inicial quanto do estado* $\{1, 3\}$*, uma vez que o evento c é o único evento observável pertencente aos conjuntos dos eventos ativos dos estados de G que compõem os estados* $\{1, 3\}$*. Finalmente, quando a ocorrência do evento c for registrada e o observador estiver no estado* $\{1, 2, 3\}$*, ele irá permanecer nesse estado. Contudo, se o evento b ocorrer, o observador retornará ao seu estado inicial.*

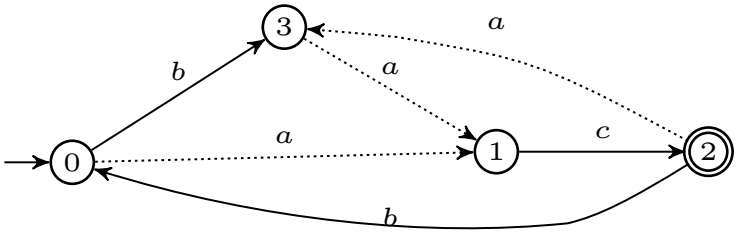

Figura 2.5: Autômato determinístico G com eventos não-observáveis do exemplo 2.1.

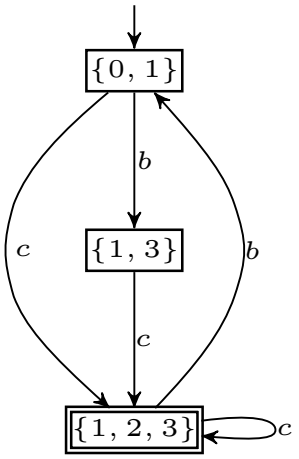

Figura 2.6: Observador do autômato G do exemplo 2.1.

Capítulo 3

Álgebra max-plus e autômatos ponderados

Álgebra Max-plus [32] é um dos muitos *semianéis* idempotentes que têm sido consideradas em vários campos da matemática. Em álgebra abstrata, um *semianel* é uma estrutura algébrica semelhante a um anel, mas sem a exigência de que cada elemento tenha um inverso aditivo. No contexto deste trabalho, essa álgebra é fundamental para representar sistemas a eventos discretos temporizados por meio de matrizes e, dessa forma, analisar diversos comportamentos.

Este capítulo está organizado da seguinte forma. Na seção 3.1 são apresentadas as definições da álgebra max-plus. Na seção 3.2 são introduzidos os conceitos de vetores e matrizes max-plus. Na seção 3.3 é mostrada a relação entre autômatos ponderados e matriz max-plus. Por fim, na seção 3.4 são considerados os conceitos de autômatos ponderados pelo tempo e é apresentado um algoritmo que calcula a máxima sublinguagem harmoniosa.

3.1 Conceitos básicos de álgebra max-plus

Primeiramente, define-se $\varepsilon := -\infty$ (elemento neutro da operação $\oplus$) e $e := 0$ (elemento neutro da operação $\otimes$). Considere que $\mathbb{R}_{max}$ seja o conjunto dos números reais. Para os elementos $a, b \in \mathbb{R}_{max}$, define-se as operações $\oplus$ e $\otimes$ da seguinte forma:

$$a \oplus b := max(a, b) \qquad e \qquad a \otimes b := a + b, \tag{3.1}$$

sendo $max(a, -\infty) = max(-\infty, a) = a$ e $a + (-\infty) = -\infty + a = -\infty$. Dessa forma, para qualquer $a \in \mathbb{R}_{max}$, então pode-se dizer que:

$$a \oplus \varepsilon = \varepsilon \oplus a = a \qquad e \qquad a \otimes \varepsilon = \varepsilon \otimes a = \varepsilon. \tag{3.2}$$

Assim como na álgebra convencional, simplifica-se a notação de modo que a operação $\otimes$ tenha prioridade sobre a operação $\oplus$. As definições (3.1) e (3.2) são ilustradas com os seguintes exemplos numéricos.

$$\begin{gathered} 2 \oplus 3 = max(2,3) = 3, \\ 2 \oplus \varepsilon = max(2,-\infty) = 2, \\ 2 \otimes \varepsilon = 2 - \infty = -\infty = \varepsilon, \\ e \oplus 3 = max(0,3) = 3, \\ 2 \otimes 3 = 2 + 3 = 5, \\ 4 \otimes -9 \oplus 6 \otimes 1 = -5 \oplus 7 = 7. \end{gathered}$$

As definições de $\oplus$ e $\otimes$ têm algumas propriedades algébricas interessantes. Para álgebra max-plus, as seguintes propriedades são válidas.

- Associatividade:

$$\forall x, y, z \in \mathbb{R}_{max}, x \oplus (y \oplus z) = (x \oplus y) \oplus z, \tag{3.3}$$

$$\forall x, y, z \in \mathbb{R}_{max}, x \otimes (y \otimes z) = (x \otimes y) \otimes z. \tag{3.4}$$

- Comutatividade:

$$\forall x, y \in \mathbb{R}_{max}, x \oplus y = y \oplus x, \tag{3.5}$$

$$\forall x, y \in \mathbb{R}_{max}, x \otimes y = y \otimes x. \tag{3.6}$$

- Distributividade:

$$\forall x, y, z \in \mathbb{R}_{max}, x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z) \tag{3.7}$$

- Elemento zero:

$$\forall x \in \mathbb{R}_{max}, x \oplus \varepsilon = \varepsilon \oplus x = x. \tag{3.8}$$

- Elemento unitário:

$$\forall x \in \mathbb{R}_{max}, x \otimes e = e \otimes x = x. \tag{3.9}$$

- Absorção em $\otimes$:

$$\forall x \in \mathbb{R}_{max}, x \otimes \varepsilon = \varepsilon \otimes x = \varepsilon. \tag{3.10}$$

- Idempotência em $\oplus$:

$$\forall x \in \mathbb{R}_{max} : x \oplus x = x. \tag{3.11}$$

- Potência:

$$\forall x^{\otimes n} := \underbrace{x \otimes x \otimes x \ldots \otimes x}_{\text{n vezes}} = n \times x. \tag{3.12}$$

3.2 Vetores e matrizes

Nesta seção, matrizes sobre $\mathbb{R}_{max}$ serão introduzidas. O conjunto de matrizes max-plus $n \times m$ é denotado por $\mathbb{R}_{max}^{n\times m}$. O elemento da matriz $A \in \mathbb{R}_{max}^{n\times m}$ na linha i e coluna j é denotado por a_{ij}, para $i \in \underline{n}$ e $j \in \underline{m}$, sendo $\underline{n} := 1, 2, ..., n$ e $\underline{m} := 1, 2, ..., m$. A Matriz A pode ser escrita como:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{pmatrix}.$$

Ocasionalmente, denotaremos o elemento a_{ij} como $[A]_{ij}$. A soma de matrizes $A, B \in \mathbb{R}_{max}^{n\times m}$, denotado por $A \oplus B$, é definida $\forall i \in \underline{n}$ e $j \in \underline{m}$ por:

$$[A \oplus B]_{ij} = a_{ij} \oplus b_{ij} = max(a_{ij}, b_{ij}) \tag{3.13}$$

Por exemplo, considere as matrizes A e B dadas por:

$$A = \begin{pmatrix} e & \varepsilon \\ 3 & 2 \end{pmatrix} \qquad e \qquad B = \begin{pmatrix} -1 & 10 \\ 1 & \varepsilon \end{pmatrix}. \tag{3.14}$$

Sendo assim, $[A \oplus B]_{11} = e \oplus -1 = max(0, -1) = 0 = e$, $[A \oplus B]_{12} = \varepsilon \oplus 10 = max(\varepsilon, 10) = 10$, $[A \oplus B]_{21} = 3 \oplus 1 = max(3, 1) = 3$ e $[A \oplus B]_{22} = 2 \oplus \varepsilon = max(2, \varepsilon) = 2$. Em notação de matriz:

$$A \oplus B = \begin{pmatrix} e & 10 \\ 3 & 2 \end{pmatrix}.$$

Note que $A, B \in \mathbb{R}_{max}^{n\times m}$ e que $A \oplus B = B \oplus A$. Para $A \in \mathbb{R}_{max}^{n\times m}$ e $\alpha \in \mathbb{R}_{max}$, o múltiplo escalar $\alpha \otimes A$ é definido por:

$$[\alpha \otimes A]_{ij} = \alpha \otimes a_{ij} \tag{3.15}$$

Seja a matriz A definida em (3.14) e considere $\alpha = 2$. Então $[2 \otimes A]_{11} = 2 \otimes e = 2 + 0 = 2$. Da mesma forma, $[2 \otimes A]_{12} = \varepsilon$, $[2 \otimes A]_{21} = 5$ e $[2 \otimes A]_{22} = 4$. Em notação matricial,

$$2 \otimes A = \begin{pmatrix} 2 & \epsilon \\ 5 & 4 \end{pmatrix}.$$

Para matrizes $A \in \mathbb{R}_{max}^{n\times l}$ e $B \in \mathbb{R}_{max}^{l\times m}$ a matriz produto $A \otimes B$ é definida como:

$$[A \otimes B]_{ik} = \bigoplus_{j=1}^{l} a_{ij} \otimes b_{jk} = \max_{j\in\underline{l}} \{a_{ij} + b_{jk}\}, \tag{3.16}$$

para $i \in \underline{n}$ e $k \in \underline{m}$. Essa definição é semelhante à álgebra convencional, trocando $+$ por max e $\times$ por $+$. Note que $A \otimes B \in \mathbb{R}_{max}^{n\times m}$, i.e., com n linhas e m colunas. Por exemplo, considere A e B definidos por (3.14). Então os elementos de $A \otimes B$ são dados por:

$$[A \otimes B]_{11} = e \otimes (-1) \oplus \epsilon \otimes 1 = max(0-1, -\infty+1) = -1$$
$$[A \otimes B]_{12} = e \otimes 10 \oplus \epsilon \otimes \epsilon = max(0+10, -\infty-\infty) = 10$$
$$[A \otimes B]_{21} = 3 \otimes (-1) \oplus 2 \otimes 1 = max(3-1, 2+1) = 3$$
$$[A \otimes B]_{22} = 3 \otimes 10 \oplus 2 \otimes \epsilon = max(3+10, 2-\infty) = 13$$

Escrevendo em notação de matriz, obtém-se:

$$A \otimes B = \begin{pmatrix} -1 & 10 \\ 3 & 13 \end{pmatrix}.$$

Note que, em geral, o produto não é comutativo. Por exemplo, para as mesmas matrizes A e B definidas acima, tem-se:

$$B \otimes A = \begin{pmatrix} 13 & 12 \\ 1 & \epsilon \end{pmatrix} \neq A \otimes B.$$

Dadas as definições de operações matriciais, pode-se apresentar as matrizes que funcionam como elemento neutro para cada operação. Considere que $\varepsilon(n,m)$ denote uma matriz $n \times m$ com todos os elementos iguais a ε, e a matriz $E(n,m)$ como uma matriz $n \times m$ definida como:

$$[E(n,m)]_{ij} := \begin{cases} e & \text{para } i = j, \\ \varepsilon & \text{caso contrário.} \end{cases} \tag{3.17}$$

Se $n = m$ então $E(n,n)$ é chamada de matriz identidade. Quando o contexto permitir, $\varepsilon(n,m)$ e $E(n,m)$ serão escritos como ε e E, respectivamente. É fácil verificar que qualquer matriz $A \in \mathbb{R}_{max}^{n\times m}$ satisfaz as seguintes propriedades:

$$A \oplus \varepsilon(n,m) = A = \varepsilon(n,m) \oplus A, \tag{3.18}$$

$$A \otimes E(m,m) = A = E(n,n) \otimes A. \tag{3.19}$$

Além disso, para $k \geq 1$ pode se dizer que:

$$A \otimes \varepsilon(m,k) = \varepsilon(n,k) \qquad e \qquad \varepsilon(k,n) \otimes A = \varepsilon(k,m). \tag{3.20}$$

Para $\mathbb{R}_{max}^{n\times m}$, a adição de matrizes $\oplus$, definida em (3.1), é associativa, comutativa, possui o elemento zero $\varepsilon(n,m)$. Para $\mathbb{R}_{max}^{n\times n}$ o produto matricial $\otimes$, definida em (3.13), é associativa, distributiva em relação a $\oplus$, possui um elemento unitário $E(n,n)$, e $\varepsilon(n,n)$ de absorção.

O transposto de um elemento $A \in \mathbb{R}_{max}^{n\times m}$, denotado por $A^\top$, é definido de modo usual como $[A^\top]_{ij} = a_{ji}$, para $i \in \underline{n}$ e $j \in \underline{m}$. Assim como na adição e na multiplicação de escalares, a operação $\otimes$ tem prioridade sobre $\oplus$, quando se trata de matrizes pertencentes a $\mathbb{R}_{max}^{n\times m}$.

Os elementos de $\mathbb{R}_{max}^{n} = \mathbb{R}_{max}^{n\times 1}$ são chamados vetores. O j-ésimo elemento de um vetor $x \in \mathbb{R}_{max}^{n}$ é denotado por x_j, mas também pode ser escrito como $[x]_j$. Um vetor em $\mathbb{R}_{max}^{n}$ com todos os elementos iguais a e é chamado de vetor unitário e é denotado por $\mathbf{u}$, tal que $[\mathbf{u}]_j = e$, $\forall j \in \underline{n}$. Note que $\alpha \otimes \mathbf{u}$ denota um vetor com elementos iguais a α, $\forall \alpha \in \mathbb{R}_{max}$. Para qualquer $j \in \underline{n}$,a j-ésimo coluna da matriz identidade $E(n, n)$ é chamada de j-ésima base vetorial de $\mathbb{R}_{max}^{n}$ e é denotada por e_j. Sendo assim o j-ésimo elemento de e_j tem o valor de e , enquanto os outros elementos são iguais a ε.

Seja $A \in \mathbb{R}_{max}^{n\times m}$ e $x \in \mathbb{R}_{max}^{m}$. Então o produto $A \otimes x$ é definido por (3.13), para $B = x$. Claramente, $A \otimes A$ e também potências de altas ordens de A são definidas apenas para matrizes quadradas, i.e., para $A \in \mathbb{R}_{max}^{n\times n}$.

Para $A \in \mathbb{R}_{max}^{n\times n}$, a k-ésima potência de A por $A^{\otimes k}$ é definida por:

$$A^{\otimes k} := \underbrace{A \otimes A \otimes ... \otimes A}_{\text{k vezes}}. \tag{3.21}$$

Por definição, $A^{\otimes 0} := E(n, n)$. Note que $[A^{\otimes k}]_{ij}$ deve ser cuidadosamente distinguido de $[a_{ij}^{\otimes k}]$. O primeiro é elemento (i, j) da k-ésima potência de A, enquanto o último é a k-ésima potência de elemento (i, j) de A.

3.3 Matrizes e autômatos ponderados

Um autômato ponderado de estados finitos, é um par $(G = (X, \Sigma, \xi, x_0, X_m), f)$ em que G é um autômato de estados finitos definido na equação (2.8) e $f : X \times \Sigma \to \mathbb{R}^+$ é uma função de peso parcial, que associa cada transição a um número real positivo. Esse peso pode ser interpretado como um custo para efetuar a transição, ou tempo em que o evento demora a ser disparado, ou ainda, o tempo máximo em que ele deve disparar. Diferencia-se do autômato temporizado com guardas [28] pois é mais genérico, já que os pesos não precisam ser necessariamente interpretados como tempo [30]. Devido à sua simplicidade, esse será o modelo utilizado na análise das propriedades da diagnosticabilidade de falhas no capítulo 4.

Na figura 3.1 é apresentado um exemplo de um autômato ponderado. Note que a função de peso é dada da seguinte forma: $f(1, a) = 1$, $f(2, b) = 2$, $f(3, c) = 1$ e $f(4, d) = 3$. Esse grafo pode ser representado por uma matriz max-plus da seguinte forma:

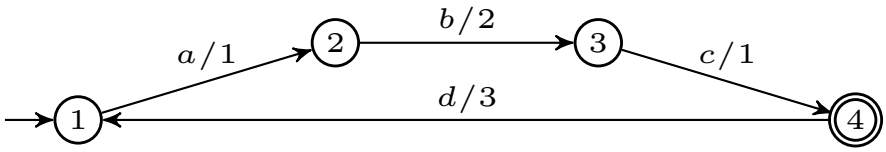

Figura 3.1: Autômato ponderado.

$$[A(n,n)]_{ij} := \begin{cases} f(j,\sigma), & para\ \xi(j,i)! \\ \varepsilon, & \text{caso contrário} \end{cases}, \tag{3.22}$$

isto é, a matriz A será quadrada tal que $n = |X|$. O elemento a_{ij} será igual ao peso da transição de j para o estado i. Se a transição não estiver definida, então o valor de a_{ij} será igual a ε. Caso haja mais de uma transição de um estado a outro, o peso que será adotado será o máximo dentre os pesos associados às transições. O motivo para o uso deste critério ficará mais claro no capítulo 4.

A matriz A que representa o autômato da figura 3.1 é dada por:

$$A = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & 3 \\ 1 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 2 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 1 & \varepsilon \end{pmatrix}$$

Lembrando que X é um conjunto de estados, considere aogra, que D seja o conjunto de arcos de um autômato ponderado G e $g : X \times \Sigma \times X \to X \times X$, uma função que relaciona uma transição de um evento σ, de um estado i para um estado j, com um arco (i,j). Caso haja mais de uma transição do estado i para j, o arco que será adotado será aquele de maior peso, conforme dito anteriormente. Para dois estados i,j a sequência de arcos $p = ((i_k,j_k) \in D : k \in \underline{m})$, tal que $i = i_1$, $j_k = i_{k+1}$, para $k < m$, e $j_m = j$ é chamado de *caminho* de i até j. Um caminho então consiste de estados $i = i_1, i_2, ..., i_m, j_m = j$ e possui *comprimento* m. O comprimento do caminho p será denotado por $|p|_l = m$. Além disso, se $i = j$, então o caminho é chamado de *circuito*. Um circuito $p = ((i_1,i_2),(i_2,i_3),...,(i_m,i_1))$ é chamado *elementar* se, restrito ao circuito, cada um dos estados tem apenas um arco de entrada e saída , isto é, se os estados i_k e i_l são diferentes para $k \neq l$. Um circuito que possui um único arco de um estado a ele mesmo é chamado de *autolaço*.

O conjunto de todos os caminhos de i até j de comprimento $m \geq 1$ é denotado por $P(i,j;m)$. Para um arco (i,j) em um autômato G, o peso de (i,j) é dado por a_{ji}, e o peso de um caminho é definido como a soma dos pesos de todos os arcos que constituem o caminho. Mais formalmente, para um caminho $p = ((i_1,i_2),(i_2,i_3),...,(i_m,i_{m+1})) \in P(i,j;m)$ com $i = i_1$ e $j = i_{m+1}$, define-se o peso de p, denotado por $|p|_w$, de acordo com a seguinte equação:

$$|p|_w := \bigotimes_{k=1}^{m} a_{i_{k+1}i_k}. \tag{3.23}$$

O *peso médio* de um caminho p é dado por $|p|_w/|p|_l$. Os caminhos em G podem ser combinados de modo a construir um novo caminho. Por exemplo, considere que $p = ((i_1, i_2), (i_2, i_3))$ e $q = ((i_3, i_4), (i_4, i_5))$ sejam dois caminhos em G. Então:

$$p \circ q = ((i_1, i_2), (i_2, i_3), (i_3, i_4), (i_4, i_5)),$$

será também um caminho em G. A operação $\circ$ é chamada de *concatenação de caminhos*. Claramente a operação não é comutativa, até mesmo quando $p \circ q$ e $q \circ p$ são definidos.

Exemplo 3.1 *Considere o autômato ponderado representado na figura 3.2. Note que O conjunto de estados do autômato é* $X = \{1, 2, 3\}$ *e o conjunto de arcos é dado por* $D = \{(1, 3), (3, 2), (2, 1), (3, 3)\}$. *É fácil verificar que o autômato ponderado da figura 3.2 é representado pela matriz A.*

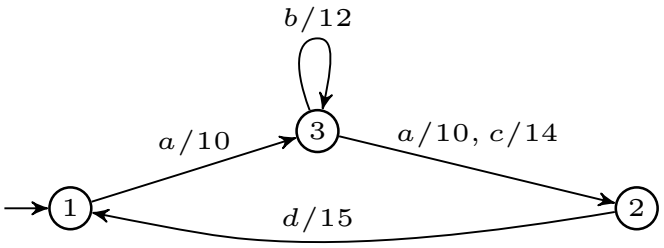

Figura 3.2: Autômato ponderado da matriz A do exemplo 3.1.

$$A = \begin{pmatrix} \varepsilon & 15 & \varepsilon \\ \varepsilon & \varepsilon & 14 \\ 10 & \varepsilon & 12 \end{pmatrix}$$

Note que $a_{23} = 14$, *uma vez que o maior peso associado ao arco* $(3, 2)$ *deve ser escolhido. O autômato da figura 3.2 consiste em dois circuitos elementares, i.e.,* $\rho = \{(1, 3), (3, 2), (2, 1)\}$ *e* $\theta = (3, 3)$. *O peso de* ρ *é dado por*

$$|\rho|_w = a_{12} + a_{23} + a_{31} = 39,$$

e o comprimento de ρ *é igual a* $|\rho|_l = 3$. *O peso do circuito* θ *é* $|\theta|_w = a_{33} = 12$ *e seu comprimento é igual a* 1.

O autômato ponderado G e as potências da sua correspondente matriz A são fortemente relacionadas. Será provado no próximo teorema que dado que existe um caminho de comprimento k do estado i para o estado j, o elemento $[A^{\otimes k}]_{ji}$ fornece o máximo peso entre todos os caminhos de comprimento k.

Teorema 3.1 *[32] Seja $A \in \mathbb{R}_{max}^{n\times n}$ uma matriz associada a um autômato ponderado G. Então, $\forall k \geq 1$,*

$$[A^{\otimes k}]_{ji} = max\{|p|_w : p \in P(i,j;k)\}.$$

Além disso, $[A^{\otimes k}]_{ji} = \varepsilon$, caso $P(i,j;k)$ seja vazio, i.e., quando não houver um caminho de comprimento k de i até j em G.

Prova A prova será feita por indução. Seja (i,j) um elemento arbitrário de $\underline{n} \times \underline{n}$. Para $k = 1$, os caminhos em $P(i,j;k)$ consistem apenas em um único arco cujo peso é dado pela definição $[A_{ji}]$. Se $[A_{ji}] = \varepsilon$, então não há arco (i,j) em G e $P(i,j;k) = \emptyset$.

Suponha agora que o teorema seja verdadeiro para k e seja $p \in P(i,j;k+1)$. Suponha ainda, sem perda de generalidade que exista no mínimo, um caminho em $P(i,j;k)$. Dessa forma p, pode ser dividido em subcaminhos de comprimento k, indo de i para algum estado l, e um caminho que consista em um único arco de l a j, ou mais formalmente:

$$p = \hat{p} \circ (l,j), \text{ tal que } \hat{p} \in P(i,j;k).$$

Considerando a hipótese de indução pode se dizer que:

$$max\{|\hat{p}|_w : \hat{p} \in P(i,j;k)\} = [A^{\otimes k}]_{li},$$

e, portanto, a expressão para o peso máximo de um caminho de i a j de comprimento $(k+1)$ será dada por:

$$\max_{l\in\underline{n}}(a_{jl} + [A^{\otimes k}]_{li}) = \bigoplus_{l=1}^{n} a_{jl} \otimes [A^{\otimes k}]_{li} = [A \otimes A^{\otimes k}]_{ji} = [A^{\otimes k+1}]_{ji}. \tag{3.24}$$

Agora retornamos ao caso em que $P(i,j;k+1) = \emptyset$, i.e., não existe um caminho de comprimento $k+1$ de i a j. Obviamente, isto implica que, para qualquer estado l, ou não existe um caminho de comprimento k de i a l ou não existe um arco de l a j (ou ambos). Sendo assim, $P(i,j;k+1) = \emptyset$ implica que, para qualquer l, no mínimo um dos valores a_{jl} ou $[A^{\otimes k}]_{li}$ é igual a ε. Como consequência:

$$[A^{\otimes k+1}]_{ji} = \varepsilon$$

o que completa a prova do teorema. □

Corolário 3.1 *[32] Para $A \in \mathbb{R}_{max}^{n\times n}$, considere que:*

$$A^{+} := \bigoplus_{k=1}^{\infty} A^{\otimes k} = A \oplus A^{\otimes 2} \oplus A^{\otimes 3} \oplus ... \tag{3.25}$$

O elemento $[A^{+}]_{ij}$ é o máximo peso de qualquer caminho de j a i (o valor de $[A^{+}]_{ij} = +\infty$ é possível). Além disso, pela equação (3.25), pode-se escrever:

$$[A^+]_{ij} = max[A^{\otimes k}]_{ij} : \forall k \geq 1,$$

em que $[A^{\otimes k}]_{ij}$ *é o máximo peso do caminho de* j *a* i *de comprimento* k*, vide teorema 3.1.*

A equação converge em um número k finito de passos desde que o autômato ponderado não possua *componentes fortemente conexas* [32].

