UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

BENJAMIN GRANDO MOREIRA

CONTROLE TOLERANTE A FALHAS DE SISTEMAS A EVENTOS DISCRETOS USANDO DIAGNOSTICADOR CONTROLADOR ATIVO

JOINVILLE
2021

BENJAMIN GRANDO MOREIRA

CONTROLE TOLERANTE A FALHAS DE SISTEMAS A EVENTOS DISCRETOS USANDO DIAGNOSTICADOR CONTROLADOR ATIVO

Tese submetida ao Curso de Pós-Graduação em Engenharia Elétrica, do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, para a obtenção do Grau de Doutor em Engenharia Elétrica.
Orientador: Prof. Dr. André Bittencourt Leal

JOINVILLE

2021

Ficha catalográfica elaborada pelo programa de geração automática da

Biblioteca Setorial do CCT/UDESC,

com os dados fornecidos pelo(a) autor(a)

Moreira, Benjamin Grando
Controle tolerante a falhas de sistemas a eventos discretos usando diagnosticador controlador ativo / Benjamin Grando Moreira. -- 2021.
135 p.

Orientador: André Bittencourt Leal
Tese (doutorado) -- Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Engenharia Elétrica, Joinville, 2021.

1. Controle tolerante a falhas. 2. Controle tolerante a falhas ativo. 3. Diagnose de falhas. 4. Controlabilidade segura. 5. Sistemas a eventos discretos. I. Leal, André Bittencourt. II. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Engenharia Elétrica. III. Título.

BENJAMIN GRANDO MOREIRA

CONTROLE TOLERANTE A FALHAS DE SISTEMAS A EVENTOS DISCRETOS USANDO DIAGNOSTICADOR CONTROLADOR ATIVO

Tese aprovada como requisito parcial para a obtenção do grau de doutor, pelo Programa de Pós-Graduação em Engenharia Elétrica (PPGEEL) da Universidade do Estado de Santa Catarina

BANCA EXAMINADORA

Prof. Dr. André Bittencourt Leal
Universidade do Estado de Santa Catarina

Membros:

Profª. Drª. Ana Teruko Yokomizo Watanabe
Universidade do Estado de Santa Catarina

Prof. Dr. Douglas Wildgrube Bertol
Universidade do Estado de Santa Catarina

Prof. Dr. Max Hering de Queiroz
Universidade Federal de Santa Catarina

Profª. Drª. Patrícia Nascimento Pena
Universidade Federal de Minas Gerais

Joinville, 26 de fevereiro de 2021.

AGRADECIMENTOS

Inicialmente agradeço à minha mãe, Maria Grando, por ter me oportunizado os estudos até a conclusão da minha graduação. Também não posso deixar de agradecer por suas várias horas de oração em cada etapa da minha vida. Agradeço à professora Anita Maria da Rocha Fernandes, por ter me incentivado a fazer o mestrado, algo que eu não cogitava na época. Agradeço também à Elisangela Maschio de Miranda, querida colega que não se encontra mais entre nós, mas que costumávamos brincar sobre quem era mais enrolado sobre fazer o doutorado. Agradeço à minha esposa, Mayara Pereira Becker, pela paciência de tantos anos para a finalização do doutorado. Dedico um agradecimento especial ao meu orientador, André Bittencourt Leal, pela atenção nas correções e explicações detalhadas sobre minhas inconsistências de entendimento durante a formulação da tese. Embora meus familiares e amigos tenham esperado pela conclusão do doutorado, foi o André que teve o trabalho e dedicação durante o processo. Por fim, agradeço aos diversos colegas com os quais pude compartilhar momentos durante a trajetória acadêmica do doutoramento, principalmente aos que tive maior contato/auxílio: Ana Teruko Yokomizo Watanabe, Renan Sebem e Thalys Eduardo Ferreira Rezende.

RESUMO

Este trabalho considera o problema do controle tolerante a falhas em sistemas a eventos discretos. É utilizada a abordagem de controle tolerante a falhas ativo e seguro, que consiste em evitar sequências de eventos que podem levar o sistema a um comportamento proibido após a ocorrência de uma falha. Para isso são utilizadas as noções de diagnosticabilidade segura e controlabilidade segura. No controle tolerante a falhas ativo é necessário diagnosticar uma falha e então realizar o chaveamento do supervisor nominal para um supervisor pós-falha, sendo esse projetado para garantir a evolução do sistema após a diagnose, bem como a controlabilidade segura. Originalmente, o conceito de controlabilidade segura é associado à diagnose segura de falhas. Para a diagnose segura, a linguagem do sistema deve ser diagnosticável e a falha precisa ser diagnosticada antes da ocorrência de uma cadeia proibida. Para a controlabilidade segura, a diagnose da falha deve ocorrer de forma a permitir que algum evento controlável possa ser desabilitado e em tempo de impedir a ocorrência da cadeia proibida. Nesta tese, introduz-se inicialmente um relaxamento da condição para a diagnose segura e, consequentemente, na controlabilidade segura, de forma a não exigir que a linguagem seja diagnosticável. As noções relaxadas são denominadas A-diagnosticabilidade segura e A-controlabilidade segura. Mostra-se então que a A-controlabilidade segura de uma linguagem é uma condição necessária e suficiente para garantir a solução para o problema do controle tolerante a falhas ativo. Como principal contribuição do trabalho está a introdução de um novo tipo de diagnosticador, denominado Diagnosticador Controlador Ativo, que atua juntamente com o supervisor do sistema com o objetivo de garantir a controlabilidade segura quando a linguagem não é A-controlável segura. A partir desse novo tipo de diagnosticador é introduzida a estrutura de CTF ativo usando o Diagnosticador Controlador Ativo.

Palavras-chave: Controle Tolerante a Falhas, Controle Tolerante a Falhas Ativo, Diagnose de falhas, Controlabilidade segura, Sistemas a Eventos Discretos.

