Universidade Federal
do Rio de Janeiro
Escola Politécnica

IMPLEMENTAÇÃO DE DIAGNOSTICADORES DE FALHAS ROBUSTOS À OBSERVAÇÃO SIMULTÂNEA DE EVENTOS

Gabriel Alcantara Costa Silva

Projeto de Graduação apresentado ao Curso de Engenharia de Controle e Automação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientador: Marcos Vicente de Brito Moreira

Rio de Janeiro
Agosto de 2013

IMPLEMENTAÇÃO DE DIAGNOSTICADORES DE FALHAS ROBUSTOS À OBSERVAÇÃO SIMULTÂNEA DE EVENTOS

Gabriel Alcantara Costa Silva

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO DE CONTROLE E AUTOMAÇÃO.

Examinado por:

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

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

Prof. Lilian Kawakami Carvalho, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
AGOSTO DE 2013

Silva, Gabriel Alcantara Costa

Implementação de diagnosticadores de falhas robustos à observação simultânea de eventos/Gabriel Alcantara Costa Silva. – Rio de Janeiro: UFRJ/ Escola Politécnica, 2013.

X, 69 p.: il.; 29, 7cm.

Orientador: Marcos Vicente de Brito Moreira

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia de Controle e Automação, 2013.

Referências Bibliográficas: p. 67 – 69.

1. Sistemas a eventos discretos. 2. Redes de Petri. 3. Autômatos. 4. Diagnose de falha. 5. Controladores lógicos programáveis. 6. Diagrama Ladder. 7. Eventos observados simultaneamente. I. Moreira, Marcos Vicente de Brito. II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia de Controle e Automação. III. Título.

Agradecimentos

Agradeço a minha família que me deu suporte e educação e me possibilitaram ser a pessoa e o profissional que sou hoje.

Agradeço aos meus amigos que tornaram a execução deste trabalho mais divertida e dinâmica.

Agradeço à Universidade Federal do Rio de Janeiro que forneceu um ensino de excelência e embasamento técnico suficiente para a realização deste trabalho.

Finalmente, agradeço ao meu professor e orientador, Marcos Vicente de Brito Moreira, por todas as horas gastas de aconselhamento e orientação.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/ UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Controle e Automação.

Implementação de diagnosticadores de falhas robustos à observação simultânea de eventos

Gabriel Alcantara Costa Silva

Agosto/2013

Orientador: Marcos Vicente de Brito Moreira

Curso: Engenharia de Controle e Automação

Na indústria, a implementação de sistemas de diagnóstico de falhas pode ser feita utilizando-se Controladores Lógico Programáveis (CLPs) que funcionam a partir da execução contínua de ciclos de varredura. Durante um mesmo ciclo de varredura, dois eventos consecutivos podem ser observados simultaneamente, como o apertar de um botão e a chegada de uma peça, por exemplo. Entretanto, esse fenômeno tem forte influência sobre o diagnosticador, que sobretudo se torna sensível à forma de implementação prática utilizada. Este trabalho realiza, então, um estudo sobre os impactos desse fenômeno no diagnóstico de falhas de sistemas a eventos discretos, assim como uma formalização para a implementação prática de diagnosticadores em diagrama ladder de sistemas sujeitos à ocorrência de observações simultâneas de eventos.

Palavras-chave: Sistemas a eventos discretos, Redes de Petri, Autômatos, Diagnose de falha, Controladores lógicos programáveis,Diagrama Ladder, Eventos observados simultaneamente.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Engineer.

IMPLEMENTATION OF ROBUST FAULT DIAGNOSERS WITH RESPECT TO SIMULTANEOUS OBSERVATION OF EVENTS

Gabriel Alcantara Costa Silva

August/2013

Advisor: Marcos Vicente de Brito Moreira

Course: Control and Automation Engineering

In Industry, the implementation of fault diagnosis systems can be done by using Programmable Logic Controllers (PLCs) that run by performing continuous execution of scan cycles. During the same scan cycle, two consecutive events can be observed simultaneously; pushing a button and the arrival of a piece are examples. However, this phenomenon has a strong influence on the diagnoser, which becomes particularly sensitive to the form of the practical implementation used. This work conducts, then, a study of the impacts of this phenomenon in fault diagnosis of discrete event systems, as well as a formalization for practical implementation of diagnosers in ladder diagram of systems subjected to the occurrence of simultaneous observations of events.

Keywords: Discrete event systems, Petri nets, Automata, Fault diagnosis, Programmable Logic Controllers,Ladder Diagram, Simultaneosly observed events,.

Sumário

Lista de Figuras

Capítulo 1

Introdução

O mundo moderno vem, há um tempo, sofrendo uma revolução em que tudo tende cada vez mais a se tornar automatizado. Sistemas automáticos estão presentes em grandes processos industriais e até nas residências de cada cidadão e se multiplicam a uma velocidade muito grande. Pode-se observar sistemas parcialmente ou totalmente automatizados em sistemas de manufatura, robôs industriais e domésticos, supervisão de tráfego, sistemas operacionais, sistemas de informação, entre outros.

Sistemas de automação são em geral movidos a eventos discretos. Eventos são ações que ocorrem instantaneamente e alteram o estado de um sistema. Como exemplos temos o sinal de um sensor de presença, um comando ou o apertar de um botão. O fato deste tipo de sistema ser movido a eventos torna difícil a análise por meio da utilização de equações diferenciais ou a diferenças e, por isso, é necessário que se forneça ferramentas para o estudo de sistemas a eventos discretos (SEDs) [1].

Os conceitos mais importantes no estudo de SEDs são os autômatos e as redes de Petri [1, 2]. Autômatos são grafos orientados, em os que vértices representam os estados e as arestas, que conectam um vértice a outro, os eventos. Os eventos representam, então, a transição de um estado do sistema para outro. As redes de petri, por sua vez, são um grafo bipartido com dois tipos de vértices, os lugares e as transições. A conexão entre dois vértices é realizada por arcos e não pode haver arcos ligando dois vértices do mesmo tipo em uma rede de petri.

A diagnose de falhas é uma área fundamental no estudo de sistemas a eventos

discretos, pois é desejável que caso uma falha não-observável ocorra, seja possível indicar a sua ocorrência e, assim, evitar maiores danos ou prejuízos ao sistema. Para se realizar o diagnóstico de falhas de um SED é construído um autômato diagnosticador. A partir da observação de sequências de eventos do sistema, os diagnosticadores identificam a ocorrência ou não de falhas. A diagnose de falhas e a construção de diagnosticadores tem recebido atenção constante na literatura, como pode ser observado nos trabalhos [3–11].

Na indústria, a implementação de controladores e diagnosticadores de sistemas a eventos discretos é realizada, principalmente, por meio da utilização de Controladores Lógico Programáveis (CLP). Um CLP funciona a partir de realização de ciclos de varredura e durante esses ciclos realiza as seguintes tarefas: (i) a leitura das entradas é realizada; (ii) o código de controle programado é executado; (iii) as variáveis de saída são atualizadas. Os CLPs suportam 5 linguagens definidas pela norma internacional IEC61131-3 [12]: (i) Diagrama de blocos de função; (ii) Diagrama ladder; (iii) Sequenciamento gráfico de funções (SFC em inglês); (iv) Texto estruturado; (v) Lista de instruções. A linguagem mais utilizada na implementação de sistemas a eventos discretos utilizando os controladores lógico programáveis é o diagrama ladder. Portanto, esta será a única linguagem abordada neste trabalho.

Embora implementação de controladores de sistemas a eventos discretos já foi bem explorada pela literatura, pode ser visto em [13–22], a implementação de diagnosticadores utilizando-se o diagrama ladder recebeu muito pouca atenção.

Um fenômeno que ocorre na implementação de diagnosticadores utilizando o diagrama ladder é a observação simultânea de eventos consecutivos e esse fenômeno pode ter forte influência no funcionamento do diagnosticador. A observação simultânea de eventos consiste na leitura simultânea de dois ou mais eventos consecutivos, durante um mesmo ciclo de varredura de um CLP. Devido às limitações do próprio CLP, o diagnosticador pode não identificar a ordem de ocorrência dos eventos, acarretando, até mesmo, a perda da capacidade do diagnosticador de indicar corretamente a ocorrência de uma falha. Eventos independentes e consecutivos como

o sinal de um sensor de nível e o apertar de um botão de comando são exemplos de eventos que possuem possibilidade de sofrer esse tipo de observação. Esse fenômeno ainda não foi abordado na literatura e será o tema central deste trabalho. Será realizado um estudo sobre os impactos da observação simultânea de eventos e também uma maneira de eliminá-los, bem como uma forma prática de implementação.

Este trabalho está organizado da seguinte forma: no capítulo 2 serão apresentados os fundamentos teóricos necessários para o entendimento deste trabalho; os conceitos de controladores lógico programáveis e a implementação de sistemas utilizando diagrama ladder são apresentados no capítulo 3. A implementação de diagnosticadores em diagrama ladder e o estudo do fenômeno de observação simultânea de eventos, assim como uma formalização para a correção dos efeitos indesejáveis ocasionados por esse fenômeno serão abordados no capítulo 4. Por fim, as conclusões e trabalhos futuros no capítulo 5.

Capítulo 2

Fundamentos Teóricos

Neste capítulo serão apresentados os fundamentos teóricos necessários para o entendimento e formulação do problema abordado neste trabalho. A teoria de sistemas a eventos discretos é a base para a estruturação do trabalho. Sistemas a eventos discretos (SEDs) são sistemas cujas realizações não podem ser modeladas de forma contínua, pois são caracterizadas por eventos discretos e instantâneos. O espaço de estados do sistema é um conjunto discreto e os eventos e transições são observados em instantes de tempo também discretos. Tal observação discreta traz algumas particularidades ao sistema, que serão abordadas ao longo do desenvolvimento do problema proposto. À medida que os eventos ocorrem, o sistema tem sua evolução e transita, assim, pelo espaço de estados. O pressionar de um botão, por exemplo, caracteriza um evento e leva o sistema a modificar o seu estado. A sequência de eventos gerada pelo sistema caracteriza a sua evolução e dita em qual estado o sistema se encontra.

A modelagem de tais sistemas não é trivial e por isso é necessária a formulação de uma teoria consistente para servir de suporte à análise e síntese de problemas envolvendo esse tipo de sistemas. Os formalismos matemáticos serão abordados a seguir.

2.1 Definições e Operações

Serão listadas a seguir alguma definições e também operações básicas no contexto de sistemas a eventos discretos a fim de fornecer um conhecimento prévio mínimo para o entendimento deste trabalho. Para um aprofundamento maior na teoria recomenda-se a leitura mais detalhada de [1].

Definição 2.1 *(Linguagem): Uma linguagem é um conjunto de sequências de comprimento finito formadas a partir dos elementos de um conjunto de símbolos ou alfabeto (E).*

Como exemplo, seja o conjunto de símbolos ou eventos $E = \{a, b, g\}$. Podemos definir uma linguagem

$$L_1 = \{\varepsilon, a, abb\},$$

consistindo de três sequências apenas, ou a linguagem

$$L_2 = \{\text{todas as sequências de comprimento 3, começando por a}\}.$$

A chave para a construção de sequências e, consequentemente, linguagens é a *concatenação.* A sequência abb em L_1 é construída a partir da concatenação das sequências ab com o evento, ou sequência de comprimento um, b. A sequência ab, por sua vez, é a concatenação dos eventos a e b. A sequência nula ε representa o *elemento identidade* da concatenação: $u\varepsilon = \varepsilon u = u$.

Dado um conjunto E, o conjunto formado por todas as sequências de comprimento finito dos elementos de E, incluindo a sequência nula ε, é denominado E^* e a operação * é chamada de *Fecho de Kleene.*

Pode-se concluir, então, que uma linguagem sobre um conjunto de eventos E é uma subconjunto de E^*.

É necessário, então, apresentar algumas terminologias para o entendimento dos conceitos de sequências, linguagens e suas operações. Seja a sequência $tuv = s$ com

t, u, e $v \in E^*$, então:

- t é um prefixo de s;
- u é uma subsequência de s;
- v é um sufixo de s.

A notação s/t será utilizada para denotar o sufixo de s depois de seu prefixo t. Se t não for prefixo de s, então s/t não é definido. Pode-se observar, também, que ε e s são prefixos, subsequências e sufixos de s.

Definição 2.2 *(Operações com linguagens):*

- *Concatenação: Seja* $L_a, L_b \subseteq E^*$*, então*

$$L_a L_b := \{s \in E^* : (s = s_a s_b) \ e \ (s_a \in L_a) \ e \ (s_b \in L_b)\},$$

  ou seja, uma sequência $s \in L_a L_b$ *se é formada por uma sequência de* L_a *concatenada com uma sequência de* L_b*.*

- *Fecho do Prefixo: Seja* $L \subseteq E^*$*, então*

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

  O Fecho de Prefixo de L *é a linguagem* $\overline{L}$ *constituída por todos os prefixos de todas as sequências de* L*. Em geral* $L \subseteq \overline{L}$ *e quando* $L = \overline{L}$ *a linguagem é denominada prefixo-fechada, ou seja, todos os prefixos de todas as sequências da linguagem também são sequências dessa mesma linguagem.*

- *Fecho de Kleene: Seja* $L \subseteq E^*$*, então*

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

- *Pós-Linguagem: Seja $L \subseteq E^*$ e $s \in L$. A pós-linguagem de L após s, cuja notação é L/s, é a linguagem formada por*

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

Por definição, temos que $L/s = \emptyset$ se $s \notin \overline{L}$.

Definição 2.3 *(Projeção): Suponha que $E_s \subset E_l$. Então, a projeção $P : E_l^* \to E_s^*$ pode ser definida como:*

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

$$P(e) := \begin{cases} e, & \text{se } e \in E_s \\ \varepsilon, & \text{se } e \in E_l/E_s \end{cases},$$

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

Podemos dizer que a operação de projeção apaga os eventos do conjunto maior (E_l) que não pertencem ao conjunto menor E_s e mantém os demais eventos do conjunto maior E_l.

O mapeamento inverso $P^{-1} : E_s^* \to 2^{E_l^*}$ também pode ser definido como

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

A projeção inversa retorna o conjunto de todas as sequências do conjunto maior (E_l) que projetam uma sequência do conjunto menor (E_s).

2.2 Autômatos

Um autômato determinístico é representado por G e constituído pela seguinte sêxtupla:

$$G = (X, \Sigma, f, \Gamma, x_0, X_m), \tag{2.1}$$

em que X representa o conjunto de estados, Σ é o conjunto de eventos, $f : X \times \Sigma \to X$ é a função de transição de estados, $\Gamma : X \to 2^{\Sigma}$ é a função dos de eventos ativos, x_0 é o estado inicial, $X_m \subseteq X$ é o conjunto de estados marcados.

Os caminhos gerados por um autômato, isto é, as possíveis sequências de nós e arestas do grafo, ou ainda as sequências de estados e transições, podem ser representados com os conceitos e formalismos de linguagens apresentados anteriormente e a conexão entre eles se dá de maneira fácil como vista a seguir.

Definição 2.4 *(Linguagem gerada):* $\mathcal{L}(G) := \{s \in \Sigma^* : f(x_0, s) \text{ é definida}\}$.

A linguagem gerada $\mathcal{L}(G)$ representa, então, todos os caminhos diretos que podem ser seguidos através do grafo de um autômato a partir de seu estado inicial.

Definição 2.5 *(Linguagem marcada):* $\mathcal{L}_m(G) := \{s \in \mathcal{L}(G) : f(x_0, s) \in X_m\}$.

A *linguagem marcada* representa todas as sequências de eventos que podem ser obtidas a partir do diagrama de estados de um autômato começando no estando inicial e alcançando um estado marcado.

Os autômatos podem ser classificados como *determinísticos* ou *não-determinísticos*. Um autômato é dito determinístico quando não existe ambiguidade na função de transição de estados e todos os eventos relacionados a essas transições estão contidos em Σ. Caso contrário, isto é, caso haja duas transições distintas partindo do mesmo estado e marcadas pelo mesmo evento ou então existe alguma transição rotulada pela transição ε, a função passa a ter uma estrutura não-determinística. O caso não determinístico será apresentado na seção a seguir.

2.2.1 Autômatos Não-Determinísticos

No desenvolvimento de sistemas a eventos discretos nem sempre é possível realizar um sistema que represente com exatidão a realidade. Para isso existe uma classe de autômatos capaz de representar a dinâmica do sistema, mesmo havendo incertezas. Para tanto, é utilizado um autômato não-determinístico. As incertezas podem

ser provenientes de limitações técnicas no sensoriamento do sistema, do desconhecimento parcial da dinâmica do sistema durante sua modelagem e também pode surgir devido à impossibilidade de se atestar, com certeza, quais efeitos determinado evento pode ter sobre o sistema em questão.

Essas características têm como consequência o fato da função de transição de estados não mais relacionar um evento com apenas um possível estado seguinte, mas sim um conjunto de estados. É necessário, também, representar os eventos ditos não-observáveis e para isso é utilizada a sequência vazia ε. Por fim, pode ser necessário expandir o estado inicial do sistema para um conjunto de possíveis estados iniciais. Assim, a definição de um *Autômato Não-determinístico* é apresentada a seguir:

$$G_{nd} = (X, \Sigma \cup \{\varepsilon\}, f_{nd}, \Gamma, x_0, X_m), \tag{2.2}$$

em que $f_{nd} : X \times \Sigma \cup \{\varepsilon\} \rightarrow 2^X$ é a funcão de transição não-determinística e $x_0 \subseteq X$ é o conjunto de estados iniciais.

Para poder caracterizar a linguagem gerada e marcada de um autômato não determinístico definimos primeiro o conjunto $\varepsilon - Reach$ de um estado x, ou seja, todos os estados que são possíveis de se alcançar a partir de x, por meio de transições rotuladas por ε. Por convenção esse conjunto será denotado por $\varepsilon R(x)$ e contém o próprio estado x. Uma vez definida a função $\varepsilon - Reach$ podemos definir a função de transição estendida do sistema $f_{nd}^{ext} : X \times \Sigma^* \rightarrow 2^X$ da seguinte forma:

$$f_{nd}^{ext}(x, \varepsilon) = \varepsilon R(x), \tag{2.3}$$

e para uma sequência $u \in \Sigma^*$ e um evento $e \in \Sigma$,

$$f_{nd}^{ext}(x, ue) := \varepsilon R(x)[\{z : z \in f_{nd}(y, e) \text{ para um estado } y \in f_{nd}^{ext}(x, u)]\}. \tag{2.4}$$

A função de transição estendida representa primeiramente todos os estados alcançáveis pela sequência u a partir do estado x . A partir desse conjunto de estados possíveis, identificam-se todos os estados y que possuem o evento e ativo, alcançando

assim o estado $z \in f_{nd}(y, e)$. Ao final, considera-se também a incerteza inerente a esse conjunto de estados, tomando-se o conjunto $\varepsilon - reach$ dos estados z. Deve-se notar que as funções f_{nd} e f_{nd}^{ext} não são idênticas e as respectivas notações devem ser respeitadas afim de se evitar ambiguidades.