Definição 3.1 *(Componentes fortemente conexas). Um conjunto de vértices* U *tal que* $U \subseteq V$ *é uma componente fortemente conexa se as seguintes condições são satisfeitas:*

1. *Para todo par de vértices* $u, v \in U$*, existe um caminho de* u *para* v *e um caminho de* v *para* u*.*

2. *O conjunto* U *é máximo em relação ao item* 1. *acima, ou seja,* $\forall v \in V \setminus U$*,* $U \cup \{v\}$ *não é uma componente fortemente conexa.*

Note que o autômato G_c da figura 3.3 possui apenas uma componente fortemente conexa formada pelos estados $\{1, 2, 3, 4, 5\}$, pois de cada um desses estados existe um caminho para acessar os outros estados que formam a componente fortemente conexa.

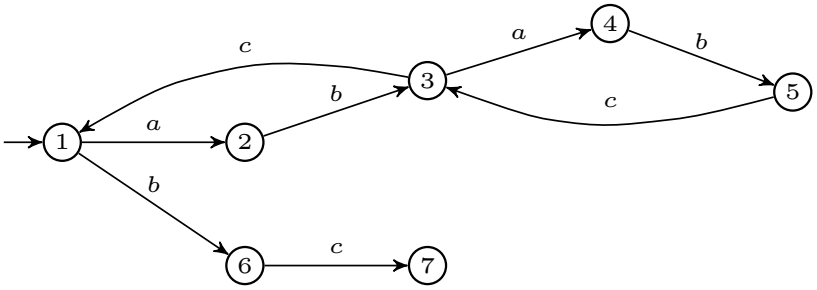

Figura 3.3: Autômato G_c

Para o autômato ponderado da figura 3.2, tem-se que a matriz A^+ não converge porque ela possui uma componente fortemente conexa $\{1, 2, 3\}$. Note que se $\bigoplus_{k=1}^{n} A^{\otimes k}$ se comporta da seguinte forma.

- $k = 1$, $A^+ = A = \begin{pmatrix} \varepsilon & 15 & \varepsilon \\ \varepsilon & \varepsilon & 14 \\ 10 & \varepsilon & 12 \end{pmatrix}$;

- $k = 2$, $A^+ = A \oplus A^{\otimes 2} = \begin{pmatrix} \varepsilon & 15 & 29 \\ 24 & \varepsilon & 26 \\ 22 & 25 & 24 \end{pmatrix}$;

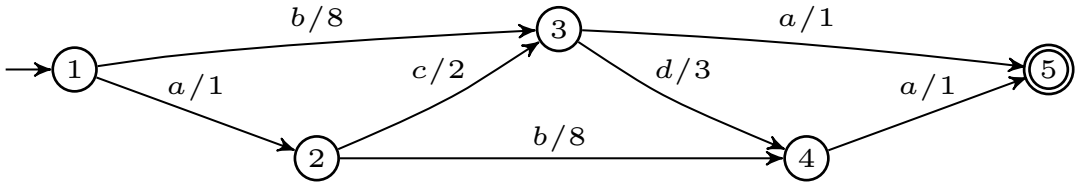

Figura 3.4: Autômato ponderado da matriz A do exemplo 3.2.

- $k = 5,\ A^{+} = A \oplus A^{\otimes 2} ... \oplus A^{\otimes 5} = \begin{pmatrix} 63 & 66 & 68 \\ 63 & 63 & 65 \\ 61 & 64 & 63 \end{pmatrix}$
- $k = 10,\ A^{+} = A \oplus A^{\otimes 2} ... \oplus A^{\otimes 10} = \begin{pmatrix} 129 & 132 & 131 \\ 126 & 129 & 131 \\ 127 & 127 & 129 \end{pmatrix}$
- $k = 100\ A^{+} = A \oplus A^{\otimes 2} ... \oplus A^{\otimes 100} = \begin{pmatrix} 1299 & 1302 & 1301 \\ 1296 & 1299 & 1301 \\ 1297 & 1297 & 1299 \end{pmatrix}$.

É possível verificar que quando $n \to \infty$, os valores da matriz também crescem indefinidamente.

Exemplo 3.2 *Considere o autômato ponderado da figura 3.4. Não é difícil verificar que a matriz A representa esse autômato ponderado.*

$$A = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 8 & 2 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 8 & 3 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 1 & 1 & \varepsilon \end{pmatrix}$$

Note que o autômato da figura 3.4 não possui elementos fortemente conexos e que o comprimento de maior caminho em G é igual a $k = 3$. Assim, de acordo com o teorema 3.1, a matriz $A^{\otimes k}$ converge para um k finito, conforme pode ser visto a seguir:

- $k = 1,\ A = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 8 & 2 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 8 & 3 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 1 & 1 & \varepsilon \end{pmatrix}$;

- $k = 2$, $A^+ = A \oplus A^{\otimes 2} = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 8 & 2 & \varepsilon & \varepsilon & \varepsilon \\ 11 & 8 & 3 & \varepsilon & \varepsilon \\ 9 & 9 & 4 & 1 & \varepsilon \end{pmatrix}$;

- $k = 3$, $A^+ = A \oplus A^{\otimes 2} \oplus A^{\otimes 3} = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 8 & 2 & \varepsilon & \varepsilon & \varepsilon \\ 11 & 8 & 3 & \varepsilon & \varepsilon \\ 12 & 9 & 4 & 1 & \varepsilon \end{pmatrix}$;

- $k = 4$, $A^+ = A \oplus A^{\otimes 2} \oplus A^{\otimes 3} \oplus A^{\otimes 4} = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 8 & 2 & \varepsilon & \varepsilon & \varepsilon \\ 11 & 8 & 3 & \varepsilon & \varepsilon \\ 12 & 9 & 4 & 1 & \varepsilon \end{pmatrix}$;

- $k = 5$, $A^+ = A \oplus A^{\otimes 2} \oplus A^{\otimes 3} \oplus A^{\otimes 4} \oplus A^{\otimes 5} = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 8 & 2 & \varepsilon & \varepsilon & \varepsilon \\ 11 & 8 & 3 & \varepsilon & \varepsilon \\ 12 & 9 & 4 & 1 & \varepsilon \end{pmatrix}$.

Conforme o esperado, a série converge para $k = 3$, isto é, para $k \geq 3$ o valor de A^+ é o mesmo que para $k = 3$.

3.4 Sistemas ponderados pelo tempo

Nesta seção, serão considerados SEDs ponderados no tempo. Nas subseções 3.4.1 e 3.4.2, serão apresentados os fundamentos básicos de sistemas ponderados pelo tempo e linguagem harmoniosa retirados de [35]. Na subseção 3.4.3 será proposto um algoritmo para o cálculo da máxima sublinguagem harmoniosa.

3.4.1 Definições

Uma estrutura mais complexa que a vista na seção 3.3 é chamada de sistema com ponderação no tempo. Nesses sistemas, o peso de cada transição é interpretado como o tempo de duração de um determinado evento associado àquela transição. Considerando que os eventos não são instantâneos, essa interpretação leva à possibilidade de sobreposição de eventos, isto é, eventos ocorrendo simultaneamente num certo

espaço de tempo. Em um sistema de manufatura, por exemplo, um braço robótico leva um tempo para carregar uma peça de uma esteira a outra. Sendo assim, é possível que durante esse tempo um outro evento ocorra no sistema. Entretanto, alguns eventos naturalmente não podem ocorrer ao mesmo tempo e essa característica é capturada numa relação chamada de relação de exclusão mútua. Esses conceitos serão esclarecidos ao decorrer da seção.

Um sistema ponderado pelo tempo é um par $(\mathcal{G} = (G_i, f_i) \in \varphi(\Sigma_i)|i \in I, h)$, em que I é um conjunto finito, $\mathcal{G}$ é uma coleção de autômatos ponderados de estados finitos dentro do conjunto de todos os autômatos ponderados $\varphi(\Sigma)$, e $h \subseteq (\cup_{i\in I}\Sigma_i) \times (\cup_{i\in I}\Sigma_i)$ é uma relação simétrica binária e reflexiva, chamada de *relação de exclusão mútua intrínseca*. Um par $(\sigma, \sigma') \in h$ se os disparos de σ e σ' são mutuamente exclusivos, i.e., se um evento estiver sob execução, o outro evento não é disparado; caso contrário, $(\sigma, \sigma') \notin h$. A razão para h ser reflexiva é que no máximo uma cópia de cada evento pode ser executada no sistema em cada instante de tempo; a razão para h ser simétrica é que a exclusão mútua é simétrica. Por isso, (σ, σ') e (σ', σ) tem o mesmo significado. O termo intrínseco é empregado, devido à relação imposta por h ser uma propriedade do sistema, que automaticamente é sempre verdadeira.

Duas condições são consideradas para lidar com esse tipo de sistema:

C1. Em cada autômato $(G_i, f_i) \in \mathcal{G}$, a ordem dos eventos em uma palavra $s \in L(G_i)$ denota a ordem de seus respectivos momentos iniciais de disparo de eventos, sendo possível que os momentos iniciais de dois disparos consecutivos em s sejam idênticos. Por exemplo, suponha que $s = abc$ e $t_a, t_b, t_c \in \mathbb{R}^+ \cup \{0\}$ como os correpondentes momentos iniciais dos eventos a, b, c em s. Então, $t_a \leq t_b \leq t_c$.

C2. Se um evento $\sigma \in \cup_{i\in I}\Sigma_i$ for compartilhado por vários componentes do autômato, então os momentos iniciais de disparo de σ em diferentes autômatos, devem ser sincronizados.

A condição C1 especifica que cada palavra de G_i impõe uma ordem parcial sobre os momentos iniciais dos eventos participantes, e a condição C2 descreve como as ordens parciais individuais são relacionadas entre si por meio de um evento comum.

Para especificar o comportamento global de um sistema ponderado pelo tempo, é necessária a composição de partes individuais de autômatos ponderados. Uma regra típica de composição foi descrita no capítulo 2, mas a maneira como se lida com eventos compartilhados não é aplicável a sistemas ponderados no tempo. Por isso é introduzida a seguinte composição paralela.

Definição 3.2 *(Composição paralela de autômatos ponderados). Dados dois autômatos ponderados* $(G_i, f_i)(i = 1, 2)$*, a composição paralela (ou produto síncrono)*

de (G_1, f_1) *e* (G_2, f_2) *é escrito como* $(G_1, f_1)||(G_2, f_2)$*, é o autômato* $((G = X, \Sigma, \xi, x_0, X_m), f)$*, em que* $X = X_1 \times X_2$*,* $\Sigma := \Sigma_1 \cup \Sigma_2$*,* $x_0 := (x_{1,0}, x_{2,0})$*,*$X_m = X_{1,m} \times X_{2,m}$*, e* $\xi : X_1 \times X_2 \times (\Sigma_1 \cup \Sigma_2) \to X_1 \times X_2$ *é definido da seguinte forma:*

$$\xi((x_1, x_2), \sigma) := \begin{cases} (\xi_1(x_1, \sigma), x_2), & \text{se } \sigma \in (\Sigma_1 - \Sigma_2) \wedge \xi_1(x_1, \sigma)! \\ (x_1, \xi_2(x_2, \sigma)), & \text{se } \sigma \in (\Sigma_2 - \Sigma_1) \wedge \xi_2(x_2, \sigma)! \\ (\xi_1(x_1, \sigma), \xi_2(x_2, \sigma)), & \text{se } \sigma \in (\Sigma_1 \cap \Sigma_2) \wedge \xi_1(x_1, \sigma)! \wedge \xi_2(x_2, \sigma)! \\ \text{não definido}, & \text{caso contrário} \end{cases}$$

e $f : X_1 \times X_2 \times (\Sigma_1 \cup \Sigma_2) \to \mathbb{R}^+$ *é definido como segue:*

$$f((x_1, x_2), \sigma) := \begin{cases} f(x_1, \sigma), & \text{se } \sigma \in (\Sigma_1 - \Sigma_2) \wedge \xi_1(x_1, \sigma)! \\ f(x_2, \sigma), & \text{se } \sigma \in (\Sigma_2 - \Sigma_1) \wedge \xi_2(x_2, \sigma)! \\ f(x_1, \sigma) \oplus f(x_2, \sigma), & \text{se } \sigma \in (\Sigma_1 \cap \Sigma_2) \wedge \xi_1(x_1, \sigma)! \wedge \xi_2(x_2, \sigma)! \\ \text{não definido}, & \text{caso contrário} \end{cases}$$

Dado $\mathcal{G} = (G_i, f_i) \in \varphi(\Sigma_i)|i \in I, h)$, chama-se (G, f, h) de *modelo centralizado* de $(\mathcal{G}, h)$. Note que a relação de exclusão mútua intrínseca h permanece inalterada depois da composição uma vez que é definida sobre eventos, sendo assim, a composição não é afetada por h.

Definição 3.3 *[35] Seja o modelo centralizado* (G, f, h)*, considere uma sequência* $s \in L(G)$*. Suponha que* $s = \sigma_1 ... \sigma_n$ *para algum* $n \in \mathbb{N}$ *e que* $s_q = \sigma_1 ... \sigma_q$*. Se* $q < 1$ *então* $s_q := \epsilon$*.*

1. *Um* selo temporal *de* s *com respeito a* (G, f, h) *é uma lista não descrescente de números reais não negativos* $\rho = (t_k^s \in \mathbb{R}^+ \cup \{0\} : k = 1, ..., n)$*, em que* $\forall q$ *e* $v \in \{1, \ldots, n\}$

$$q < v \wedge (\sigma_q, \sigma_v) \in h \Rightarrow t_q^s + f(\xi(x_0, s_{q-1}), \sigma_q) \leq t_v^s.$$

2. *O* tempo de execução *de* s *em relação a* ρ *é definido como:*

$$T_{(G,f,h)}(s, \rho) := max\{t_1^s + f(x_0, \sigma_1), ..., t_n^s + f(\xi(x_0, s_{n-1}), \sigma_n))\}.$$

3. *Seja* $\Theta_{(G,f,h)}(s)$ *o conjunto de todos os selos temporais de* s *e defina*

$$v_{(G,f,h)}(s) := min_{\rho \in \Theta_{(G,f,h)}(s)} T_{(G,f,h)}(s, \rho).$$

4. *Para qualquer* $W \subseteq L_m(G)$*,*

$$\omega(G, f, h, W) := sup_{s \in W} v_{(G,f,h)}(s)$$

é definido como o makespan *de W m relação a (G, f, h). Por convenção, $\omega(G, f, h, \emptyset) := \infty$.*

Cada t_k^s em um *selo temporal* é interpretado como o momento inicial do evento σ_k executado no estado $\xi(x_0, s_{k-1})$ e $t_k^s + f(\xi(x_0, s_{k-1}), \sigma_k)$ como o momento final da execução de σ_v. Se $(\sigma_q, \sigma_v) \in h$ e $q < v$, então, para que a execução de σ_v possa ser iniciada, a execução de σ_q deve ter sido finalizada. Sendo assim, tem-se que $t_q^s + f(\xi(x_0, s_{q-1}), \sigma_q) \leq t_v^s$. O *tempo de execução* $v_{(G,f,h)}(s)$ é interpretado como o tempo mínimo requerido para finalizar a execução de s, que é atingível se todo evento participante não controlável disparar imediatamente sempre que for eleito para disparar.

Para exemplificar os conceitos abordados nesta seção, considere o autômato $G \in \phi(\{a, b, c\})$ ilustrado na figura 3.5, para o qual tem-se a seguinte relação de exclusão mútua intrínseca: $h = \{(a, b), (b, c)\}$. Nesse autômato, a legenda "$a/2$" na transição $\xi(x_0, a) = x_1$ significa $f(x_0, a) = 2$ e segue da mesma forma para as outras transições. A lista $\omega_1 = (t_a, t_c, t_b) = (0, 2, 3)$ é uma *selo temporal* para $s = acb$ uma vez que $t_a + f(x_0, a) = 2 < t_b$, $t_c + f(x_1, c) = 3 < t_b$. A lista $\omega_2 = (t_a, t_c, t_b) = (0, 0, 2)$ é também um *selo temporal* para s pois $t_a + f(x_0, a) = 2 < t_b$ e $t_c + f(x_1, c) = 1 < t_b$. Pode-se verificar que não há outra *selo temporal* tal que $T_{(G,f,h)}(s, \omega) < T_{(G,f,h)}(s, \omega_2)$. Sendo assim, o *tempo de execução* de s é dado por $v_{(G,f,h)}(s) = t_b + f(x_2, b) = 5$.

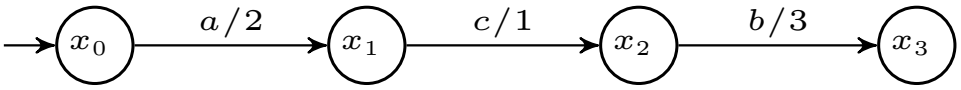

Figura 3.5: Autômato ponderado pelo tempo G.

3.4.2 Linguagem harmoniosa

Seja $\Sigma = \Sigma_c \cup \Sigma_{uc}$ uma partição de Σ, em que Σ_c e Σ_{uc} denotam respectivamente, os conjuntos de eventos *controláveis* e *não controláveis*.

Definição 3.4 *Sejam o modelo centralizado (G, f, h) e K uma linguagem tal que $\bar{K} \subseteq L_m(G)$. Então, K é dito controlável em relação a (G, f, h) se*

$$\bar{K}\Sigma_{uc} \cap L(G) \subseteq \bar{K} \tag{3.26}$$

De acordo com a definição 3.4, tem-se que o conceito de controlabilidade é diretamente estendido do padrão original definido em [36], para sistemas ponderados no tempo. Retomando o paradigma de controle de Ramadge-Wonham, há duas considerações importantes: que os disparos dos eventos são *instantâneos* e *assíncronos*.

O último significa que não mais do que um evento pode ser disparado em cada instante de tempo. Seja $\phi(\Delta)$, o conjunto dos autômatos que tem Δ como seu conjunto de eventos. Então, uma especificação $E \in \phi(\Delta)$ com $\Delta \subseteq \Sigma$ pode ser interpretado como uma especificação de uma ordem sequencial de disparo de eventos. Quando cada evento tem uma duração de disparo não nula, os disparos de dois diferentes eventos podem se sobrepor. Sendo assim, nenhuma dessas afirmações básicas são verdadeiras, o que sugere que é necessário prover uma nova interpretação para a especificação E antes de se falar sobre controle supervisório. Dada uma palavra $s = ab \in L_m(G)$, ela pode ser interpretada de duas maneiras:

1. O momento inicial de disparo de a deve preceder o momento inicial de disparo de b;

2. O momento inicial de b se dá depois do disparo do evento a ter sido completado.

A fim de lidar com ambas as interpretações unificadamente, associa-se E com a relação binária simétrica $h_E \subseteq \Delta \times \Delta$, chamada de *relação de exclusão mútua forçada*. A diferença entre h e h_E é que h é uma propriedade intrínseca de G e sempre ocorre, enquanto h_E é imposta por um usuário e nem sempre ocorre. Sendo assim, pode ser necessária uma estratégia de controle para forçar que h_E aconteça. Por exemplo, dados dois eventos não controláveis $a, b \in \Sigma_{uc}$, se $(a, b) \in h$ então é sabido que as execuções de a e b nunca poderão se sobrepor, havendo ou não um supervisor. Se $(a, b) \in h_E$, então, em geral, é necessário um supervisor para prevenir o sistema de ir para um estado onde a e b possam se sobrepor. Para um melhor entendimento desses conceitos, faz-se necessária a seguinte definição vista em [35].

Definição 3.5 *(Linguagem harmoniosa). Dados o modelo centralizado (G, f, h), uma sublinguagem $K \subseteq L(G)$ e uma relação de exclusão mútua forçada $h_E \subseteq \Delta \times \Delta$ com $\Delta \subseteq \Sigma$, então K é dita harmoniosa em relação a (G, f, h) e h_E se $\forall s \in \bar{K}$, $a, b \in \Sigma_{uc}$ e $s' \in \Sigma_{uc}^{\star}$:*

$$sas'b \in \bar{K} \wedge a \neq b \Rightarrow [(a, b) \notin h_E \vee (\forall c \in s'b)(a, c) \in h]. \tag{3.27}$$

Em outras palavras, para que K seja harmoniosa em relação a (G, f, h) e h_E é necessário que quaisquer dois eventos não controláveis diferentes, a e b, conectados por um caminho de eventos não controláveis s', "obedeçam" à relação de exclusão mútua forçada h_E no sentido de que se $(a, b) \notin h_E$ ou a e b sejam separados por um outro evento não controlável $c \in s'b$, de tal sorte que a execução de a seja finalizada antes que c seja disparado (i.e., $(a, c) \in h$). Note que se $(a, c) \in h$, então o momento inicial de execução de b não é mais longo do que a execução de c, que significa que a e b devem ser mutuamente exclusivos. Se nenhuma dessas duas

condições forem verdadeiras, não é possível interferir na execução de a e b depois da execução de s, uma vez que são eventos não controláveis e conectados por um caminho s' também não controlável. Como resultado, as execuções de a e b podem se sobrepor, o que viola a relação de exclusão mútua forçada h_E. Note que os eventos não controláveis são o foco principal, uma vez que o disparo dos eventos controláveis pode ser controlado, evitando assim a sobreposição indesejada.

Dada uma relação de exclusão mútua forçada $h_E \subseteq \Delta \times \Delta$, com $\Delta \subseteq \Sigma$, considere $\{K_j \subseteq L(G) : j \in J\}$, sendo J um conjunto finito e K_j o conjunto de todas as sublinguagens em relação a (G, f, h) e h_E. A partir da definição 3.5, é possível mostrar [37] que $\cup_{j \in J} K_j$ é também harmoniosa em relação a (G, f, h). Sendo assim, a harmonia é fechada com relação à união. Sejam (G, f, h) um modelo centralizado e (E, h_E) uma especificação com $E \in \varphi(\Delta)$, tal que $\varphi(\Delta)$ seja o conjunto de todos os autômatos que possuam Δ como alfabeto. Defina as seguintes classes de linguagem:

- $\mathcal{H}(G, f, h, E, h_E) = \{K \subseteq L_m(G)||L_m(E)$: K é harmonioso em relação a (G, f, h) e h_E $\}$,
- $\mathcal{C}(G, f, h, E, h_E) = \{K \subseteq L_m(G)||L_m(E)$: K é controlável em relação a (G, f, h) e harmonioso em relação a (G, f, h) e h_E $\}$.

Considere $K \in \mathcal{C}(G, f, h, E, h_E)$ uma sublinguagem controlável harmoniosa de (G, f, h) e (E, h_E). Uma vez que a controlabilidade e a harmonia são fechadas com relação à união, existe $K_* \in \mathcal{C}(G, f, h, E, h_E)$ tal que $\forall K \in \mathcal{C}(G, f, h, E, h_E)$, $K \subseteq K_\star$. A linguagem $K_\star$ é chamada de *sublinguagem harmoniosa controlável suprema* de (G, f, h) e (E, h_E) sendo denotada por $K^{\uparrow CH} = sup\mathcal{C}(G, f, h, E, h_E)$. Para computá-la é necessário o seguinte teorema.

Teorema 3.2 *Suponha que $K \subseteq L(G)$ seja regular, i.e., reconhecida por um autômato de estado finito $H := (Z, \Sigma, \gamma, z_0, Z_m)$. Então K é harmoniosa em relação a (G, f, h) e h_E se, e somente se, não existe estado $z \in Z$ alcançável a partir de um z_0 tal que existem eventos $a, b \in \Sigma_{uc}$ e $s \in \Sigma_{uc}^\star$ tais que:*

$$a \neq b \wedge \gamma(z, asb)! \wedge (a, b) \in h_E \wedge (\forall c \in sb)(a, c) \notin h \tag{3.28}$$

De acordo com o teorema 3.2, se K for não harmoniosa, então para um conjunto Z finito, é possível procurar todos os estados z que violam a harmonia e removê-los de H. Quando um estado é removido, todas as transições de entrada e saída desse estado também são removidas. A sublinguagem resultante deve ser a maior sublinguagem de K harmoniosa em relação a (G, f, h) e h_E. Uma vez que a especificação (E, h_E) é dada, considere $K = L_m(G||(E)$ como a linguagem que marca o autômato $G||E$. Então, pode-se computar a maior sublinguagem harmoniosa $K^{\uparrow H} = K'$,

tal que $K' \subseteq K$. Após isso, pode-se calcular a sublinguagem controlável suprema $K^{\uparrow C} = K''$, com $K'' \subseteq K$, em relação a (G, f, h) utilizando o procedimento proposto em [38]. Logo $K'' = K^{\uparrow CH} = sup\mathcal{C}(G, f, h, E, h_E)$. É possível que alguns estados sejam removidos do autômato H' que marca a linguagem gerada por K'. Note que a linguagem K' é harmoniosa e portanto H' não possui estados que violam a harmonia. Então H'' também não possui que violam a harmonia, uma vez que $K'' \subseteq K'$.