ABSTRACT

This work considers the safe controllability of Discrete Event Systems. The safe active fault tolerant control is considered, and the problem consists of avoiding sequences of events that after a fault occurrence could lead the system to a forbidden behavior. According to this notions of safe diagnosability and safe controllability are used. In the active fault tolerant control, it is necessary to diagnose a fault and then perform a switch to a post-fault supervisor, being this supervisor projected to guarantee the system evolution after the diagnosis and also the safe controllability. Originally the concept of safe controllability is associated to safe diagnosability. In terms of the safe diagnosis, the system language should be diagnosable, and the fault need be diagnosed before the occurrence of a forbidden string. For safety controllability, the fault diagnosis should occur in a way that permits a controllable event to be disabled to avoid forbidden strings. This thesis introduces a less rigorous version of the safe diagnosability condition and therefore, to the safe controllability in order to not demand that the language need be diagnosable. The notions of the less rigorous version are called A-Safe diagnosability and A-safe controllability. It is then shown that the A-safe controllability of a language is a necessary and sufficient condition to guarantee the solution to the problem of active fault tolerant control. As the main contribution of this work, a new diagnose type called Active Diagnoser Controller is introduced, that acts together with the supervisor of the system to guarantee safe controllability when the language is not A-Safe controllable. From this new type of diagnoser, the active fault tolerant control structure is introduced using the Active Diagnoser Controller.

Keywords: Fault tolerant control, Active Fault tolerant control, Fault diagnosis, Safe controllability, Discrete Event Sytems.

LISTA DE FIGURAS

LISTA DE DEFINIÇÕES

LISTA DE PROPOSIÇÕES

LISTA DE TEOREMAS

LISTA DE ABREVIATURAS

A-CSD	A-controlabilidade segura pelo diagnosticador
ADP	(*Active Diagnosis Problem*) Problema da Diagnose Ativa
A-DSD	A-diagnosticabilidade segura pelo diagnosticador
A-CS	A-controlabilidade segura
A-DS	A-diagnosticabilidade segura
ASDP	(*Active Safe Diagnosis Problem*) Problema da Diagnose Segura Ativa
CI	Condição de Impedimento
CSI	Controlabilidade Segura Irrestrita
GASR	Grupo de Automação de Sistemas e Robótica
RSI	Reconhecedor Seguro Interrompido
SED	Sistemas a Eventos Discretos
TCS	Teoria de Controle Supervisório

LISTA DE SÍMBOLOS

A_l	Autômato rotulador de falhas
A_{rsi}	Autômato rotulador seguro interrompido
A_s	Autômato rotulador seguro de falhas
ε	Cadeia vazia
$G_1 \| G_2$	Composição paralela de G_1 e G_2
$\|\|s\|\|$	Comprimento de uma cadeia s
st	Concatenação das cadeias s e t
$\mathscr{D}$	Condição de diagnosticabilidade de uma falha
G_d	Autômato diagnosticador
G_{ds}	Autômato diagnosticador seguro
G_{dsa}	Autômato diagnosticador seguro ativo
$\wedge$	Operador "e"
$\exists$	Existe
f	Evento de falha, $f \in \Sigma_{uo} \cap \Sigma_{uc}$
Σ^*	Fecho de Kleene: conjunto de todas as possíveis cadeias de comprimento finito sobre Σ, incluindo a cadeia vazia ϵ
$\overline{s}$	Fecho de prefixo de uma cadeia s
G	Autômato finito determinístico, $G = (X, \Sigma, \delta, \Gamma, x_0, X_m)$
$\widehat{G}$	Autômato observador dos eventos controláveis
G_{dca}	Autômato diagnosticador controlador ativo
G^f	Modelo da planta sob influência de uma falha f
$\mathscr{K}_f$	Linguagem ilegal pós-falha

L_m	Linguagem marcada por um autômato, $L_m \subseteq L$
L	Linguagem gerada por um autômato, $L \subseteq \Sigma^*$
L^{CNS}	Máxima linguagem controlável, normal, prefixo-fechada e segura
L^{safe}	Máxima linguagem segura
$Obs(G)$	Observador de G em relação a Σ_o
$\Omega(s)$	Mais longo prefixo da cadeia s cujo último evento da cadeia é controlável
$\forall$	Para todo
P_c	Projeção $\Sigma^* \rightarrow \Sigma_c^*$
Φ	Conjunto de cadeias proibidas pós-falha
$\tilde{\Phi}$	Conjunto de cadeias evitáveis pós-falha
P_i	Projeção $\Sigma^* \rightarrow \Sigma_i^*$
P_o	Projeção $\Sigma^* \rightarrow \Sigma_o^*$
P_o^+	Mapeamento perceptivo para os eventos observáveis do diagnosticador
L/s	Pós-linguagem de L após s
$t < s$	t é prefixo estrito de s
$\overline{L}$	Fecho de prefixo de L: conjunto de todos os prefixos das cadeias de L
$G_1 \times G_2$	Composição produto de G_1 e G_2
P_o^{-1}	Projeção inversa $\Sigma_o^* \rightarrow 2^{\Sigma^*}$
$\Psi_L(f)$	Conjunto de todas as cadeias de L que terminam com o evento f
Q_{dsa}^B	Conjunto de maus estados do diagnosticador seguro ativo
Q_{dsa}^C	Conjunto de estados certo de falha do diagnosticador seguro ativo
Q_{dsa}^{FC}	Conjunto dos primeiros estados certos de falha do diagnosticador seguro ativo
Q_{dsa}^N	Conjunto de estados normais do diagnosticador seguro ativo
Q_{dsa}^U	Conjunto de estados incertos de falha do diagnosticador seguro ativo
Q_{ds}^B	Conjunto de maus estados do diagnosticador seguro
Q_{ds}^C	Conjunto de estados certos de falha do diagnosticador seguro
Q_{ds}^{FC}	Conjunto dos primeiros estados certos de falha do diagnosticador seguro
Q_{ds}^N	Conjunto de estados normais do diagnosticador seguro
Q_{ds}^U	Conjunto de estados incertos de falha do diagnosticador seguro