Podemos agora definir as linguagens gerada e marcada por um autômato não-determinístico, como a seguir

$$\mathcal{L}(G_{nd}) = \{s \in \Sigma^* : (\exists x \in x_0)[f_{nd}^{ext}(x, s) \text{ é definida}]\}, \tag{2.5}$$

$$\mathcal{L}_m(G_{nd}) = \{s \in \mathcal{L}(G_{nd}) : (\exists x \in x_0)[f_{nd}^{ext}(x, s) \cap X_m \neq \emptyset]\}. \tag{2.6}$$

2.2.2 Operações com autômatos

Nesta seção serão apresentadas as operações sobre autômatos. Dentre elas estão as operações unárias e as de composição.

As operações unárias alteram o diagrama de estados de um autômato, porém sem alterar o conjunto de eventos Σ. A partir das definições 2.4 e 2.5, pode-se observar que é possível deletar de G todos os estados que não são acessíveis ou alcançáveis a partir de x_0 por uma sequência pertencente a $\mathcal{L}(G)$, sem alterar a linguagem gerada e marcada por G. Ao se apagar um estado de G, todas as transições associadas a esse estado também serão apagadas. A operação de se apagar todos os estados de um autômato G que não são acessíveis a partir do estado inicial é representada por $Ac(G)$ e pode ser definida como a seguir:

Definição 2.6

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

sendo:

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

$$X_{ac,m} = X_m \cap X_{ac},$$

$$f_{ac} = f|_{X_{ac}\times\Sigma\to X_{ac}},$$

em que $f|_{X_{ac}\times\Sigma\to X_{ac}}$ denota a função de transição com domínio restrito a $X_{ac} \times \Sigma$.

Outra operação unária consiste em eliminar todos os estados não coacessíveis de um autômato. Um estado $x \in X$ é dito ser coacessível, se existe uma sequência de eventos possíveis no diagrama de estados de G que levem o autômato do estado x para um estado marcado. A parte coacessível de um autômato $CoAc(G)$ pode ser definida como:

Definição 2.7

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

sendo:

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

$$x_{0,coac} = \begin{cases} x_0 & \text{se } x_o \in X_{coac} \\ \textit{indefinido,} & \textit{caso contrário} \end{cases},$$

$$f_{coac} = f|_{X_{coac}\times\Sigma\to X_{coac}}.$$

Já as operações de composição são realizadas entre autômatos e têm como propósito produzir sistemas complexos a partir da composição de módulos menos complexos. As duas operações que serão apresentadas nesse trabalho são o Produto e a Composição Paralela.

Na operação de produto, denotada por $\times$, apenas as transições sincronanizadas nos autômatos de origem serão realizadas no autômato resultante da composição. Em contrapartida, na Composição Paralela, denotada por $||$, eventos pertencentes a apenas um dos autômatos são autorizados a ocorrer. Isto é, eventos privados que representam o comportamento interno do autômato podem ocorrer.

Definição 2.8 *O produto entre dois autômatos G_1 e G_2 é definido como:*

$$G_1 \times G_2 := Ac(X_1 \times X_2, \Sigma_1 \cup \Sigma_2, f_{1\times2}, \Gamma_{1\times2}, (x_{01}, x_{02}), X_{m1} \times X_{m2}), \qquad (2.7)$$

em que:

$$f_{1\times 2}((x_1,x_2),e) := \begin{cases} (f_1(x_1,e), f_2(x_2,e)) & \textit{se } e \in \Gamma_1(x_1)\cap\Gamma_2(x_2) \\ \textit{indefinido,} & \textit{caso contrário} \end{cases},$$

e $\Gamma_{1\times 2}(x_1,x_2) = \Gamma_1(x_1)\cap\Gamma_2(x_2)$.

Definição 2.9 *A Composição Paralela entre dois autômatos* G_1 *e* G_2 *é definida como:*

$$G_1||G_2 := Ac(X_1\times X_2, \Sigma_1\cup\Sigma_2, f_{1||2}, \Gamma_{1||2}, (x_{01},x_{02}), X_{m1}\times X_{m2}), \tag{2.8}$$

em que:

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

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

Na composição paralela os eventos comuns poderão ocorrer somente se estiverem sincronizados entre os dois autômatos, isto é, os enventos em $\Sigma_1\cap\Sigma_2$ somente podem ocorrer simultaneamente. Os eventos que representam o comportamento individual e independente de cada autômato, eventos em $\Sigma_1\backslash\Sigma_2$ e $\Sigma_2\backslash\Sigma_1$, não são submetidos a tal restrição e podem ser executados se ativos nos seus respectivos autômatos. A composição paralela e o produto também são válidas para autômatos não-determinísticos.

Sejam os autômatos G_1 e G_2 com conjuntos de eventos $\Sigma_1 = \{a,b,c\}$ e $\Sigma_2 = \{a,b,d\}$ representados pelas figuras 2.1 e 2.2. O resultado do produto $G_1\times G_2$ pode ser visto na figura 2.3. Observa-se que o autômato resultante do produto de dois

autômatos, em que pelo menos um deles é não determinístico, também pode ser não-determiístico, como exemplificado.

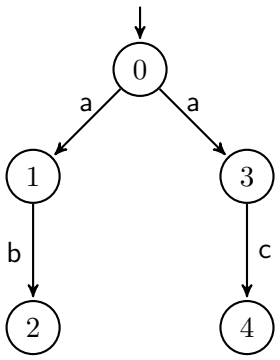

Figura 2.1: Autômato G_1.

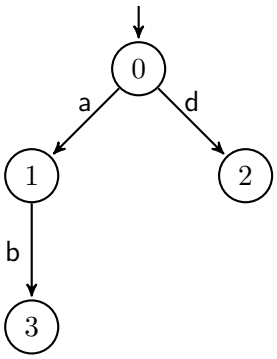

Figura 2.2: Autômato G_2.

A figura 2.4 representa a composição paralela $G_1||G_2$ e tem como resultado um autômato não-determinístico, mas que possibilita a ocorrência de eventos privados de G_1 e G_2.

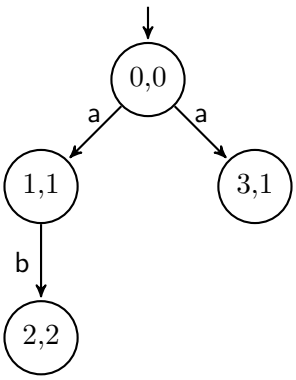

Figura 2.3: Produto $G_1 \times G_2$.

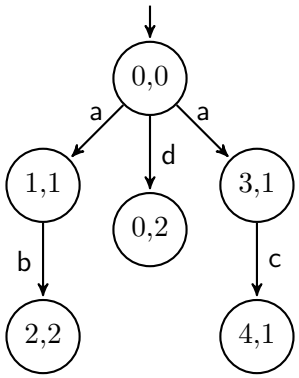

Figura 2.4: Composição Paralela $G_1||G_2$.

2.2.3 Autômato Observador

Em sistemas a eventos discretos nem sempre é possível ter acesso a ocorrência de todos os eventos que constituem o sistema. Essa impossibilidade de observação dos eventos pode ser traduzida pela inviabilidade técnica ou financeira de se sensoriar todo o sistema, ou mesmo, no caso de sistemas distribuídos, pelo fato de cada parte do sistema ter acesso a, apenas, uma parte do conjunto de eventos. Portanto, podemos particionar o conjunto de eventos de um autômato em $\Sigma = \Sigma_o \cup \Sigma_{uo}$, sendo Σ_o o conjunto de eventos observáveis ao sistema e Σ_{uo} o conjunto de eventos não observáveis.

Os autômatos observadores estão presentes então para rastrear o sistema original, gerar uma observação do sistema e, assim, estimar o seu estado atual. Os observadores são particularmente úteis para o caso de autômatos com observação parcial de eventos e, também, para o caso de autômatos não-determinísticos. O autômato observador é então um autômato determinístico que tem linguagem equivalente à projeção observável do sistema original e gera uma estimativa do seu estado atual. O autômato observador será denotado como $Obs(G_{nd})$ ou mesmo G_{obs}. Foi proposto em [1] uma formalização para a representação do *autômato observador*, como visto a seguir.

Seja $G_{nd} = (X, \Sigma \cup \{\varepsilon\}, f_{nd}, x_0, X_m)$, um autômato não-determinístico, seu observador $Obs(G_{nd}) = (X_{obs}, \Sigma, f_{obs}, x_{0,obs}, X_{m,obs})$ pode ser obtido da seguinte forma:

Algoritmo 2.1

Passo 1: *Defina* $x_{0,obs} = \varepsilon R(x_0)$ *e faça* $X_{obs} = \{x_{0,obs}\}$.

Passo 2: *Para cada* $B \in X_{obs}$ *e* $e \in \Sigma$, *defina*

$$f_{obs}(B, e) := \varepsilon R(\{x \in X : (\exists x_e \in B)[x \in f_{nd}(x_e, e)]\}),$$

caso $f_{obs}(x_e, e)$ *seja definido para algum* $x_e \in B$, *inclua o estado* $f_{obs}(B, e)$ *em* X_{obs}. *Caso contrário,* $f_{obs}(B, e)$ *não é definido.*

Passo 3: *Repetir o Passo 2 até que a toda a parte acessível de* $Obs(G_{nd})$ *for contruída.*

Passo 4: *Faça* $X_{m,obs} := \{B \in X_{obs} : B \cap X_m \neq \emptyset\}$.

O algoritmo 2.1 tem como resposta um autômato que realiza o papel de observador externo, ou seja, a partir do estado inicial ele estima em quais possíveis estados o autômato original G pode se encontrar após a observação de uma sequência de eventos. O observador evoluí através de todos os estados que o autômato G que podem ser alcançados, por meio de eventos não-observáveis ou devido ao comportamento não determinístico do sistema.

Três importantes propriedades de $Obs(G)$ devem ser destacadas: (i) $Obs(G_{nd})$ é um autômato determinístico, (ii) $\mathcal{L}(Obs(G_{nd})) = \mathcal{L}(G_{nd})$ e (iii) $\mathcal{L}_m(Obs(G_{nd})) = \mathcal{L}_m(G_{nd})$

2.3 Diagnose de falhas

Na análise de sistemas que possuem observação parcial de eventos é desejável que se possa estimar a ocorrência ou não de um evento, ou ainda afirmar, com certeza, a ocorrência do evento. A partir da observação da sequência de eventos realizada por um autômato, é possível inferir sobre a ocorrência de eventos não-observáveis e, assim, realizar alguma ação a respeito, ou mesmo, no caso de uma falha, uma ação corretiva.

Para se detectar, com eficiência, a ocorrência de um evento não-observável é necessário que se possa indica-lá após a ocorrência de um número finito de eventos, posteriores à ocorrência desse evento não-observável. Entretanto, dada uma linguagem gerada por um autômato, não é sempre possível afirmar que qualquer evento não-observável poderá ser *diagnosticado* nessa linguagem.

É então, que se introduz o conceito de *diagnosticabilidade* de uma linguagem em relação a um evento ou falha não-observável, para indicar se é possível afirmar precisamente a ocorrência ou não, de um evento ou falha não-observável. Para a formalização desse conceito, consideraremos:

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

Pode-se, então, definir a diagnosticabilidade de uma linguagem formalmente [3].

Definição 2.10 *Seja L uma viva e prefixo-fechada gerada por um autômato G. Então L será dita diagnosticável em relação à projeção P_o e $\Sigma_f = \{\sigma_f\}$ se a seguinte condição for verificada:*

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

sedo D a condição de diagnose expressa por

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

2.3.1 Diagnosticadores

Diagnosticadores são utilizados para a partir da observação de sequências de eventos do sistema, identificar a ocorrência ou não de falhas não observáveis. A notação

utilizada neste trabalho para representar o autômato diagnosticador será $Diag(G)$ ou mesmo G_d. Com a observação parcial de eventos o conjunto de eventos observados pelo diagnosticador fica limitado a apenas uma projeção P_o do conjunto Σ, definida por $P_o : \Sigma^* \rightarrow \Sigma_o^*$ e, neste trabalho, será considerada apenas a diagnose de um tipo de falha não observável σ_f. Pode-se, então, definir G_d como:

$$G_d = (X_d, \Sigma_o, f_d, \Gamma_d, x_{0d}), \tag{2.9}$$

e pode ser construído a partir dos seguintes passos: (i) obtenha a composição paralela $G||A_l$, sendo A_l o autômato responsável por rotular com F estados em que σ_f já ocorreu e N estados que σ_f ainda não tenha ocorridos e pode ser observado na figura 2.5; (ii) calcule $Obs(G||A_l)$.

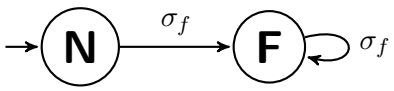

Figura 2.5: Autômato rotulador A_l.

As seguintes características podem então ser observadas:

- Todos os estados posteriores a um estado que já recebeu um rótulo de falha F também terão esse rótulo propagado. Após a ocorrência de uma falha, o diagnosticador deverá continuar indicando a ocorrência de falha nos estados subsequentes.

- Se todos os rótulos de um estado de $Diag(G)$ possuem o rótulo N, então podemos afirmar com certeza que o evento σ_f não ocorreu até o presente momento. Denomina-se então esse estado como um estado normal ou de não-falha.

- Analogamente se o estado possuir todos os rótulos F, esse será um estado certo (ou de falha) e o diagnosticador pode afirmar a ocorrência do evento σ_f. Se a sequência $t \in P_o[\mathcal{L}(G)]$ for observada e $f_d(x_{0,d}, t)$ é um estado certo, então tem-se que todas as sequências contidas em $P^{-1} \cap \mathcal{L}(G)$ deverão conter

o evento σ_f. Caso contrário um estado de G contido em $f_d(x_{0,d}, t)$ deverá ter o rótulo N.

- Se o estado atual de $Diag(G)$ contém pelo menos um rótulo N e um rótulo F, então não o diagnosticador não pode confirmar a ocorrência ou a não ocorrência do evento não observável σ_f. Esse tipo de estado é considerado um estado de incerto.

A construção do diagnosticador só tem propósito se após um número finito de eventos observados for possível afirmar com certeza se uma falha ocorreu ou não. Ou seja, somente deve-se construir o diagnosticador, caso a linguagem gerada pelo autômato G original for diagnosicável em relação a falha não observável σ_f.

Capítulo 3

Implementação em Ladder

De acordo com a definição da ABNT (Associação Brasileira de Normas Técnicas), um controlador lógico programável é um equipamento eletrônico digital com hardware e software compatíveis com aplicações industriais. Segundo a NEMA (National Electrical Manufacturers Association), é um aparelho eletrônico digital que utiliza uma memória programável para armazenar internamente instruções e para implementar funções específicas, tais como lógica, sequenciamento, temporização, contagem e aritmética, controlando, por meio de módulos de entradas e saídas, vários tipos de máquinas ou processos, ou seja, é um equipamento idealizado especificamente para reduzir a complexidade de implementação e manutenção de sistemas de controle e automação.

Os CLPs interagem com a planta através de entradas e saídas que podem ser tanto digitais, quanto analógicas. As entradas analógicas se encontram em uma faixa contínua de tensão ou corrente elétrica. Um exemplo de entrada digital, com valores lógicos 0 ou 1, é o apertar de um botão que envia um sinal elétrico indicando que o botão foi apertado ou um sensor de presença que só tem a capacidade de indicar a presença ou não de uma peça, pessoa ou objeto em questão. Um exemplo muito encontrado em sistemas de controle de processos para entradas analógicas são os sensores ou transdutores de temperatura, que transformam a escala da temperatura medida em valores, normalmente, de 4 a 20 mA. Atuadores se conectam às saídas do controlador e podem ser atuadores pneumáticos, válvulas e relés elétricos, por

exemplo.

Por essas características os CLPs se tornaram a principal ferramenta para a implementação de controladores a eventos discretos na indústria e suportam 5 linguagens definidas pela norma internacional IEC61131-3 [12]:

1. Diagrama de blocos de função;
2. Diagrama ladder;
3. Sequenciamento gráfico de funções (SFC em inglês);
4. Texto estruturado;
5. Lista de instruções.

Neste trabalho será abordada a Linguagem de Diagrama Ladder por ser a mais utilizada na indústria e estar disponível em praticamente todos os CLPs.

3.1 Ciclo de varredura

Durante a execução de um código programado em um controlador lógico programável, o dispositivo realiza continuamente ciclos de varredura. Cada ciclo de varredura pode ser divido em três etapas distintas. Na primeira etapa é realizada a leitura das entrada. Em seguida, na segunda etapa, o código programado responsável pelo processamento das entradas em si é executado e, finalmente, na terceira etapa as variáveis de saída são atualizadas.

Durante a primeira etapa, o CLP armazena todos os valores lógicos das variáveis de entrada em sua memória interna e esses valores continuarão constantes no decorrer do ciclo de varredura até uma nova inicialização do ciclo e uma nova leitura dos valores. Esse fato tem grande influência na estruturação e gerenciamento do problema proposto neste trabalho. Posteriormente o CLP executa o código programado e, a partir da lógica executada, atualiza os valores nas variáveis de saída. Com todas as etapas realizadas, o controlador inicia, assim, o processo novamente

e executa um novo ciclo. O tempo necessário para a realização de cada ciclo de varredura é denominado tempo de varredura.

No contexto de diagnose de falhas de sistemas a eventos discretos, um CLP pode ser utilizado na implementação do diagnosticador do sistema e da malha de controle simultaneamente. Além de se reduzir gastos em hardware, essa implementação dupla também traz como benefício que os comandos internos da malha de controle se tornam observáveis para o diagnosticador, sem a necessidade de sensoriamento desses eventos ou mesmo uma comunicação entre o dispositivo de diagnose e o de controle.

3.2 Diagrama Ladder

O diagrama ladder é uma das cinco linguagens definidas pela norma IEC61131-3 [12] e consiste, basicamente, de contatos e bobinas que são organizados de forma a representar funções lógicas. O diagrama ladder também suporta componentes de cálculo mais complexos, como somadores, comparadores e temporizadores, possibilitando assim a composição dos mais variados tipos de sistemas. Apenas as bobinas e os contatos têm relevância para este trabalho e serão apresentados nesta seção.

3.2.1 Contatos

Os contatos representam uma parte fundamental da linguagem ladder, pois eles são os responsáveis por indicar o valor lógico das variáveis de entradas ou das variáveis internas do diagrama. A linguagem possui vários tipos de contatos como *normalmente abertos* (NA), *normalmente fechados* (NF), *detector de borda de subida* (P) e *detector de borda de descida* (N).

Os contatos NA funcionam verificando intermitentemente o estado lógico do bit associado a ele e caso o valor seja igual a 0, o contato assume valor falso, abrindo a linha do diagrama ladder em que está posicionado. Se a variável possuir valor igual a 1, o contato irá assumir valor lógico verdadeiro e dará continuidade à linha. Na

figura 3.1 é apresentada a simbologia usualmente adotada para um contato NA.

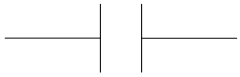

Figura 3.1: Contato Normalmente Aberto.