Exemplo 3.3 *Como ilustração, considere o autômato G da figura 3.6. Suponha que* $\Sigma_c = \{a, b, e\}$, $\Sigma_{uc} = \{c, d, p, q\}$, $h = \{(a, a), (b, b), (c, c), (d, d), (e, e), (p, p), (q, q)\}$ $h_E = \{(p, q), (a, b)\}$. *O autômato que marca a linguagem harmoniosa suprema de G é apresentado na figura 3.7. Note que, a partir do estado 5, existe um caminho* $s = \epsilon \in \Sigma_{uc}^*$ *entre p e q que viola a harmonia de acordo com o teorema 3.2. Por isso, estado* 5 *foi removido de G.*

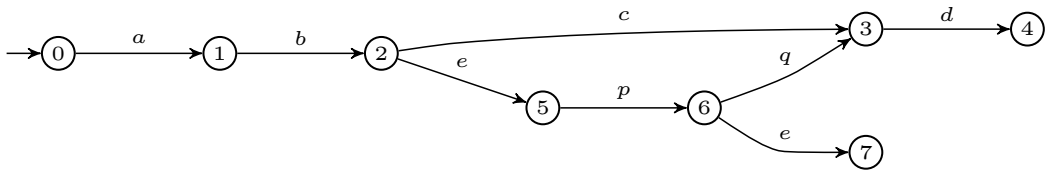

Figura 3.6: Autômato do exemplo 3.3 .

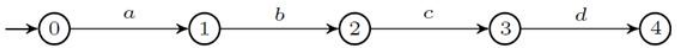

Figura 3.7: Autômato que marca a linguagem harmoniosa suprema do exemplo 3.3.

3.4.3 Computando $K^{\uparrow H}$

Nesta subseção será proposto o cálculo da sublinguagem harmoniosa suprema. Quando uma dada linguagem K não é harmoniosa, é possível encontrar a "maior" sublinguagem de K que seja harmoniosa, em que "maior" deve ser entendido em termos de conjunto de inclusão. Seja C_H o conjunto de todas as linguagens L que atendem ao teorema 3.2. É possível mostrar que a sublinguagem harmoniosa suprema [35] é dada por:

$$K^{\uparrow H} = \bigcup_{L \in C_H} L.$$

Sendo assim, tem se que, no "pior" caso: $K^{\uparrow H} = \emptyset$, uma vez que $\emptyset \in C_H$. Por outro lado, se K for harmoniosa, então $K^{\uparrow H} = K$.

Em [35], Su et al. introduzem o conceito de autômatos ponderados pelo tempo e de linguagem harmoniosa. No entanto, não é apresentado um método para calcular

a sublinguagem harmoniosa suprema. Neste trabalho será proposto um algoritmo para a obtenção da sublinguagem harmoniosa suprema de uma linguagem regular K.

Seja K uma linguagem regular e seja G o autômato que marca essa linguagem. Um algoritmo para a obtenção de um autômato H tal que $L_m(H) = K^{\uparrow H}$ será proposto a seguir.

Algoritmo 3.1 *Cálculo da sublinguagem harmoniosa suprema*

- Entradas*:*
  - *Autômato* $G = (X_G, \Sigma_G, \xi_G, \Gamma_G, x_{0,G}, X_{m,G})$ *tal que* $L_m(G) = K$
  - *Partição* Σ_c, Σ_{uc} *de* Σ
  - *Relações de exclusão mútua* h *e* h_E
- Saída*:*
  - *Autômato* $H = (X_H, \Sigma_H, \xi_H, \Gamma_H, x_{0,H}, X_{m,H})$ *tal que* $L_m(H) = K^{\uparrow H}$
- *Passo 1: Construa o autômato* $G_{uc} = (X_{uc}, \Sigma_{uc}, \xi_{uc}, \Gamma_{uc}, x_{0,uc}, x_{0,uc})$ *tal que* $\xi(x_{0,uc}, e) = x_{0,uc}$, $\forall e \in \Sigma_{uc}$, *isto é,* G_{uc} *é um autômato composto por apenas um estado marcado e com autolaços rotulados com todos os eventos de* Σ_{uc}.
- *Passo 2: Forme o conjunto* $h_{E_{uc}} = \{(p, q) \in h_E : p, q \in \Sigma_{uc}\}$.
  - *Se* $h_{E_{uc}} = \emptyset$ *então* $H = G$. *Caso contrário, forme o autômato* $G_h = (X_{G_h}, \Sigma, \xi_{G_h}, \Gamma_{G_h}, x_{0,G_h}, X_{m,G_h})$ *tal que* $X_{G_h} = X_G$, $\xi_{G_h} = \xi_G$, $\Gamma_{G_h} = \Gamma_G$, $x_{0,G_h} = x_{0,G}$, $x_{m,G_h} = \emptyset$ *e vá para o passo 3.*
- *Passo 3: Seja* $|h_{E,uc}| = l$, $h_{E_{uc}} = \{(p_1, q_1), (p_2, q_2), \ldots, (p_l, q_l)\}$. *Para* $i = 1, \ldots,$ *selecione* $(p_i, q_i) \in h_{E,uc}$, *de tal forma que* $(p_i, q_i) \notin h$. *Caso* $(p_i, q_i) \in h$, $\forall i = 1, ..., l$ *então* $H = G$. *Caso contrário, vá ao passo seguinte.*
- *Passo 4: Para* i *variando de 1 a* l, *forme o conjunto* $X_{p_i} = \{x \in X_{G_h} : p_i \in \Gamma_{G_h}(x) \wedge \forall s \in \Sigma^* : f(x_0, s) \in x, \exists s' \in \bar{s}, s'p \notin \bar{s}.\}$, *isto é, o conjunto* X_{p_i} *é obtido varrendo-se o autômato* G *a partir do estado inicial até encontrar os estados* x, *tais que* $p_i \in \Gamma_{G_h}(x)$ *pela primeira vez.*
- *Passo 5: Forme os autômatos* $G_{p_i} = \{X_{G_h}, \Sigma, \xi_{G_h}, \Gamma_{G_h}, x_{0,p_i}, \emptyset\}$, $x_{0,p_i} \in X_{p_i}$ *e forme os autômatos* $G_{p_i,q_i} = \{X_{G_h}, \Sigma, \xi_{G_h}, \Gamma_{G_h}, x_{0,p_i}, X_{m,i}\}$, *em que* $X_{m,i} = \{x \in X_{G_h} : q_i \in \Gamma_{G_h}(x)\}$. *Compute* $G_{v_i} = trim(G_{p_i,q_i})$

- *Passo 6: Verifique se existe algum elemento* $(p_i, \sigma_{uc}) \in h$, $\sigma_{uc} \in \Sigma_{uc}$. *Caso exista, forme um novo autômato* $G_{huc} = (X_{huc}, \xi_{huc}, \Gamma_{huc}, x_{0,huc}, x_{0,huc})$, *tal que* $X_{huc} = X_{uc}$, $\xi_{huc} = \xi_{uc}$, $\Gamma_{huc} = \Gamma_{uc}$, $x_{0,huc} = x_{0,uc}$, $\xi(x_{0,huc}, e) = x_{0,huc}$, $\forall e \in \Sigma_{uc} \setminus \{\sigma_{uc}\}$. *Caso contrário,* $G_{huc} = G_{uc}$.

- *Passo 7: Compute* $G'_{v_i} = trim(G_{v_i} \times G_{huc})$

  - *Se* G'_{v_i} *for um autômato vazio,* $H = G$.
  - *Caso contrário, remova o estado* x *de* G_h. *Repita os passos de 3 a 8 tal que* $h_{E_{uc}} = \{(q_1, p_1), (q_2, p_2), \ldots, (q_l, p_l)\}$, *isto é, trocando* p_i *por* q_i.

O algoritmo 3.1 termina quando todos os estados x forem verificados, e removidos ou não de G_h. Após a verificação, $H = Ac(G_h)$.

Prova Em termos de linguagem, para determinar a sublinguagem harmoniosa suprema de $L(G)$ de acordo com o teorema 3.2, é necessário remover de L todas sequências $s' = s_h t$ tal que $t \in L$, isto é, todas as sequências que possuem s_h como prefixo, tal que $s_h = psq$ de modo que $\exists p \neq q$, $\{(p, q) \in h_E$, $p, q \in \Sigma_{uc}$, $s \in \Sigma^{\star}_{uc}$ e $(\forall c \in sq)(p, c) \notin h\}$. Para remover essas sequências s' que violam a harmonia, é necessário remover todo estado z alcançável do estado inicial, tal que $\gamma(z, s_h)$ seja definida. Por essa razão, o algoritmo 3.1 busca todos os estados z de G que possuem p como evento ativo e marca os estados z_m, tais que $(p, q) \in h_E$ e $p, q \in \Sigma_{uc}$. A partir daí, é construído o autômato G_v, tal que $\forall s_v \in L_m(G_v)$, $\xi(z, s_v q)$ é definida em G. Também é construído o autômato G_{huc}, que possui apenas um estado com autolaços com todos os eventos não controláveis, exceto os eventos c tal que $(p, c) \notin h$. Se a operação produto entre G_v e G_{huc} não resultar em um autômato vazio, então haverá uma sequência s_h que viola as condições de harmonia. Portanto cada estado z em que isso ocorre deve ser removido de G para se determinar o autômato H tal que $L(H)$ é a sublinguagem harmoniosa suprema. □

Observação 3.1 *(Análise da complexidade computacional). Note que os autômatos formados no algoritmo 3.1 são obtidos por meio de alguma operação unária com o autômato de entrada* G. *Então no pior caso, possuem* $|X|$ *estados e* $|X||\Sigma|$ *transições. Logo, se for necessário que n autômatos sejam construídos, a complexidade é* $O(n \times |X|)$, *isto é, linear com o número de estados do autômato de entrada* G. *O exemplo a seguir ilustra os passos do algoritmo 3.1.*

Exemplo 3.4 *Seja* K *uma linguagem marcada pelo autômato* G *representado na figura 3.8. Suponha que* $\Sigma_c = \{a, b, e\}$, $\Sigma_{uc} = \{c, d, p, q, u\}$, $h = \{(a, a), (b, b), (c, c), (d, d), (e, e), (p, p), (q, q), (u, u), (p, c)\}$ $h_E = \{(p, q), (a, b)\}$. *O autômato* G_{euc} *é mostrado na figura 3.9. Após computar* G_{euc}, *o algoritmo procura em* h_E *por elementos* $(e_1, e_2) \in \Sigma_{uc}$. *No exemplo, há apenas um elemento com*

todos os eventos não controláveis: (p, q)*. Com base nisso, deve-se buscar quais estados a partir do estado inicial de* G *têm como evento ativo o evento* p *e construir os autômatos* G_{p_i}*,* $i = 1, 2, \ldots, l$*. Como os estados 1 e 5 possuem* p *como evento ativo, dois autômatos são computados:* G_{p_1} *e* G_{p_2}*, sendo que o primeiro tem o estado 1 como inicial e o último o estado 5, conforme visto nas figuras 3.10 e 3.11. Então são marcados os estados de* G_{p_1} *e* G_{p2} *que têm o evento* q *em seu conjunto de eventos ativos, i.e., os estados* 4, 7 *e* 10 *dando origem aos autômatos* $G_{p_1q_1}$ *e* $G_{p_2q_2}$ *respectivamente. Os autômatos* $G_{p_1q_1}$ *e* $G_{p_2q_2}$ *são representados nas figuras 3.12 e 3.13, respectivamente. Após isto calcula-se* $G_{v_1} = trim(G_{p_1q_1})$ *e* $G_{v_2} = trim(G_{p_2q_2})$*, mostrados nas figuras 3.14 e 3.15, respectivamente. Note que* (p, c) *está em* h*. Logo o caminho de* p *até* q *passando por* c *não viola a harmonia já que existe uma exclusão mútua entre* p *e* c*. Portanto, o evento* c *deve ser removido de* G_{euc}*, dando origem ao autômato* G_{huc} *mostrado na figura 3.16. Sendo assim, o autômato do produto entre* G_{v_1} *e* G_{huc} *"para" no estado 2 de* G_{v_1} *porque* G_{huc} *não contém* c*, e portanto,* $G'_{v_1} = trim(G_{v_1} \times G_{huc})$ *é vazio, já que nunca alcançará o estado marcado e portanto o estado inicial de* G_{v_1} *não será removido de* G_h*. Porém o mesmo não ocorre com o estado 5. O autômato* $G'_{v_2} = trim(G_{v_2} \times G_{huc})$ *não é vazio e é mostrado na figura 3.17. Isto ocorre porque existe um caminho não controlável até o estado marcado, tal que todos os eventos neste caminhos estão em* G_{huc}*. Logo, o estado 5 deve ser removido de* G*.*

Neste momento o algoritmo repete todo este procedimento invertendo a busca, isto é, procura-se um caminho de q *para* p *que fere a harmonia. Isto ocorre no estado 4 que é removido de* G*. Sendo assim, computa-se o autômato* G_h*, que é o próprio autômato* G *porém com os estados que ferem a harmonia removidos. O autômato* H *que marca a linguagem harmoniosa suprema de* G *é apresentado na figura 3.18.*

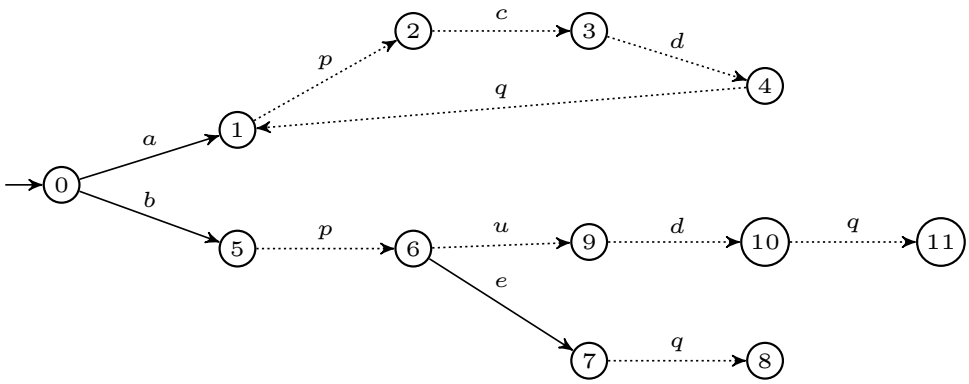

Figura 3.8: Autômato G do exemplo 3.4.

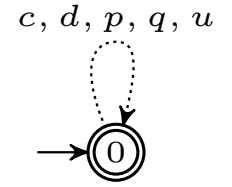

Figura 3.9: Autômato G_{euc} exemplo 3.4.

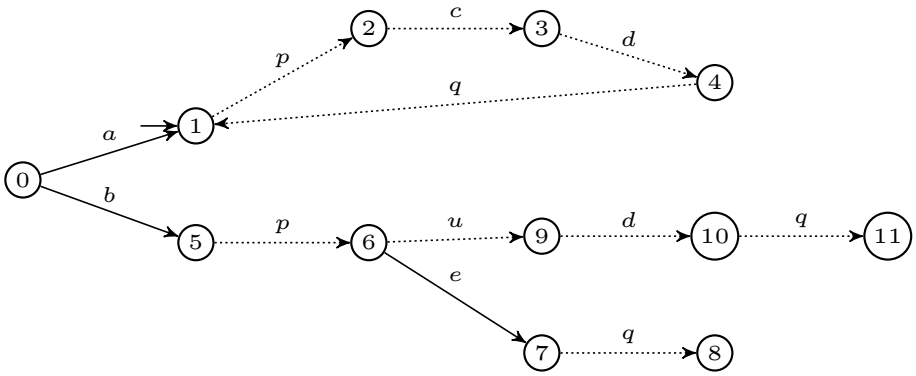

Figura 3.10: Autômato G_{p_1} do exemplo 3.4.

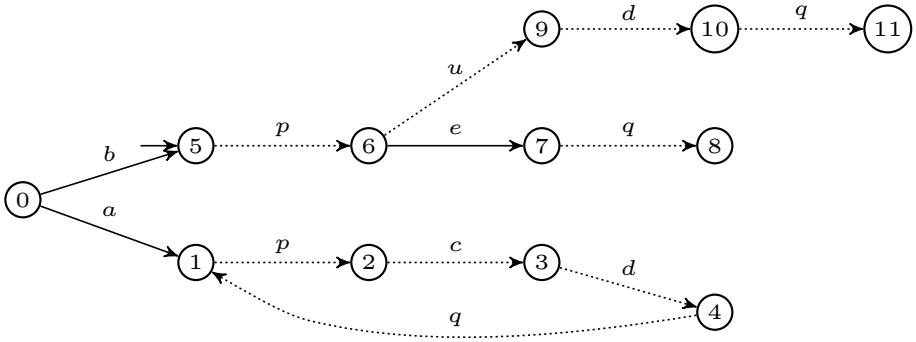

Figura 3.11: Autômato G_{p_2} do exemplo 3.4.

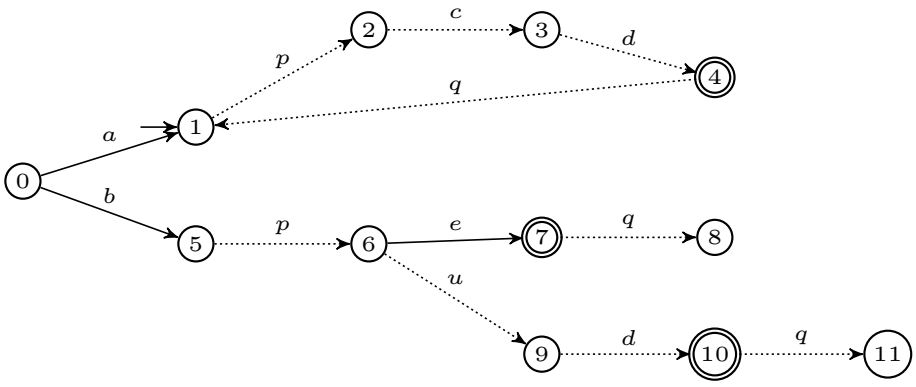

Figura 3.12: Autômato $G_{p_1q_1}$ do exemplo 3.4.

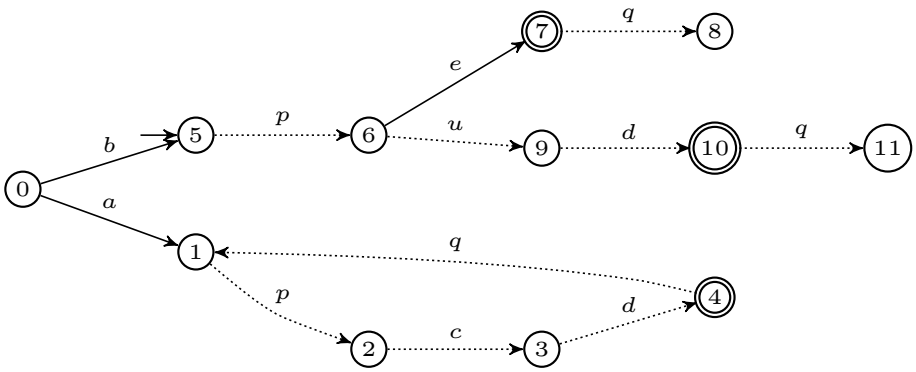

Figura 3.13: Autômato $G_{p_2q_2}$ do exemplo 3.4.

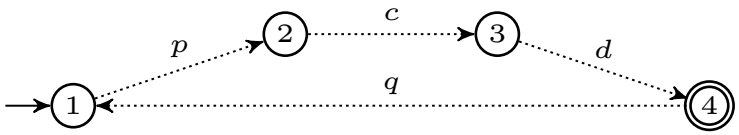

Figura 3.14: Autômato G_{v_1} do exemplo 3.4.

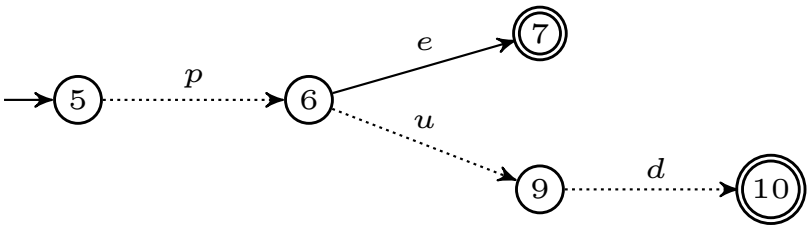

Figura 3.15: Autômato G_{v_2} do exemplo 3.4.

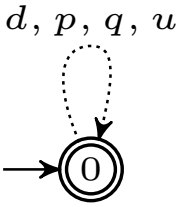

Figura 3.16: Autômato G_{huc} do exemplo 3.4.

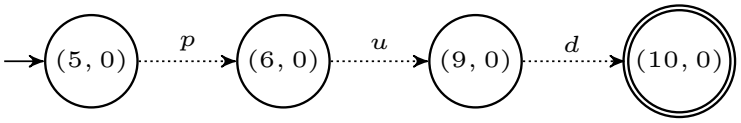

Figura 3.17: Autômato G'_{v_2} do exemplo 3.4.

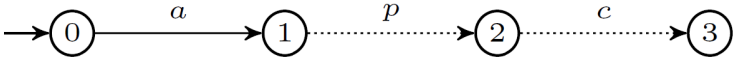

Figura 3.18: Autômato H do exemplo 3.4.

Capítulo 4

Diagnose de falhas de sistemas a eventos discretos modelados por autômatos ponderados

O problema da diagnose de falhas tem sido amplamente abordado nos últimos anos, principalmente em sistemas que utilizam o modelo global do sistema a eventos discretos (DES) [18]. As metodologias desenvolvidas para a diagnose de falhas de SEDs podem ser aplicadas não só a sistemas em que o modelo por eventos discretos é o mais apropriado (por exemplo, redes de comunicação e sistemas de computação e de manufatura), como também a diversos sistemas dinâmicos de variáveis contínuas (SDVC) [23], uma vez que esses sistemas podem também ser modelados como SEDs dependendo do grau de abstração [39]. Neste capítulo são apresentados fundamentos da diagnose de falhas bem como uma nova abordagem para verificação da diagnosticabilidade.

Sendo assim, este capítulo está estruturado da seguinte forma: na seção 4.1 é introduzido o problema de diagnose de falhas. Na seção 4.2 é estudada a diagnose de falhas baseada no autômato diagnosticador G_d. Na seção 4.3 é apresentada uma nova abordagem para a verificação da diagnosticabilidade de um SED baseada no diagnosticador de teste, chamado G_t, que é obtido a partir da composição paralela entre o autômato diagnosticador G_d e o autômato G_ℓ. Na seção 4.4, é considerado o problema da diagnose de falhas de SEDs modelados por autômatos com ponderação. Na seção 4.5 é apresentado um exemplo de análise de diagnosticabilidade de uma célula de manufatura. Comentários finais sobre esse capítulo são feitos na seção 4.6

4.1 Diagnosticabilidade

A diagnosticabilidade é uma propriedade que determina, a partir do modelo de um sistema, se é possível detectar e localizar a ocorrência de um evento de falha após a ocorrência de um número finito de eventos.

Na diagnose de falhas em SEDs, é importante atentar para o fato de que as falhas a serem diagnosticadas são eventos não observáveis, isto é, eventos cujas ocorrências não podem ser registradas por sensores ou o evento ocorre em um local remoto, mas não é comunicado ao lugar que está sendo modelado. Além disso, a ocorrência de falhas altera o comportamento do sistema, porém, não necessariamente, leva o sistema a uma parada; por exemplo, em sistemas de manufatura, a ocorrência de uma falha pode levar a uma degradação dos indicadores de eficácia global dos equipamentos (disponibilidade, eficiência e qualidade).

Portanto, são construídos sistemas para a diagnose de falhas, cujo objetivo é informar a ocorrência de falhas tendo como base apenas os eventos que foram registrados pelos sensores, isto é, os eventos observáveis. O projeto desses sistemas requer, em primeiro lugar, a construção de um modelo a eventos discretos do sistema que capture tanto o comportamento normal quanto o comportamento do sistema levando-se em consideração a ocorrência da falha. A segunda parte do projeto é calcada em um arcabouço teórico desenvolvido nas duas últimas décadas e consiste no desenvolvimento de um conjunto de regras a serem seguidas para a identificação e a diagnose de falhas.