S/G	supervisor S controlando G
$S \wedge DCA$	Ação conjunta do supervisor S e do DCA
Σ	Alfabeto da linguagem
Σ_c	Conjunto de eventos controláveis, $\Sigma_c \subseteq \Sigma$
Σ_o	Conjunto de eventos observáveis, $\Sigma_o \subseteq \Sigma$
Σ_{uc}	Conjunto de eventos não controláveis, $\Sigma_{uc} \subseteq \Sigma$
Σ_{uo}	Conjunto de eventos não observáveis, $\Sigma_{uo} \subseteq \Sigma$
Σ^+	$\Sigma^* - \{\epsilon\}$
S_i^{deg}	Supervisor pós-falha/degradado
$K^{\uparrow N}$	Suprema sublinguagem normal para K
:	Tal que
$\widetilde{Obs}(G_{ci})$	Calcula o observador e determina as condições de impedimento dos estados
$\emptyset$	Conjunto vazio
ξ	Cadeia proibida, $\xi \in \Phi$

SUMÁRIO

1 INTRODUÇÃO

Técnicas para o controle de sistemas diferem de acordo com o tipo do sistema a ser tratado. Um tipo de sistema é o Sistema a Eventos Discretos (SED), cuja mudança de estado é dependente da ocorrência de eventos em qualquer instante de tempo e seus modelos são obtidos por diversos formalismos. Para Baillieul e Samad (2015), a ênfase em uma representação rigorosa e formal dos SEDs é necessária para garantir a operação correta e segura do sistema, especialmente no contexto de sistemas autônomos, onde o sistema precisa que suas tarefas sejam executadas corretamente e sem gerar riscos às suas estruturas ou ambiente. Mesmo com um projeto cuidadoso, a complexidade dos sistemas tecnológicos modernos faz com que estes estejam cada vez mais vulneráveis a falhas.

Falhas são consideradas mudanças abruptas no comportamento de um sistema e podem ocasionar consequências indesejadas, podendo a magnitude dessas consequências variar entre uma diminuição de desempenho do sistema, até ocasionar riscos ao equipamento, produção, meio-ambiente ou a seres humanos envolvidos na utilização do sistema. Sem a ação apropriada, uma falha pode conduzir para consequências graves e inseguras. Para garantir a segurança do sistema e garantir que a planta continue em funcionamento após a ocorrência de uma falha são projetados supervisores com Controle Tolerante a Falhas (CTF).

A tolerância a falhas é a habilidade de um supervisor manter os objetivos de controle mesmo na ocorrência de uma falha. O projeto de sistemas tolerantes a falhas considera o comportamento nominal da planta na ausência da falha e o comportamento degradado da planta após a ocorrência da falha. Na literatura são encontradas duas estratégias alternativas para obter um sistema em malha fechada tolerante a falhas, sendo elas denominadas CTF passivo e CTF ativo.

No CTF passivo, um supervisor é projetado para corresponder adequadamente à ocorrência da falha, ou seja, é projetado um único controlador que pode lidar satisfatoriamente com ambos os modelos de planta (comportamento nominal e o comportamento degradado). No CTF passivo um desempenho degradado é aceito para permitir o funcionamento do sistema sujeito à falha.

No CTF ativo, um diagnosticador identifica a falha e comunica essa falha a um coordenador para posterior alteração da política de controle. Sendo assim, é projetado

um controlador para a planta nominal, outro controlador para a planta degradada e um diagnosticador para detectar a falha e ativar o controlador apropriado.

Sendo assim, o CTF passivo permite utilizar o mesmo controlador nas situações com falha e sem falha, podendo ser considerado como um supervisor robusto. O CTF ativo é comparado (por analogia) ao controle adaptativo e necessita que o controlador adapte a lei de controle se uma falha acontecer. Tanto a abordagem de CTF ativo, quanto o CTF passivo, são campos de pesquisa ativos (SÜLEK; SCHMIDT, 2013; MOOR, 2016; Raman; Sreenivas, 2020; BLANKE et al., 2016; BORUTZKY, 2021). Uma revisão sobre o CTF e Sistemas a Eventos Discretos (SEDs) é apresentada em Fritz e Zhang (2018).

Esta tese considera o problema do CTF para sistemas modelados por SEDs, mais especificamente a abordagem de CTF ativa para SEDs modelados com autômatos. Segundo Moor (2016), dependendo das classes de sistema em consideração, o controle CTF passivo pode impor limitações inaceitáveis ao comportamento de malha fechada nominal. Segundo Fritz e Zhang (2018), a abordagem passiva pode resultar em um controlador complexo, grande e em restrições indesejadas no comportamento nominal, que limita as possibilidades de aplicação. A vantagem é que não precisa da diagnose da falha. Essas visões são corroboradas por Niguez, Amari e Faure (2015), que afirmam que o método passivo não é aplicável para todos os tipos de falhas, enquanto o método ativo é aplicável para um maior número de falhas, mas exige o uso de um diagnosticador.

Para o CTF ativo é preciso inicialmente diagnosticar a ocorrência de falhas para, posteriormente, alterar a política de controle de forma apropriada. A diagnose de falhas em SEDs é, por si só, uma importante área de estudo, que recebe atenção constante, com diversos conceitos e métodos propostos na literatura, sendo também associada com outras importantes propriedades de SEDs, como a observabilidade (YIN; LAFORTUNE, 2015).