O contato normalmente fechado funciona de maneira inversa ao contato NA. Quando a variável associada tem valor igual a 0, o contato NF fecha a linha e assume valor lógico verdadeiro e quando a variável assume valor igual a 1, o contato se abre e a linha assume valor lógico falso. Na figura 3.2 é apresentada a simbologia usualmente adotada para um contato NF.

Figura 3.2: Contato Normalmente Fechado.

Os contatos *detector de borda de subida* (P) e *detector de borda de descida* (N) funcionam de maneira diferenciada dos NA e NF. O contato P é fechado quando o nível lógico do sinal passa de 0 para 1 e, de maneira inversa, funciona o contato N, detectando a variação de 1 para 0. Os contatos P e N funcionam detectando apenas a subida ou descida do sinal e armazenam esse valor durante todo o ciclo de varredura, independente do comportamento da variável nesse intervalo de tempo. As simbologias utilizadas para reprensentar os contatos P e N estão ilustradas nas figuras 3.3 e 3.4 respectivamente.

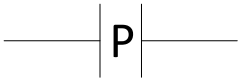

Figura 3.3: Contato tipo P.

Após o término do ciclo de varredura, os contatos N e P retornam seu valor lógico para falso e somente irão se fechar novamente se detectarem uma nova borda do sinal. No caso do contato N, a borda observada é a de descida e no caso do contato P, a de subida.

Em sistemas a eventos discretos, muitos dos sinais recebidos são impulsos elétricos, representando que determinado evento ou ação ocorreram e por isso os

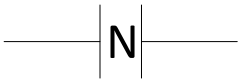

Figura 3.4: Contato tipo N.

contatos do tipo P e N são muito importantes na modelagem e implementação de sistemas de controle e diagnose de falhas em ladder. Eles são capazes de detectar e armazenar a borda de subida ou descida de um sinal e utilizar posteriormente esse valor lógico durante o ciclo de varredura, sem perda de informação.

3.2.2 Bobinas

Outro componente básico no desenvolvimento de um sistema programado em diagrama ladder são as bobinas. Elas têm o papel de acionar ou atualizar o valor de outros componentes. As bobinas são responsáveis por modificar o estado lógico das variáveis booleanas e, para isso, contam com os tipos Simples, SET e RESET.

As bobinas simples respondem diretamente à lógica da linha em que estão situadas. Caso a lógica seja verdadeira, a bobina se tornará energizada, atribuindo valor lógico 1 à sua variável associada e caso a lógica antecedente à bobina seja falsa, a bobina será desernegizada, atribuindo valor 0 à sua variável associada. A simbologia usualmente utilizada para representar as bobinas simples está representada na figura 3.5

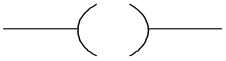

Figura 3.5: Bobina Simples.

As bobinas do tipo SET e RESET, cuja simbologias usuais estão representadas nas figuras 3.6 e 3.7 respectivamente, têm um outro tipo de funcionamento. As bobinas SET e RESET funcionam em conjunto para alterar o valor de uma variável. A variável deve, então, possuir os dois tipos de bobinas associados a ela, para poder ter o seu valor alterado tanto para falso, quanto para verdadeiro. A bobina SET é responsável por alterar o valor lógico da variável associada para 1 e esse valor continuará positivo até que uma bobina RESET seja energizada, alterando, assim,

o valor da variável para 0.

Figura 3.6: Bobina tipo SET.

Figura 3.7: Bobina tipo RESET.

3.2.3 Implementação de autômatos em linguagem ladder

Métodos descrevendo a conversão de códigos de controle para diagrama ladder são um tópico muito explorado na literatura, como pode ser visto em [13–22]. Dois problemas de implementação prática de autômatos em linguagem ladder são o efeito avalanche e a observação simultânea de eventos.

O efeito avalanche consiste na habilitação inadequada de transições consecutivas. Esse efeito é causado geralmente quando um número arbitrário de transições de estado consecutivas são executadas na ocorrência de um único evento. A figura 3.8 ilustra um autômato G_{aval}, em que é possível ocorrer o efeito avalanche. Um exemplo de implementação em diagrama ladder desse autômato pode ser visto na figura 3.9. Ao se analisar o diagrama ladder, é possível observar que na ocorrência do evento a, o autômato teria a sua evolução do estado inicial 0 para o estado 1, já que tanto o contato associado ao estado inicial quanto ao evento a estariam fechados. Entretanto, no decorrer da execução do programa, a segunda linha também teria a sua evolução realizada, executando, assim, a transição do estado 1 para o estado 2. Isso se deve ao fato de que o evento a, que habilita ambas as transições, continua ativo durante todo o ciclo de varredura. Portanto, o sistema que, com a ocorrência de apenas um evento a, deveria evoluir do estado inicial até o estado 1 somente, é levado até o estado 2.

É possível observar, então, que o efeito avalanche poderia ser evitado, para o sistema em questão, com a troca da ordem das linhas do diagrama ladder demonstrado

na figura 3.9. Ao se realizar essa troca de linhas, impossibilita-se que a transição do estado 1 para o estado 2 seja realizada indevidamente, pois como a condição de que o sistema se encontre no estado 1, só será satisfeita após a leitura da linha responsável por essa transição. O sistema não mais iria executar ambas as transições na ocorrência de apenas um evento a.

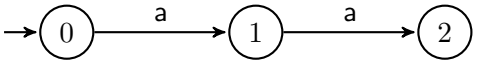

Figura 3.8: Autômato G_{aval} que ilustra possível efeito avalanche.

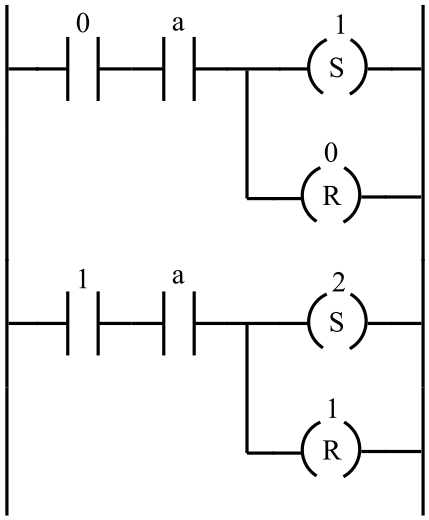

Figura 3.9: Diagrama ladder simplificado para a implementação de G_{aval}.

A observação simultânea de eventos, por sua vez, ocorre quando dois eventos consecutivos são observados em apenas um ciclo de varredura do CLP. Esse fenômeno pode ser possível quando não há dependência causal entre os eventos como, por exemplo, o apertar de um botão e a chegada de uma peça. A figura 3.10 representa um autômato no qual os eventos a e b podem ser observados simultaneamente em apenas um ciclo de varredura. A figura 3.11 ilustra uma implementação em diagrama ladder para esse autômato. Pode ser visto que se o sistema observar a e b no mesmo ciclo de varredura, o sistema irá evoluir do estado inicial 0 para o estado 1 e em seguida do estado 1 para o estado 2, que é o comportamento esperado do sistema.

Enquanto que na implementação em diagrama ladder da figura 3.13, referente ao autômato representado pela figura 3.12, é possível perceber que a evolução do

sistema se daria de maneira equivocada do estado 0 para o estado 1 na ocorrência da sequência ab. Como a transição do estado 0 para o estado 1 aparece logo na primeira linha, caso ocorra a e b simultaneamente, o sistema irá entender que o evento b ocorreu e realizar essa mudança de estados. No estado 1, não existe nenhum evento ativo e, por isso, a ocorrência de a seria ignorada. Portanto, o autômato que deveria evoluir do estado 0 para o estado 2 e em seguida para o estado 3 em um mesmo ciclo de varredura, realizará apenas uma transição para o estado 1.

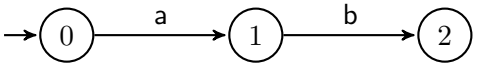

Figura 3.10: Autômato G_{obs_1} para ilustrar a observação simultânea dos eventos a e b.

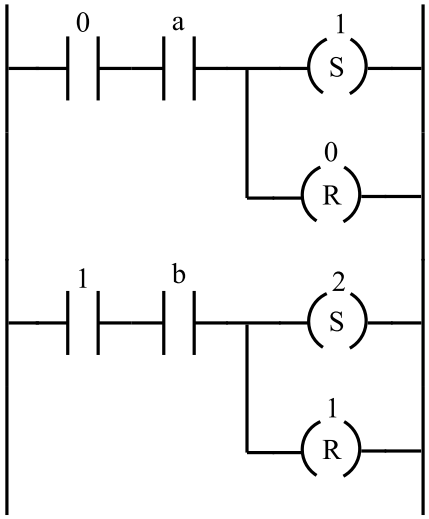

Figura 3.11: Diagrama ladder simplificado para a implementação de G_{obs_1}.

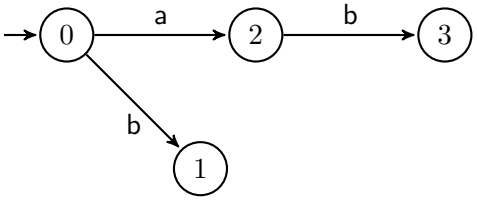

Figura 3.12: Autômato G_{obs_2} para ilustrar a observação simultânea dos eventos a e b.

É possível constatar que ambos os fenômenos tem como consequência a execução de mais de uma transição de estados em um mesmo ciclo de varredura. Entretanto,

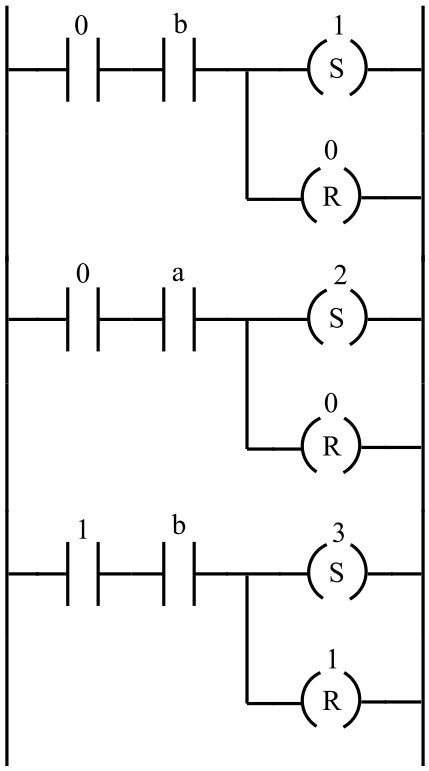

Figura 3.13: Diagrama ladder simplificado para a implementação de G_{obs_2}.

são dois fenômenos diferentes e com causas diferentes. Na observação simultânea de eventos, a execução de mais de uma transição de estados faz parte do comportamento esperado do diagrama ladder, enquanto o efeito avalanche produz indesejadamente a execução múltipla de transições de estados. Além disso, a observação múltipla de eventos em um mesmo ciclo de varredura pode fazer com que o sistema evolua para estados que não deveria.

Como o efeito da observação simultânea de eventos ainda não foi explorado pela literatura, neste trabalho serão apresentados métodos de implementação de diagramas ladder que levam em consideração o efeito avalanche e não ambos os fenômenos. Será, então, verificado sua robustez em relação à observação múltipla de eventos. Um método que leva em conta efeito avalanche foi abordado primeiramente em [13, 14]. Nesses artigos, o método apresentado realiza a identificação de eventos e a transição de estados concomitantemente e utiliza uma ordenação específica das linhas para tentar evitar o efeito avalanche. Porém, para sistemas complexos, com a presença de ciclos, o método perde a sua eficácia. Esse método será denominado,

neste trabalho, por método FH.

Uma metodologia robusta ao efeito avalanche foi proposta em [15] e [16] e o método proposto em [16] será denominado, neste trabalho, por método xMB.

O método MB se divide em quatro etapas:

- **Módulo M1** - representa a inicialização da rede de Petri, ou seja, a marcação inicial.
- **Módulo M2** - associado à identificação da ocorrência de enventos externos.
- **Módulo M3** - associado às condições para a ocorrência das transições.
- **Módulo M4** - descreve a evolução dos estados ativos, ou seja, a dinâmica da função de transição.
- **Módulo M5** - define quais alarmes e/ou ações serão realizadas.

Módulo M1

O módulo de inicialização é responsável por definir o estado inicial do sistema. É importante que esse módulo seja executado uma única vez durante o processo de inicialização e após sua execução ele seja ignorado nos ciclos subsequentes. Para isso, o ciclo dispõe de apenas uma linha com um contato NF associado a uma variável binária B_0 e duas bobinas de Set, sendo uma associada ao estado inicial do sistema e outra à própria varíavel B_0. Assim, após o primeiro ciclo de varredura, o sistema terá a sua marcação incial realizada.

Módulo M2

O módulo de identificação da ocorrência de eventos tem um papel muito importante no decorrer deste trabalho, pois ele é o responsável por detectar a ocorrência de todos os eventos a cada ciclo de varredura, independentemente de serem observados simultaneamente ou não.

A ocorrência de eventos está associada às bordas de subida ou descida dos sensores da planta ligados ao diagnosticador. Para detectar tais sinais utiliza-se, como

já foi apresentado, contatos do tipo *detector de borda de subida* (tipo P) ou *detector de borda de descida* (tipo N). Os contatos ficarão acionados, então, durante um ciclo de varredura completo.

Módulo M3

O módulo de disparo de transições é responsável por habilitar todas as transições que tenham as suas condições satisfeitas. Ao se utilizar um módulo separado apenas para a habilitação das transições, é possível organizar de maneira estruturada todas as transições. Uma transição será disparada somente quando todas as suas condições forem satisfeitas. Com a utilização de comparadores pode-se expressar todo tipo de condições.

Módulo M4

O módulo da dinâmica é responsável por atualizar os estados após a ocorrência de transições que se tornaram habilitadas no módulo de disparo das transições. Ou seja, é o módulo reponsável pela evolução dos estados do sistema.

Módulo M5

O módulo de alarmes é responsável por realizar uma ação ou acionar um alarme toda vez que uma falha é detectada. O módulo possui, então, um número de linhas igual ao número de tipos de falha modeladas pelo sistema.

Organização do diagrama ladder

Os cinco módulos apresentados devem ser implementados na ordem apresentada para que o procedimento tenha consistência em todas as suas aplicações, ou seja, (i) módulo de inicialização, (ii) módulo de ocorrência de eventos, (iii) módulo de ocorrência das transições, (iv) módulo da evolução da dinâmica e (v) módulo de acionamento de alarmes e/ou ações.

A ordem dos módulos garante que as as condições das transições sejam verificadas antes da evolução do sistema e não concomitantemente, evitando assim o efeito avalanche, ou seja, evita que duas transições consecutivas sejas disparadas, caso ambas sejam ativadas pelo mesmo evento e utiliza toda uma formalização e uma estruturação do diagrama ladder para isso. Porém, caso ocorra a observação simultânea dos eventos o sistema deverá ter a capacidade de discernir se as transições consecutivas habilitadas são decorridas da observação múltipla de eventos durante um mesmo ciclo de varredura ou se é fruto de uma indesejável possibilidade de efeito avalanche. O próximo capítulo irá explorar então se os métodos apresentados são robustos, ou não, a esse fenômeno.

Capítulo 4

Implementação de diagnosticadores com observação simultânea de eventos

A observação simultânea de eventos pode ocorrrer quando dois eventos são registrados pelo CLP no mesmo ciclo de varredura. Quando ambos os eventos se encontram de forma consecutiva no grafo do autômato que modela o sistema, podem ocorrer erros na diagnose de falha do sistema.

Mesmo que a probabilidade de ocorrência de observação simultânea de eventos seja pequena, ela é possível, principalmente se considerarmos eventos como o apertar de um botão ou como um comando de pausa ou parada, que podem ocorrer a qualquer momento e são acionados por agentes externos.

Portanto, é necessário indicar, com clareza, quais os eventos têm a possibilidade de sofrer observação simultânea em um mesmo ciclo de varredura. Para isso, será utilizada a notação Σ_{so} para denotar o conjunto de sequências de eventos que podem sofrer esse tipo de observação. Os eventos que constituem as sequências de Σ_{so} devem pertencer a Σ_o, uma vez que todos os eventos devem ser observáveis. É importante notar que não é suficiente indicar apenas quais os eventos têm possibilidade de observação simultânea, mas sim, a sequência em que eles podem ser observados,

indicando a ordem de observação.

Neste capítulo será apresentado o problema a partir de alguns casos e serão analisados seus impactos no diagnóstico de falhas.

4.1 Análise de casos

4.1.1 Caso 1

Seja o autômato G_1, representado na figura 4.1, com $\Sigma_o = \{a, b, c\}$, $\Sigma_{uo} = \{\sigma_f\}$ e $\Sigma_{so} = \{ab\}$. Considera-se que existam apenas dois caminhos partindo do estado inicial e em apenas um deles há a presença da falha não-observável σ_f. Os dois caminhos serão denominados neste texto como caminho de comportamento normal e caminho de comportamento de falha do sistema. Somente no caminho de funcionamento normal estão presentes os eventos a e b, que formam a sequência $ab \in \Sigma_{so}$.

A linguagem gerada pelo autômato G_1 é diagnosticável em relação à falha σ_f e, analisando G_1, nota-se que, caso a observação do evento c, a partir do estado inicial, seja realizada, é possível indicar a ocorrência da falha σ_f. Dessa forma, o diagnosticador de falhas de G_1 é ilustrado na figura 4.2.

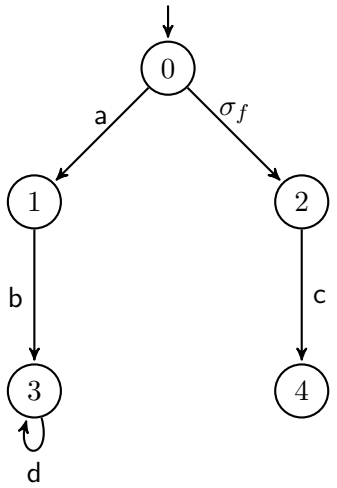

Figura 4.1: Autômato G_1.

Na ocorrência da observação simultânea da sequência ab, o comportamento esperado do sistema é evoluir do estado inicial para o estado 1 e, em seguida, para o

estado 3. Enquanto que o comportamento esperado do diagnosticador é a evolução do estado $(0N, 2F)$ para o estado $1N$ e, em seguida, para o estado $3N$. Entretanto, a evolução realizada pelo diagnosticador, em uma implementação prática, pode diferir do comportamento esperado, já que, o CLP iria observar, no caso de observação simultânea de eventos, a ocorrência de a e b, perdendo, assim, a informação da ordem de ocorrência. A análise das implementações práticas será realizada a seguir.

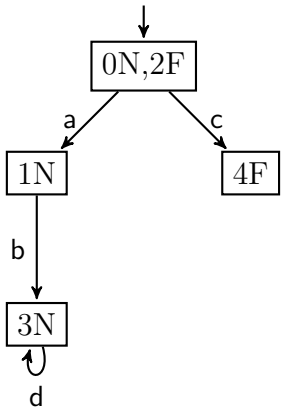

Figura 4.2: Diagnosticador de falhas de falhas de G_1.

Na figura 4.3 estão representadas duas implementações do diagnosticador de falhas do autômato G_1, utilizando o método FH, apresentado na seção 3.2.3. A diferença entre as figuras 4.3(a) e 4.3(b) consiste na troca das linhas 1 e 4.