Suponha que o conjunto de eventos Σ de G possa ser particionado da forma: $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$, sendo que Σ_o representa o conjunto de eventos observáveis de Σ e o conjunto Σ_{uo} representa o conjunto de eventos não-observáveis de Σ. Seja $\Sigma_f \subseteq \Sigma_{uo}$ o conjunto de eventos de falha. Além disso, suponha que o conjunto de eventos de falha Σ_f também possa ser particionado da seguinte forma:

$$\Sigma_f = \bigcup_{i=1}^{l} \Sigma_{f_i},$$

em que Σ_{f_i} representa o conjunto de eventos de falha do mesmo tipo e ℓl é o número de tipo de falha. Seja a linguagem gerada por G, $\mathcal{L}(G)$, representada por L. A diagnose de um evento de falha pertencente ao conjunto Σ_{f_i} pode ser realizada de duas formas distintas: centralizada e descentralizada. Na arquitetura centralizada, um evento de falha do conjunto Σ_{f_i} é diagnosticado se puder ser identificado a partir de observações apenas do conjunto Σ_o. Na arquitetura descentralizada, observações de eventos são distribuídas ao longo de m diagnosticadores locais, sendo que cada diagnosticador possui seu próprio conjunto de eventos observáveis, além de não se comunicarem entre si ou com um coordenador. Um evento de falha pertencente ao

conjunto Σ_{f_i} é diagnosticado quando pelo menos um diagnosticador local identifica sua ocorrência. A arquitetura descentralizada é conhecida como codiagnose.

Para construir dispositivos para diagnose de falhas, é necessário entender o conceito de *diagnosticabilidade*. A noção de diagnosticabilidade está baseada na possibilidade de se detectar qualquer tipo de falha em um sistema com um atraso finito utilizando-se somente as ocorrências de eventos observáveis registradas.

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

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

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

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

A hipótese A1 é feita considerando que o sistema está sempre em operação. A hipótese A2 é necessária para evitar que a ocorrência do evento associado à falha possa vir a não ser detectada caso o sistema fique preso em um ciclo de estados ligados por eventos não observáveis após a sua ocorrência. Essa hipótese será removida ainda neste capítulo, sendo tais ciclos referidos como escondidos. A hipótese A3 é feita por simplicidade, uma vez que, para cada conjunto de eventos de falhas de um mesmo tipo, é necessário criar um rótulo diferente; os fundamentos relacionados à análise da diagnosticabilidade são, contudo, os mesmos daqueles empregados para um único tipo de falha.

Para a definição de diagnosticabilidade as seguinte definições serão utilizadas:

Definição 4.1

- *A linguagem de L após s, denotada por L/s, é definida como*

$$L/s = \{t \in \Sigma^* : st \in L\}$$

- *Suponha que $\Psi(\Sigma_f)$ denote o conjunto de todas as sequências de L que terminam com o evento σ_f associado à falha que se deseja diagnosticar. Formalmente, se s_f denota o último evento de uma sequência s, então,*

$$\Psi(\Sigma_f) = \{s \in L : s_f \in \Sigma_f\}$$

Com um pequeno abuso de notação, dada uma sequência s, a relação de pertinência $\Sigma_f \in s$ pode ser usada para denotar que $\bar{s} \cap \Psi(\Sigma_f) \neq \emptyset$, na qual $\bar{s}$ denota o fecho do prefixo de s.

- *Sequência que contém uma falha: A sequência $s \in L$ é uma sequência que contém uma falha se $\Sigma_f \in s$.*

Informalmente, diz-se que a linguagem gerada por um autômato é diagnosticável em relação a uma projeção $P_o : \Sigma^* \rightarrow \Sigma_o^*$ e um conjunto de eventos de falhas Σ_f se a ocorrência de qualquer evento de Σ_f puder ser detectada após um atraso finito da ocorrência dessa falha usando somente sequências de eventos observáveis. Formalmente, a diagnosticabilidade de uma linguagem é definida da seguinte forma.

Definição 4.2 *Seja L a linguagem gerada por um autômato G e suponha que L seja viva e prefixo-fechada. Então L será diagnosticável em relação a P_o e $\Sigma_f = \{\sigma_f\}$ se a condição for verificada:*

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

sendo a seguinte condição de diagnose D expressa por:

$$(\forall w \in P_o^{-1}[P_o(st)] \cap L)(\Sigma_f \in w).$$

Dependendo de como as informações sobre a evolução dinâmica do sistema são disponibilizadas, isto é, centralizada em um único sistema de aquisição ou distribuída como no caso de redes de comunicação, sistemas de manufaturas, e sistemas elétricos de potência, podem-se definir duas estruturas para a diagnose de falhas em SED: codiagnosticadores e diagnosticadores centralizados. Nos codiagnosticadores a leitura dos sensores não é centralizada, mas sim distribuída em diferentes módulos. Esta estrutura está fora do escopo deste trabalho. A diagnose de falhas baseada no diagnosticador centralizado, que utiliza um único diagnosticador que tem acesso a todos os eventos observáveis do sistema, será vista a seguir.

4.2 Diagnose de falhas baseada no autômato diagnosticador

Em alguns casos pode ser necessário determinar a ocorrência de um ou mais eventos não-observáveis, em um SED parcialmente observado. Essa hipótese é motivada pela possibilidade de tal SED possuir um evento que representa uma falha. Nos casos mais simples, a falha é observada por um sensor, e portanto, é um evento observável e sua ocorrência pode ser detectada imediatamente. Porém, em muitos casos, não existe um sensor capaz de perceber a ocorrência dessa falha, e portanto, tal falha deve ser modelada utilizando um evento não-observável em um autômato parcialmente observado, por exemplo. A teoria de diagnose de falhas de SEDs, permite verificar

se a ocorrência de uma falha em um SED pode ser detectada, ou não. Além disso, é possível construir um dispositivo capaz de informar a ocorrência (quando possível) de um determinado evento não-observável (falha). Esse dispositivo é denominado de diagnosticador.

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

$$G_d = (X_d, \Sigma_o, \xi_d, \Gamma_d, x_{0d}),$$

e pode ser construído em dois passos:

(i) obtenha a composição paralela $G_\ell = G||A_\ell$, sendo A_ℓ o autômato rotulador de dois estados mostrado na figura 4.1;

(ii) calcule $G_d = Obs(G_\ell)$.

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

Exemplo 4.1 *Para ilustrar a construção de diagnosticadores, considere o autômato G da figura 4.2. O autômato G_ℓ representado na figura 4.3 é formado a partir da composição paralela entre G e A_ℓ, i.e. $G_\ell = G||A_\ell$. Note que há uma divisão do estado 5 de G em $(5, N)$ e $(5, Y)$ em G_ℓ. Isso é devido à existência de duas sequências distintas, uma que contém a falha $\sigma_f ab$, e outra que não contém a falha, ba, que leva o sistema do estado inicial até o estado 5. O diagnosticador, mostrado na figura 4.4, é obtido a partir da construção do observador de G_ℓ, i.e., $G_d = Obs(G_\ell)$.*

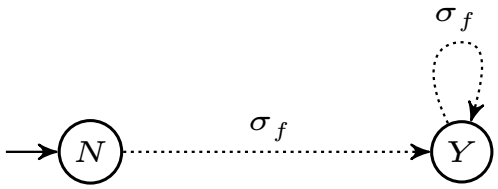

Figura 4.1: Autômato rotulador.

Considerando a construção de $G_d = Obs(G||A_\ell)$, é fácil notar que se o diagnosticador tiver certeza da ocorrência da falha, todos os estados seguintes permanecem indicando a falha. Contudo, é possível para um diagnosticador mudar de um estado de não-falha para duvidoso ou certo. Portanto, é possível classificar os estados do diagnosticador da seguinte forma.

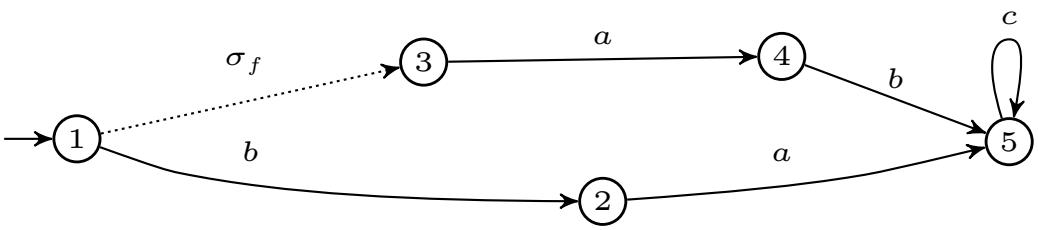

Figura 4.2: Autômato G do exemplo 4.1.

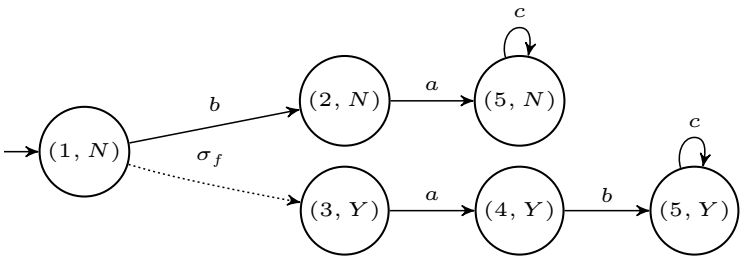

Figura 4.3: Autômato G_ℓ do exemplo 4.1.

Definição 4.3 *Um estado $x_d \in X_d$ é denominado certo (de falha), se $\ell = Y$ para todo $x_\ell \in x_d$, e normal (ou de não-falha) se $\ell = N$ para todo $x_\ell \in x_d$. Se existir (x, ℓ), $(y, \tilde{\ell}) \in x_d$, x não necessariamente distinto de y, tal que $\ell = Y$ e $\tilde{\ell} = N$, então x_d é um estado incerto de G_d.*

Utilizando as definições 4.2 e 4.3, é possível estabelecer as seguintes relações entre os estados do diagnosticador e as sequências da linguagem gerada por G.

Lema 4.1 *Seja $x_d = \xi_d(x_{0_d}, s)$.*

(i) Se x_d for um estado certo, então para todo $\omega \in [P_0^{-1}(s)] \cap L$, $\Sigma_f \in \omega$.

(ii) Se x_d for um estado incerto, então existirão sequências $s_1, s_2 \in L$ tais que $\Sigma_f \in s_1$ e $\Sigma_f \notin s_2$, porém $P_o(s_1) = P_o(s_2)$ e $\xi_d(x_{0_d}, P_o(s_1)) = \xi_d(x_{0_d}, P_o(s_2)) = x_d$.

Uma consequência imediata da definição 4.2 e do lema 4.1 é que a linguagem gerada por G será diagnosticável em relação a Σ_f e P_o se, e somente se, o diagnosticador sempre alcançar um estado certo para toda sequência arbitrariamente longa de L que contiver o evento σ_f. Isso não irá ocorrer se, e somente se, existir uma sequência de L que faça com que o diagnosticador fique preso indefinidamente em um laço formado por estados incertos. Para que resultados mais expressivos sobre esse problema possam ser enunciados, considere as seguintes definições.

Definição 4.4 *Seja $L(G, x_1) = \{v \in \Sigma^* : (\exists uv \in L)[\xi(x_0, u) = x_1 \wedge uv \in L]\}$. Um conjunto de estados $\{x_1, x_2, \ldots, x_n\} \subseteq X$ forma um ciclo em um autômato G, se existirem uma sequência $s = \sigma_1\sigma_2 \ldots \sigma_n \in L(G, x_1)$ tais que $\xi(x_\ell, \sigma_\ell) = x_{\ell+1}$, $\ell = 1, \ldots, n-1$ e $\xi(x_n, \sigma_n) = x_1$.*

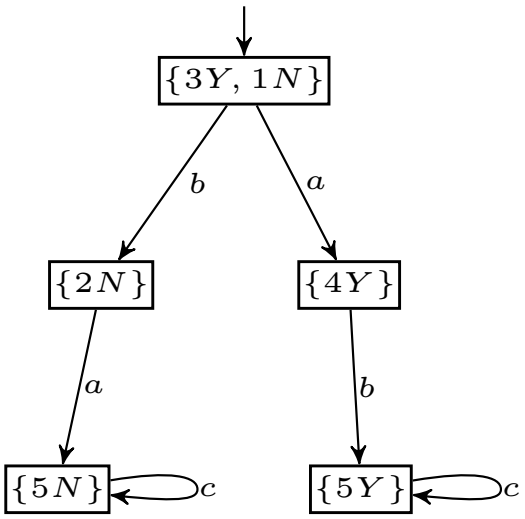

Figura 4.4: Autômato G_d do exemplo 4.1.

Definição 4.5 *Um conjunto de estados incertos* $\{x_{d_1}, x_{d_2}, \ldots, x_{d_p}\} \subset X_d$ *forma um ciclo indeterminado se as seguintes condições forem satisfeitas:*

1. $x_{d_1}, x_{d_2}, \ldots, x_{d_p}$ *forma um ciclo em* G_d
2. $\exists(x_\ell^{k_\ell}, Y), (\tilde{x}_\ell^{r_\ell}, N) \in x_{d_\ell}$, $x_\ell^{k_\ell}$ *não necessariamente distinto de* $\tilde{x}_\ell^{r_\ell}$, $\ell = 1, 2, \ldots, p$, $k_l = 1, 2, \ldots, m_\ell$, $r_\ell = 1, 2, \ldots, \tilde{m}_\ell$ *de tal sorte que as sequências de estados* $\{x_\ell^{k_\ell}\}$, $\ell = 1, 2, \ldots, p$, $k_\ell = 1, 2, \ldots, m_\ell$ *e* $\{\tilde{x}_\ell^{r_\ell}\}$, $\ell = 1, 2, \ldots, p$, $r_\ell = 1, 2, \ldots, \tilde{m}_\ell$ *podem ser rearranjadas para formar ciclos em* G*, cujas sequências correspondentes* s *e* $\tilde{s}$*, formados com os eventos que definem a evolução dos ciclos, têm como projeção* $\sigma_1\sigma_2 \ldots \sigma_p$*, em que* $\sigma 1, \sigma_2, \ldots \sigma_p$ *são definidos de acordo com a definição 4.4.*

Dentro das hipóteses para a diagnosticabilidade, a hipótese A2 era necessária para evitar que a ocorrência do evento não fosse detectada caso o sistema ficasse preso em um ciclo de estados ligados por eventos não observáveis após sua ocorrência. Essa hipótese será removida após a seguinte definição [14].

Definição 4.6 *(Ciclos escondidos e ciclos escondidos indeterminados). Seja* $x_d = \{x_1\ell_1, x_1\ell_1, \ldots, x_n\ell_n\}$ *um estado* G_d*. Então, existirá um ciclo escondido em* x_d *para algum* $\{i_1, i_2, \ldots, i_k\} \subseteq \{1, 2, \ldots, n\}$*, se as seguintes condições forem verdadeiras:*

C1. $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ *formam um ciclo em* G*;*

C2. $\sigma_{i_1}, \sigma_{i_2}, \ldots, \sigma_{i_k} \subseteq \Sigma_{uo}$*, em que* $\sigma_{i_1}, \sigma_{i_2}, \ldots, \sigma_{i_k}$ *tal que* $\xi(x_{i_j}, \sigma_{i_j}) = x_{i_{j+1}}, j = 1, 2, \ldots, k-1$*, e* $\xi(x_{i_k}, \sigma_{i_k}) = x_{i_1}$.

Se x_d *for um estado incerto de* G_d *e além das condições C1 e C2, a seguinte condição também for satisfeita,*

C3. $\ell_{i_j} = Y, j = 1, 2, \ldots, k,$

então x_d tem um ciclo escondido indeterminado.

A ideia por trás das definições de ciclos escondidos e ciclos escondidos indeterminado é a seguinte. Note que as hipóteses C1 e C2 garantem que $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ formam um ciclo de estados conectados com eventos não-observáveis. Seja uma sequência $s = s_o(\sigma_{i_1}, \sigma_{i_2}, \ldots, \sigma_{i_k})^n \in L$ ($n \in \mathbb{N}$). Sem perda de generalidade, considere que o último evento de s_0 seja observável. Suponha inicialmente que $\sigma_f \notin s$ e que não há uma sequência de falha s' tais que $P_o(s) = P_o(s')$. Nesse caso, existe um estado x_d^N tal que $\{x_{i_1}N, x_{i_2}N, \ldots, x_{i_k}N\} \subseteq x_d^N$. Considere que $\sigma_f \in s_o$ e $\xi(x_{0,\ell}, s_o) = x_\ell^Y$, tal que ξ_ℓ seja a função de transição de $G_\ell = G||A_\ell$ sendo $x_{0,\ell}$ o estado inicial x_ℓ^Y um estado certo de G_ℓ. Além disso, considere que não haja sequências normais s'' tal que $P_o(s) = P_o(s'')$. Então haverá um estado certo x_d^Y de G_d tal que $(x_\ell^Y \cup \{x_{i_1}Y, x_{i_2}Y, \ldots, x_{i_k}Y\} \subseteq x_d^Y)$. Por outro lado, se existir uma sequência normal s'' (de comprimento limitado ou não) tal que $\xi(x_{0,\ell}, s_o) = x_\ell^N$, em que x_ℓ^N é um estado normal de G_ℓ, e $P_o(s) = P_o(s'')$, então haverá um estado incerto x_d^{YN} em G_d tal que $(x_\ell^Y \cup \{x_{i_1}Y, x_{i_2}Y, \ldots, x_{i_k}Y\} \cup x_\ell^N \subseteq x_d^{YN})$. Consequentemente, de acordo com a definição 4.6, existem ciclos escondidos nos estados x_d^N e x_d^Y de G_d e um ciclo escondido indeterminado em x_d^{YN}. Note que na verificação da diagnoticabilidade, um estado x_d^Y garante que a falha ocorreu e um estado x_d^N que a falha não ocorreu. Sendo assim, a existência de ciclos escondidos em estados normais ou certos de G_d não afeta a diagnosticabilidade. Por outro lado, a existência de ciclos indeterminados escondidos implica que a linguagem não é diagnosticável, uma vez que existem duas sequências: uma sequência de falha s de comprimento ilimitado, e uma sequência normal, s'' de comprimento finito de tal forma que $P_o(s) = P_o(s'')$. É por isso que os ciclos escondidos, formados com estados de G que são rotulados por Y, em algum estado incerto de G_d são denominados ciclos escondidos indeterminados.

Os ciclos escondidos indeterminados são rotulados como *ihc* (do inglês indeterminate hidden cycle) e os ciclos escondidos em estados normais, em estados certos ou em estados incertos cujos ciclos escondidos não sejam formados por estados certos de G_d são rotulados simplesmente como *hc*.

Uma conseqüência imediata da definição 4.6 é que nenhuma das hipótes feitas em SAMPATH et al. [2] é necessária, devido às seguintes razões vistas em [14].

1. Como é mostrado na sequência, a condição de diagnosticabilidade pode ser expressa em termos de ciclos escondidos indeterminados, portanto, permite remover a hipótese A2.

2. Com a definição de ciclos escondidos e ciclos escondidos indeterminados, também é possível remover hipótese A1 da seguinte forma: se para algum

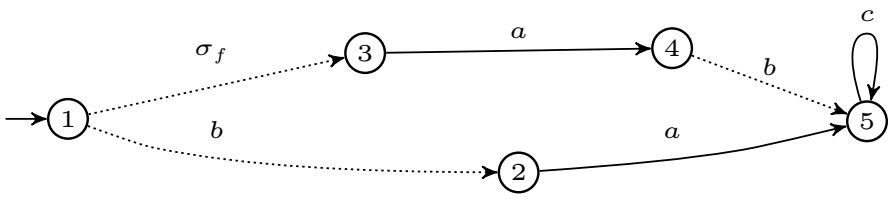

Figura 4.5: Autômato G do exemplo 4.2.

estado y de G, $\Gamma(y) = \emptyset$, então substitui-se G por um novo autômato $G' = (X', \Sigma', \xi', \Gamma', x_0)$, em que $\Sigma' = \Sigma \cup \{\sigma_u\}$, sendo σ_u um evento não-observável, $\xi'(x, .) = \xi(x, .)$, para todo $x \neq y$ e $\xi'(y, \sigma_u) = y$. Observe que as linguagens geradas por G e G' têm a mesma projeção P_o, o que não altera a diagnosticabilidade de L. A conseqüência desse procedimento é que um ciclo escondido rotulado com o evento σ_u será formado em algum estado de G_d.

Utilizando as definições 4.2, 4.4, 4.5, 4.6 e o lema 4.1, pode-se enunciar a seguinte condição necessária e suficiente para a diagnosticabilidade de uma linguagem.

Teorema 4.1 *[14] Uma linguagem L gerada por autômato G será diagnosticável em relação a uma projeção P_o e $\Sigma_f = \{\sigma_f\}$ se, e somente se, o seu diagnosticador G_d não tiver ciclos indeterminados (observados e escondidos).*

Exemplo 4.2

- *Como exemplo, considere o autômato da figura 4.2 e seu diagnosticador, mostrado na figura 4.4. Note que G_d não possui ciclos indeterminados. Assim, pode-se concluir que a linguagem L é diagnosticável com relação a P_o e Σ_f.*

- *Considere agora o autômato G cujo diagrama de transição de estados é mostrado na figura 4.5. Suponha que $\Sigma = \{a, b, c, \sigma_f\}$, $\Sigma_o = \{a, c\}$, $\Sigma_f = \{\sigma_f\}$. O diagnosticador de G pode ser visto na figura 4.6. Observe que o estado $\{5N, 5Y\}$ é um estado incerto e forma um ciclo em G_d, já que $\xi_d(\{5N, 5Y\}, c) = \{5N, 5Y\}$. Além disso, existe uma sequência $s = \sigma_f abc^n, n \in \mathbb{N}$ de falha e uma sequência $s' = bac^n, n \in \mathbb{N}$ que não contém a falha, que possuem a mesma projeção $P_o(s) = P_o(s') = ac^n, n \in \mathbb{N}$. Consequentemente s é uma sequência ambígua e, portanto, a linguagem L não é diagnosticável com relação a P_o e Σ_f.*

Observação 4.1 *A presença de ciclos em G_d formados somente por estados incertos não implica necessariamente que L não seja diagnosticável com relação a P_o e Σ_f. Para que a impossiblidade de se diagnosticar uma falha seja caracterizada, é necessário que G possua um ciclo de estados formado após a ocorrência da falha, que seja correspondente ao ciclo de estados incertos em G_d. Tal fato será ilustrado no exemplo a seguir.*

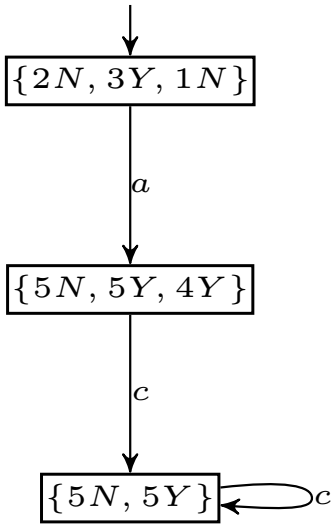

Figura 4.6: Diagnosticador centralizado de G do exemplo 4.2.

Exemplo 4.3 *[19] Considere um SED e o seu correspondente diagnosticador mostrados nas figuras 4.7 e 4.8, respectivamente, tal que $\Sigma_f = \{\sigma_f\}$ denota o evento a ser diagnosticado, sendo o único evento não-observável. Este diagnosticador tem um ciclo de estados incertos. No entanto, não se pode formar um ciclo no sistema de entradas que aparecem nos estados de incerteza no diagnosticador e têm o rótulo Y. O único ciclo do sistema que pode fazer com que o diagnosticador permaneça no seu ciclo de estados incertos é $7 \to 11 \to 12 \to 7$, e todos esses estados têm rótulo N no correspondente estado do diagnosticador. Sendo assim, o ciclo de estados incertos no diagnosticador não é indeterminado. Devido à ausência de ciclos indeterminados, pode-se dizer que a ocorrência do evento σ_f no sistema é sempre diagnosticável. Além disso, se o evento σ_f ocorrer, o diagnosticador deixará o ciclo de estados incertos e entrará no estado $\{6Y\}$ sob observação do evento t.*

Os exemplos anteriores não consideram a presença de ciclos escondidos em autômatos diagnosticadores. Isto será ilustrado a seguir.