Sobre a diagnose em SEDs, em Lin (1994) é apresentada uma abordagem de diagnose de falhas offline. Em seguida, Sampath et al. (1995) introduziram a diagnose online a partir da construção de um autômato diagnosticador. Posteriormente, os métodos de diagnose foram estendidos para as arquiteturas modulares (CONTANT; LAFORTUNE; TENEKETZIS, 2006) e descentralizada (DEBOUK; LAFORTUNE; TENEKETZIS, 2000; QIU; KUMAR, 2006; WANG; YOO; LAFORTUNE, 2007). Revisões das técnicas de diagnóstico de falhas para SEDs podem ser encontradas em Lafortune (2015), Zaytoon e Lafortune (2013), Ekanayake et al. (2019).

Além da aplicação em diversas arquiteturas, diversos tipos de abordagens foram aplicadas para a diagnose de falhas, como autômatos temporizados (JIANG; KUMAR, 2004; DERBEL et al., 2006; JÚNIOR et al., 2016), SEDs fuzzy (LIU; QIU, 2009) e autômatos estocásticos (CHEN; KUMAR, 2013; LIU; QIU, 2008; THORSLEY; TENEKETZIS, 2005). Outras noções de diagnosticabilidade também foram apresentadas, com condições mais fracas de diagnosticabilidade (THORSLEY; TENEKETZIS, 2005; ZHAO; LIU; LIU, 2017; LIN et al., 2020) e condições mais fortes de diagnosticabilidade, como a diagnosticabilidade segura (PAOLI; LAFORTUNE, 2005; LIU, 2015; LIU; QIU, 2008).

Com relação à diagnosticabilidade segura de falhas, esta é vista como a primeira etapa para obter um sistema tolerante a falhas (PAOLI; LAFORTUNE, 2005; LIU et al., 2020). Um importante aspecto para o CTF ativo é detectar a falha com antecedência suficiente para ter a chance de atingir os objetivos pós-falha prescritos.

Por ser um desvio do comportamento desejado, uma falha pode levar o sistema para um problema grave de segurança. Nesse sentido, pode ser necessário que determinada sequência de eventos após a ocorrência da falha não ocorra, de forma a prevenir que a falha desenvolva outras falhas e que possam causar riscos à segurança. Para tanto, no modelo da planta deve existir um evento controlável, após a cadeia responsável pela diagnose, e antes da execução da sequência proibida, sendo essa propriedade chamada de controlabilidade segura.

A controlabilidade segura em SEDs objetiva que o sistema seja capaz de, após a ocorrência de uma falha, evitar zonas proibidas por motivo de segurança. Para impedir a execução de cadeias proibidas após a falha, Paoli, Sartini e Lafortune (2011) propuseram a noção de controlabilidade segura de falhas a partir da diagnose segura online de falhas. Em Watanabe et al. (2017b), a controlabilidade segura de SEDs é obtida por uma abordagem de diagnose e prognose online.

Uma vez que a controlabilidade segura utiliza alguma instrução (evento controlável) que possa impedir a ocorrência de uma cadeia proibida, esse impedimento pode interromper a evolução do sistema. Para permitir a continuidade do sistema, uma alternativa é a utilização do CTF ativo, que chaveia um supervisor pós-falha que aplica outra política de controle ao sistema, podendo, por exemplo, acionar componentes redundantes ao componente com a falha.

A noção de CTF ativo de SEDs apresentada por Paoli, Sartini e Lafortune (2011)

realiza o chaveamento de um controle nominal para um reconfigurado quando uma falha é detectada, com a propriedade de continuar a operação de forma segura. Para o CTF proposto, os autores utilizam um tipo especial de diagnosticador, denominado diagnosticador seguro, o qual detecta a falha de forma segura e altera o comportamento do sistema. Sequências de eventos indesejadas no sistema são evitadas com a alteração do comportamento nominal para um novo comportamento, em que não ocorra a sequência indesejada.

Nesta tese considera-se o problema da controlabilidade segura para sistemas que não apresentam as condições para controlabilidade segura. Para isso, é alterada a estrutura de CTF ativo de Paoli, Sartini e Lafortune (2011) para fazer uso de um novo tipo de diagnosticador, denominado Diagnosticador Controlador Ativo (DCA). Essa estrutura propõe formas de ação para o CTF seguro mesmo quando o sistema não apresenta as condições de controlabilidade segura. Diferente de Moor (2016), Kumar e Takai (2012) que exploraram condições de existência para supervisores nominais e pós-falha para o CTF ativo, esta tese se preocupa em garantir a controlabilidade segura durante as incertezas com relação à ocorrência de falhas.

É importante diferenciar duas abordagens em relação à diagnose ativa de falha. Na primeira, são projetados sinais que, quando aplicados ao sistemas, aumentam a quantidade de informação para diagnosticabilidade. Esses sinais podem ser aplicados para o sistema quando alguma métrica de desempenho indica uma operação anormal, ou é parte da rotina de diagnóstico, possivelmente periódica, para verificar se a falha ocorreu ou não. Dessa maneira, a diagnose ativa de falhas envolve manipular o sistema para ativamente investigar se a falha ocorreu. Esse significado de ativo é diferente do contexto de CTF ativo, que representa a capacidade do controlador reconfigurar a estratégia de controle. Em Heirung e Mesbah (2019) é encontrada uma revisão de técnicas para essa abordagem de diagnose ativa. Na segunda abordagem, introduzida por Sampath, Lafortune e Teneketzis (1998), a diagnose ativa é utilizada no sentido de obter um controle que imponha ao sistema um comportamento que permita alcançar as condições de diagnosticabilidade de uma linguagem. Essa segunda abordagem é a explorada no contexto da tese. Em Paoli e Lafortune (2005), a diagnose ativa foi estendida para o caso da diagnose segura, sendo denominada diagnose segura ativa.