Caso haja a ocorrência da sequência ab em um mesmo ciclo de varredura, a implementação da figura 4.3(a) realizaria a evolução do estado inicial $(0N, 2F)$ para o estado $1N$ (linha 2), já que os contatos referentes ao estado inicial e ao evento a estariam com valor lógico 1. Em seguida, com o fechamento do contato referente ao estado $1N$, na linha 4, o diganosticador iria evoluir para o estado $3N$, pois o evento b também estaria ativo no mesmo ciclo de varredura. A implementação da figura 4.3(a) não sofreu, então, impactos com o fenômeno de observação múltipla de eventos em um mesmo ciclo de varredura.

Entretanto, a troca de linhas realizada na figura 4.3(b) torna essa implementação sensível à observação da sequência ab em um único ciclo de varredura. Pode-se constatar que no momento da execução da linha 2, a evolução do estado $(0N, 2F)$ para o estado $1N$ ainda não foi realizada e, por isso, a linha 2 não terá sua transição

realizada, dado que o contato referente ao estado $1N$ estaria aberto. Sendo assim, o diagnosticador evoluiria do estado $(0N, 2F)$ com a execução da linha 4 e iria ignorar totalmente a ocorrência do evento b. Portanto, com a observação simultânea de eventos o diagnosticador ficaria preso no estado $1N$, pois, caso o diagnosticador continue sua evolução, apenas eventos d seriam observados e apenas o evento b é ativo nesse estado.

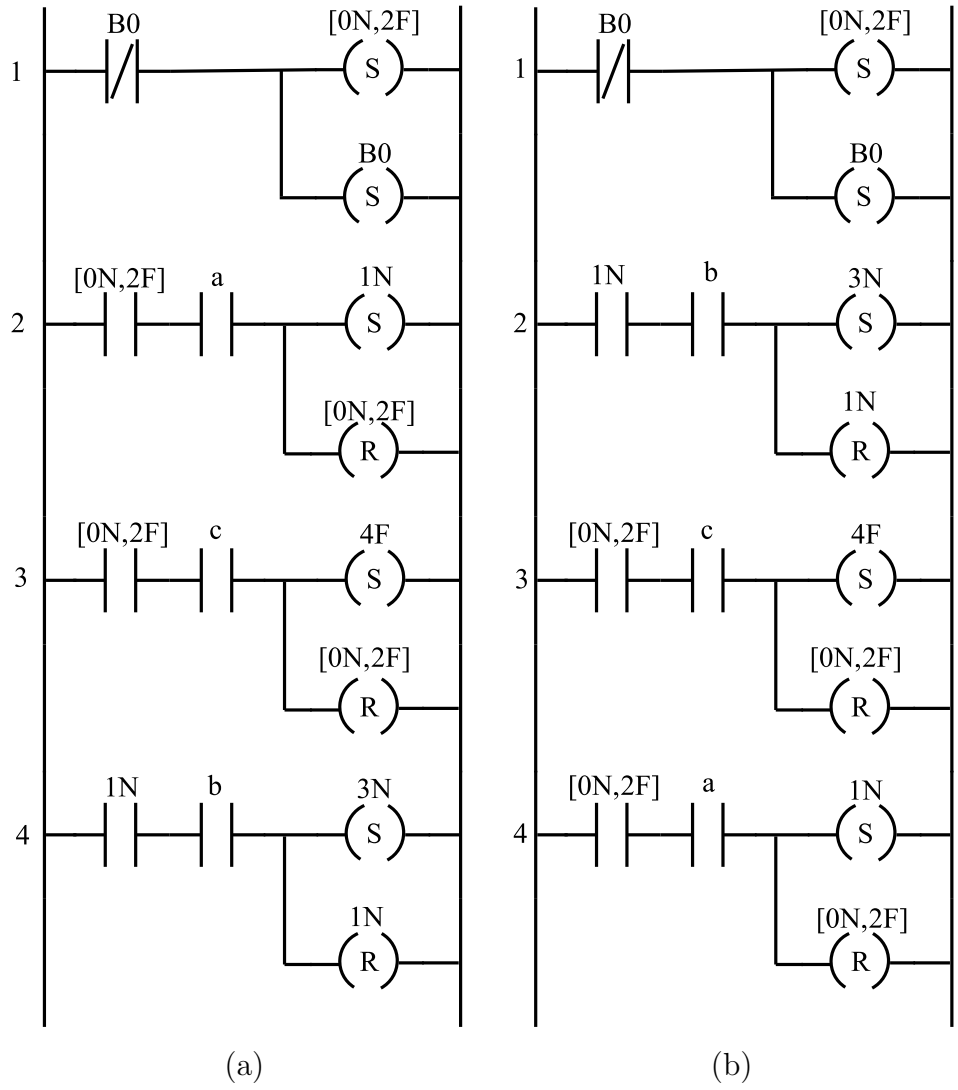

Figura 4.3: Diagramas ladder utilizando o método FH para G_1.

A figura 4.4 exemplifica a implementação do diagnosticador do autômato G_1 utilizando o método MB. A separação dos módulos de identificação de eventos e do módulo de disparo das transições tem um efeito indesejável para o diagnosticador, pois o ele não é capaz de distinguir se as duas transições consecutivas que são habi-

litadas (linhas 5 e 7), na observação simultânea da sequência ab, são provenientes de um possível efeito avalanche ou da própria observação simultânea de eventos. Sendo assim, na observação de a e b em um mesmo ciclo de varredura, o diagnosticador teria apenas a transição $t1$ habilitada (linha 5), independente da ordenação das linhas no diagrama ladder. O diagnosticador iria, então, executar somente a transição referente ao evento a e ignoraria a ocorrência do evento b. Novamente caso o sistema continue sua evolução, o diagnosticador ficaria travado indefinidamente no estado $1N$, dado que apenas o evento b é ativo nesse estado e somente a ocorrência de eventos d seria observada.

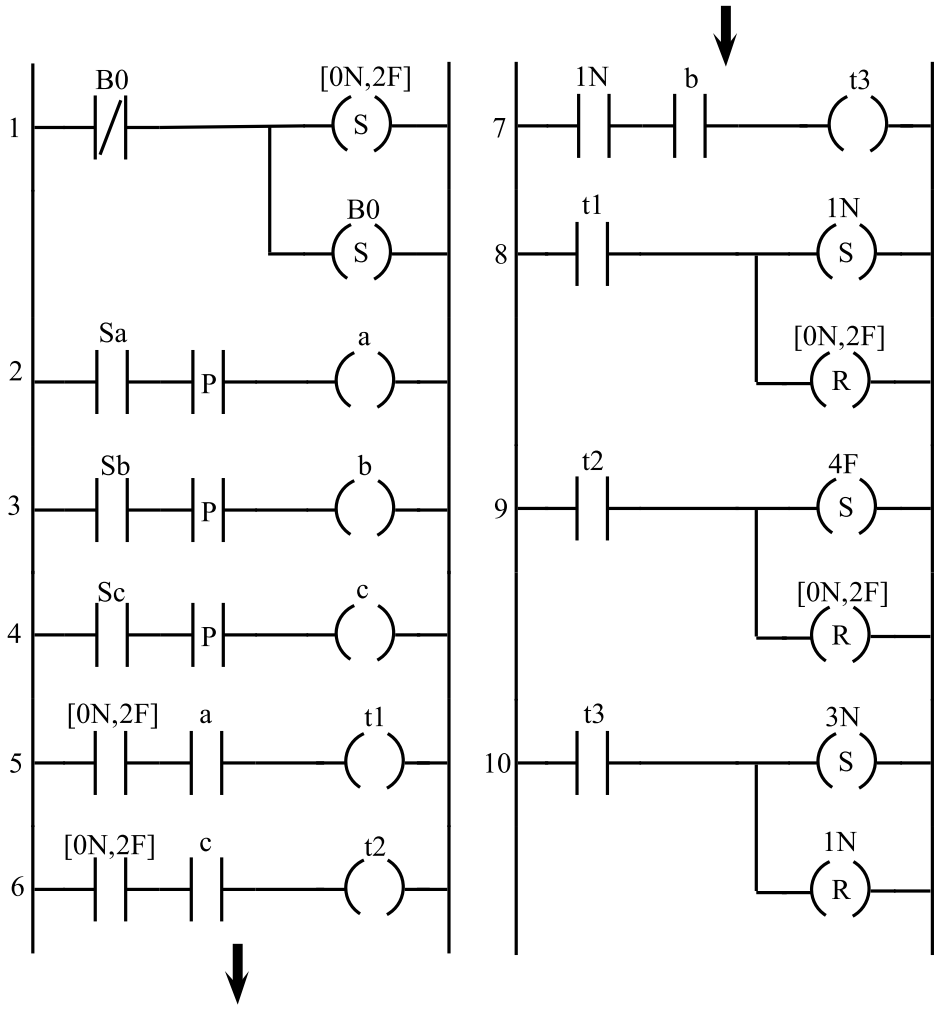

Figura 4.4: Diagrama ladder utilizando o método MB para G_1.

4.1.2 Caso 2

Seja o autômato G_2, representado na figura 4.5, com $\Sigma_o = \{a, b\}$, $\Sigma_{uo} = \{\sigma_f\}$ e $\Sigma_{so} = \{ab\}$. Novamente, a nomeclatura de caminho de funcionamento normal e de falha será utilizada para representar os dois caminhos possíveis a partir do estado inicial 0. Entretanto, no autômato G_2, os eventos a e b, que formam a sequência $ab \in \Sigma_{so}$, estão presentes no caminho de comportamento normal do sistema e o evento b, também, aparece no caminho de falha. Esse fato terá influência na análise do diagnosticador de G_2. A linguagem gerada pelo autômato G_2 é diagnosticável

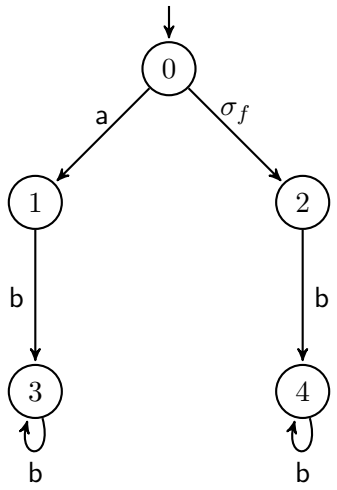

Figura 4.5: Autômato G_2.

em relação à falha σ_f e, analisando G_2, nota-se que, caso a observação do evento b, a partir do estado inicial, seja realizada, é possível indicar a ocorrência da falha σ_f. Dessa forma, o diagnosticador de falhas de G_2 é ilustrado na figura 4.6. O diagnosticador tem como característica que ambos os eventos que formam a sequência $ab \in \Sigma_{so}$, estão ativos no estado inicial. O comportamento esperado do diangosticador, na observação simultânea de a e b, é a evolução até o estado $3N$, passando pelo estado $1N$. Entretanto, em uma implementação prática, o conflito gerado pela leitura de ambos os eventos ativos no estado inicial, pelo CLP, pode gerar erros na evolução do diagnosticador.

Na figura 4.7 estão representadas duas implementações do diagnosticador de falhas do autômato G_2, utilizando o método FH. A diferença entre as figuras 4.7(a)

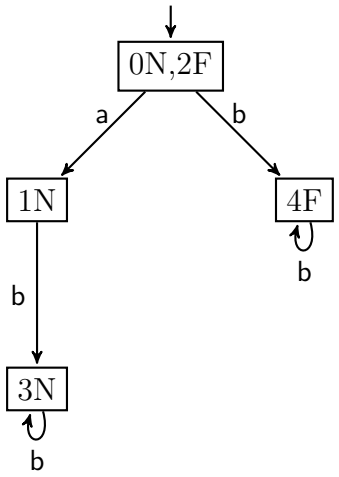

Figura 4.6: Diagnosticador de falhas de falhas de G_2.

e 4.7(b) consiste na troca das linhas 2 e 3. Caso o CLP observe a sequência ab em um único ciclo de varredura, a implementação da figura 4.7(a) teria todas as condições da linha 2 satisfeitas e o diagnosticador evoluiria, assim, do estado inicial $(0N, 2F)$ para o estado $1N$. Após essa transição de estado, a linha 4 teria, então, sua evolução realizada, dado que tanto os contatos referentes ao estado $1N$ e ao evento b estariam, agora, fechados nesse mesmo ciclo de varredura. Portanto, o diagnosticador que deveria terminar a sua evolução no estado $3N$, ignoraria a ocorrência do evento a, evoluiria para um estado errado e ainda indicaria uma falha que não ocorreu.

A figura 4.8 exemplifica a implementação do diagnosticador do autômato G_3 utilizando o método MB. Caso haja a observação simultânea dos eventos a e b, as transições t_1 e t_2 (linhas 4 e 5), associadas à evolução a partir do estado inicial $[0N, 2F]$ para os estados $[1N]$ e $[4F]$ respectivamente, se encontrariam habilitadas no módulo de identificação da ocorrência de eventos externos. Com a execução de t_1 e t_2, o sistema atingiria um cenário totalmente absurdo e se encontraria nos estados $1N$ e $4F$ simultaneamente ao final do ciclo de varredura.

4.1.3 Caso 3

Seja o autômato G_3, representado na figura 4.9, com $\Sigma_o = \{a, b, c, d, e\}$, $\Sigma_{uo} = \{\sigma_f\}$ e $\Sigma_{so} = \{ab\}$. Novamente, a nomeclatura de caminho de funcionamento normal e de

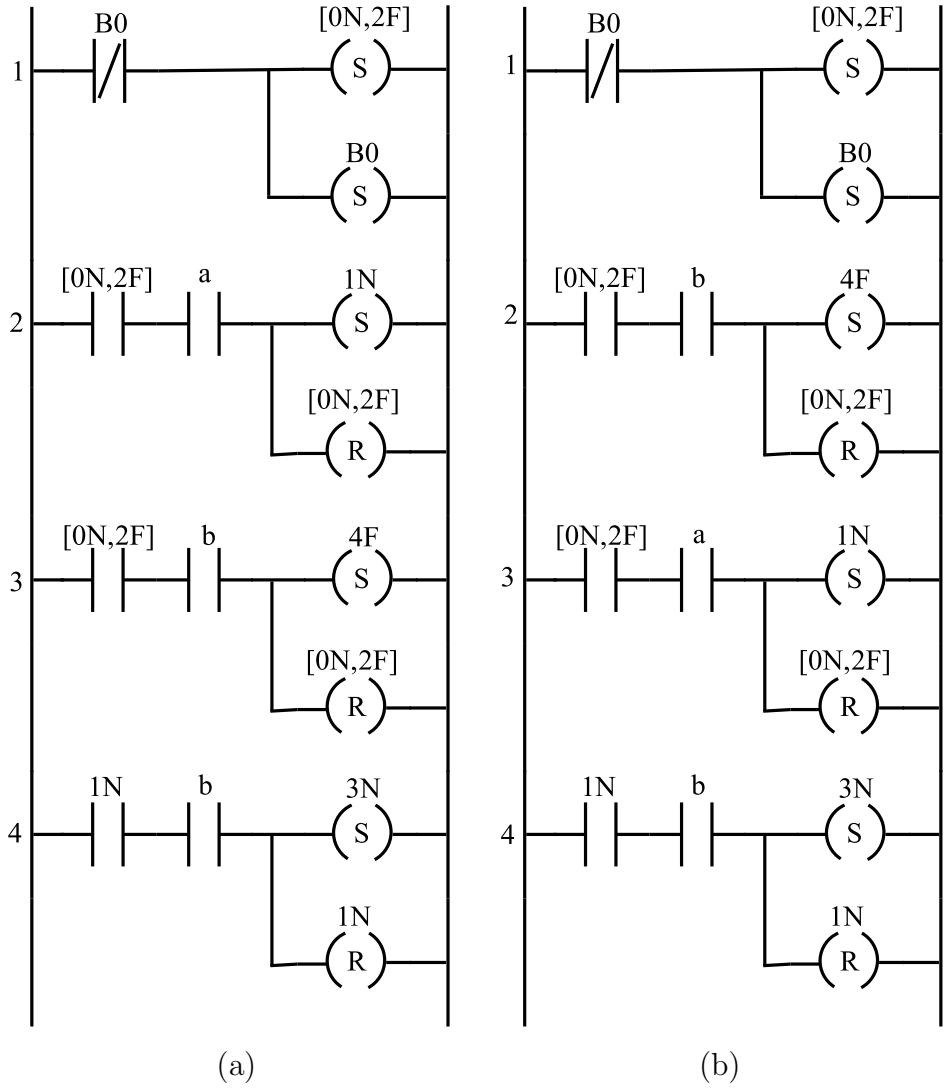

Figura 4.7: Diagramas ladder utilizando o método FH para G_2.

falha será utilizada para representar os dois caminhos possíveis a partir do estado inicial 0. Entretanto, no autômato G_3, os eventos a e b, que formam a sequência $ab \in \Sigma_{so}$, estão presentes tanto no caminho de comportamento normal do sistema, quanto no caminho de comportamento de falha. Esse fato terá influência na análise do diagnosticador de G_3.

A linguagem gerada pelo autômato G_3 é diagnosticável em relação à falha σ_f. Analisando o diagnosticador de G_3, ilustrado na figura 4.10, nota-se que, caso a observação da evento e, no estado $(3N, 6F)$, seja realizada, é possível indicar a ocorrência da falha σ_f. É possível perceber que a presença da sequêcia $ab \in \Sigma_{so}$ em ambos os caminhos de G_3, se traduziu como um único caminho no diagnosticador.

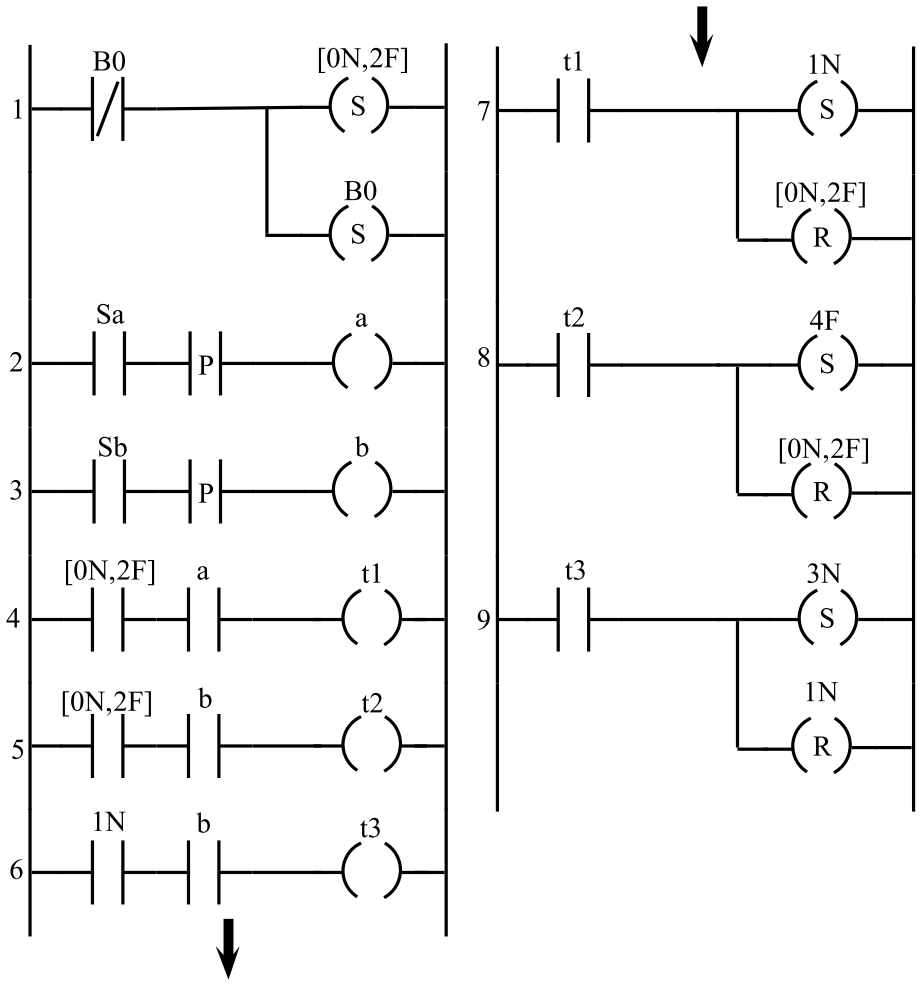

Figura 4.8: Diagrama ladder utilizando o método MB para G_2.