Exemplo 4.4 *Para ilustrar o resultado do teorema 4.1 para ciclos escondidos, considere o autômato $G = (X, \Sigma, \xi, \Gamma, x_0, X_m)$ cujo diagrama de transição de estados está representado na figura 4.9. Suponha que $\Sigma = \{a, b, c, d, \sigma, \sigma_f\}, \Sigma_o = \{c, d\}, \Sigma_{uo} = \{a, b, \sigma, \sigma_f\}$ e $\Sigma_f = \{\sigma_f\}$. Considere a projeção $P_o : \Sigma^* \to \Sigma_o^*$ e Σ_f. O diagnosticador parcial G_d correspondente ao conjunto de eventos observáveis Σ_o está representado na figura 4.10, de onde se pode ver que G_d tem um ciclo escondido indeterminado no estado $\{3N, 4N, 6Y\}$. Como consequência, L é não diagnosticável em relação a P_o e Σ_f. A justificativa para não diagnosticabilidade de L com relação a P_o é a existência da sequência $s = a\sigma_f ca^n, n \in \mathbb{N}$, que contém o evento de falha σ_f e que possui a mesma projeção que uma sequência normal $s' = ac$, isto é,*

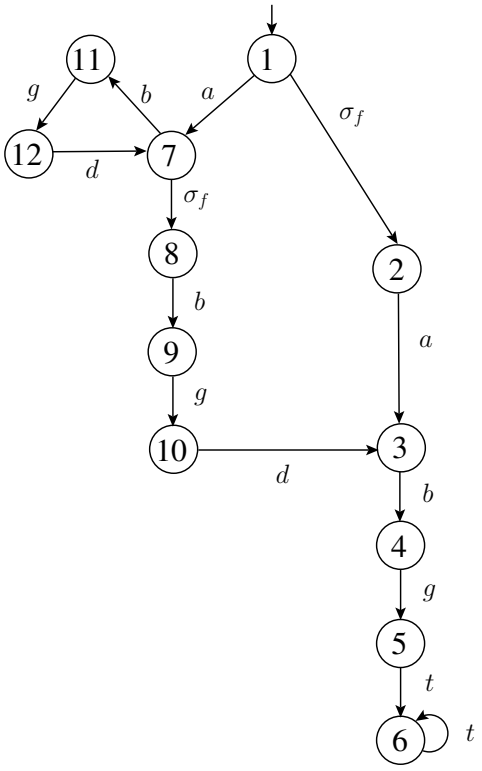

Figura 4.7: Autômato G do exemplo 4.3.

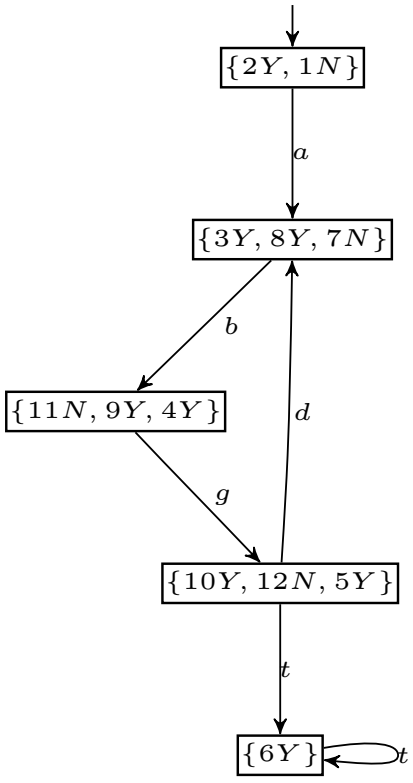

Figura 4.8: Diagnosticador centralizado de G do exemplo 4.3.

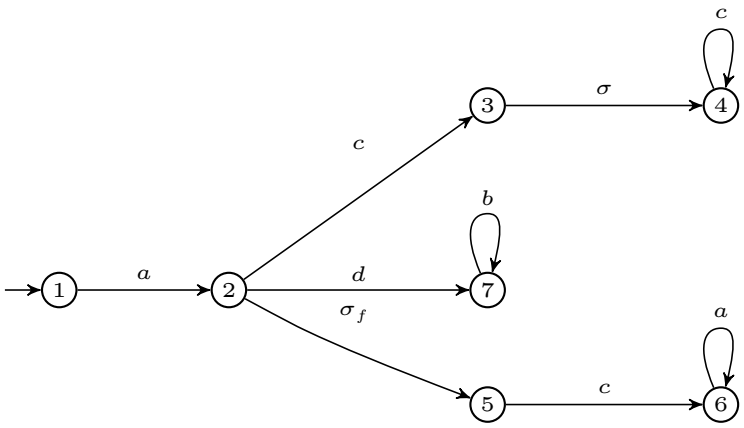

Figura 4.9: Autômato G do exemplo 4.4.

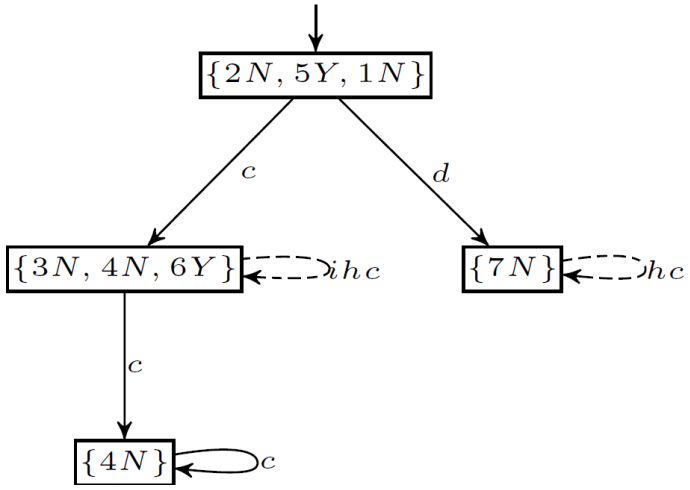

Figura 4.10: Diagnosticador G_d de G do exemplo 4.4 para $\Sigma_o = \{c, d\}$.

$P_o(s) = P_o(s') = c$; consequentemente s é uma sequência ambígua. Note que se o conjunto de eventos observáveis for $\Sigma_o^{'} = \{a, c, d\}$ L é diagnosticável em relação a $P_o^{'} : \Sigma^* \to \Sigma_o^{'}$ e Σ_f, com base no teorema 4.1, uma vez que o autômato $G_d^{'}$ representado na figura 4.11 não possui ciclos indeterminados (observados ou escondidos).

4.3 Uma nova abordagem para verificação da diagnosticabilidade usando diagnosticadores

4.3.1 Diagnosticabilidade de sistemas centralizados

O presente trabalho apresenta uma nova abordagem para a verificação da diagnosticabilidade de falhas de SEDs. Ao invés da verificação baseada no autômato, G_d, propõe-se a construção de um novo autômato G_t, denominado diagnosticador de teste. A primeira vantagem dessa abordagem em relação à apresentada na seção anterior é que de acordo com o teorema 4.1 é necessário procurar por ciclos inde-

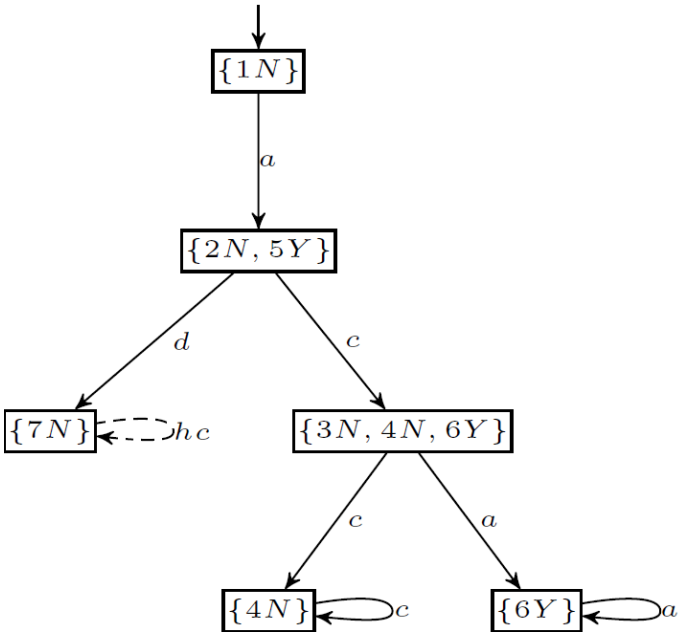

Figura 4.11: Diagnosticador G'_d de G do exemplo 4.4 para $\Sigma'_o = \{a, c, d\}$.

terminados em G_d. Porém, para tanto, segundo a definição 4.5, é preciso procurar um conjunto de estados incertos em G_d e verificar se esses estados levam a ciclos apropriados em G. Isto é, o autômato G_d não carrega informação suficiente para determinar se uma linguagem L gerada por um autômato G será diagnosticável em relação a uma projeção P_o e Σ_f.

Neste trabalho, iremos apresentar um novo autômato, G_t, que carrega toda informação necessária para a verificação da diagnosticabilidade. Uma outra vantagem será na diagnose aplicada a autômatos ponderados que será vista mais adiante.

O autômato G_t será construído da seguinte forma:

Algoritmo 4.1 *(Cálculo do autômato diagnosticador de teste G_t)*
Passo 1: Calcule $G_\ell = G||A_\ell$
Passo 2: Calcule $G_d = Obs(G_\ell)$;
Passo 3: $G_t = G_d||G_\ell$

A partir da definição de G_t, pode-se enunciar o seguinte resultado.

Lema 4.2 $L(G_t) = L(G_\ell) = L(G)$.

Prova Note que o autômato G_ℓ possui a mesma linguagem de G, uma vez que $L(G_\ell) = L(G||A_\ell) = L(G) \cap P_f^{-1}[L(A_\ell)]$, sendo $P_f : \Sigma^* \to \Sigma_f^*$, uma vez que o autômato rotulador A_ℓ possui apenas o evento de falha $\Sigma_{A_\ell} = \Sigma_f = \{\sigma_f\} \subseteq \Sigma_G$. Portanto a composição paralela tem apenas a função de rotular os estados de G, mas não altera a linguagem de G. O cálculo de G_d é obtido computando-se $Obs(G_\ell)$. De acordo com o que foi visto na seção 4.2, a linguagem de um observador é a projeção

da linguagem de G em um certo conjunto de eventos observáveis Σ_o. Portanto, $L(G_d) = L(Obs(G_\ell)) = P_o[L(G_\ell)] = P_o[L(G)]$. Considerando que o conjunto de eventos de G_d também é igual ao conjunto de eventos observáveis $\Sigma_o \subseteq \Sigma$, então a composição paralela entre G_d e G_ℓ gerará a linguagem de G. Portanto,

$$L(G_t) = L(G_d||G_\ell) = P_o^{-1}[L(G_d)] \cap L(G_\ell) = P_o^{-1}[P_o[L(G)]] \cap L(G) = L(G).$$

□

Exemplo 4.5 *Para ilustrar a construção do autômato G_t, considere o autômato G_ℓ e G_d das figuras 4.3 e 4.4, respectivamente. A composição paralela $G_t = G_d||G_\ell$ pode ser vista na figura 4.12. Note que os estados de G_t são da forma (x_d, x_ℓ). Sendo assim, é possível ocorrer as seguintes combinações:*

- *(Y, Y) que significa que G_d está certo de que a falha ocorreu e G_ℓ está mostrando que a falha ocorreu;*
- *(N, N) que significa que G_d está certo de que a falha não ocorreu e G_ℓ está mostrando que a falha não ocorreu;*
- *(YN, Y) que significa que G_d está em dúvida se a falha ocorreu ou não, porém G_ℓ está mostrando que a falha ocorreu;*
- *(YN, N) que significa que G_d está em dúvida se a falha ocorreu ou não, porém G_ℓ está mostrando que a não falha ocorreu;*

As combinações (N, Y) e (Y, N) são impossíveis, já que G_ℓ não pode mostrar que a falha ocorreu (resp. não ocorreu) e G_d afirmar o contrário, uma vez que G_d é obtido a partir de G_ℓ. Para esclarecer esse fato, o seguinte lema pode ser anunciado.

Lema 4.3 *Todo estado (x_d, x_ℓ) de G_t satisfaz à seguinte condição: $x_\ell \subseteq x_d$.*

Prova Como $G_d = Obs(G_\ell)$, então conforme dito na seção 2.3, o observador é construído com base no alcance não-observável dos estados em G_ℓ. Logo todos os estados x_ℓ de G_ℓ possuem ao menos um correspondente estado x_d em G_d. Uma vez que os eventos comuns de G_d e G_ℓ são Σ_o e os eventos privados de G_ℓ são Σ_{uo}, esses estados são consequentemente unidos em G_ℓ. Dessa forma $x_\ell \subseteq x_d$. □

Utilizando a definição de G_t, pode-se enunciar o seguinte teorema.

Teorema 4.2 *Uma linguagem L, gerada por um autômato G, será diagnosticável em relação à projeção P_o e $\Sigma_f = \{\sigma_f\}$ se, e somente se, não existirem componentes fortemente conexas formadas por estados (x_d, x_ℓ), tais que x_d seja incerto e x_ℓ seja certo.*

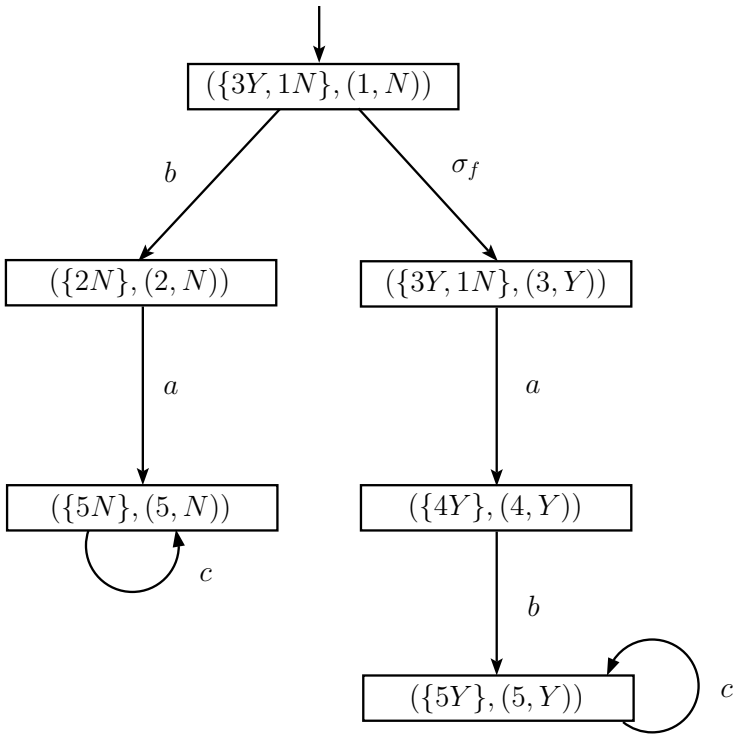

Figura 4.12: Autômato G_t correspondente a G da figura 4.2 .

Prova ($\Rightarrow$) Suponha que exista uma componente fortemente conexa formada por estados (x_{d_1}, x_{ℓ_1}), (x_{d_2}, x_{ℓ_2}), $\ldots$, (x_{d_n}, x_{ℓ_n}) sendo x_{d_i}, $i = 1, \ldots, n$, incerto e x_ℓ certo. De acordo com o lema 4.2, $L(G_t) = L(G_\ell) = L(G) = L$ e, portanto, $\exists st \in L : s \in \Sigma_f, |t| > n$. Surgem então duas possibilidades:

- $x_{d_1} = x_{d_2} = x_{d_3} = \ldots = x_{d_n} = x_d$. Isso significa que os estados (x_{d_1}, x_{ℓ_1}), (x_{d_2}, x_{ℓ_2}), $\ldots$, (x_{d_n}, x_{ℓ_n}) são conectados por eventos não-observáveis, uma vez que esses eventos são privados de G_ℓ. Além disso, devido ao lema 4.3, $x_{\ell_i} \in x_d$, $i = 1, 2, \ldots, n$ que junto com o fato de que x_{ℓ_i} são estados certos, implica que existe um ciclo escondido indeterminado em x_d.

- Existe $\{i_1, i_2, \ldots, i_p\} \subseteq \{1, \ldots, n\}$ tal que $x_{d_{i_k}} \neq x_{d_{i_l}}$, $k \neq l$, $k, l \in \{1, 2, \ldots, p\}$. Uma vez que x_{ℓ_i}, $i = 1, 2, \ldots, n$ são certos, e $L(G_t) = L$, então existe uma sequência arbitrariamente longa $s_Y = st \in L$ tal que $s \in \Psi(\Sigma_f)$ e $|t| \geq n$, para todo $n \in \mathbb{N}$. Além disso, uma vez que $x_{d_{i_k}}$ $i = 1, 2, \ldots, p$, são estados incertos, como provado em [2], existe uma sequência $s_N \in L$ tal que $P_o(s_Y) = P_o(s_N)$, logo L não é diagnosticável em relação a P_o e Σ_f.

($\Leftarrow$) Considere que L seja não diagnosticável em relação a P_o and Σ_f. Sendo assim, existem duas sequências: uma sequência ilimitada $s_Y = st$, $s \in \Psi(\Sigma_f)$, e $|t| > n$ para todo $n \in \mathbb{N}$ e uma sequência não necessariamente ilimitada s_N, tal que $\Sigma_f \notin s_N$ que satisfaz $P_o(s_Y) = P_o(s_N)$. Considere $|X_d||X_\ell| = q$ e $n > q$. Então

$\xi_t(x_{t_0}, s_Y) = (x_d, x_\ell)$, x_ℓ certo, e (x_d, x_ℓ) já existe em G_t, sendo assim, forma um ciclo, e, consequentemente, uma componente fortemente conexas em G_t. Considere agora que x_d é certo. Uma vez que, depois de entrar em um ciclo, um estado certo não pode se tornar incerto novamente, então qualquer sequência $s \in L$ tal que $P_o(s) = P_o(s_Y)$ será certa, o que contradiza hipótese de que existe s_N, $\Sigma_f \notin s_N$ tal que $P_o(s_Y) = P_o(s_N)$. Portanto, a componente x_d deve ser incerta para todos os estados na componente fortemente conexa. □

O resultado apresentado no teorema 4.2 será ilustrado pelo exemplo a seguir.

Exemplo 4.6

- *Considere o autômato G da figura 4.2. Os correspondentes autômatos G_d e G_ℓ estão representados nas figuras 4.4 e 4.3, respectivamente, e o autômato $G_t = G_d||G_\ell$ está mostrado na figura 4.12. A partir da figura 4.12, é possível concluir que a linguagem L gerada por G será diagnosticável em relação à projeção P_o e Σ_f pois não existem componentes fortemente conexas formadas por estados (x_d, x_ℓ) em que x_d é incerto e x_ℓ é certo.*

- *Considere, agora, o autômato G da figura 4.5 e seu diagnosticador G_d da figura 4.6. O autômato $G_t = G_d||G_\ell$ é representado na figura 4.13. Note que há uma componente fortemente conexa formada pelo estado $(\{5N, 5Y\}, (5, Y))$, e como essa componente é formada por um estado (x_d, x_ℓ) em que x_d incerto e x_ℓ certo, então a linguagem L gerada por G não será diagnosticável em relação à projeção P_o e Σ_f. Perceba que a conclusão da não diagnosticabilidade foi exatamente igual a do exemplo 4.2, porém não foi necessário procurar em G por sequências ambíguas, o que é uma vantagem.*

- *Finalmente considere o autômato da figura 4.9 e o diagnosticador G_d da figura 4.10. O autômato G_t correspondente é representado na figura 4.14. Note que o estado $(\{3N, 4N, 6Y\}, (6Y))$ forma um componente fortemente conexa na qual seu estado (x_d, x_ℓ) é tal que x_d é incerto e x_ℓ é certo. Logo a linguagem de G é não diagnosticável. Perceba que no estado $\{3N, 4N, 6Y\}$ do autômato diagnosticador da figura 4.9 há um ciclo escondido indeterminado. Esse ciclo escondido ocorre porque o evento a é não-observável. Pela construção de G_t esse ciclo escondido torna-se visível no estado $(\{3N, 4N, 6Y\}, (6Y))$,*

4.4 Diagnosticabilidade de falhas de SEDs modelados por autômatos ponderados

No projeto de um diagnosticador de falhas de SEDs, o primeiro passo é verificar se a linguagem gerada por um autômato G é diagnosticável ou não em relação a

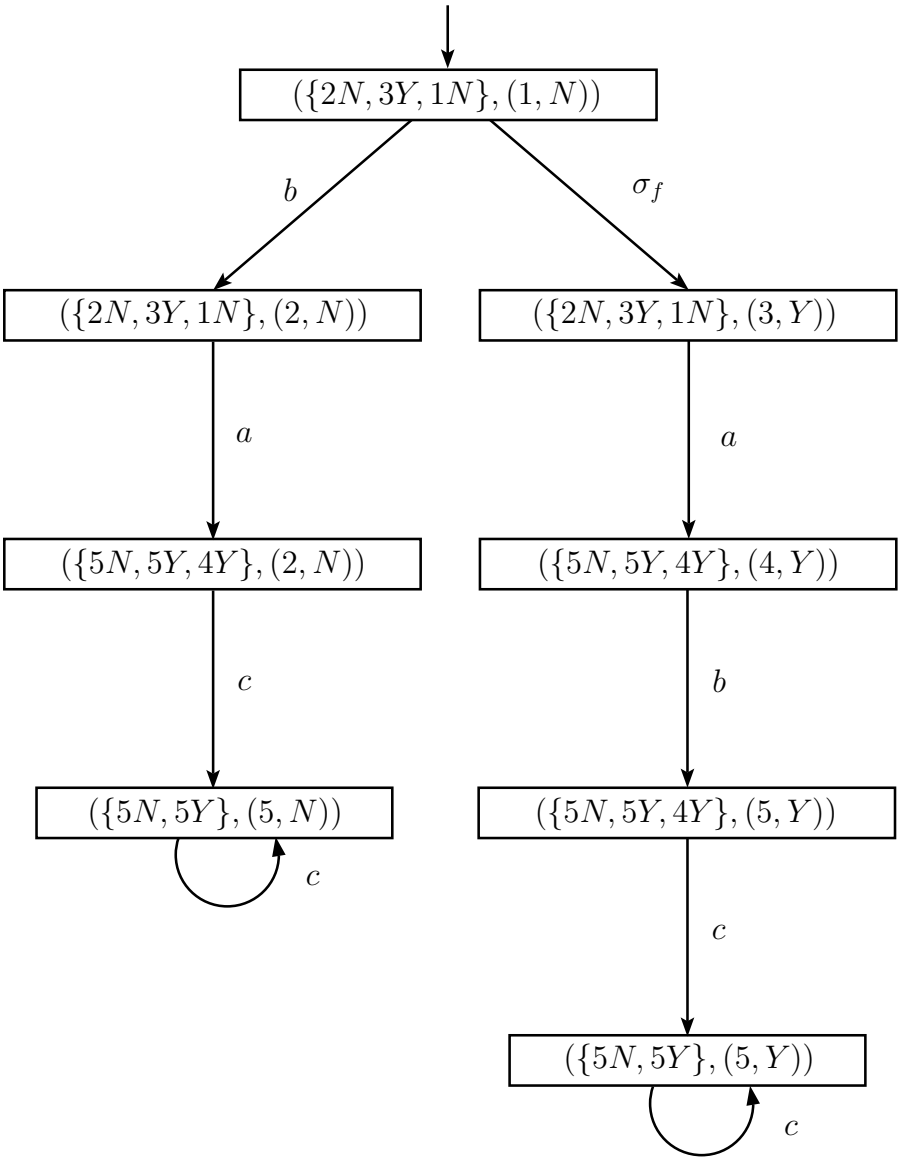

Figura 4.13: Autômato G_t correspondente a G da figura 4.5.