O DCA objetiva garantir a controlabilidade segura do sistema em situações de

incerteza em relação à falha e permitir o chaveamento de supervisores pós-falha após a confirmação da falha, aproveitando a arquitetura de CTF ativa. Durante a incerteza em relação à falha, o DCA é responsável por auxiliar o supervisor e desabilitar eventos para prevenir cadeias proibidas de ocorrerem, ou seja, o DCA restringe as ações permitidas pelo supervisor, com o objetivo de garantir a controlabilidade segura. Uma proposta ilustrando esse controle foi apresentada inicialmente em Moreira e Leal (2020).

Além da estrutura de CTF ativo utilizando o DCA, nesta tese também são apresentados relaxamentos das condições da diagnosticabilidade e controlabilidade segura, consideradas coerentes com a noção de controlabilidade segura. Esses relaxamentos permitem que sistemas que não atendam as condições da noção de diagnosticabilidade ainda possam ser considerados controláveis seguros. Esses relaxamentos foram denominados, respectivamente, A-diagnosticabilidade segura e A-controlabilidade segura. Uma vez que a proposta do DCA é auxiliar o supervisor no controle seguro do sistema quando esse não apresenta a condição de A-controlabilidade segura, uma nova condição é introduzida, denominada controlabilidade segura irrestrita.

1.1 OBJETIVOS

1.1.1 Objetivo geral

Apresentar novos métodos e abordagens para o problema do controle tolerante a falhas ativo e da controlabilidade segura de SEDs modelados por autômatos de estados finitos.

1.1.2 Objetivos específicos

Para alcançar o objetivo geral da tese, foram estabelecidos como objetivos específicos:

1. Propor modificações em conceitos relacionados com a controlabilidade segura a fim de obter noções com condições de atendimento mais relaxadas do que as existentes na literatura;

2. Formalizar os conceitos propostos;

3. Estabelecer condições necessárias e suficientes para o atendimento das propriedades introduzidas; e

4. Introduzir um nova estrutura de controle tolerante a falhas ativo.

1.2 RESUMO DAS CONTRIBUIÇÕES

A partir da pesquisa sobre diagnose de falhas e controlabilidade segura foram apresentadas as seguintes contribuições:

1. Introdução da noção de A-diagnosticabilidade segura: relaxamento das condições para a diagnosticabilidade segura introduzida por Paoli e Lafortune (2005), permitindo que linguagens não diagnosticáveis sejam A-diagnosticáveis seguras;

2. Introdução da noção de A-controlabilidade segura: relaxamento da condição de controlabilidade segura introduzida por Paoli, Sartini e Lafortune (2011), alterando a condição de uma linguagem ser diagnosticável segura para a condição da linguagem ser A-diagnosticável segura. Mostrou-se que a A-controlabilidade segura apresenta as condições necessárias e suficientes para garantir a solução para o problema do CTF ativo;

3. Apresentação das noções de A-diagnosticabilidade e A-controlabilidade segura pelo diagnosticador: discussão sobre situações quando a A-diagnosticabilidade e a A-controlabilidade de um sistema não são obtidas pela linguagem do sistema (utilizada para a obtenção do supervisor) e sim por uma linguagem com mais informações, sendo essas informações utilizadas pelo diagnosticador;

4. Introdução da noção de controlabilidade segura irrestrita: condição mínima que uma linguagem precisa atender para permitir a A-controlabilidade segura; e

5. Introdução do diagnosticador controlador ativo: estrutura que adapta o diagnosticador-controlador de Paoli, Sartini e Lafortune (2011) e que permite a controlabilidade segura quando uma linguagem não é A-controlável segura. O diagnosticador controlador ativo também permite que as informações utilizadas para a diagnose possam ser utilizadas para o controle seguro de forma a auxiliar o supervisor.

1.3 ORGANIZAÇÃO DOS CAPÍTULOS

Este documento está estruturado da seguinte forma. No Capítulo 2 é feita uma revisão dos conceitos da teoria de linguagens e autômatos em SEDs. No Capítulo 3 é feita

uma revisão bibliográfica de conceitos e fundamentos relacionados à diagnose online de falhas em SEDs modelados por autômatos. Entre os fundamentos de diagnose de falhas destacam-se a diagnose de falhas e a diagnose segura de falhas. No Capítulo 4 é feita uma revisão bibliográfica dos conceitos e fundamentos relacionados ao controle tolerante a falhas em SEDs, com destaque para a abordagem de CTF ativo. No Capítulo 5 são apresentados relaxamentos das condições utilizadas na controlabilidade segura, sendo introduzidas as noções de A-diagnosticabilidade segura e A-controlabilidade segura, bem como a noção de controlabilidade segura irrestrita. O Capítulo 6 apresenta a contribuição principal da tese, que adapta a arquitetura de CTF ativo de Paoli, Sartini e Lafortune (2011) para fazer uso de um novo diagnosticador proposto na tese, denominado Diagnosticador Controlador Ativo (DCA). Para obtenção do DCA são definidos e ilustrados todos os passos necessários e avaliado o impacto causado no controle a partir da colaboração do DCA. O Objetivo do DCA é garantir a controlabilidade segura do sistema, podendo fazer uso de informações próprias do processo de diagnose, e que não estão disponíveis para o supervisor do sistema. Por fim, no Capítulo 6, são realizadas conclusões sobre o trabalho e apresentadas ideias de trabalhos futuros.

36

2 SISTEMAS A EVENTOS DISCRETOS E TEORIA DE CONTROLE SUPERVISÓRIO

Sistemas a Eventos Discretos (SEDs) são sistemas cujos estados assumem valores discretos e a dinâmica de transição de estados é dirigida pela ocorrência de eventos discretos assíncronos no tempo. Diversos tipos de sistemas podem ser representados como SEDs, tais como os sistemas de manufatura, telecomunicações e sistemas de transporte.