O que não deverá gerar nenhum conflito na observação de a e b em um único ciclo de varredura, como o conflito apresentado no caso 2, em que ambos os eventos pertencentes a sequência ab se encontravam ativos no estado inicial. O comportamento esperado do diagnosticador de G_3, caso haja a observação da sequência ab, é evoluir do estado inicial $(0N, 2F)$ para o estado $(1N, 4F)$ e, em seguida, para o estado $(3N, 6F)$.

Na figura 4.11 estão representadas duas implementações do diagnosticador de falhas do autômato G_3, utilizando o método FH. A diferença entre as figuras 4.11(a) e 4.11(b) consiste na troca das linhas 2 e 3.

Na figura 4.11(a), pode-se observar que a observação simultânea dos eventos a e b não terá impacto no diagnóstico de falhas, pois o diagnosticador iria realizar as transições de estados do estado inicial $(0N, 2F)$ para o estado $(1N, 4F)$ e, em

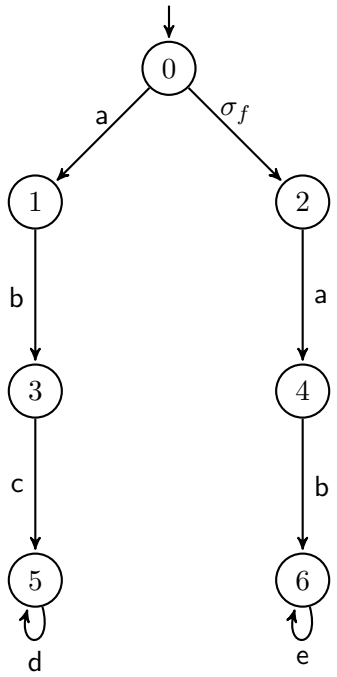

Figura 4.9: Autômato G_3.

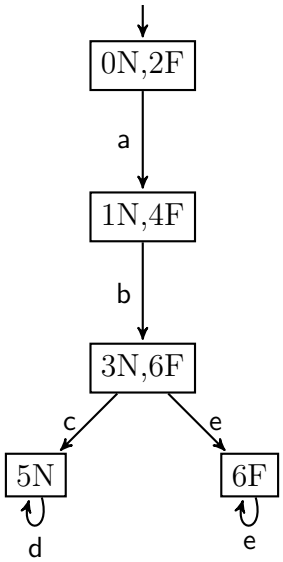

Figura 4.10: Diagnosticador de falhas de falhas de G_3.

seguida, para o estado $(3N, 6F)$, no mesmo ciclo de varredura. Porém, ao se trocar a ordem das linhas responsáveis pelas transições de estado do estado inicial, como ilustrado na figura 4.11(b), o diagnosticador não seria capaz de perceber a ocorrência do evento b, pois na execução da linha 2, que tem o estado $(1N, 4F)$ como condição para sua execução, a evolução para o esse estado ainda não teria sido realizada e o diagnosticador ainda se encotraria no estado $(0N, 2f)$. Portanto, o diagnosticador

evoluiria, somente, até o estado $(1N, 4F)$ (linha 3). Com a ocorrência de novos eventos (c ou e), o diagnosticador não realizaria mais nenhuma transição, pois somente o evento b está ativo no estado em que o diagnosticador se encontra.

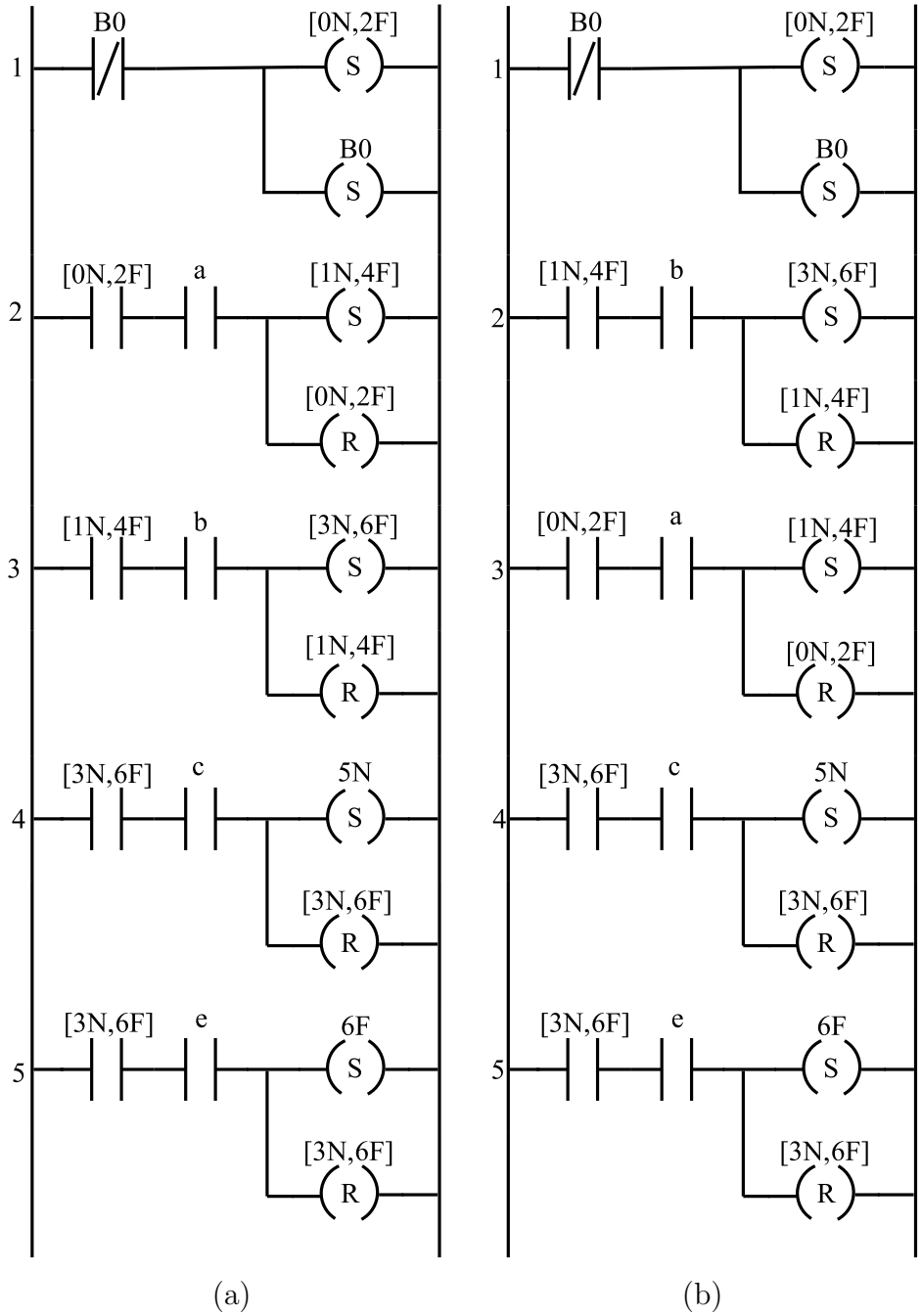

Figura 4.11: Diagramas ladder utilizando o método FH para G_3.

A figura 4.12 exemplifica a implementação do diagnosticador do autômato G_3 utilizando o método MB. Mais uma vez, o diagnosticador não consegue discernir se as múltiplas transições consecutivas que estão habilitadas no mesmo ciclo de varre-

dura são provenientes da observação simultânea de eventos ou de um possível efeito avalanche. Portanto, em vez de executar a transição t_1 e, em seguida, t_2 no mesmo ciclo, o diagnosticador irá executar a transição t_1, do estado inicial $(0N, 2F)$ para o estado $(1N, 4F)$, ignorando assim a ocorrência do evento b. Caso o diagnosticador continue a observar o autômato G_3, ele não conseguiria sair do estado $(1N, 4F)$, pois somente o evento b está ativo nesse estado e o diagnosticador iria observar os eventos c ou e no próximo ciclo.

4.1.4 Caso 4

Seja o autômato G_4, representado na figura 4.13, com $\Sigma_o = \{a, b, c, d\}$, $\Sigma_{uo} = \{\sigma_f\}$ e $\Sigma_{so} = \{ab, ba\}$. A nomeclatura de caminho de funcionamento normal e de falha será utilizada para representar os dois caminhos possíveis a partir do estado inicial 0. No autômato G_4, os eventos a e b, pertencentes às sequências de eventos do conjunto Σ_{so}, estão presentes tanto no caminho de comportamento normal do sistema, quanto no caminho de comportamento de falha. Entretanto, no caminho de falha essa sequência ocorre na ordem inversa do caminho normal e, portanto, ambas as sequências ab e ba devem ser incluídas em Σ_{so}. Por hipótese, foi considerada a possibilidade de observação simultânea tanto de ab, quanto de ba, dado que se ba não tiver essa possibilidade, o caso 4 se reduz ao caso 2.

A linguagem gerada pelo autômato G_4 é diagnosticável em relação à falha σ_f e, analisando G_4, nota-se que, caso a observação do evento b, a partir do estado inicial, seja realizada, é possível indicar a ocorrência da falha σ_f. Dessa forma, o diagnosticador de falhas de G_4 é ilustrado na figura 4.14. O diagnosticador de G_4 possui ambos os eventos que constituem as sequências de Σ_{so} ativos em seu estado inicial. O comportamento esperado do diagnosticador é, caso ocorra a sequência ab, evoluir até o estado $3N$, passando pelo estado $1N$. Na ocorrência de ba, em um mesmo ciclo de varredura, a evolução será do estado inicial para o estado $4F$ e, em seguida, para o estado $5F$ e o diagnosticador deverá indicar a ocorrência de σ_f.

É importante ressaltar que, em uma implementação prática, o CLP não é capaz

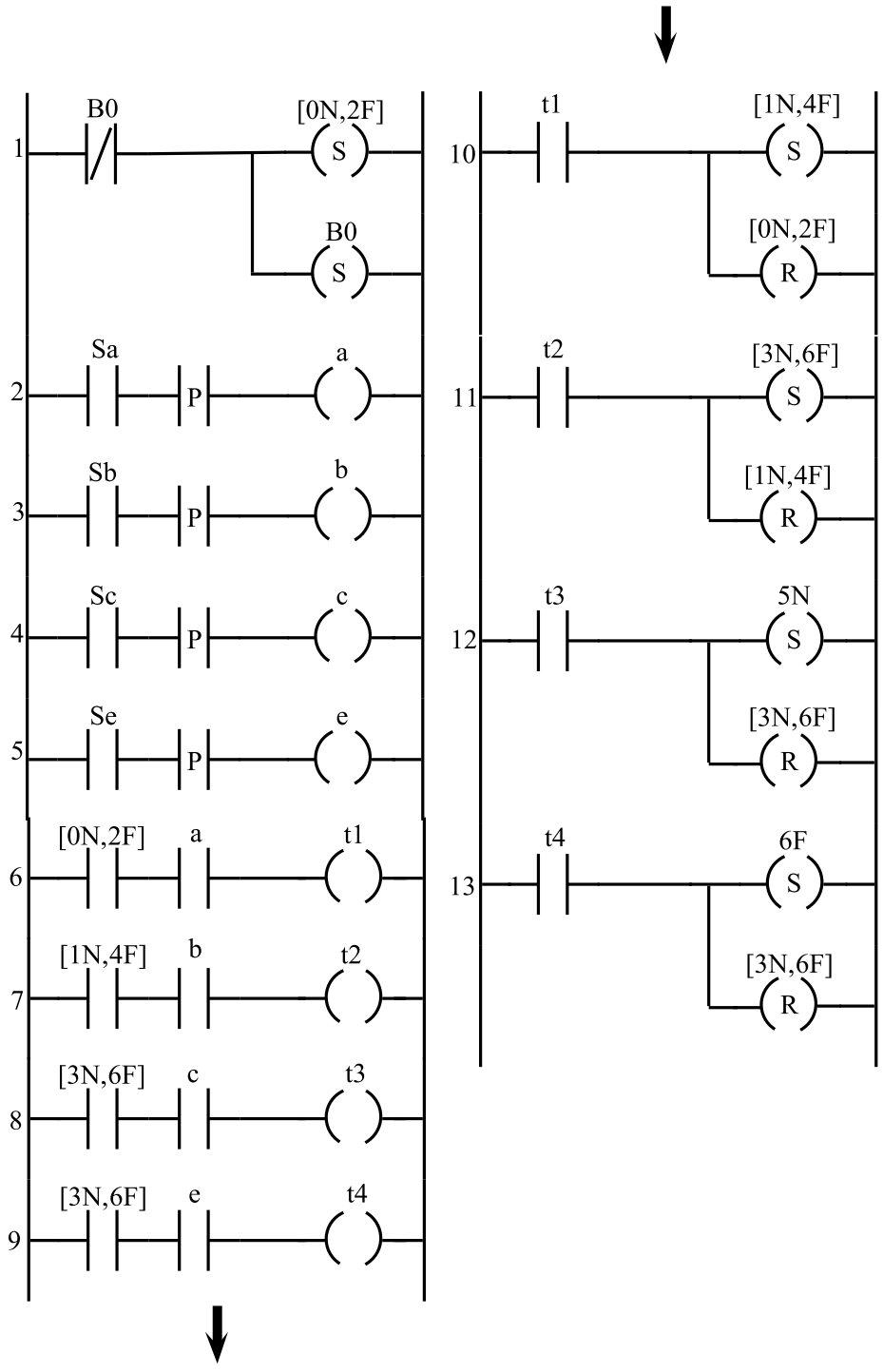

Figura 4.12: Diagrama ladder utilizando o método MB para G_3.

de distinguir a ocorrência das sequências ab ou ba se os eventos a e b forem observados em um único ciclo de varredura. O CLP conseguirá identificar apenas a ocorrência de a e b e perderá a informação da ordem dos eventos. Esse fato tem consequências no diagnóstico correto da falha σ_f para o caso 4.

Na figura 4.15 estão representadas duas implementações do diagnosticador de

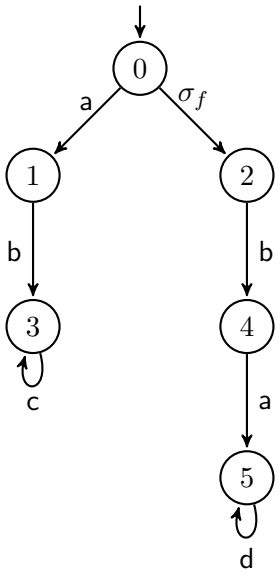

Figura 4.13: Autômato G_4.

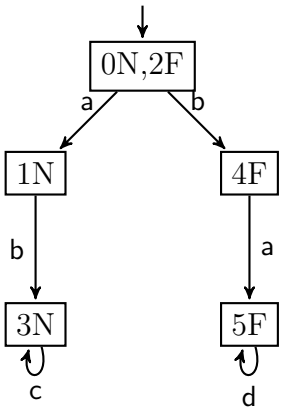

Figura 4.14: Diagnosticador de falhas de G_4.

falhas do autômato G_4, utilizando o método FH. A diferença entre as figuras 4.15(a) e 4.15(b) consiste na troca das linhas 2 e 3.

Na implementação da figura 4.15(a) é possível constatar que, caso a sequência ab seja observada em um mesmo ciclo de varredura, o diagnosticador terá a evolução da linha 2 realizada, dado que o estado $[0N, 2F]$ e o evento a estão ativos. Em seguida, com o contato referente ao estado $1N$ fechado na linha 4, o diagnosticador terá sua evolução para o estado $3N$ realizada. Esse comportamento é a evolução correta do sistema para a observação de ab.

Entretanto, com a observação de ba, as mesmas linhas do diagrama ladder seriam ativadas e a evolução do sistema se daria da mesma maneira. Ou seja, o diagnosti-

cador que deveria ter a sua evolução até o estado $5F$ e indicar a ocorrência da falha, terminaria no estado $3N$, sem indicar nenhuma falha.

Já na implementação da figura 4.15(b), o caminho que o diagnosticador iria escolher independentemente da ocorrência de ab ou ba seria o caminho de falha. Ou seja, independente da ordem de observação dos eventos, a linha 2 e, em seguida a linha 4, do diagrama da figura 4.15(b) teriam sua evolução realizada. Portanto, mesmo que a sequência ab seja observada, em um único ciclo de varredura, o diagnosticador irá sempre indicar que a falha não-observável ocorreu e terminar o ciclo de varredura no estado $5F$, o que é um erro grave. Esse comportamento do diagnosticador de sempre escolher um caminho preferencial nas duas implementações é devido à perda de informação da ordem de ocorrência dos eventos, no caso de observação no mesmo ciclo de varredura.

A figura 4.16 exemplifica a implementação do diagnosticador do autômato G_4 utilizando o método MB. Nessa implementação ambas as linhas 4 e 5 teriam todas as suas condições satisfeitas, no caso de observação simultânea de eventos. O evento a seria responsável por executar a transição t_1, a partir do estado inicial e o evento b executaria a transição t_2. Ao final do ciclo de varredura o diagnosticador iria se encontrar no estado $1N$ e no também $4F$ simultaneamente, indicando erroneamente a ocorrência de σ_f e não teria capacidade de sair desse estado de erro.