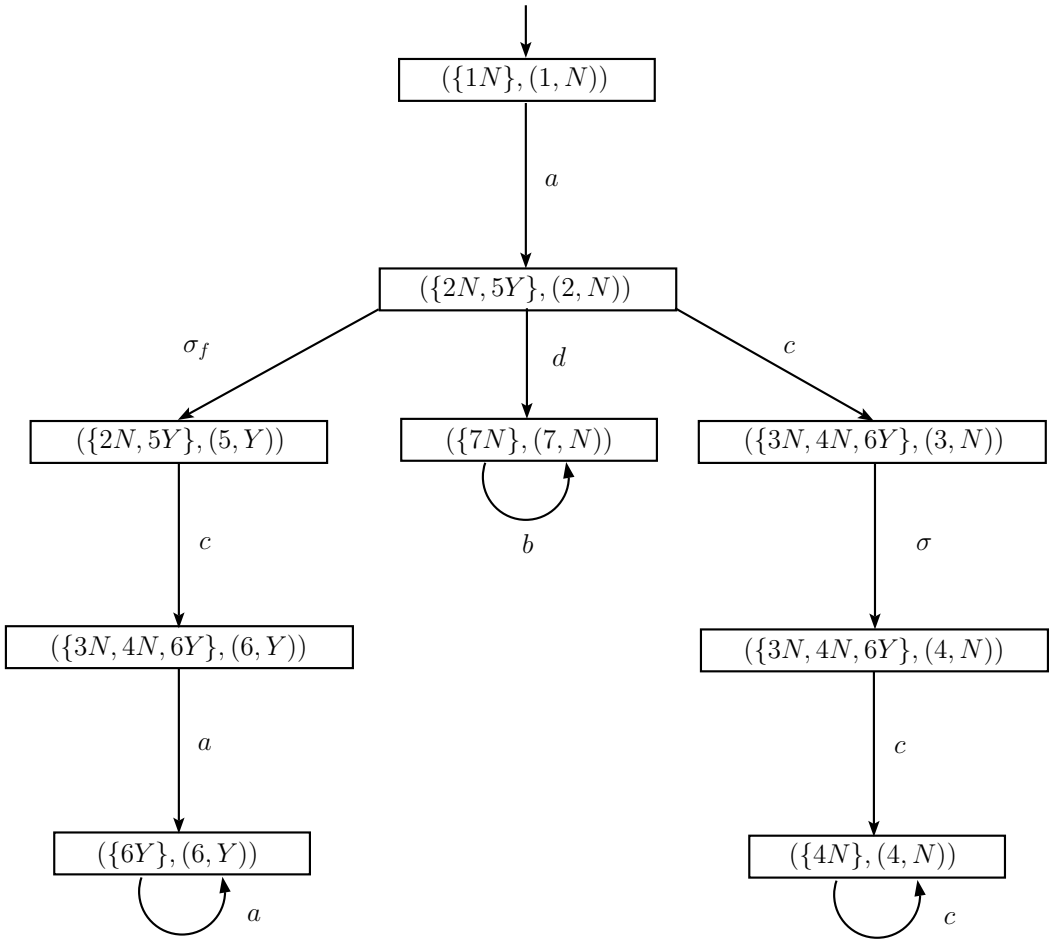

Figura 4.14: Diagnosticador G_t de G do exemplo 4.4 para $\Sigma_o = \{c, d\}$.

uma projeção P_o e Σ_f, isto é, saber se o sistema é capaz de diagnosticar a falha em um número finito de ocorrência de eventos. Porém, um aspecto pouco analisado e igualmente importante é, dado que uma linguagem seja diagnosticável, saber, após uma falha ocorrer, qual o tempo necessário para que o diagnosticador chegue num estado certo de falha (apenas rotulado com Y). Ou melhor ainda, qual o tempo máximo necessário para o sistema ter certeza de que a falha ocorreu.

Em um sistema real, uma falha pode comprometer toda uma linha de produção. Porém somente ter certeza de que a falha ocorreu não é o suficiente; por exemplo, componentes podem queimar ou peças desalinharem antes da ocorrência da falha ser detectada. Por isso, é importante analisar o tempo de detecção como um dos parâmetros para qualificar a diagnosticabilidade; eventualmente, baseado nesse parâmetro, será possível definir um coeficiente de segurança. Assim sendo, simplesmente modelar um SED como um autômato de estados finitos não terá essa informação. Tal informação pode ser obtida acrescentando às transições, ponderação de tempo. Para calcular o tempo máximo para diagnosticar a falha é necessário computar novos autômatos, chamados de autômatos de falha G_{f_i}, $i = 1, 2, \ldots, m, m \in \mathbb{N}$. Seja G_p um autômato ponderado que gera uma linguagem diagnosticável em relação a uma projeção P_o e Σ_f. Sejam G_{dp} e $G_{\ell p}$, tais que os autômatos diagnosticador e rotulador ponderado associados a G_p são dados por: $G_{dp} = (G_d, f_d)$ e $G_{\ell p} = (G_\ell, f_\ell)$, sendo f_p e f_ℓ as funções de ponderação associadas aos autômatos G_d e G_ℓ, respectivamente. Note que G_{dp} e $G_{\ell p}$ são computados da mesma forma que G_d e G_ℓ, conforme descrito na seção 4.2, porém seguindo as regras de composição paralela de autômatos ponderados descritas na seção 3.3 e a regra de construção do observador de estados descrita na seção 2.3.

Os autômatos de falha G_{f_i} podem ser calculados de acordo com o seguinte algoritmo.

Algoritmo 4.2 *Cálculo dos autômatos de falha* G_{f_i}

- Entradas*:*
  - *Autômatos* G_p, G_{dp} *e* $G_{\ell p}$
- Saída*:*
  - *Autômatos* G_{f_i}, $i = 1, 2, \ldots, m, m \in \mathbb{N}$
- *Passo 1: Compute* $G^m_{\ell p}$ *de* $G_{\ell p}$ *marcando todos os estados de* $G_{\ell p}$ *que possuam apenas rótulos* Y*.*
- *Passo 2: Compute* G^m_{dp} *de* G_{dp} *marcando todos os estados certos de* G_{dp}*.*
- *Passo 3: Compute* $G^m_{tp} = G^m_{dp} || G^m_{\ell p}$*.*

- *Passo 4: Apague todas as transições de "saída" dos estados marcados x_m em G^m_{tp}, isto é, faça $\Gamma(x_m) = \emptyset$, $\forall x_m \in X^m_{tp,m}$ e calcule $G^m_{tp,trim} = trim(G^m_{tp})$*

- *Passo 5: procurar em $G^m_{tp,trim}$ por estados x_i tal que $\sigma_f \in \Gamma(x_i)$. Suponha que existam m estados que satisfaçam tal condição. Para cada $i = 1, 2, \ldots, m$:*

  - *Passo 5.1: Construa novos autômatos: $G_{tp,i} = (X^i_{tp}, \Sigma^i_{tp}, \xi^i_{tp}, \Gamma^i_{tp}, x^i_{tp,0}, X^i_{m,tp}), i = 1, 2, \ldots, m$. Em que $X^i_{tp} = X^m_{tp,trim}$, $\Sigma^i_{tp} = \Sigma^m_{tp,trim}$, $\xi^i_{tp} = \xi^m_{tp,trim}$, $\Gamma^i_{tp} = \Gamma^m_{tp,trim}$, $x^i_{tp,0} = \xi^m_{tp}(x_i, \sigma_f)$ e $X^i_{m,tp} = X^m_{tp,trim}$. Para cada $G_{tp,i}$ calcule um novo autômato:*
  - *Passo 5.2:*

$$G_{f_i} = trim(G_{tp,i}) \tag{4.1}$$

Teorema 4.3 *Seja A_i a matriz max-plus associada a G_{f_i}, $i = 1, 2, \ldots, m$. Então:*

(a) O tempo t_{f_i} que o autômato G_{f_i} leva para diagnosticar a falha é igual ao elemento de A^+_i, a_{ij} tal que $i = x_{m_{f_i}}$ e $j = x_{0_{f_i}}$ somado à ponderação associada à falha t_{s_i}, em que $x_{0_{f_i}}$ é o estado inicial e $x_{m_{f_i}}$ é um estado marcado de G_{f_i}. Se houver n estados marcados, $t_{f_i} = (a_{ij_1} \oplus a_{ij_2} \oplus \ldots \oplus a_{ij_n}) \otimes t_{s_i}$.

(b) O tempo máximo t_f para diagnosticar a falha $t_f = \bigoplus_{i=1}^m t_{f_i}$.

Prova O teorema 4.3 é um resultado direto do teorema 4.1. As matrizes A^+_i, $i = 1, 2, \ldots, m$, possuem elementos a_{ij} que tem o valor do peso máximo do caminho do estado j até o estado i. Se houver mais de um estado marcado em G_{f_i}, o valor de t_{f_i} é o máximo entre cada peso referente ao caminho máximo do estado inicial até cada estado marcado de G_{f_i} somado com a ponderação da falha. A ponderação da falha é levada em conta pois não se sabe ao certo em quanto tempo dentro da estimativa a falha ocorrerá. De posse do valor de t_{f_i} para cada G_{f_i}, o valor do tempo máximo t_f para diagnosticar a falha será o máximo dentre os valores de t_{f_i}, $i = 1, 2, \ldots, m$. □

Exemplo 4.7 *Para ilustrar o procedimento descrito anteriormente, considere o autômato ponderado cujo diagrama de transições é mostrado na figura 4.15. Primeiramente calcula-se $G_{\ell p}$ e marcam-se os estados rotulados com Y, conforme mostrado na figura 4.16. Após isso, calcula-se G_{dp} e marcam-se os estados rotulados com Y, conforme mostrado na figura 4.17. O passo seguinte é obter o autômato G^m_{tp}, representado na figura 4.18, resultado da composição paralela entre G^m_{dp} e $G^m_{\ell p}$. Note que a linguagem é diagnosticável em relação a P_o e Σ_f pois G^m_{tp} não possui componente fortemente conexa formada por estados (x_d, x_ℓ) em que x_d incerto e x_ℓ certo. A falha σ_f aparece no conjunto de eventos ativos de dois estados em G_{tp}, $(\{2Y, 1N\}, (1N))$ e $(\{3Y, 8Y, 7N\}, (7N))$. Note que há uma componente fortemente conexa em G^m_{tp} formada pelos estados $(\{3Y, 8Y, 7N\}, (7, N))$,*

$(\{10Y, 12N, 5Y\}, (12, N))$ *e* $(\{11N, 9Y, 4Y\}, (11, N))$. *No entanto, tal fato não viola a diagnosticabilidade já que seus estados* x_ℓ *são normais* (N). *Porém, isso colocaria em sério risco o cálculo do tempo máximo para diagnosticar a falha caso os estados* $(\{3Y, 8Y, 7N\}, (7, N))$, $(\{10Y, 12N, 5Y\}, (12, N))$ *e* $(\{11N, 9Y, 4Y\}, (11, N))$ *tivessem sido escolhidos como os estados iniciais de* G_{f_i}, *uma vez que esses estados formam uma componente fortemente conexa. Assim, para se calcular* G_{f_1} *e* G_{f_2}, *primeiramente, remove-se o auto-laço no estado marcado* $(\{6Y\}, (6Y))$ *de* G^m_{tp}. *O estado inicial de* G^m_{tp} *é trocado para o estado* $(\{3Y, 8Y, 7N\}, (8Y))$ *e efetua-se a operação trim para se obter* G_{f_1}, *mostrado na figura 4.19. A matriz max-plus correspondente a* G_{f_1}, *assim como sua matriz max-plus de peso máximo,* A_1 *e* A_1^+, *respectivamente, são dadas por:*

$$A_1 = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 3 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 4 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & 3 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & 1 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 2 & \varepsilon & 4 & 2 & \varepsilon \end{pmatrix} \quad e \quad A_1^+ = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 3 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 4 & 1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 8 & 5 & 4 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 11 & 8 & 7 & 3 & \varepsilon & \varepsilon & \varepsilon \\ 12 & 9 & 8 & 4 & 1 & \varepsilon & \varepsilon \\ 15 & 12 & 11 & 7 & 4 & 2 & \varepsilon \end{pmatrix}.$$

Os índices da matriz A_1 *que correspondem a cada estado de* G_{f_1} *são mostrados na tabela 4.1. Note que o elemento* $a_{71} = 15$ *corresponde ao peso máximo indo do estado inicial* $(\{3Y, 8Y, 7N\}, (8Y))$ *até o estado marcado* $(\{6Y\}, (6Y))$. *Considerando que o peso da falha* $t_s = 1$, *então* $t_{f_1} = 15 + 1 = 16$. *De modo semelhante a* G_{f_1} *obtem-se* G_{f_2}, *representado na figura 4.20. As correspondentes matrizes são dadas por:*

$$A_2 = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 9 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 3 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 1 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 4 & 2 & \varepsilon \end{pmatrix} \quad e \quad A_2^+ = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 9 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 12 & 3 & \varepsilon & \varepsilon & \varepsilon \\ 13 & 4 & 1 & \varepsilon & \varepsilon \\ 16 & 7 & 4 & 2 & \varepsilon \end{pmatrix}.$$

Os índices da matriz A_2 *que correspondem a cada estado de* G_{f_2} *são mostrados na tabela 4.2. Note que o elemento* $a_{51} = 16$ *corresponde ao peso máximo indo do estado inicial* $(\{2Y, 1N\}, (2Y))$ *até o estado marcado* $(\{6Y\}, (6Y))$. *Considerando que o peso da falha* $t_s = 1$, *então* $t_{f_1} = 16 + 1 = 17$. *Portanto, o tempo máximo para diagnose* $t_f = t_{f_1} \oplus t_{f_2} = max(t_{f_1}, t_{f_2}) = 17$.

Note que para o exemplo 4.7, os autômatos G_{f_1} e G_{f_2} não possuem componentes fortemente conexas. Isso de fato ocorre para todo G_{f_i}, $i = 1, 2, \ldots, m$, conforme mostrado a seguir.

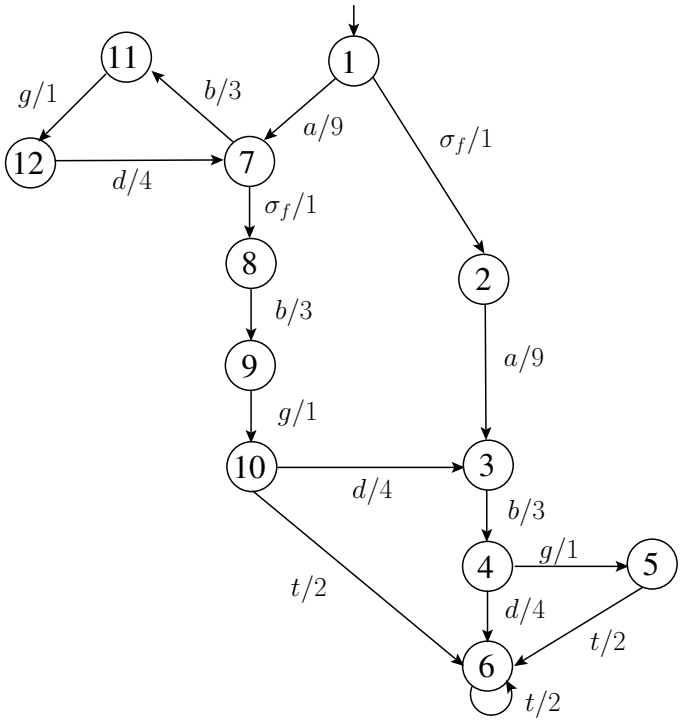

Figura 4.15: Autômato ponderado G_p para o exemplo 4.7.

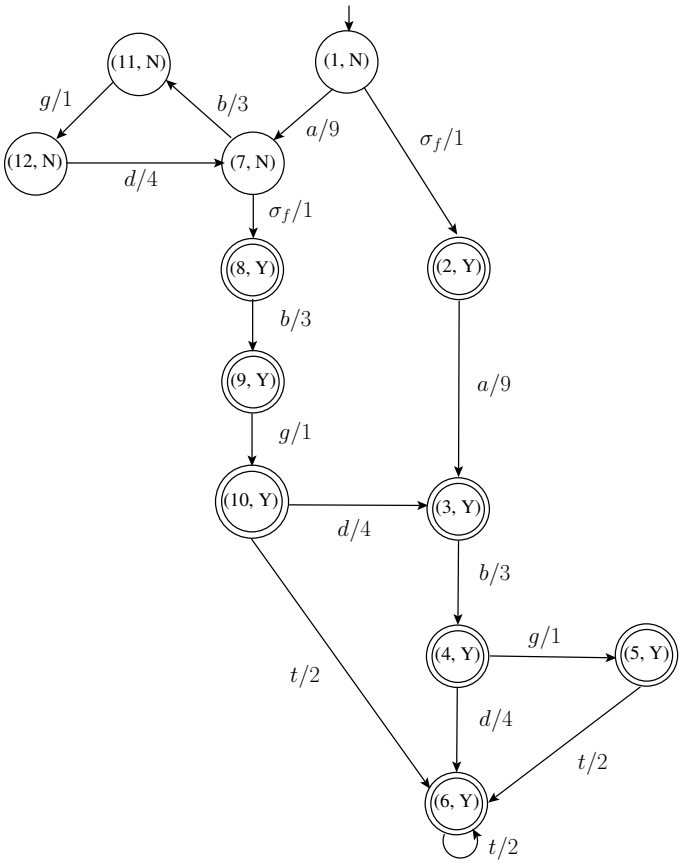

Figura 4.16: Autômato ponderado $G_{\ell p}^{m}$ para o exemplo 4.7.

Tabela 4.1: Relação entre os estados de G_{f_1} e os índices da matriz max-plus A_1 correspondente.

Estado	Índice da matriz A_1
$(\{3Y, 8Y, 7N\}, (8, Y))$	1
$(\{11N, 9Y, 4Y\}, (9, Y))$	2
$(\{10Y, 12N, 5Y\}, (10, Y))$	3
$(\{3Y, 8Y, 7N\}, (3, Y))$	4
$(\{11N, 9Y, 4Y\}, (4, Y))$	5
$(\{10Y, 12N, 5Y\}, (5, Y))$	6
$(\{6Y\}, (6, Y))$	7

Tabela 4.2: Relação entre os estados de G_{f_2} e os índices da matriz max-plus A_2 correspondente.

Estado	Índice da matriz A_2
$(\{2Y, 1N\}, (2, Y))$	1
$(\{3Y, 8Y, 7N\}, (3, Y))$	2
$(\{11N, 9Y, 4Y\}, (4, Y))$	3
$(\{10Y, 12N, 5Y\}, (5, Y))$	4
$(\{6Y\}, (6, Y))$	5

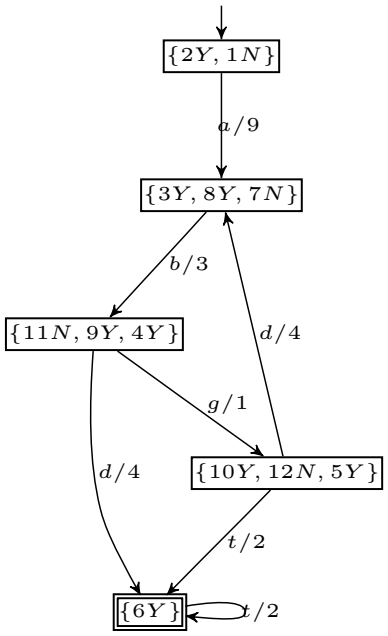

Figura 4.17: Autômato ponderado G_{dp}^{m} para o exemplo 4.7.

Teorema 4.4 *Suponha que a linguagem gerada por um autômato ponderado G_p seja diagnosticável em relação à projeção P_o e $\Sigma_f = \{\sigma_f\}$. Então os autômatos $G_{f_i}, i = \{1, 2, \ldots, m\}$, formados no algoritmo 4.2, não possuem componentes fortemente conexas.*

Prova Como a linguagem gerada por G_p é diagnosticável em relação à projeção P_o e $\Sigma_f = \{\sigma_f\}$, de acordo com o teorema 4.2 o autômato G_{tp} não possui componentes fortemente conexas (x_d, x_ℓ), tais que x_d incerto e x_ℓ certo. Contudo deve se considerar a possibilidade de haver componentes fortemente conexas formadas por estados dos seguintes tipos:

1. x_d certo e x_ℓ certo
2. x_d incerto e x_ℓ normal
3. x_d normal e x_ℓ normal

Uma componente fortemente conexa formada por x_d certo e x_ℓ certo não é possível, pois pela construção de $G_{f_i}, i = \{1, 2, \ldots, m\}$ todas transições saindo de estados marcados são removidas. Além disso, também por construção, como o estado inicial de G_{f_i} é sempre um estado pós-falha, então todos os estados de G_{f_i} são do tipo (x_d, x_ℓ), em que x_ℓ certo. Sendo assim as possibilidades 2 e 3 também são excluídas. Portanto os autômatos $G_{f_i}, i = \{1, 2, \ldots, m\}$ não possuem componentes fortemente conexas. □

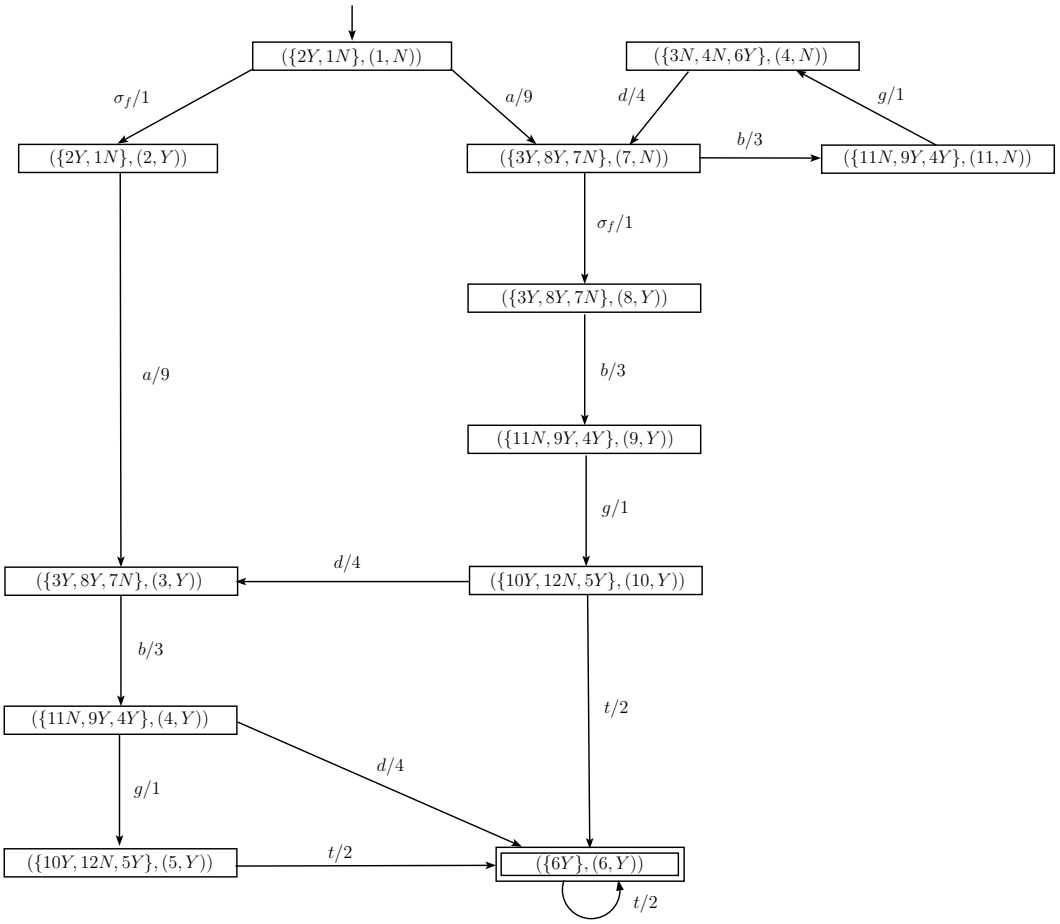

Figura 4.18: Autômato ponderado G_{tp}^m para o exemplo 4.7.

O teorema 4.4 o motivo pelo qual o estado inicial de G_{f_i} foi escolhido como sendo um estado $x_f = \xi_{tp}(x_i, \sigma_f)$, e não x_i tal que $\sigma_f \in \Gamma(x_i)$, uma vez que antes da falha é possível que G_{tp}^m possua uma componente fortemente conexa mas nunca após a falha. A ausência de componentes fortemente conexas em G_{f_i} para uma linguagem diagnosticável de G_p em relação a projeção P_o e Σ_f é crucial para convergência da matriz de pesos máximos A_i^+, conforme será visto no corolário a seguir.