Dentre os diversos formalismos existentes para a modelagem de SEDs, destacam-se as redes de Petri e a teoria de linguagens e autômatos, pois estes possuem um arcabouço matemático que permite tratar aspectos relacionados com a diagnose de falhas, assim como do controle supervisório de SEDs, que são temas abrangidos nesta tese. Assim, por ser de domínio dos pesquisadores do Grupo de Automação de Sistemas e Robótica (GASR) da UDESC, grupo no qual esta tese foi desenvolvida, adotou-se o paradigma de modelagem baseado na teoria de linguagens e autômatos. Assim, os principais conceitos relacionados a este tema são brevemente apresentados a seguir.

2.1 LINGUAGENS

Um SED é associado com um conjunto de eventos Σ, sendo esse conjunto conhecido como *alfabeto*, e as sequências de eventos formam palavras, também chamadas de cadeias. O conjunto de todas as palavras de comprimento finito que representam o comportamento de um SED é chamado de linguagem, a qual é denotada por L. A palavra que não possui nenhum evento é chamada de *cadeia vazia* e é denotada por ε.

A operação de base para a formação de palavras e linguagens é a concatenação de símbolos de um alfabeto Σ. Por exemplo, a linguagem $L = \{a, cb, abc\}$ pode ser formada pela concatenação dos símbolos que compõem o alfabeto $\Sigma = \{a, b, c\}$.

O conjunto de todas as cadeias possíveis de serem formadas com os símbolos do alfabeto Σ, incluindo o ε, é chamado de *fecho de Kleene* de Σ, que é denotado por Σ^* e definido como: $\Sigma^* = \{\varepsilon\} \cup \Sigma \cup \Sigma\Sigma \cup \Sigma\Sigma\Sigma$..... O conjunto de todas as possíveis cadeias do alfabeto, sem incluir ε, é denotado por Σ^+ e definido por $\Sigma^+ = \Sigma^* - \{\varepsilon\}$.

Seja s uma cadeia e $tuv = s$ com $t, u, v \in \Sigma^*$, então t é chamado de prefixo de s, u é chamado de subcadeia de s e v é chamado de sufixo de s. Sejam duas cadeias, s e t, a notação s/t é utilizada para denotar o sufixo de s após t. Se t não é um prefixo de s, então

s/t não é definida. O comprimento de uma cadeia é denotado por $\|s\|$ e indica número de eventos da cadeia, incluindo repetições. Seja L uma linguagem sobre Σ e seja $s \in L$, a pós-linguagem de L após s, denotada por L/s, é definida como $L/s = \{t \in \Sigma^* : st \in L\}$.

Dadas as cadeias $t, s \in \Sigma^*$, t é um prefixo de s ($t \leq s$) se existe $u \in \Sigma^*$ tal que $tu = s$. Denota-se $t < s$, um caso especial de $t \leq s$, cujo $s \neq t$, o qual é dito que t é prefixo estrito de s. Dado um evento $\sigma \in \Sigma$ e uma cadeia $s \in \Sigma^*$, é usada a notação $\sigma \in s$ para expressar que σ aparece pelo menos uma vez em s.

A concatenação de duas linguagens L_1 e L_2 é formada concatenando todas as sequências de L_1 com todas as sequências de L_2. Formalmente, $L := L_1L_2 = \{s = s_1s_2 : s_1 \in L_1 \text{ e } s_2 \in L_2\}$.

O fecho do prefixo (prefixo-fechamento) é denotado por $\overline{L}$ e é o conjunto de todos os prefixos das sequências de L. Formalmente, $\overline{L} := \{s \in \Sigma^* : (\exists t \in \Sigma^*)[st \in L]\}$. Uma linguagem $L = \overline{L}$ é dita ser prefixo-fechada.

Seja uma linguagem L e um evento $\sigma \in \Sigma$, $\Psi_L(\sigma)$ é o conjunto de todas as sequências em L que terminam com o evento σ. Formalmente, $\Psi_L(\sigma) = \{s\sigma \in L : s \in \Sigma^*, \sigma \in \Sigma\}$.

A projeção natural consiste em apagar de uma cadeia os eventos que não pertencem ao alfabeto no qual se está projetando a cadeia. Formalmente, a projeção P_i para dois alfabetos Σ_i^* e Σ^*, sendo $\Sigma_i^* \subseteq \Sigma^*$, é definida por $P_i : \Sigma^* \to \Sigma_i^*$, com as seguintes propriedades:

$$P_i(\varepsilon) := \varepsilon$$

$$P_i(\sigma) := \begin{cases} \sigma & \text{se } \sigma \in \Sigma_i \\ \varepsilon & \text{se } \sigma \in \Sigma \backslash \Sigma_i \end{cases}$$

$P_i(s\sigma) := P_i(s)P_i(\sigma)$ para $s \in \Sigma^*, \sigma \in \Sigma$.

A projeção inversa é denotada por P_i^{-1} e é definida por: $P_i^{-1} : \Sigma_i^* \to 2^{\Sigma^*}$, em que $P_i^{-1}(t) := \{s \in \Sigma^* : P_i(s) = t\}$.

As projeções podem ser estendidas para linguagens de forma natural aplicando a projeção a todas as cadeias da linguagem. Assim, seja $L \subseteq \Sigma^*$, então $P_i(L) := \{t \in \Sigma_i^* : (\exists s \in L)[P_i(s) = t]\}$. Seja $L_s \subseteq \Sigma_i^*$, então $P_i^{-1}(L_s) := \{s \in \Sigma^* : (\exists t \in L_s)[P_i(s) = t]\}$.