4.1.5 Caso 5

Seja o autômato G_5, representado na figura 4.17, com $\Sigma_o = \{a, b, c\}$, $\Sigma_{uo} = \{\sigma_f\}$ e $\Sigma_{so} = \{ab, ba\}$. A nomeclatura de caminho de funcionamento normal e de falha será utilizada para representar os dois caminhos possíveis a partir do estado inicial 0. No autômato G_5, os eventos a e b, pertencentes às sequências de eventos do conjunto Σ_{so}, estão presentes tanto no caminho de comportamento normal do sistema, quanto no caminho de comportamento de falha. No caminho de falha, essa sequência ocorre na ordem inversa do caminho normal e, portanto, ambas as sequências ab e ba devem ser incluídas em Σ_{so}. Entretanto, a projeção P_o dos caminhos de comportamento

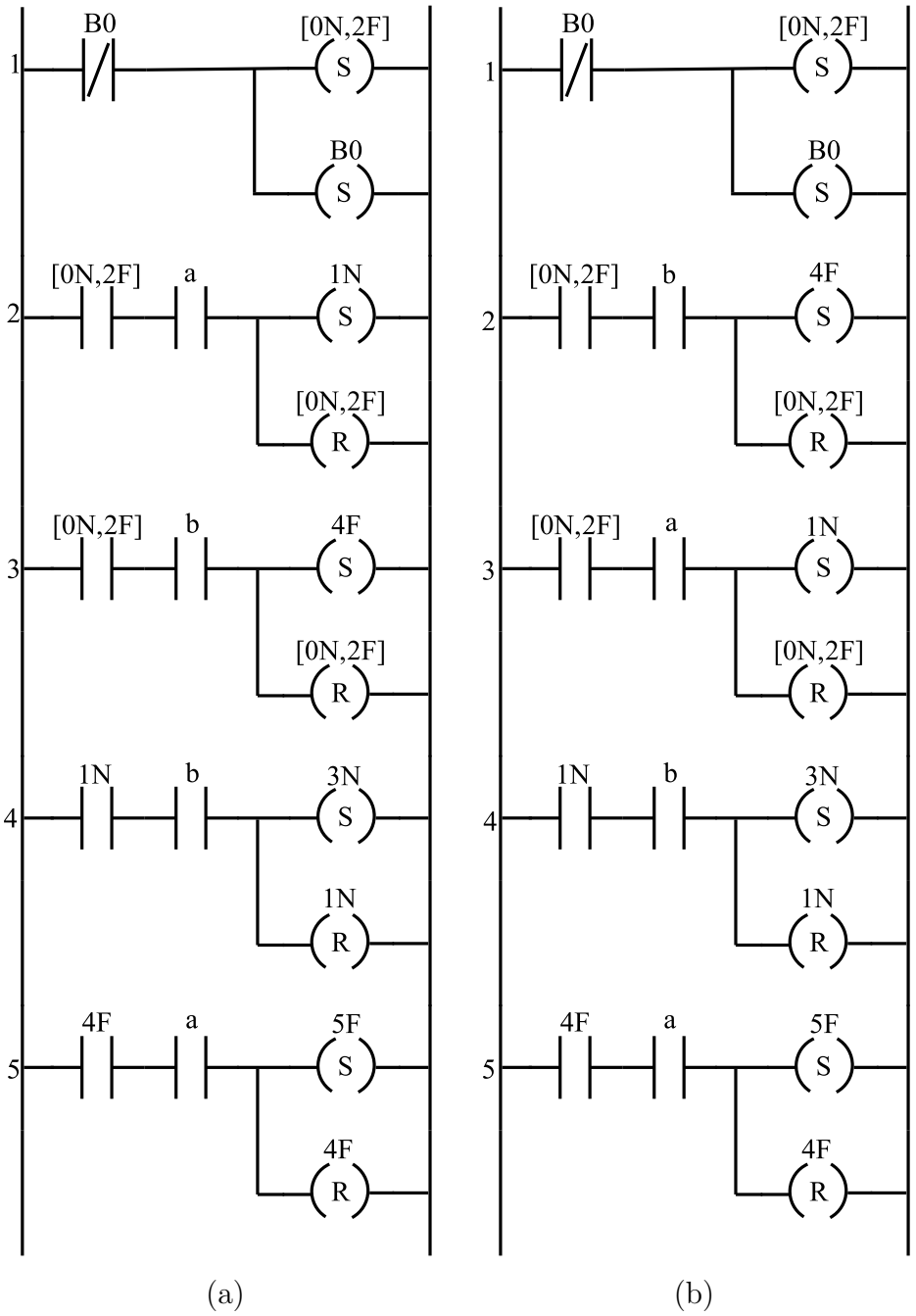

Figura 4.15: Diagramas ladder utilizando o método FH para G_4.

normal e de falha são diferenciadas, apenas, pela inversão da ordem dos eventos das sequências pertencentes a Σ_{so}.

A linguagem gerada pelo autômato G_5 é diagnosticável em relação à falha σ_f e, analisando G_5, nota-se que, caso a observação do evento b, a partir do estado inicial, seja realizada, é possível indicar a ocorrência da falha σ_f. Dessa forma, o diagnosticador de falhas de G_5 é ilustrado na figura 4.18. O diagnosticador apresenta

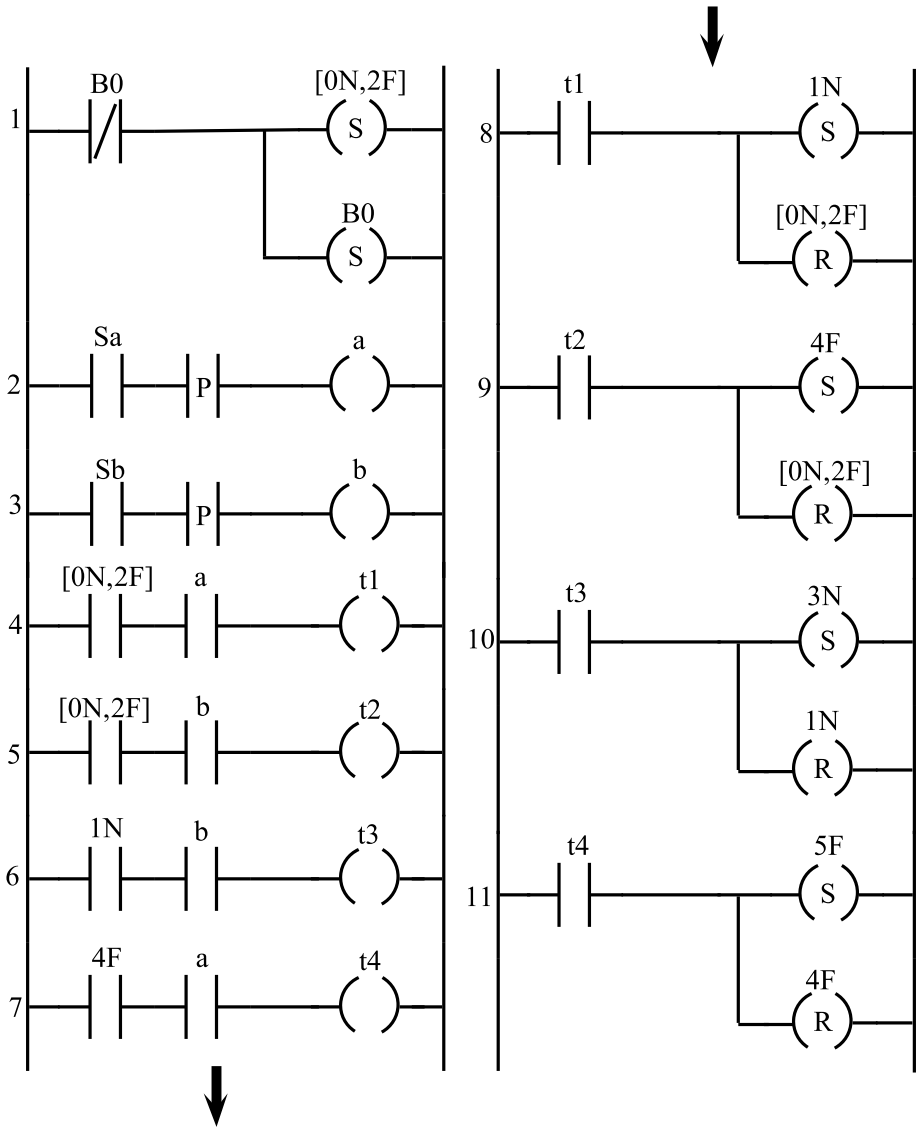

Figura 4.16: Diagramas ladder utilizando o método MB para G_4.

ambos os eventos a e b ativos no estado inicial $[0N, 2F]$ e caso ocorra a observação simultânea desses eventos, o diagnosticador pode evoluir de maneira equivocada dependendo de sua implementação.

Na figura 4.19 estão representadas duas implementações do diagnosticador de falhas do autômato G_45, utilizando o método FH e na figura [**?**] está representada a implementação do diagnosticador do autômato G_5 utilizando o método MB. Os diagramas dos diagnosticadores do caso 5 são idênticos aos diagramas apresentados no caso 4, pois a única diferença entre os autômatos G_4 e G_5 são os ciclos de eventos d e c, no final do caminho de falha de cada autômato. Portanto, não há a

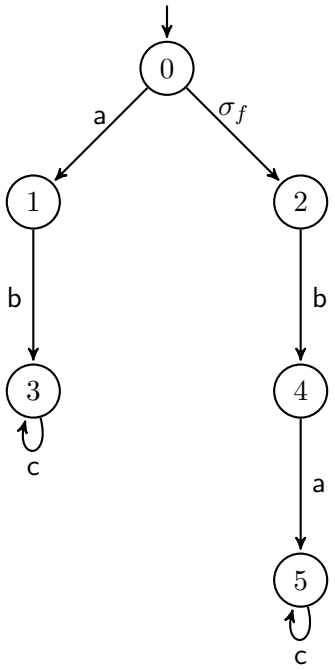

Figura 4.17: Autômato G_5.

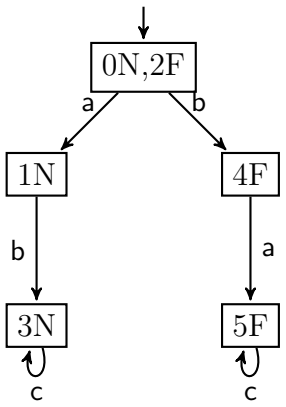

Figura 4.18: Diagnosticador de falhas de G_5.

necessidade de se representar essas transições de estados no diagrama ladder, pois elas não realizam uma mudança no estado no diagnosticador

4.1.6 Caso 6

Seja o autômato G_6, representado na figura 4.21, com $\Sigma_o = \{a, b, c\}$, $\Sigma_{uo} = \{\sigma_f\}$ e $\Sigma_{so} = \{ab, ba\}$. A nomeclatura de caminho de funcionamento normal e de falha será utilizada para representar os dois caminhos possíveis a partir do estado inicial 0. No autômato G_6, os eventos a e b, pertencentes às sequências de eventos do conjunto

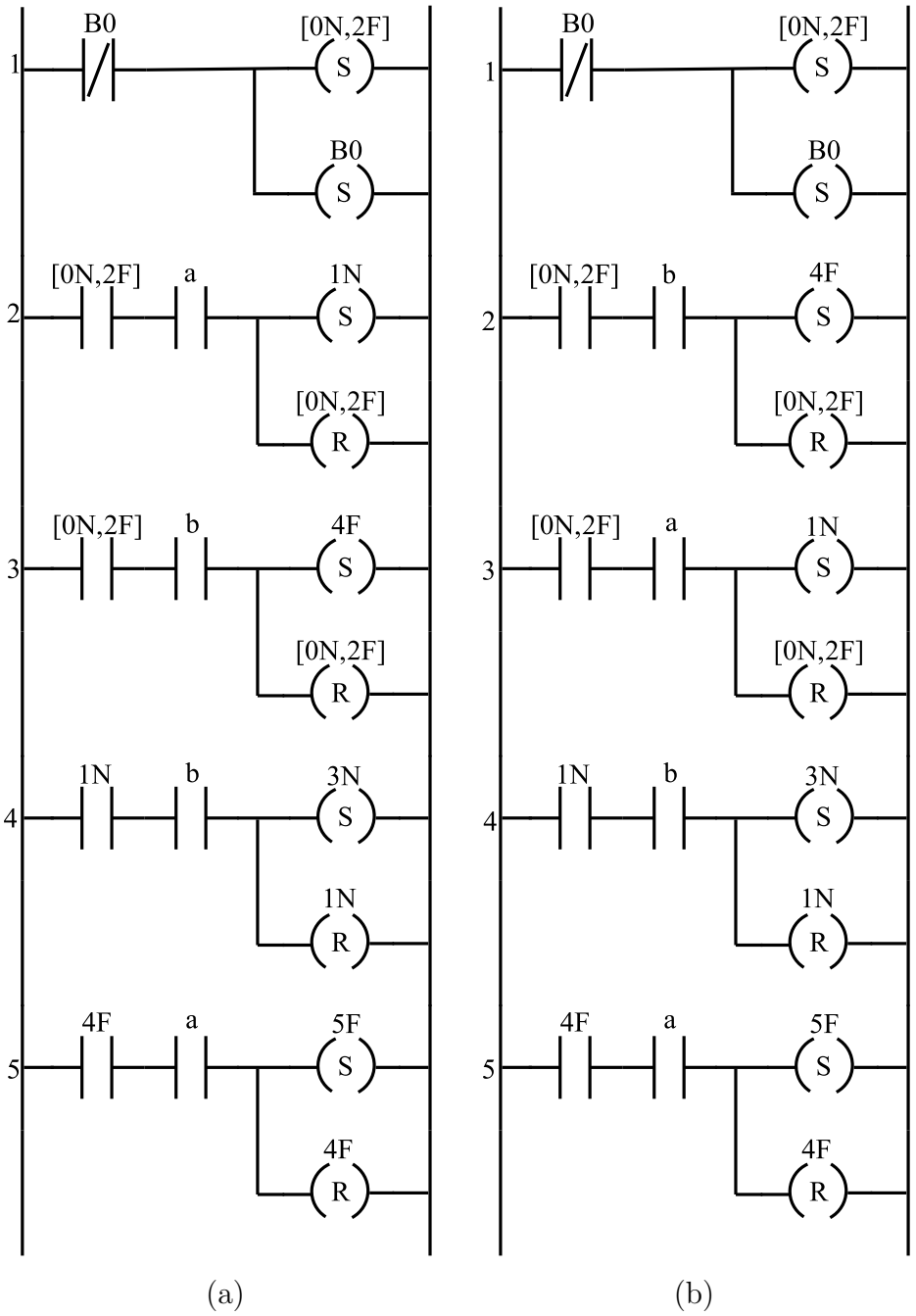

Figura 4.19: Diagramas ladder utilizando o método FH para G_5.

Σ_{so}, estão presentes tanto no caminho de comportamento normal do sistema, quanto no caminho de comportamento de falha. No caminho de falha, essa sequência ocorre na ordem inversa do caminho normal e, portanto, ambas as sequências ab e ba devem ser incluídas em Σ_{so}. No autômato G_6, a falha σ_f foi deslocada e não ocorre mais no estado inicial, mas sim, no estado 4.

A linguagem gerada pelo autômato G_6 é diagnosticável em relação à falha σ_f e,

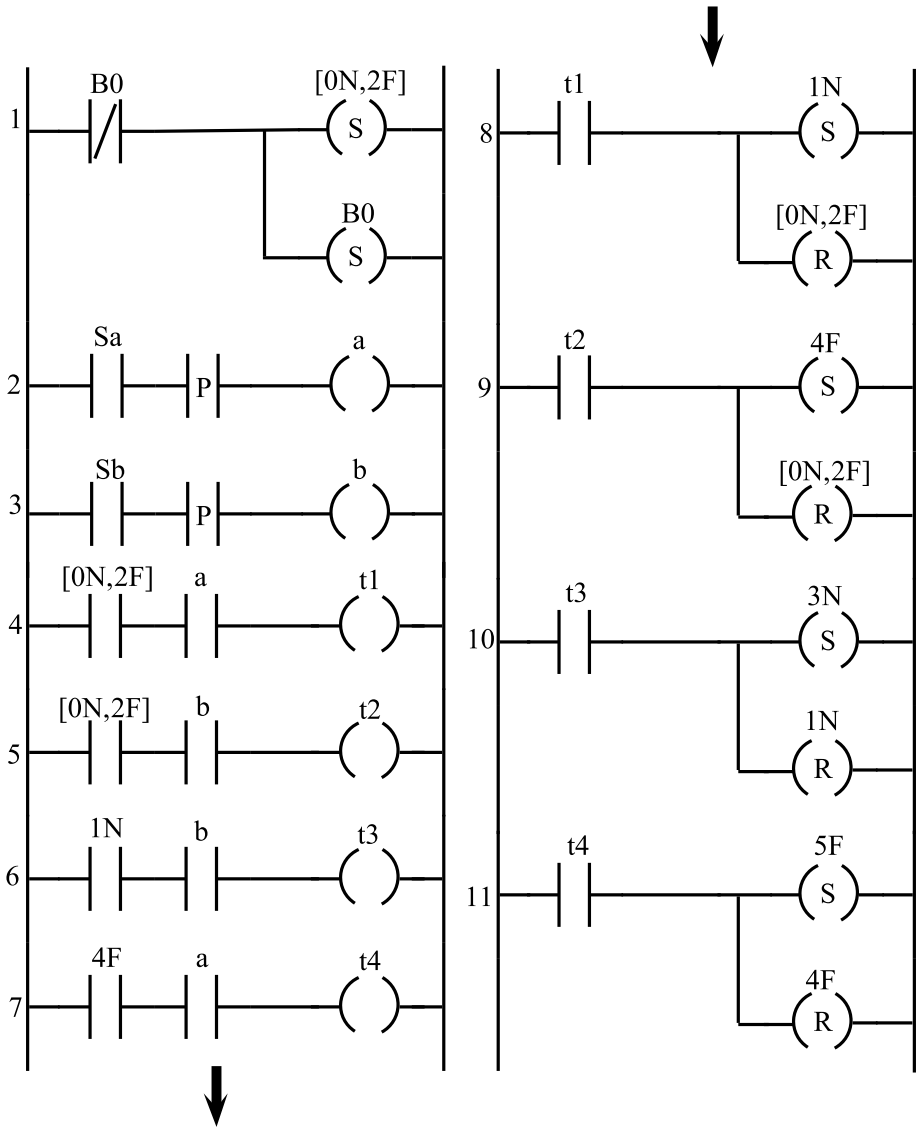

Figura 4.20: Diagramas ladder utilizando o método MB para G_5.

analisando G_6, nota-se que, caso a observação da sequência *bad*, a partir do estado inicial, seja realizada, é possível indicar a ocorrência da falha σ_f. Dessa forma, o diagnosticador de falhas de G_6 é ilustrado na figura 4.22. O diagnosticador apresenta, mais uma vez, ambos os eventos com possibilidade de observação simultânea ativos no estado inicial, o que já foi constatado que gera problemas no diagnóstico de falhas com implementação em diagrama ladder. O comportamento esperado do diagnosticador perante a ocorrência da sequência *ab* é evoluir do estado inicial $0N$ para o estado $1N$ e, em seguida, para o estado $3N$ em um mesmo ciclo de varredura do CLP. Já o comportamento esperado frente a ocorrência da sequência *ba* é a partir

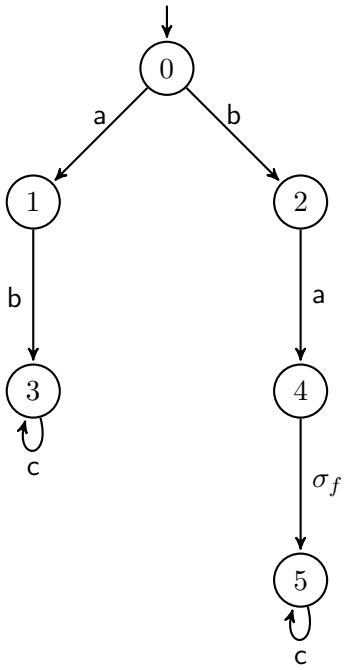

Figura 4.21: Autômato G_6.

do estado inicial, evoluir até o estado $(4N, 5F)$, passando pelo estado $2N$, em um único clico.