Corolário 4.1 *Suponha que uma linguagem gerada por um autômato G_p seja diagnosticável em relação à projeção P_o e $\Sigma_f = \{\sigma_f\}$. Então a matriz A_i^+ associada ao autômato $G_{f_i}, i = \{1, 2, \ldots, m\}$ sempre existe.*

Prova Conforme visto na seção 3.3, a matriz A_i^+ não converge se o autômato ponderado associado à matriz A possui ciclos. Pelo teorema 4.4, se a linguagem gerada por um autômato G_p for diagnosticável em relação à projeção P_o e $\Sigma_f = \{\sigma_f\}$, então os autômatos $G_{f_i}, i = \{1, 2, \ldots, m\}$ não possuirão componentes fortemente conexas, e portanto, A_i^+ sempre convergirá. □

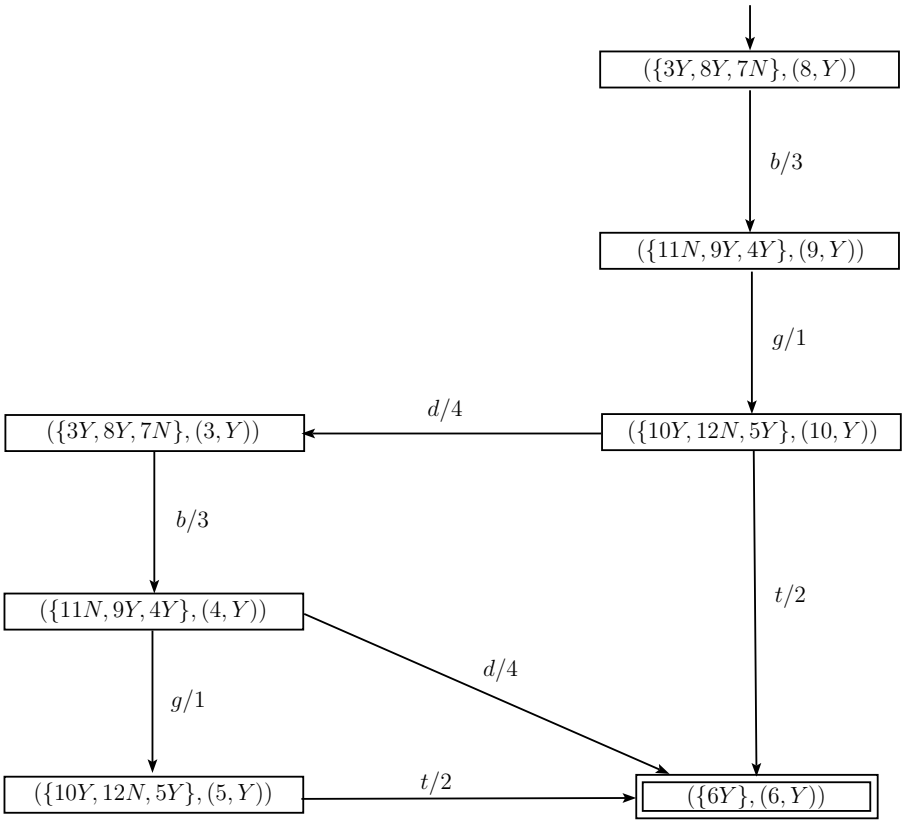

Figura 4.19: Autômato ponderado G_{f_1} do exemplo 4.7.

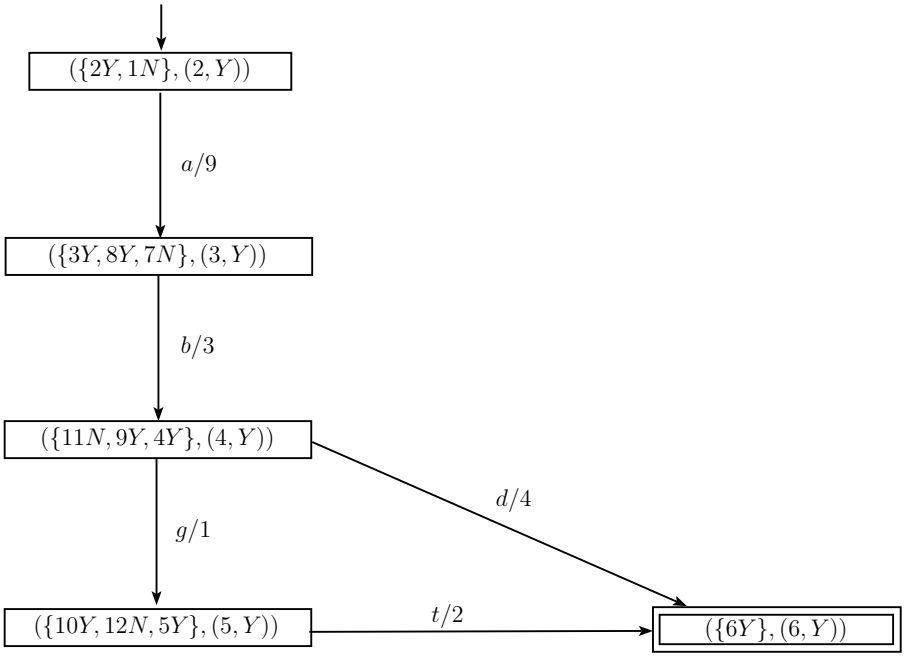

Figura 4.20: Autômato ponderado G_{f_2} do exemplo 4.7.

Observação 4.2 *O corolário 4.1 é uma das razões para se utilizar G_{tp} ($G_{f_i}, i = \{1, 2, \ldots, m\}$ mais precisamente) e não apenas G_{dp} da tradicional diagnose centralizada para a análise de diagnose de falhas em autômatos. Primeiramente, $L(G_d) \neq L(G)$. Então a linguagem de G_{dp} não representa a linguagem gerada pelo autômato G_p. E, mesmo se a linguagem for diagnosticável, uma matriz max-plus associada ao autômato G_{dp} poderia não convergir se G_{dp} possuisse ciclos, como por exemplo na figura 4.17, diferentemente de $G_{f_i}, i = \{1, 2, \ldots, m\}$.*

Exemplo 4.8 *Para ilustrar os resultados do teorema 4.4, considere o autômato ponderado G_p da figura 4.21, cujos autômatos G^m_{dp} e $G^m_{\ell p}$ estão mostrados nas figuras 4.22 e 4.23, respectivamente e a composição paralela $G^m_{tp} = G^m_{dp}||G^m_{\ell p}$ está representada na figura 4.24. Note que, como G^m_{tp} não possui componentes fortemente conexas por estados (x_d, x_ℓ), x_d incerto e x_ℓ certo, então a linguagem gerada por G_p é diagnosticável em relação à projeção P_o e $\Sigma_f = \{\sigma_f\}$. Note que as transições $[(\{5Y\}, (5Y)), d, (\{3Y\}, (3Y))]$,, $[(\{3Y\}, (3Y)), a, (\{5Y\}, (5Y))]$, $[(\{3Y\}, (3Y)), c, (\{4Y\}, (4Y))]$ e $[(\{4Y\}, (4Y)), d, (\{5Y\}, (5Y))]$, serão removidas de G^m_{tp} para a construção de G_{f_1}, representado na figura 4.25, uma vez que G^m_{tp} possui uma componente fortemente conexa formada justamente pelos estados marcados (estados certos, rotulados apenas por Y). Note que o estado $(\{3Y\}, (3Y))$ não será mais alcançável e não aparecerá em G_f por causa da operação trim. A matriz max-plus A e a matriz de pesos máximos A^+ referente a G_{f_1} apenas confirmam o que se nota visualmente na figura 4.25. Considerando o peso da falha $t_s = 2$, então $t_f = 2 + 3 = 5$. A relação entre os estados G_{f_1} e os índices da matriz A é mostrada na tabela 4.3.*

$$A = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon \\ 3 & \varepsilon & \varepsilon \\ 1 & \varepsilon & \varepsilon \end{pmatrix} \quad A^+ = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon \\ 3 & \varepsilon & \varepsilon \\ 1 & \varepsilon & \varepsilon \end{pmatrix}$$

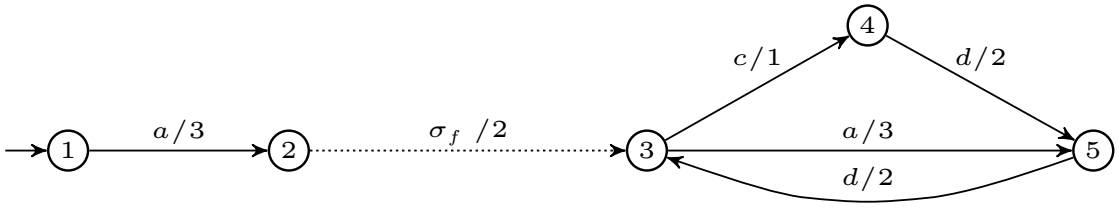

Figura 4.21: Autômato ponderado G_p do exemplo 4.8.

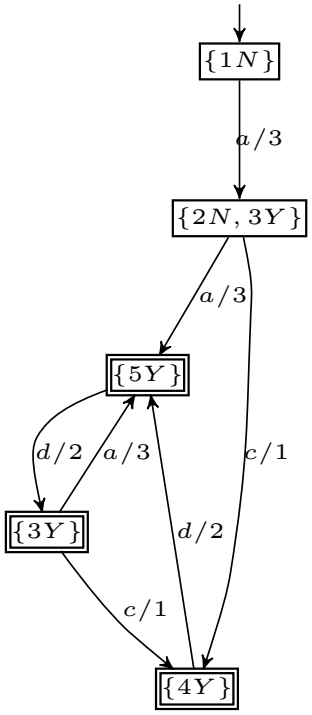

Figura 4.22: Autômato ponderado G^m_{dp} do exemplo 4.8.

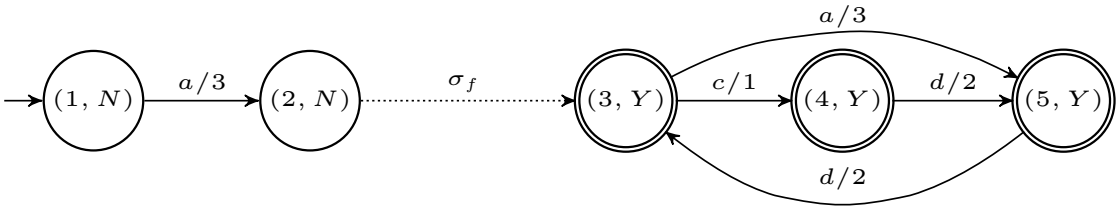

Figura 4.23: Autômato ponderado $G^m_{\ell p}$ do exemplo 4.8.

4.5 Análise da diagnosticabilidade em um sistema de manufatura

Para ilustrar os resultados apresentados neste trabalho, será considerado um sistema composto por uma linha de montagem de peças em que cada peça é formada por uma base metálica prensada a uma tampa plástica. O sistema é formado por três esteiras, sendo uma de entrada, E_{in}, uma de saída, E_{out}, e a terceira de descarte de bases metálicas defeituosas, E_d. As peças de metal são levadas por um braço robótico R para a esteira de entrada. No início da esteira de entrada, um conjunto de sensores verifica a altura da peça. Se a altura da peça estiver correta, ela é levada para a esteira de saída, contudo se a peça for maior que o desejado, ela é empurrada por um atuador pneumático, *atuador*1, para a esteira de descarte de peças defeituosas. Após as peças serem montadas, um atuador pneumático, *atuador*2, empurra a peça para fora da esteira de saída até um depósito. O evento de falha considerado é

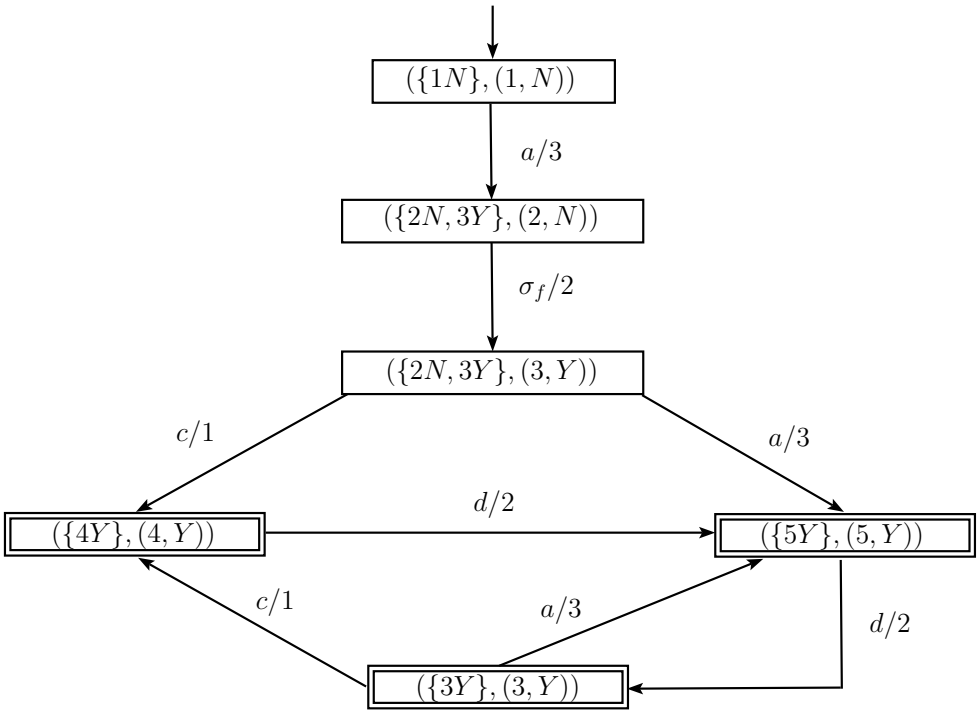

Figura 4.24: Autômato ponderado G^m_{tp} do exemplo 4.8.

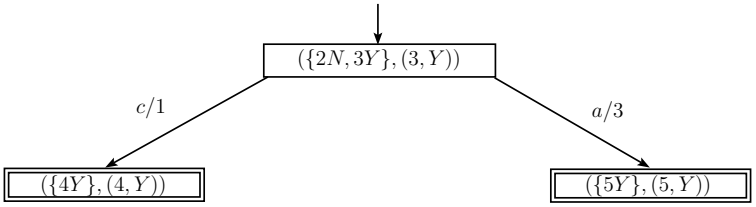

Figura 4.25: Autômato ponderado G_{f_1} do exemplo 4.8.

um defeito no atuador que leva a peça para a esteira de descarte. Esse sistema é ilustrado na figura 4.26.

Duas configurações serão consideradas para esse sistema, incluindo o controle, de forma que em uma configuração a falha é diagnosticável e na outra não. Dessa maneira, serão propostos dois modelos diferentes para o sistema. Em seguida, a diagnosticabilidade de cada modelo será examinada.

De maneira geral, o sistema foi modelado como mostra a figura 4.27. A descrição dos eventos e seus respectivos pesos podem ser acompanhados a partir da tabela 4.5. O sistema considerado não funciona com base no tempo. Os pesos (informação temporal) foram inseridos para permitir o cálculo do tempo máximo para diagnoticar a falha. O modelo do sistema pode ser descrito da seguinte forma: todas as esteiras estão ligadas em todos os momentos, o sistema inicia no estado 0, e então, o braço robótico apanha uma peça e a coloca na esteira de entrada, E_{in}, modelado pelo evento p. Em seguida a peça passa pelo sensor de altura e é identificada como uma peça normal, evento a, ou como uma peça defeituosa, evento a_d.

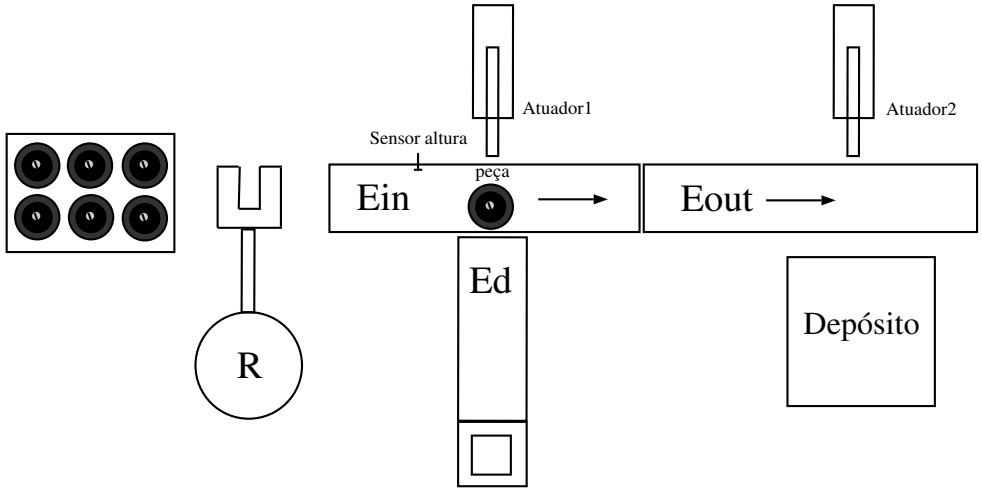

Figura 4.26: Linha de montagem.

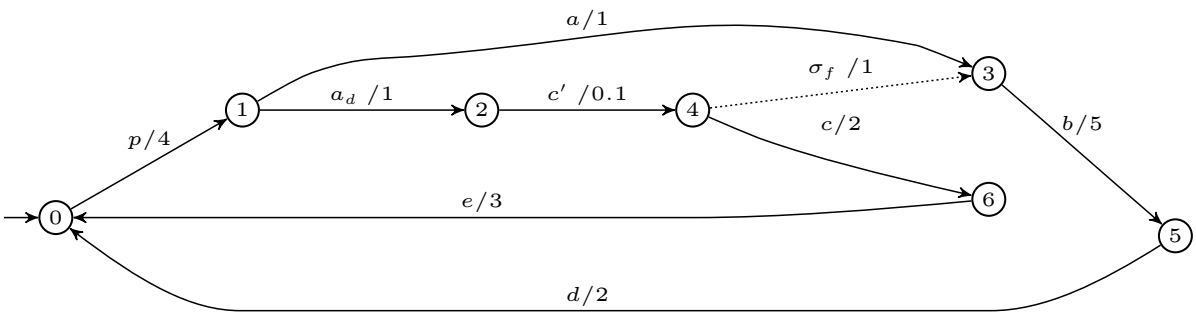

Figura 4.27: Modelo em autômato do comportamento controlado da planta.

Tabela 4.3: Relação entre os estados de G_{f_1} do exemplo 4.8 e os índices da matriz max-plus A correspondente.

Estado	Índice da matriz A
$(\{2N, 3Y\}, (3, Y))$	1
$(\{5Y\}, (5, Y))$	2
$(\{4Y\}, (4, Y))$	3

Tabela 4.4: Descrição dos eventos do autômato que modela a planta controlada.

Evento	Peso	Descrição
c	2	$atuador1$ empurra a peça defeituosa
a	1	peça normal
a_d	1	peça defeituosa
p	4	chegada de peça
b	5	peça é transferida de E_{in} para E_{out}
σ_f	1	$atuador1$ falha
c'	0.1	comando para $atuador1$
e	3	peça defeituosa é descartada
d	2	peça deixa o sistema pela esteira de saída

Serão considerados dois conjuntos de eventos não-observáveis: $E_{uo,1} = \{b, c, d, e, \sigma_f\}$ para a configuração não diagnosticável e o conjunto de eventos não observáveis $E_{uo,2} = \{b, c, e, \sigma_f\}$ configuração diagnosticável. A configuração não diagnosticável será tratada como caso 1 e a configuração diagnosticável será tratada como caso 2, neste trabalho. Note que a única diferença da configuração diagnosticável para a configuração não diagnosticável é que o evento d passa de observável para não-observável.

4.5.1 Análise da diagnosticabilidade com base no modelo 1

Para a construção do diagnosticador, deve-se seguir os passos descritos na seção 4.2. Primeiramente atentamos para a composição paralela entre G_p (figura 4.27) e A_ℓ (figura 4.28). Esta composição terá como resultado o autômato $G^m_{\ell p}$ mostrado na figura 4.29. Para analisar a diagnosticabilidade é necessário calcular o G^m_{dp}, isto é, observador de $G^m_{\ell p}$, lembrando que o conjunto de eventos não observáveis para o caso 1 é $E_{uo} = \{b, c, d, e, \sigma_f\}$. O autômato G_{dp} é mostrado na figura 4.30. Logo depois, computa-se $G^m_t = G^m_{dp} || G^m_\ell$, representado na figura 4.31. Note que a linguagem de G_p é não diagnosticável. Isso ocorre porque há uma componente fortemente conexa formada por estados (x_d, x_ℓ), sendo x_d incerto e x_ℓ certo. Esta componente é formada pelos estados: $(\{2N, 2Y\}, (2Y))$, $(\{3Y, 3N, 0N, 0Y, 5Y, 5N\}, (3Y))$,

Tabela 4.5: Descrição dos estados do autômato da figura 4.27 que modela a planta controlada .

Estado	Descrição
0	Nenhuma peça no sistema
1	peça está em E_{in} antes do $atuador1$
2	peça defeituosa na esteira E_{in}
3	peça normal na esteira E_{in}
4	peça defeituosa está em frente ao $atuador1$
5	uma peça está na esteira E_{out}
6	uma peça está na esteira E_d

$(\{3Y, 3N, 0N, 0Y, 5Y, 5N\}, (0Y))$, $(\{1N, 1Y\}, (1Y))$, $(\{3Y, 0N, 4N, 0Y, 5Y, 6Y, 4Y$ $, 6N\}, (6Y))$, , $(\{3Y, 0N, 4N, 0Y, 5Y, 6Y, 4Y, 6N\}, (5Y))$, $(\{3Y, 0N, 4N, 0Y, 5Y, 6Y, 4Y,$ $6N\}, (3, Y))$, $(\{3Y, 0N, 4N, 0Y, 5Y, 6Y, 4Y, 6N\}, (4, Y))$, $(\{3Y, 0N, 4N, 0Y, 5Y, 6Y, 4Y,$ $6N\}, (0, Y))$, $(\{3Y, 3N, 0N, 0Y, 5Y, 5N\}, (5, Y))$.

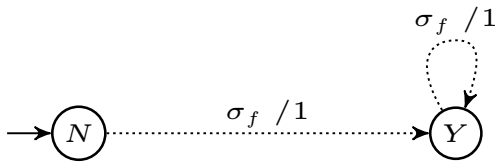

Figura 4.28: Autômato rotulador para o sistema de manufatura.

4.5.2 Análise da diagnosticabilidade com base no modelo 2

Tomando o conjunto de eventos não observáveis é $E_{uo} = \{b, c, e, \sigma_f\}$ e repetindo o procedimento da subseção anterior tem-se $G^m_{\ell p}$ e G^m_{dp} diferentes dos anteriores, conforme mostrado nas figuras 4.32 e 4.33, respectivamente. Em seguida, computa-se $G^m_{tp} = G^m_{dp}||G^m_{\ell p}$, representado na figura 4.34. Note que a linguagem de G_p é diagnosticável pois não há componente fortemente conexa em G_t formada por estados (x_d, x_ℓ): x_d incerto e x_ℓ certo. Portanto, pode-se calcular o tempo máximo para diagnose, utilizando o autômato G_{f_1} mostrado na figura 4.35 e sua matriz A max-plus correspondente. A relação entre os índices da matriz A e os estados de G_{f_1} é mostrada na tabela 4.6. Considerando que $t_s = 1$, claramente por meio de A^+ percebe-se que $t_f = 7 + 1 = 8$.

$$A = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon \\ 5 & \varepsilon & \varepsilon \\ \varepsilon & 2 & \varepsilon \end{pmatrix} \quad e \quad A^+ = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon \\ 5 & \varepsilon & \varepsilon \\ 7 & 2 & \varepsilon \end{pmatrix}$$

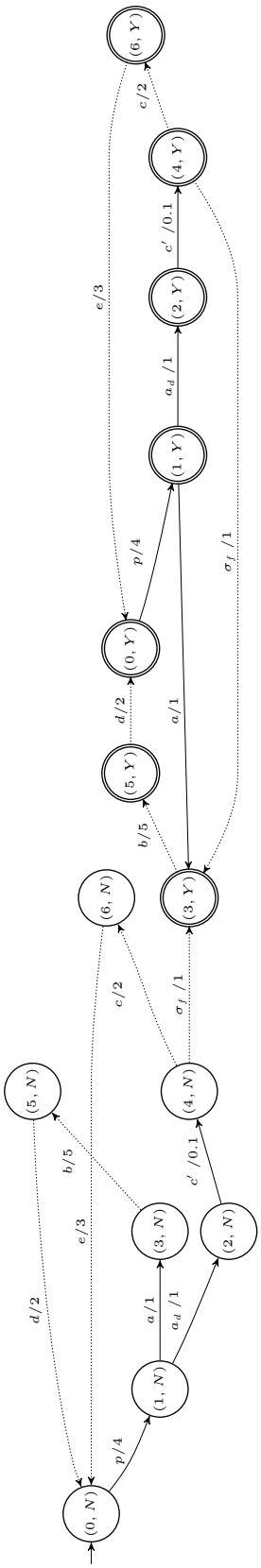

Figura 4.29: Autômato $G^m_{\ell p}$ para o caso 1.