2.2 AUTÔMATOS

O comportamento de SEDs pode ser modelado por autômatos de estados finitos. Formalmente, um autômato é representado a partir da sêxtupla $G = (X, \Sigma, \delta, x_0, \Gamma, X_m)$, sendo que X denota o espaço de estados, Σ o conjunto de eventos, x_0 o estado inicial do sistema, $X_m \subseteq X$ é o conjunto dos estados marcados e δ é a função de transição de estados, denotada por $\delta : X \times \Sigma \rightarrow X$. Se a função de transição for estendida ($\delta : X \times \Sigma^*$), ela opera sobre cadeias ao invés de um único evento (se for informada uma cadeia como parâmetro, é utilizada a função de transição estendida e se for informado um evento é utilizada a função de transição). $\Gamma : X \rightarrow 2^{\Sigma}$ é a função de eventos ativos. $\Gamma(x)$ é chamado de conjunto de eventos ativos de G no estado x e representa o conjunto de todos os eventos σ tal que $\delta(x, \sigma)$ é definida.

Um autômato G é dito determinístico se a função de transição resulta unicamente em um estado. Em outras palavras, em um estado não podem existir duas transições ou mais com o mesmo evento.

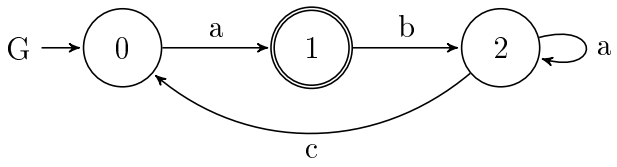
O comportamento de um SED modelado por um autômato G pode ser representado pela linguagem gerada $L(G)$ e a linguagem marcada $L_m(G)$. A linguagem prefixo-fechada gerada por G é definida pela função $L(G) = \{s \in \Sigma^* : \delta(x_0, s) \text{ é definida}\}$. A linguagem marcada é definida pela função $L_m(G) = \{s \in L(G) | \delta(x_0, s) \in X_m\}$ e contém todos os caminhos que começam do estado inicial e terminam em um estado marcado. É utilizado $L(G, x)$ para descrever a linguagem L a partir do estado $x \in X$ do autômato G.

Para representação gráfica dos autômatos, os estados são representados por circunferências e conectados por arcos rotulados com símbolos que representam os eventos. Os arcos representam as transições do estado, sendo arcos com traçado contínuo representações para eventos observáveis e o arco tracejado utilizado para representar eventos não observáveis. O estado inicial é indicado por uma seta apontada para ele. Os estados marcados são identificados por duas circunferências concêntricas e estão relacionados ao cumprimento de uma tarefa a ser completada. Quando um evento não gera a mudança de estado ele é chamado de auto-laço ou, no inglês, *selfloop*. O Exemplo 1 ilustra os conceitos, definidos até o momento, relacionados com autômatos.

Exemplo 1 *(Autômato determinístico).* O autômato determinístico $G = (X, \Sigma, \delta, x_0, \Gamma, X_m)$ representado na Figura 1 é composto por $X = \{0, 1, 2\}$, $x_0 = 0$, $X_m = \{1\}$ e

$\Sigma = \{a, b, c\}$. As funções de transição do autômato são: $\delta(0, a) = 1$, $\delta(1, b) = 2$, $\delta(2, a) = 2$ e $\delta(2, c) = 0$.

Figura 1 – Autômato G para o Exemplo 1



Fonte: Elaborado pelo autor (2021).

Um estado é acessível se existe uma cadeia que a partir do estado inicial alcança o estado em questão. Formalmente, $x \in X$ é acessível se $\exists s \in L(G)$ tal que $\delta(x_0, s) = x$. A componente acessível de G com respeito a x é dada por $Ac(G, x) = (X_{ac}, \Sigma, \delta_{ac}, x_{0,ac}, \Gamma_{ac}, X_{m,ac})$, onde $X_{ac} = \{x' \in X : (\exists s \in \Sigma^*)(\delta(x_0, s) = x' \text{ é definido})\}$, $\delta_{ac} = \delta|_{X_{ac} \times \Sigma \to X_{ac}}$ e $X_{m,ac} = X_m \cap X_{ac}$. A função $Ac(G)$ é a operação que elimina os estados não acessíveis de G.

Um estado é co-acessível (ou não bloqueante) se existe uma cadeia que, a partir desse estado, alcança um estado marcado. Formalmente, $x \in X$ é co-acessível se $\exists s \in \Sigma^*$ tal que $\delta(x_0, s) \in X_m$.

A operação de composição síncrona (também chamada de composição paralela) de n autômatos $G_i = (X_i, \Sigma_i, \delta_i, \Gamma_i, x_{0,i}, X_{m,i})$, com $i = \{1, ..., n\}$, resulta no autômato $G = \|G_i$, i = 1,...,n, e é definida por $G_1 \| G_2 := Ac(X_1 \text{ x } X_2, \Sigma_1 \cup \Sigma_2, \delta_{1\|2}, \Gamma_{1\|2}, (x_{0,1}, x_{0,2}), X_{m,1} \text{ x } X_{m,2})$, em que:

$$\delta_{1\|2}((x_1, x_2), \sigma) = \begin{cases} (\delta_1(x_1, \sigma), \delta_2(x_2, \sigma)) & \text{se } \sigma \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ (\delta_1(x_1, \sigma), x_2) & \text{se } \sigma \in \Gamma_1(x_1) \setminus \Sigma_2 \\ (x_1, \delta_2(x_2, \sigma)) & \text{se } \sigma \in \Gamma_2(x_2) \setminus \Sigma_1 \\ indefinida & \text{de outra forma} \end{cases}$$

A função de eventos ativos $\Gamma_{1\|2}$ para $G_1 \| G_2$ é obtida por: $\Gamma_{1\|2}((x_1, x_2)) = [\Gamma_1(x_1) \cap \Gamma_2(x_2)] \cup [\Gamma_1(x_1) \setminus \Sigma_2] \cup [\Gamma_2(x_2) \setminus \Sigma_1]$.