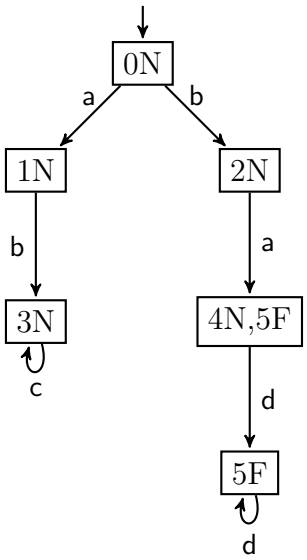

Figura 4.22: Diagnosticador de falhas de G_6.

Na figura 4.1.6 estão representadas duas implementações do diagnosticador de falhas do autômato G_6, utilizando o método FH. A diferença entre as figuras 4.1.6(a) e 4.1.6(b) consiste na troca das linhas 2 e 3.

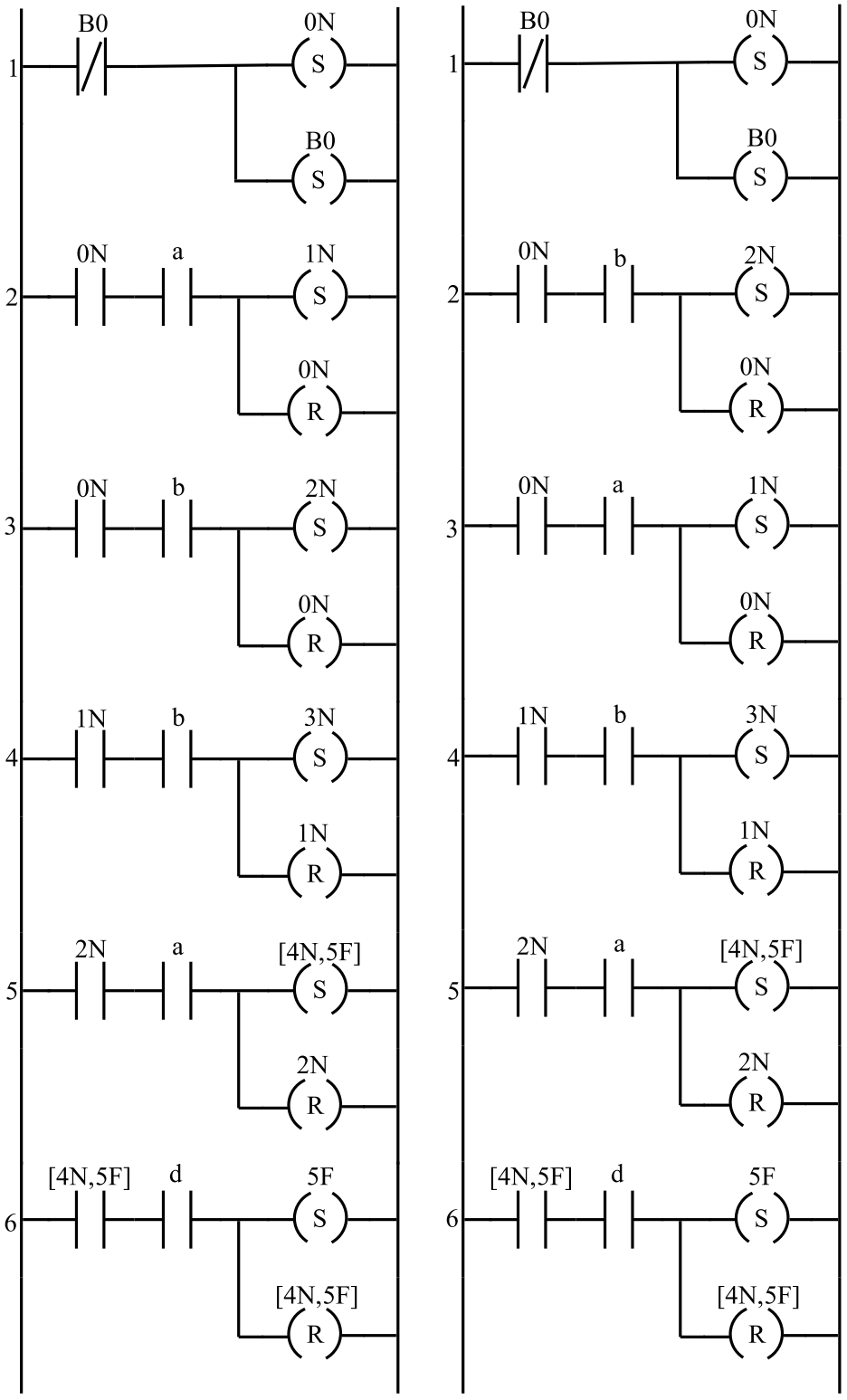

Figura 4.23: Diagramas ladder utilizando o método FH para G_6.

Na implementação da figura 4.1.6(a) é possível constatar que, caso a sequência *ab* seja observada em um mesmo ciclo de varredura, o diagnosticador terá a evolução

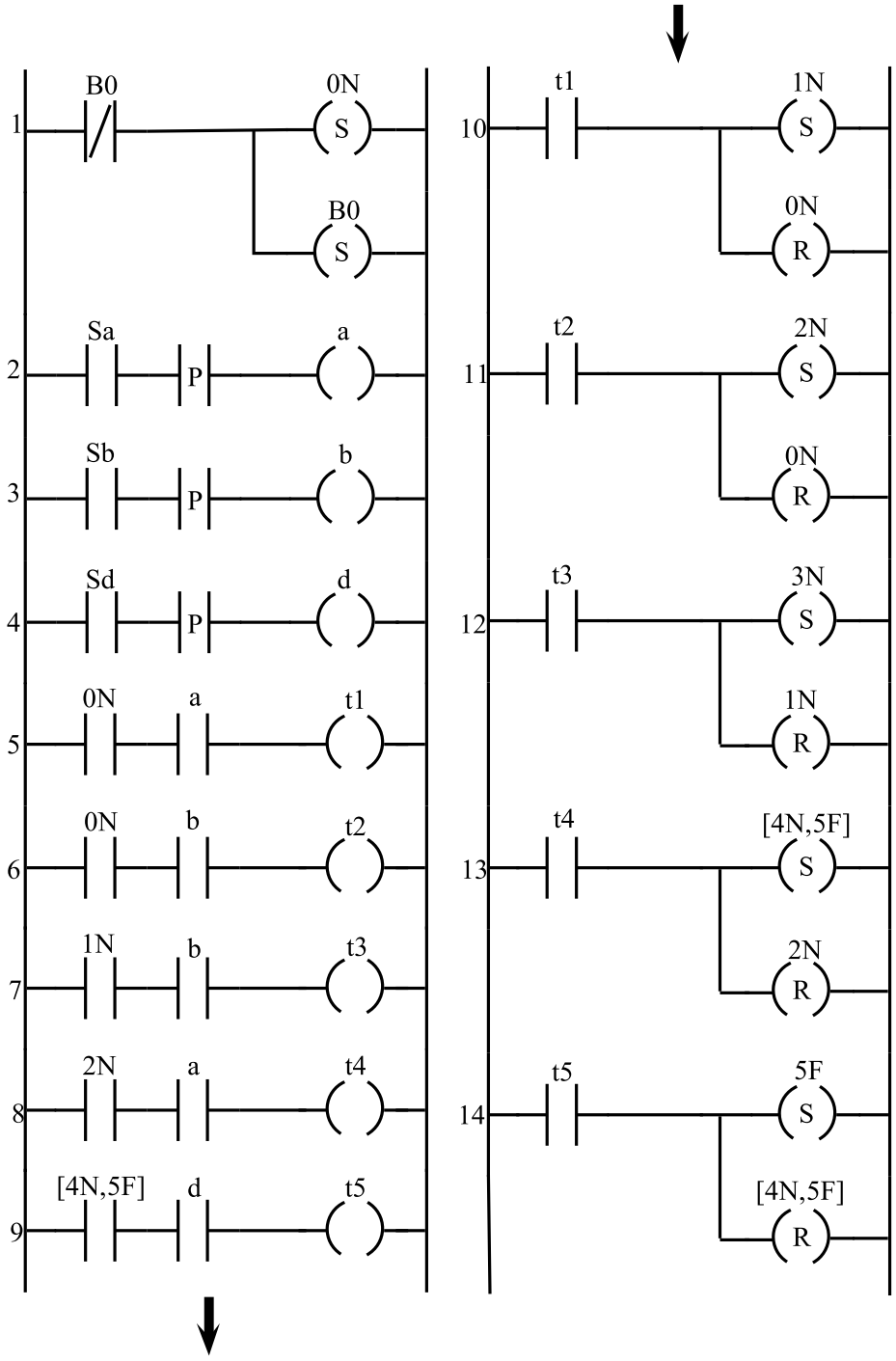

Figura 4.24: Diagrama ladder utilizando o método MB para G_6.

da linha 2 realizada, dado que o estado $0N$ e o evento a estão ativos. Em seguida, a transição de estados da linha 4, que leva o diagnosticador do estado $1N$ para o estado $3N$, será executada, visto que após a execução da linha 2 do diagrama, tanto o contato referente ao estado $1N$, quanto o contato referente ao evento b estariam fechados no mesmo ciclo de varredura. Caso a sequência observada no mesmo ciclo de

varredura seja ba, o diagnosticador realizará a mesma evolução apresentada frente a sequência ab. Portanto, independente da ordem dos eventos, o diagnosticador sempre seguirá o caminho de comportamento normal do sistema, o que caracteriza um erro grave.

Com a troca de linhas realizada na figura 4.1.6(b), o sistema iria evoluir, na observação simultânea de eventos, do estado inicial $0N$ para o estado $2N$ e em seguida para o estado $(4N, 5F)$. Mais uma vez a ordem dos eventos é indiferente ao CLP e o diagnosticador apresenta um caminho preferencial de operação, porém oposto ao apresentado pela figura 4.1.6(a).

A figura 4.24 exemplifica a implementação do diagnosticador do autômato G_6 utilizando o método MB. Nessa implementação ambas as linhas 4 e 5 teriam todas as suas condições satisfeitas, no caso de observação simultânea de eventos. O evento a seria responsável por executar a transição t_1, a partir do estado inicial e o evento b executaria a transição t_2. Ao final do ciclo de varredura, o diagnosticador iria se encontrar no estado $1N$ e no estado $2N$ simultaneamente e não teria capacidade de sair desse estado de erro.

4.2 Método para a obtenção de um diagnosticador robusto à observação simultânea de eventos

A ocorrência de observação simultânea de eventos consecutivos em um mesmo ciclo de varredura mostrou ter grande influência no diagnóstico de falhas em sistemas a eventos discretos, sobretudo em suas implementações práticas. Existe, então, uma necessidade de se formalizar um método para evitar os efeitos indesejáveis da ocorrência desse fenômeno. Será apresentada, neste trabalho, uma solução para evitar os efeitos indesejados da observação simultânea de eventos em dois passos: (i) introdução de um evento que modele a observação de ambos os eventos simultaneamente; (ii) alteração do módulo de identificação de ocorrência de eventos apresen-

tado por [16], para tornar o diagnosticador capaz de distinguir entre a observação individual dos eventos e a observação simultânea dos mesmos.

Um dos motivos para que a observação simultânea de eventos em um único ciclo de varredura tenha impactos no diagnóstico de falhas é que ela não foi modelada na hora da concepção e projeto do sistema. Assim, o primeiro passo na resolução dessa questão, é identificar os pontos onde é possível haver a observação mais de um evento em um mesmo ciclo de varredura e assim introduzir um evento que modele essa ocorrência. A figura 4.25 ilustra a introdução desse evento. O novo evento será denotado por π e representa a ocorrência da sequência de eventos consecutivos que podem ser observados simultaneamente. Dependendo da configuração do sistema, ele pode denotar também a ocorrência da sequência de eventos com observação simultânea em ordem invertida. O autômato resultante será denominado G_π e as suas características mais específicas serão abordadas no estudo de casos nesta seção.

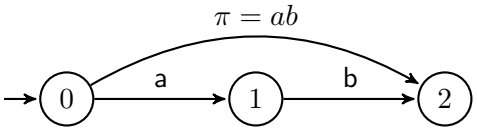

Figura 4.25: Introdução da modelagem da observação simultânea.

Seja um autômato G, com $\Sigma_{uo} = \{\sigma_{u_1}\}$, $\Sigma_{so} = \{ab, ba, a\sigma_u b, cde, dec, dce\}$. Neste exemplo, seriam criados dois novos eventos π para esse autômato. Primeiro um evento $\pi_1 = ab = ba = a\sigma_u b$ e um segundo evento $\pi_2 = cde = dec = dce$. No exemplo acima, $a\sigma_u b$ foi modelada por π_1, enquanto $b\sigma_u a$ foi considerada, por hipótese, sem a possibilidade de ser observada em um único ciclo. O evento π_2 ilustra uma sequência com mais de dois elementos sendo representada por um evento que modela a observação simultânea.

É importante ressaltar que a introdução do evento π não é suficiente para retirar todos os efeitos causados pela observação simultânea de eventos. O fato do evento π representar a observação dos eventos simultaneamente, não exclui a habilitação desses eventos durante a execução do diagrama ladder. Pelo contrário, ele gera uma ambiguidade adicional, já que agora estão habilitados tanto os eventos

individualmente, quanto o evento π.

Para cancelar tal ambiguidade, é proposto então o segundo passo da solução, uma formalização adicional durante a identificação de eventos no método proposto por [16]. A figura 4.26 demonstra a nova inicialização de eventos para o caso de haver possibilidade de observação simultânea dos eventos a e b. É necessário distinguir entre a observação de apenas um evento ou dois eventos simultaneamente. Isso é obtido desabilitando-se os eventos individuais na observação simultânea de ambos.

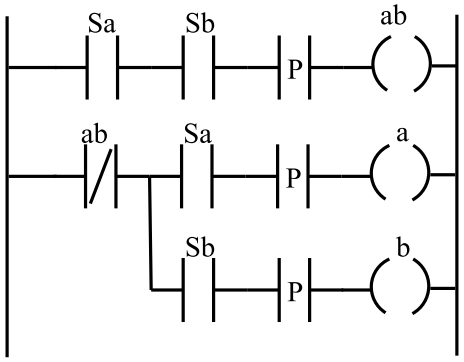

Figura 4.26: Módulo de identificação de eventos alterado.

4.2.1 Caso 1

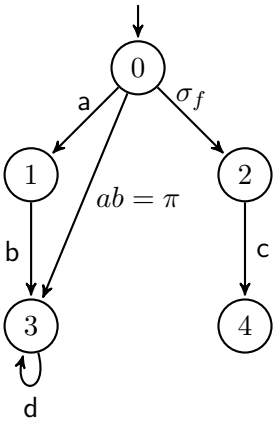

Figura 4.27: Autômato G_{π_1}.

Seja G_{π_1}, representado na figura 4.27, com $\Sigma_o = \{a, b, c, \pi\}$, $\Sigma_{uo} = \{\sigma_f\}$, o autômato resultante da metodologia proposta para se evitar os efeitos da observação simultânea de eventos consecutivos. O evento π representa a observação simultânea

da sequência $ab \in \Sigma_{so}$ e está presente somente no caminho de comportamento normal de G_{π_1}. A introdução do evento π não acarretou na perda da diagnosticabilidade da linguagem gerada por G_{π_1} em relação à σ_f, dado que o evento π não altera nenhuma sequência s com $\sigma_f \in s$. O diagnosticador de G_{π_1} pode ser observado na figura 4.28.

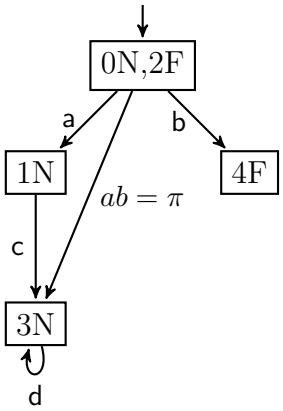

Figura 4.28: Diagnosticador de falhas de G_{π_1}.

4.2.2 Caso 2

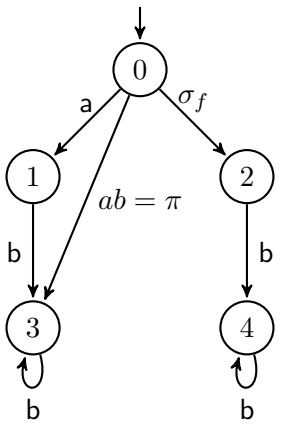

Figura 4.29: Autômato G_{π_2}.

Seja G_{π_2}, representado na figura 4.29, com $\Sigma_o = \{a, b, \pi\}$, $\Sigma_{uo} = \{\sigma_f\}$, o autômato resultante da metodologia proposta para se evitar os efeitos da observação simultânea de eventos consecutivos. O evento π representa a observação simultânea da sequência $ab \in \Sigma_{so}$ e está presente somente no caminho de comportamento normal de G_{π_2}. A introdução do evento π não acarretou na perda da diagnosticabilidade da

linguagem gerada por G_{π_2} em relação à σ_f, dado que o evento π não altera nenhuma sequência s com $\sigma_f \in s$. O diagnosticador de G_{π_2} pode ser observado na figura 4.30.

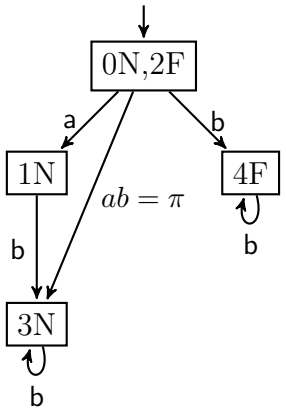

Figura 4.30: Diagnosticador de falhas de G_{π_2}.

4.2.3 Caso 3

Seja G_{π_3}, representado na figura 4.31, com $\Sigma_o = \{a, b, c, d, e, \pi\}$, $\Sigma_{uo} = \{\sigma_f\}$, o autômato resultante da metodologia proposta para se evitar os efeitos da observação simultânea de eventos consecutivos. O evento π representa a observação simultânea da sequência $ab \in \Sigma_{so}$ e está presente tanto no caminho de comportamento normal de G_{π_3}, quanto no seu comportamento de falha. A introdução do evento π não acarretou na perda da diagnosticabilidade da linguagem gerada por G_{π_3} em relação à σ_f, dado que o evento π representa apenas ab e por isso a perda de informação da ordem dos eventos não observáveis pelo CLP não altera as projeções dos eventos observáveis tanto no caminho de comportamento normal, quanto no caminho de falha. O diagnosticador de G_{π_3} pode ser observado na figura 4.32.