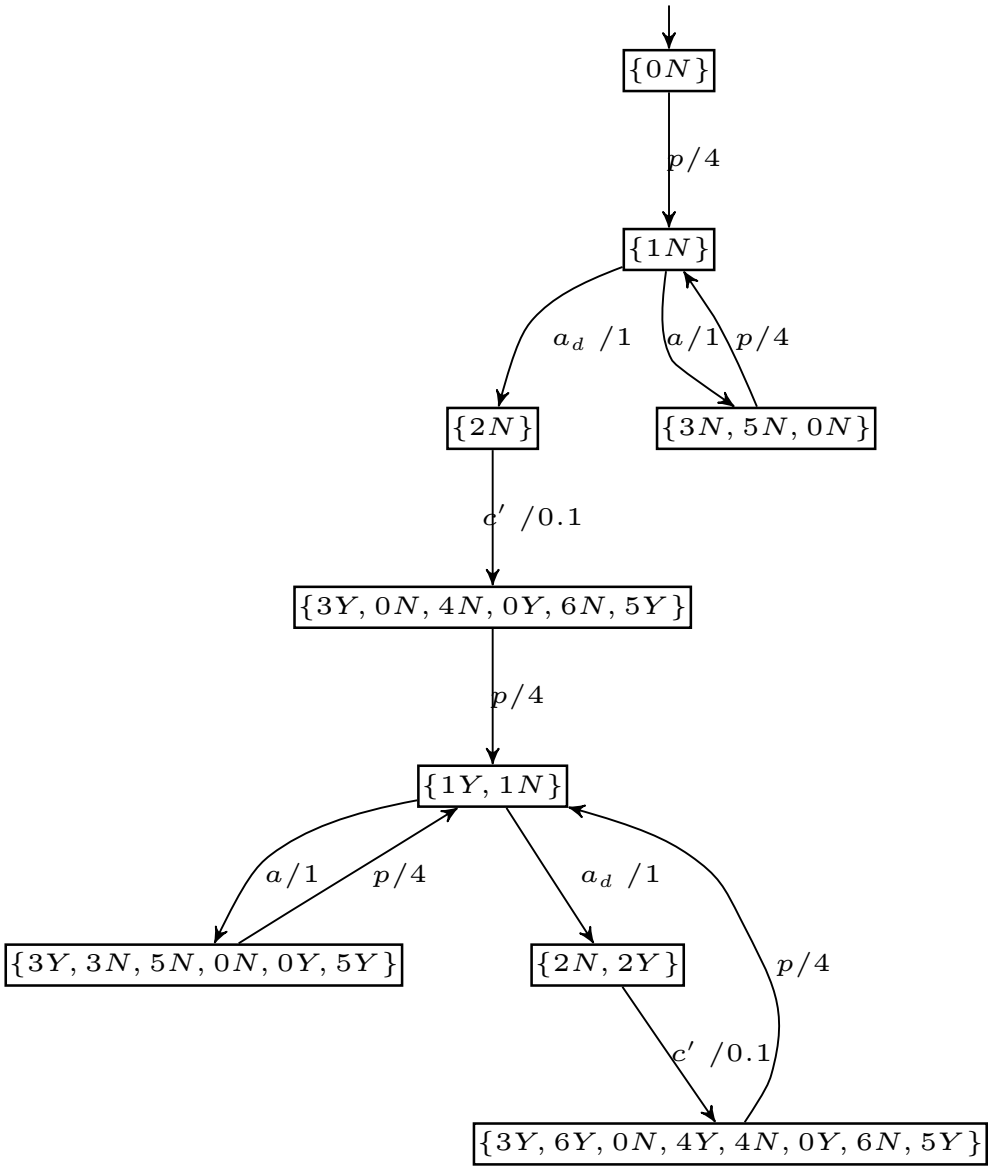

Figura 4.30: Diagnosticador G_{dp}^{m} para o caso 1.

Tabela 4.6: Relação entre os estados de G_f da figura 4.35 e os índices da matriz max-plus A correspondente.

Estado	Índice da matriz A
$(\{3Y, 4N, 6N, 5Y0N\}, (3, Y))$	1
$(\{3Y, 4N, 6N, 5Y0N\}, (5, Y))$	2
$(\{0Y\}, (0, Y))$	3

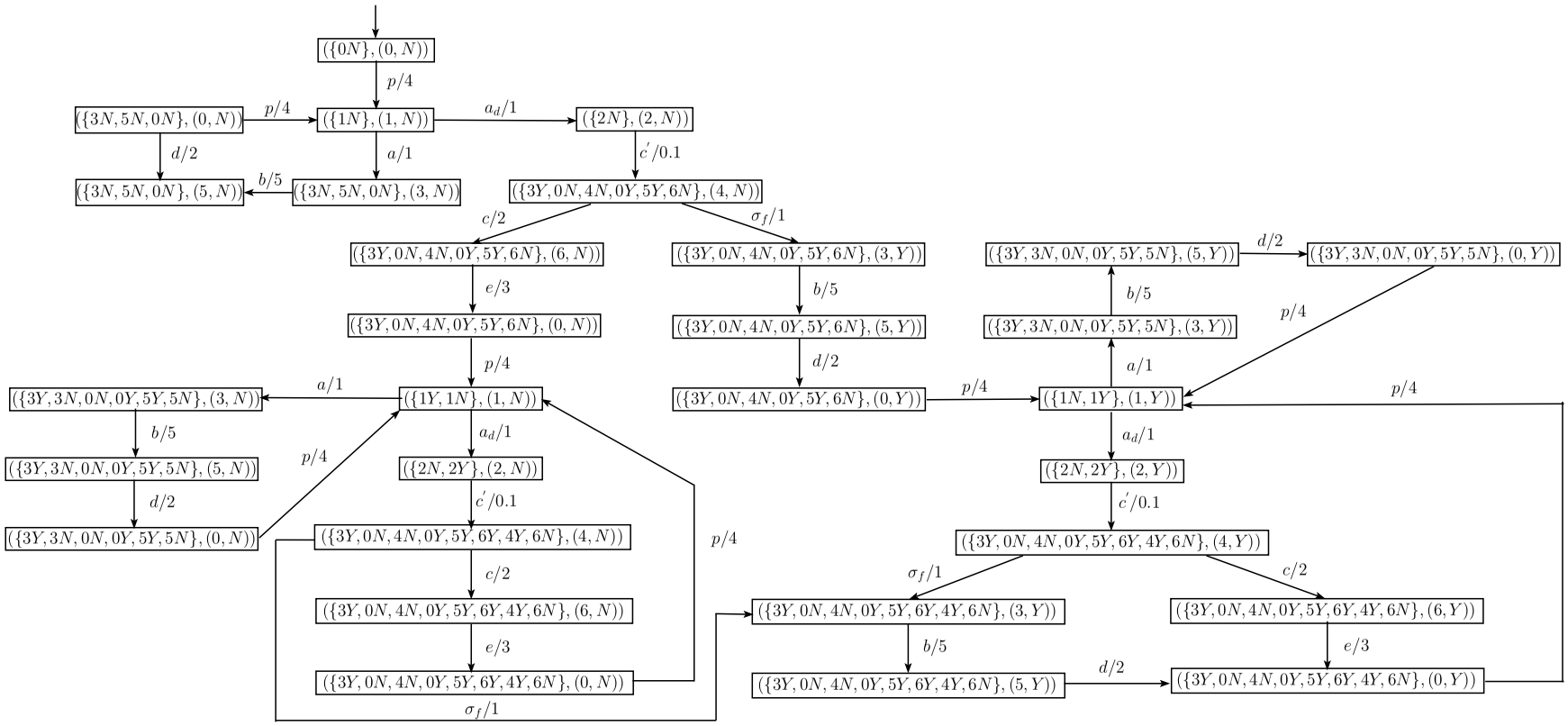

Figura 4.31: Diagnosticador G_{tp}^{m} para o caso 1.

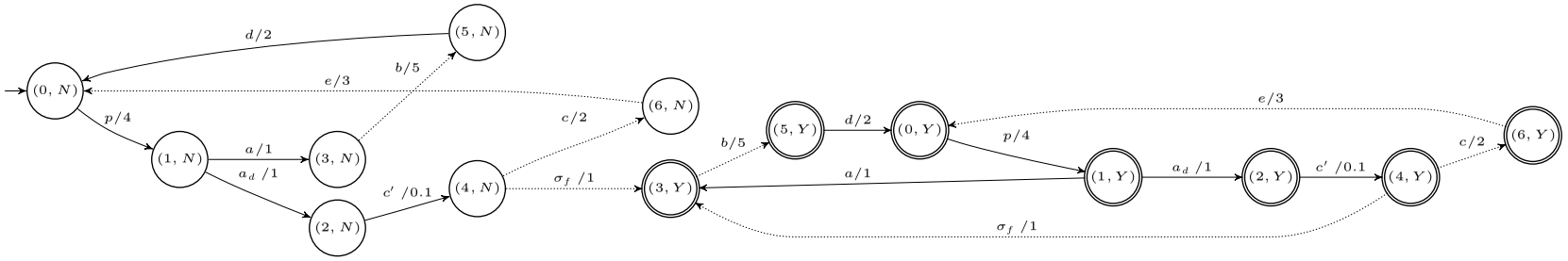

Figura 4.32: Autômato $G^m_{\ell p}$ para o caso 2.

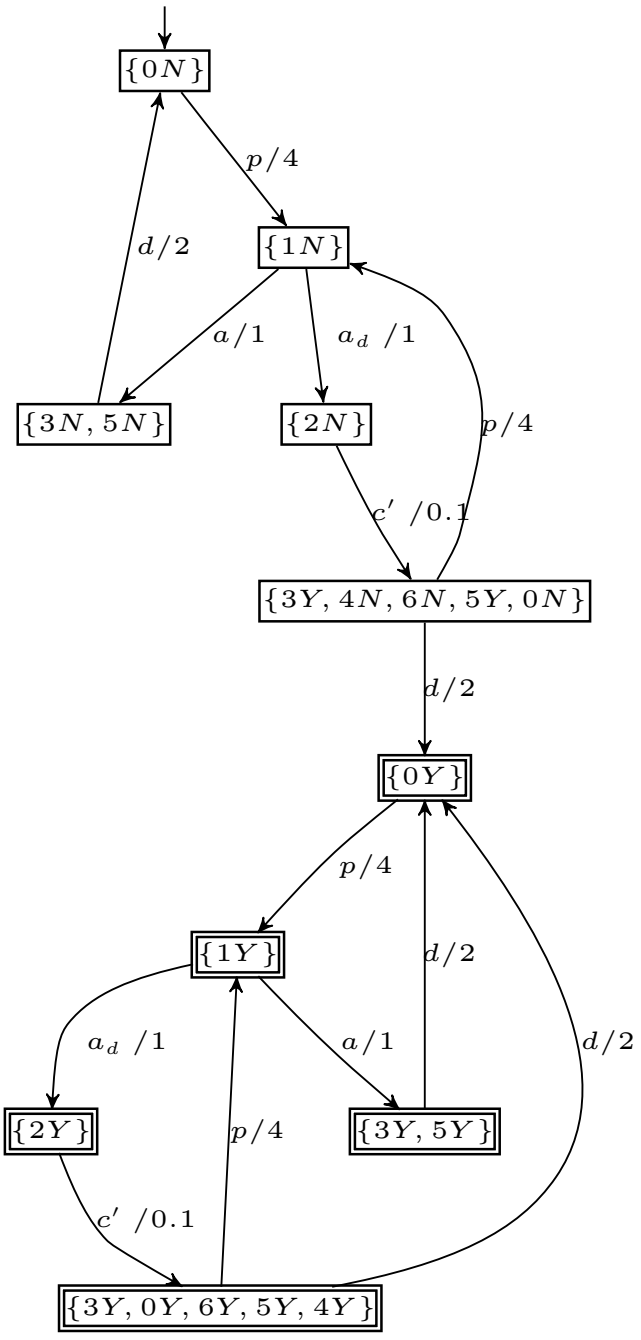

Figura 4.33: Diagnosticador G^m_{dp} para o caso 2.

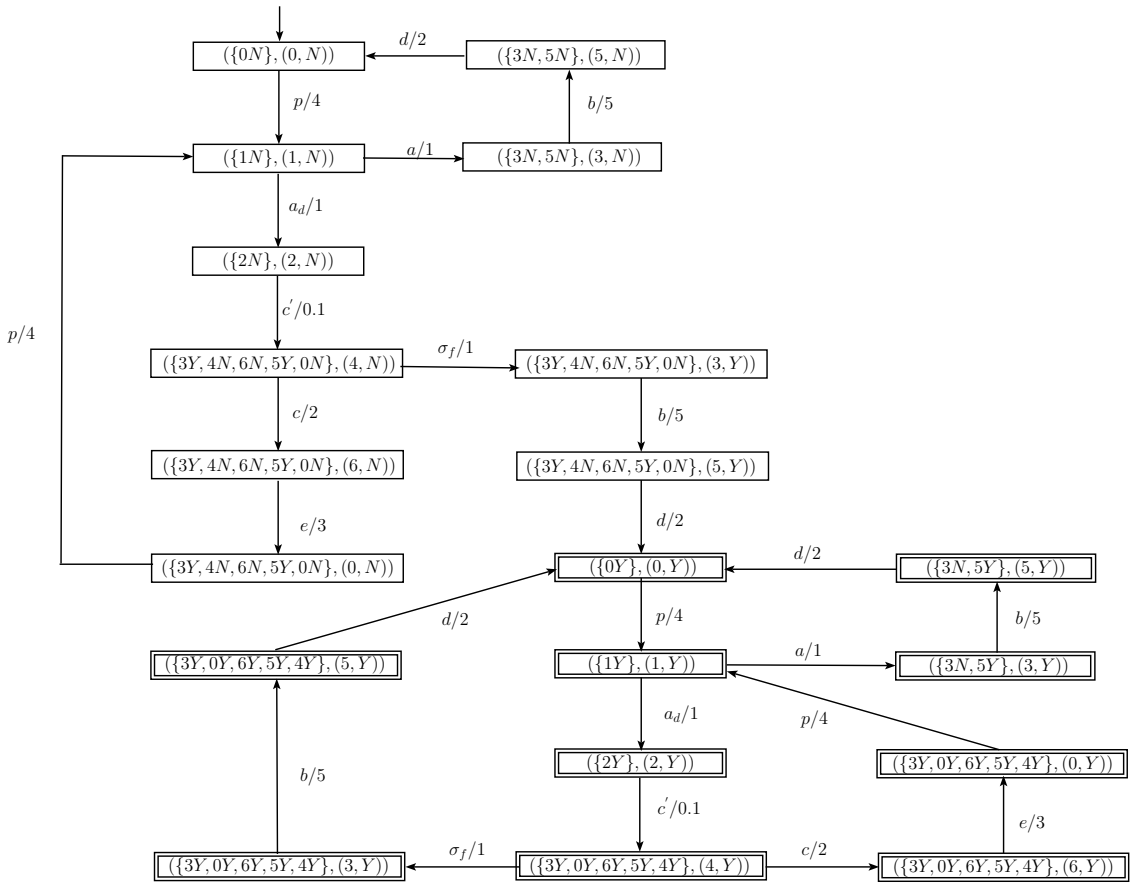

Figura 4.34: Diagnosticador G_{tp}^m para o caso 2.

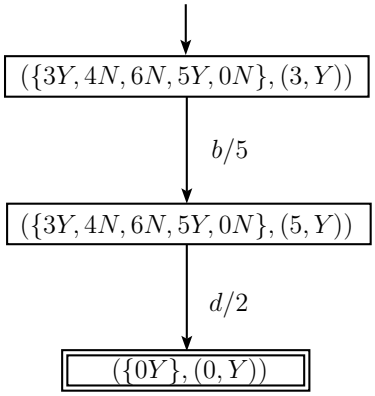

Figura 4.35: Autômato G_{f_1} para o caso 2.

4.6 Comentários finais

Como pode ser observado no exemplo da seção anterior, o diagnosticador de teste é capaz de informar sobre a diagnosticabilidade da linguagem de um sistema modelado por um autômato ponderado. Essa verificação é feita por meio da busca por componentes fortemente conexas. Além disso, esse diagnosticador utiliza as ponderações e os conceitos de álgebra max-plus para estimar o tempo para se diagnosticar a falha. Logo, fica clara a utilidade desse diagnosticador de teste.

Capítulo 5

Conclusões e Trabalhos Futuros

Neste trabalho foi proposta uma nova abordagem para a verificação da diagnosticabilidade de falhas de SEDs utilizando um novo autômato, chamado de diagnosticador de teste, que substitui a busca por ciclos indeterminados pela busca de componentes fortemente conexas que possui a vantagem de ter complexidade linear [27]. Na verificação da diagnosticabilidade por meio do diagnosticador proposto por SAMPATH et al. [2] é preciso procurar um conjunto de estados incertos no diagnosticador e verificar se esses estados levam a ciclos apropriados no autômato que modela a planta. Isto é, apenas o diagnosticador em [2] não carrega informação suficiente para determinar se uma linguagem é diagnosticável ou não. O diagnosticador de teste, por outro lado, já carrega toda informação necessária para a verificação da diagnosticabilidade de uma determinada linguagem que modela o comportamento da planta.

Uma outra contribuição deste trabalho é o desenvolvimento de um método que combina o diagnosticador de teste proposto com resultados da álgebra max-plus, para calcular o tempo máximo para a diagnose de falhas. Em sistema reais, apenas a certeza de que ela ocorreu pode não ser suficiente para o bom funcionamento da planta. Por isso, por razões de segurança, deve-se também estimar o tempo máximo para a diagnose da falha.

Considerando o contexto de sistemas ponderados pelo tempo, foi também desenvolvido neste trabalho um algoritmo que computa a máxima linguagem harmoniosa. Trabalhos recentes [35, 37] introduziram o conceito de máxima linguagem harmoniosa sem porém desenvolver um método sistemático para computá-la. Nesses sistemas, esse cálculo é importante porque essa linguagem marca a maior linguagem do sistema ponderado pelo tempo em que não há sobreposição de eventos indesejados.

Pode-se citar como continuações imediatas deste trabalho: (i) verificação da diagnosticabilidade usando o diagnosticador de teste para sistemas descentralizados; (ii) o desenvolvimento de um algoritmo para diagnose de falhas com o modelo de autômatos ponderados pelo tempo, isto é, incluindo relação de exclusão mútua.

Outros trabalhos podem ser desenvolvidos mais à frente, como a utilização do diagnosticador de teste para a diagnose robusta e a utilização do diagnosticador de teste no processo de recuperação de SEDs após uma falha ter sido detectada.

Referências Bibliográficas

[1] LIN, F. "Diagnosability of discrete event systems and its applications", *Discrete Event Dynamic Systems: Theory and Applications*, v. 4, n. 1, pp. 197–212, 1994.

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

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

[4] SAMPATH, M., LAFORTUNE, S., TENEKETZIS, D. "Active diagnosis of discrete event systems", *IEEE Transactions on Automatic Control*, v. 43, n. 7, pp. 908–929, 1998.

[5] 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, pp. 33–86, 2000.

[6] SAMPATH, M., LAFORTUNE, S., TENEKETZIS, D. "A Hybrid Aproach to Failure Diagnosis of Industrial Systems", *Proc. of the American Control Conference*, v. 3, n. 7, pp. 2077–2082, 2001.

[7] SENGUPTA, R., TRIPAKIS, S. "Decentralized diagnosability of regular languages is undecidable", *In Proc. 41st IEEE Conference on Decision and Control*, v. 1, n. 1, pp. 423–428, 2002.

[8] ZAD, S. H., KWONG, R. H., WONHAM, W. M. "Fault diagnosis in discrete-event systems: Framework and model reduction", *IEEE Transactions on Automatic Control*, v. 48, n. 7, pp. 1199–1212, 2003.

[9] SU, R., WONHAM, W. "A model of component consistency in distributed diagnosis", *International Workshop on Discrete Event Systems*, v. 16, n. 1, pp. 427–432, 2004.

[10] SU, R., WONHAM, W. M. "Global and local consistencies in distributed fault diagnosis for discrete-event systems", *IEEE Transactions on Automatic Control*, v. 32, n. 12, pp. 1923–1935, 2005.

[11] PAOLI, A., LAFORTUNE, S. "Diagnosability analysis of a class of hierarchical state machines", *Discrete Event Dynamic Systems: Theory and Applications*, v. 18, n. 3, pp. 385–413, 2008.

[12] 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.

[13] MOREIRA, M. V., JESUS, T. C., BASILIO, J. C. "Polynomial time verification of decentralized diagnosability of discrete event systems", *IEEE Transactions on Automatic Control*, v. 56, n. 7, pp. 1679–1684, 2011.

[14] 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.

[15] BASILIO, J. C., SOUZA LIMA, S. T., LAFORTUNE, S., et al. "Computation of minimal event bases that ensure diagnosability", *Discrete Event Dynamic Systems: Theory and Applications*, v. 22, n. 3, 2012.

[16] BASILIO, J. C., LAFORTUNE, S. "Robust codiagnosability of discrete event systems", *American control conference*, v. 10, pp. 2202–2209, 2009.

[17] CARVALHO, L., MOREIRA, M., BASILIO, J. "Generalized robust diagnosability of discrete event systems", *In Proc. IFAC World Congress*, v. 18, n. 1, pp. 8737–8742, 2011.

[18] ZAYTOON, J., LAFORTUNE, S. "Overview of Fault Diagnosis Methods for Discrete Event Systems", *Annual Reviews in Control*, v. 37, n. 2, pp. 308–320, 2013.

[19] CASSANDRAS, C. G., LAFORTUNE, S. *Introduction to Discrete Events Systems*. 2nd ed. New York, NY : USA, Springer, 2008.

[20] MURATA, T. "Petri Nets: Properties, Analysis and Applications", *Proceedings of the IEEE*, v. 77, n. 4, pp. 541–580, 1989.

[21] PETERSON, J. L. *Petri net theory and the modeling of systems*. Upper Saddle River, NJ, Prentice Hall, 1981.

[22] DAVID, R., ALLA, H. "Petri nets for Modeling of Dynamic Systems - A Survey", *Automatica*, v. 30, pp. 175–202, 1995.

[23] NUNES, C. E. V., BASILIO, J. C., SOTOMAYOR, O. A. "Diagnóstico de falhas em uma unidade de separação água-óleo-gás usando um modelo a eventos discretos", *XIX Congresso Brasileiro de Automática*, pp. 2547–2554, 2012.

[24] NUNES, C. E. V. *Diagnóstico de falhas em uma unidade de separação água-óleo-gás usando um modelo a eventos discretos*. Dissetação de mestrado, UFRJ, COPPE, 2012.

[25] VALIANT, G. L. "Graph-theoretic properties in computational complexity", *Journal of Computer and System Sciences*, v. 13, n. 3, pp. 278–285, 1976.

[26] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., et al. *Introduction to Algorithms*. Cambridge, MA, MIT Press, 2007.

[27] TARJAN, R. "Depth first search and linear graph algorithms", *SIAM Journal of Computer*, v. 1, n. 2, pp. 146–160, 1972.

[28] ALUR, R., DILL, D. L. "Optimal paths in weighted timed automata", *Theoret. Comput. Sci.*, v. 126, n. 2, pp. 183–235, 1994.

[29] TRIPAKIS, S. "Fault diagnosis for timed automata", *In Proc. Int. Conf. on Formal Techniques in Real Time and Fault Tolerant Systems.*, v. 24, n. 3, pp. 205–224, 2002.

[30] SU, R., VAN SCHUPPEN, J., ROODA., J. E. *Supervisory control of partially observed weighted discrete-event systems*. Relatório técnico, Eindhoven University of Technology, Department of Mechanical Engineering, Eindhoven, 2010.

[31] SU, R., VAN SCHUPPEN, J., ROODA., J. E. "Coordinated distributed time optimal supervisory control", *American Control Conference*, v. 11, pp. 907–912, 2013.

[32] HEIDERGOTT, B., OLSDER, G. J., , et al. *Max Plus at Work*. New Jersey, NJ, Princeton University Press, 2006.

[33] KOPROWSKI, A., WALDMANN, J. "Max/Plus Tree Automata for Termination of Term Rewriting", *Acta Cybernetica*, v. 19, n. 2, pp. 357–392, 2009.

[34] VIENNOT, G. X. "Heaps of pieces, I: Basic definitions and combinatorial lemmas", *in Combinatoire Énumérative*, v. 2, pp. 321–350, 1997.

[35] SU, R., VAN SCHUPPEN, J., ROODA., J. E. "The synthesis of time optimal supervisors by using heaps-of-pieces", *IEEE Transactions on Automatic Control*, v. 57, n. 1, pp. 105–118, 2012.

[36] WONHAM, W. M., RAMADGE, P. J. "On the supremal controllable sublanguage of a given language", *SIAM Journal on Control and Optimization*, v. 25, n. 3, pp. 637–659, 1987.

[37] SU, R., VAN SCHUPPEN, J., ROODA., J. E. *The synthesis of time optimal supervisors by using heaps-of-pieces*. Relatório técnico, Eindhoven University of Technology, Department of Mechanical Engineering, Eindhoven, 2009.

[38] BOUZAN, B. P. *Algoritmos em tempo polinomial para verificação da observabilidade e normalidade de linguagens regulares*. Dissertação de mestrado, UFRJ, COPPE, 2013.

[39] BASILIO, J. C., 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.