4.2.4 Caso 4

Seja G_{π_4}, representado na figura 4.33, com $\Sigma_o = \{a, b, c, d, \pi\}$, $\Sigma_{uo} = \{\sigma_f\}$, o autômato resultante da metodologia proposta para se evitar os efeitos da observação simultânea de eventos consecutivos. O evento π representa a observação simultânea das sequências ab e $ba \in \Sigma_{so}$ e ab está presente no caminho de comportamento

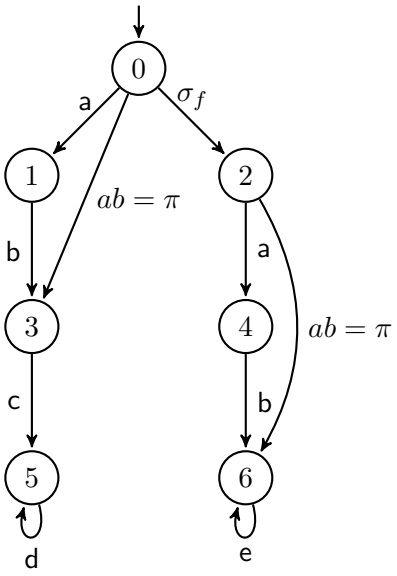

Figura 4.31: Autômato G_{π_3}.

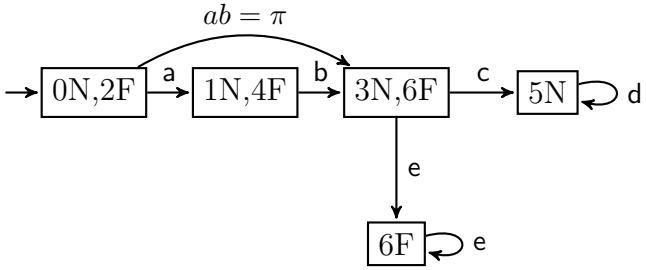

Figura 4.32: Diagnosticador de falhas de G_{π_3}.

normal de G_{π_4}, enquanto *ba* está presente no seu comportamento de falha. A introdução do evento π não acarretou na perda da diagnosticabilidade da linguagem gerada por G_{π_4} em relação à σ_f, uma vez que as projeções observáveis do caminho de comportamento normal e do caminho de falha não diferem entre si apenas pelas sequências representadas por π. O diagnosticador de G_{π_4} pode ser observado na figura 4.34.

4.2.5 Caso 5

Seja G_{π_5}, representado na figura 4.35, com $\Sigma_o = \{a, b, c, \pi\}$, $\Sigma_{uo} = \{\sigma_f\}$, o autômato resultante da metodologia proposta para se evitar os efeitos da observação simultânea de eventos consecutivos. O evento π representa a observação simultânea das sequências ab e $ba \in \Sigma_{so}$ e ab está presente no caminho de comportamento nor-

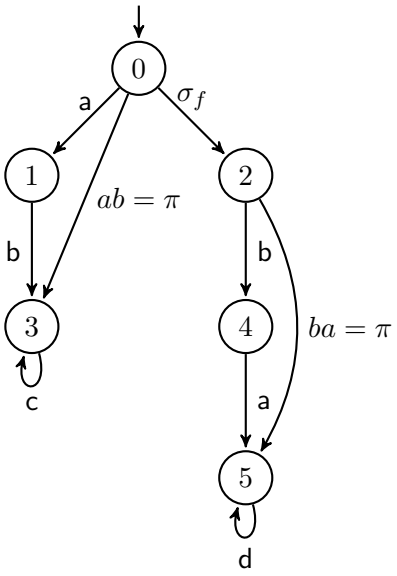

Figura 4.33: Autômato G_{π_4}.

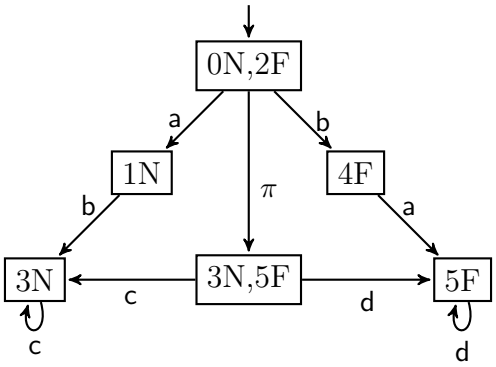

Figura 4.34: Diagnosticador de falhas de G_{π_4}.

mal de G_{π_5}, enquanto *ba* está presente no seu comportamento de falha. A introdução do evento π acarretou na perda da diagnosticabilidade da linguagem gerada por G_{π_5} em relação à σ_f, pois as projeções observáveis do caminho de comportamento normal e do caminho de falha diferem entre si apenas pelas sequências representadas por π. O diagnosticador de G_{π_5} pode ser observado na figura 4.36 e verifica-se que a partir da observação do evento π, o diagnosticador não será capaz de sair do estado de incerteza $3N, 5F$.

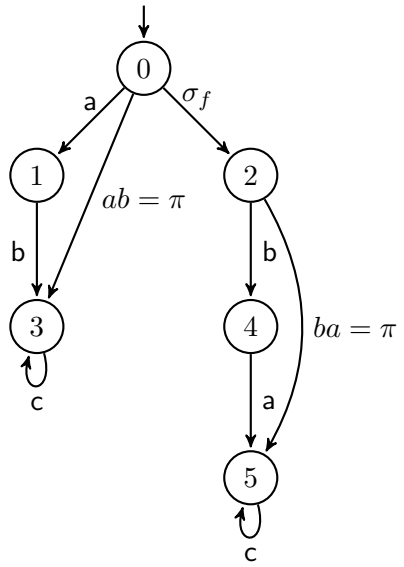

Figura 4.35: Autômato G_{π_5}.

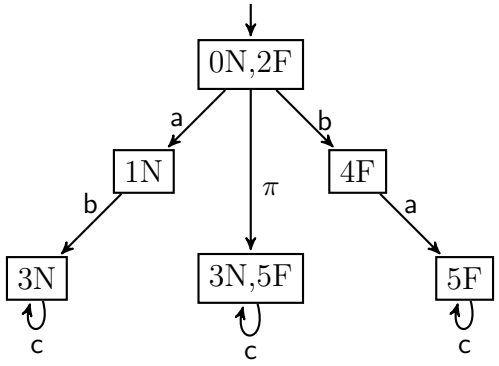

Figura 4.36: Diagnosticador de falhas de G_{π_5}.

4.2.6 Caso 6

Seja G_{π_6}, representado na figura 4.37, com $\Sigma_o = \{a, b, c, d, \pi\}$, $\Sigma_{uo} = \{\sigma_f\}$, o autômato resultante da metodologia proposta para se evitar os efeitos da observação simultânea de eventos consecutivos. O evento π representa a observação simultânea das sequências ab e $ba \in \Sigma_{so}$ e ab está presente no caminho de comportamento normal de G_{π_6}, enquanto ba está presente no seu comportamento de falha. A introdução do evento π gerou um autômato não-determinístico, porém não acarretou na perda da diagnosticabilidade da linguagem gerada por G_{π_6} em relação à σ_f, dado que as projeções observáveis do caminho de comportamento normal e do caminho de falha não diferem entre si apenas pelas sequências representadas por π. O diagnosticador

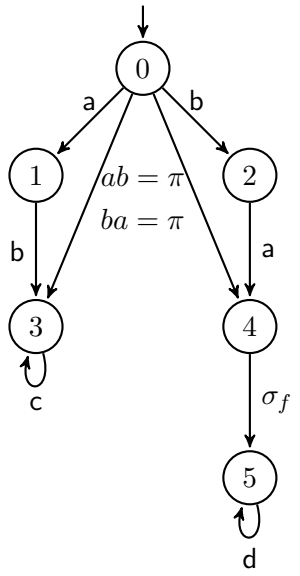

Figura 4.37: Autômato G_{π_6}.

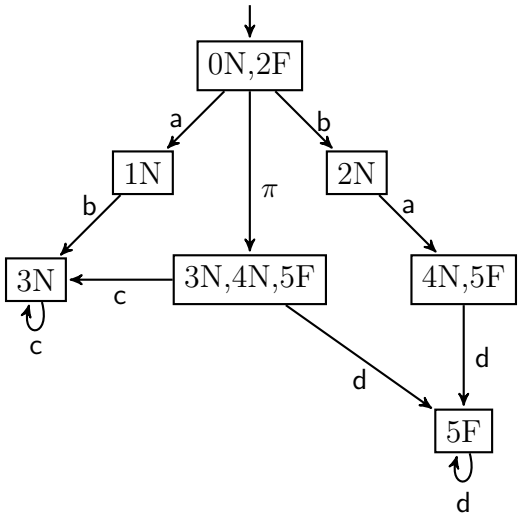

Figura 4.38: Diagnosticador de falhas de G_{π_6}.

de G_{π_6} pode ser observado na figura 4.38.

4.2.7 Diagramas Ladder

Com a introdução do evento π e da formalização adicional no módulo de identificação de ocorrências de eventos o diagnosticador consegue identificar a observação simultânea de eventos e, ainda, não encontra nenhuma ambiguidade entre a sequência observada simultaneamente e os eventos constituintes dessa sequência individualmente. Para ilustrar utilizaremos o diagnosticador de G_{π_4}, ilustrado pela figura

4.33, e será verificado que o método é eficaz.

Na figura 4.39, é possível observar a implementação em diagrama ladder proposta para o autômato G_{π_4} e caso o não ocorra a observação simultânea de a e b o sistema irá se comportar normalmente, pois a bobina ab não será energizada e a presença do contato normalmente fechado associado à ab na linha responsável pela identificação dos eventos a e b (linha 3) não irá interferir na habilitação desses eventos. Porém, se ambos os eventos ocorrerem no mesmo ciclo de varredura, a bobina ab será energizada e o sistema habilitará o evento ab. Os eventos a e b não serão individualmente habilitados devido ao contato normalmente fechado associado à ab. Portanto o diagnosticador não irá se confundir em qual decisão tomar.

Como os eventos foram habilitados de maneira correta, caso ocorra o evento a, somente a evolução de $(0N, 2F)$ para $1N$ será executada. Enquanto que com a ocorrência do evento b, o diagnosticador iria executar a transição t_2 e ir para o estado $4F$. E, finalmente, com a observação simultânea de ambos os eventos, o diagnosticador iria do estado inicial para o estado $3N, 5F$, como modelado.

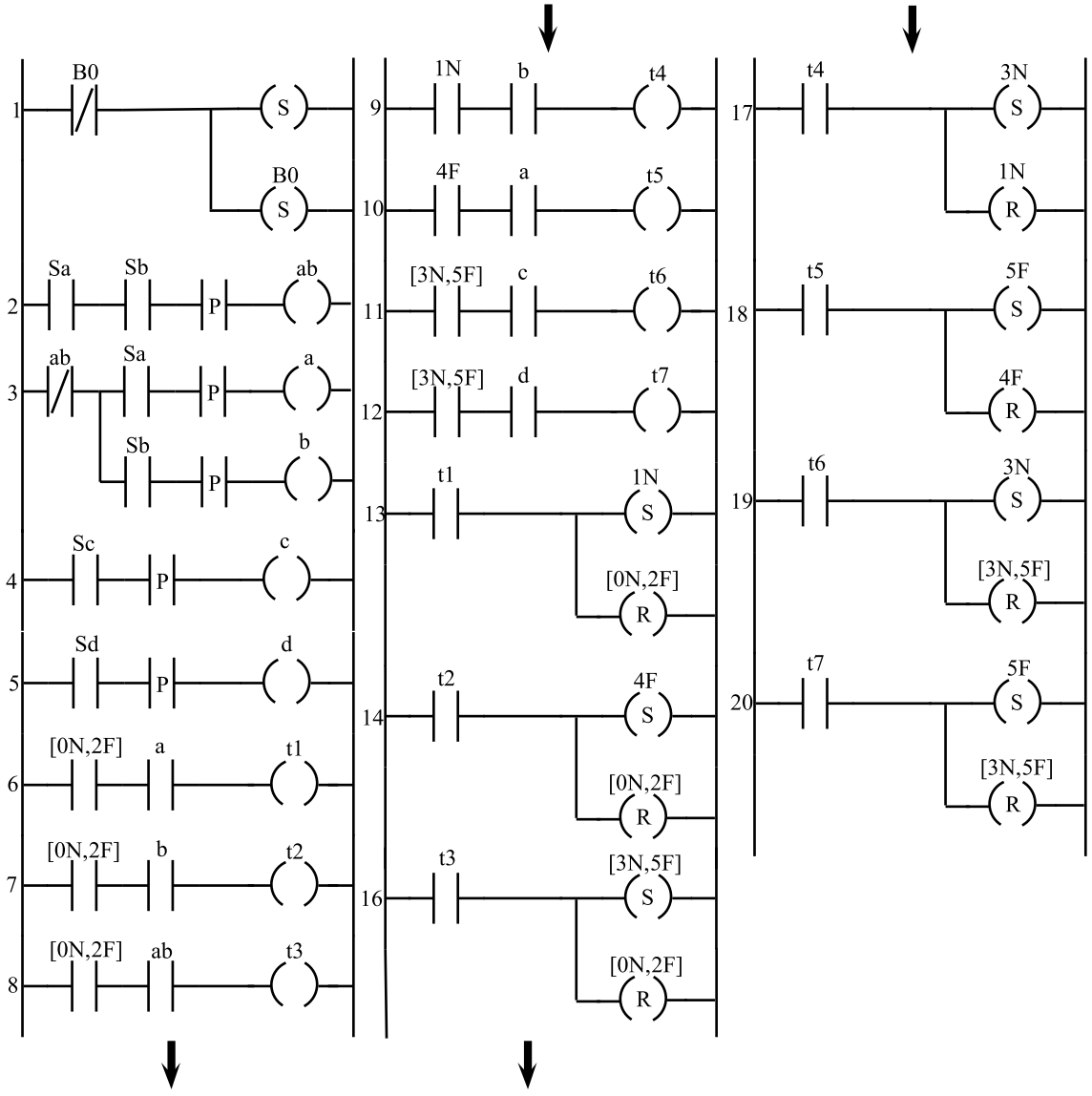

Figura 4.39: Diagrama ladder para o diagnosticador de G_{π_4} com a metodologia proposta.

Capítulo 5

Conclusão

Neste trabalho foi realizado um estudo sobre a observação simultânea de eventos consecutivos em um mesmo ciclo de varredura e sua influência na construção de diagnosticadores. O fenômeno foi analisado e separado em seus casos base e, com isso, averiguado quais eram os impactos que a observação simultânea de eventos tem sobre um sistema real. Foi constatado que a implementação prática é sensível ao método utilizado, principalmente à ordenação das linhas, na ocorrência do fenômeno de observação simultânea de eventos consecutivos em um único ciclo de varredura.

O diagrama ladder, por ser o mais utilizado no desenvolvimento de sistemas de controle de SEDs, foi analisado neste trabalho para a implementação de diagnosticadores. Tanto a metodologia proposta em [13, 14], quanto o método proposto por [16] não se mostraram robustas à observação simultânea de eventos. Foi proposto, então, uma formalização adicional ao método apresentado por [16], permitindo a correta identificação dos eventos ocorridos, sua habilitação e também a correção de ambiguidades geradas pela observação simultânea de eventos em um mesmo ciclo de varredura.

A metodologia proposta se mostrou eficaz em praticamente todos os casos apresentados. Porém, caso exista uma liguagem gerada por um autômato G, diagnosticável em relação à σ_f e duas sequências s_1 e $s_2 \in \mathcal{L}(G)$, com $P_o(s_1) \neq P_o(s_2)$, $\sigma_f \in s_1$ e $\sigma_f \notin s_2$, ambas as sequências s_1 e s_2 possuam subsequências que tenham possibilidade de observação simultânea de eventos e, em s_1, a subsquência com essa

possibilidade possui ordem inversa à subsequência presente em s_2.Portanto, devido à incapacidade do CLP de identificar a ordem de ocorrência dos eventos com observação simultânea em s_1 e s_2, tem-se que $P_o(s_1) = P_o(s_2)$ e, assim, a linguagem gerada por G se tornará não diagnosticável em relação à σ_f.

Como trabalhos futuros podemos citar a investigação da robustez de diagnosticadores de falhas com arquitetura descentralizada em relação à observação simultânea de eventos e o desenvolvimento de um método para identificar a possibilidade de observação simultânea de eventos.

Referências Bibliográficas

[1] CASSANDRAS, C. G., LAFORTUNE, S. *Introduction to Discrete Event Systems*. Springer, 2007.

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

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

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

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

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

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

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

[9] WANG, Y., YOO, T.-S., LAFORTUNE, S. "Diagnosis of discrete event systems using decentralized architectures", *Discrete Event Dyn Syst*, v. 17, pp. 233–263, 2007.

[10] QIU, W., KUMAR, R. "Decentralized Failure Diagnosis of Discrete Event Systems", *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND*

CYBERNETICS—PART A: SYSTEMS AND HUMANS,, v. 36, n. 2, March 2006.

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

[12] JOHN, K., TIEGELKAMP, M. *IEC 61131-3: Programming Industrial Automation Sytems*. New York, NY, USA, Springer, 2005.

[13] FABIAN, M., HELLGREN, A. "PLC-based implementation of supervisory control for discrete event systems", *37th IEEE Conference on Decision and Control*.

[14] HELLGREN, A., FABIAN, M., LENNARTSON, B. "On the execution of sequential fucntion charts", *Control Engineering Practice*, v. 13, pp. 1283–1293, 2005.

[15] MOREIRA, M. V., BOTELHO, D. S., BASILIO, J. C. "Ladder Diagram Implementation of Control Interpreted Petri Nets: a State Equation Approach", *4th IFAC Workshop on Discrete-Event System Design*, pp. 95–90, 2009.

[16] MOREIRA, M. V., BASILIO, J. C. "Bridging the gap between design and implementation of discrete event controllers, Aprovado para publicação", *IEEE Transactions on Automation Science and Engineering*, 2013.

[17] UZAM, M., JONES, A. H., AJLOUNI, N. "Conversion of Petri Nets Controllers for Manufacturing Systems into Ladder Logic Diagrams", *IEEE Conference on Emerging Technologies and Factory Automation*, pp. 649–655, 1996.

[18] UZAM, M., JONES, A. H. "Discrete Event Control System Design Using Automation Petri Nets and their Ladder Diagram Implementation", *Int J Adv Manuf Technol*, v. 14, pp. 716–728, 1998.

[19] JONES, A., UZAM, M., AJLOUNI, N. "Design of discrete event control systems for programmable logic controllers using T-timed Petri nets", *IEEE Int. Symp. Computer-Aided Control System Design*, pp. 212–217, 1996.

[20] JIMENEZ, I., LOPEZ, E., RAMIREZ, A. "Synthesis of Ladder diagrams from Petri nets controller models", *IEEE International Symposium on Intelligent Control*, pp. 225–230, 2001.

[21] PENG, S. S., ZHOU, M. C. "Ladder diagram and Petri-net-based discrete-event control design methods", *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, v. 34, pp. 523–531, 2004.

[22] UZAM, M. "A general technique for the PLC-based implementation of RW supervisors with time delay functions", *Int J Adv Manuf Technol*, v. 62, pp. 687–704, 2012.