Caso exista mais do que uma transição com determinado evento em um estado do autômato, esse é chamado de não determinístico e sua função de transição é definida pela função $\delta : X \text{ x } \Sigma \to 2^X$. Para converter um autômato não determinístico em um

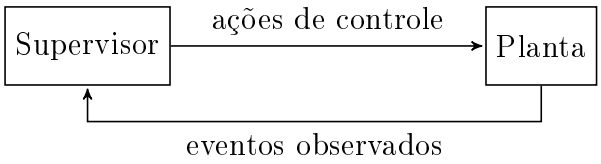
autômato determinístico é utilizado um autômato denominado observador. Para análise do algoritmo para construir o observador de um autômato não determinístico, consultar a página 88 em Cassandras e Lafortune (2009). Esta tese utiliza o observador quando existem eventos não observáveis no sistema, sendo essa questão apresentada na Subseção 2.3.1.

2.3 CONTROLE SUPERVISÓRIO DE SISTEMAS A EVENTOS DISCRETOS

A Teoria de Controle Supervisório (TCS) é um paradigma para síntese de controle para SEDs e foi introduzida por Ramadge e Wonham (1989). A TCS estabelece um método formal para cálculo de supervisores minimamente restritivos e não bloqueantes para SEDs.

A Figura 2 ilustra a arquitetura centralizada de controle. Nela o supervisor observa os eventos que ocorrem na planta e, em resposta a essas observações, age sobre a planta de forma a impedir que certos eventos aconteçam a fim de garantir o cumprimento de especificações de controle impostas pelo projetista.

Figura 2 – Arquitetura centralizada de controle



Fonte: Elaborado pelo autor (2021).

Considerando que um SED é modelado por um autômato G, esse autômato modela o comportamento não controlável do sistema, indicando todas as situações possíveis do sistema sem controle. Para controlar esse comportamento é possível obter um supervisor S. Nesse sentido, G representa a planta, enquanto S representa o controlador (supervisor) para o sistema e o que se busca é a planta sob supervisão do controlador (S/G[1]).

Um supervisor é definido formalmente como uma função $S : L(G) \rightarrow 2^{\Sigma}$. Para cada cadeia de eventos observada na planta, o supervisor associa um conjunto de eventos habilitados para essa cadeia. Ou seja, após a ocorrência de $s \in L(G)$, os eventos habilitados pelo supervisor são dados por $S(s)$.

[1] S/G é o resultado em malha fechada da política de controle S controlando G.

A notação S/G é utilizada para representar S controlando G, cuja linguagem gerada por S/G é dada por:

1. $\varepsilon \in L(S/G)$;
2. $[(s \in L(S/G))$ e $(s\sigma \in L(G))$ e $(\sigma \in S(s))] \Leftrightarrow [s\sigma \in L(S/G)]$.

A linguagem marcada por S/G é dada por $L_m(S/G) = L(S/G) \cap L_m(G)$.

Para o cálculo do supervisor, o comportamento da planta é representado por um autômato G e por suas linguagens gerada $L(G)$ e marcada $L_m(G)$, sendo $L_m(G) \subseteq L(G)$. O supervisor é obtido a partir de um conjunto de restrições E especificadas em Σ^* e que visam impedir comportamentos indesejados. Para isso o supervisor possui a capacidade de desabilitar (ou não habilitar) a ocorrência de eventos que conduziriam ao comportamento não desejado (como um bloqueio, por exemplo).

O controle de um evento (habilitação ou desabilitação do evento) depende que esse evento seja controlável e com isso o conjunto de eventos é particionado em eventos controláveis (Σ_c) e eventos não controláveis (Σ_{uc}), sendo $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$. Eventos controláveis são aqueles que um controlador tem controle, como ligar e desligar um equipamento, enquanto eventos não controláveis são aqueles que sua ocorrência não pode ser impedida pelo controlador, como a ativação ou desativação de um sensor.

Na Definição 1 é apresentada a condição necessária e suficiente para que um comportamento possa ser obtido em malha fechada (S/G), considerando que todos os eventos de G são observados por S.

Definição 1 *(Controlabilidade).* Sejam K e L linguagens definidas sobre um conjunto de eventos Σ, em que $K \subseteq L$ e $L = \overline{L}$. Seja Σ_{uc} um subconjunto de Σ. Diz-se que K é controlável e.r.a. L e Σ_{uc} se $\overline{K}\Sigma_{uc} \cap L \subseteq \overline{K}$.

A verificação da controlabilidade de uma linguagem é utilizada para determinar a existência de um supervisor. Sendo verificada a existência de um supervisor, é possível aplicar o conceito de controlabilidade para o projeto de supervisores utilizando o Teorema 1.

Teorema 1 *(Teorema da controlabilidade).* Seja $G = (X, \Sigma, \delta, x_0, \Gamma, X_m)$, em que

$\Sigma_{uc} \subseteq \Sigma$ é o conjunto de eventos não-controláveis. Seja $K \subseteq L(G)$ no qual $K \neq \emptyset$. Existe um supervisor S tal que $L(S/G) = \overline{K}$ se, e somente se, $\overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K}$.

Para a TCS, diversas extensões da estrutura monolítica foram propostas. Algumas dessas extensões buscam restringir o escopo da observação do supervisor através dos conceitos de observabilidade e normalidade, como em Cieslak et al. (1988) e Lin e Wonham (1988), ou supervisores robustos que adaptam o controle (LIN, 1993).

Uma vez que esse trabalho trata do controle de sistemas sob influência de falhas e que essas falhas são eventos não observáveis, a seguir é abordada a questão do controle sob observação parcial.

2.3.1 Controle sob observação parcial

Para o controle sob observação parcial introduz-se inicialmente algumas noções. O conjunto de eventos Σ é particionado em eventos observáveis (Σ_o) e eventos não observáveis (Σ_{uo}), sendo $\Sigma = \Sigma_o \cup \Sigma_{uo}$. Quando as projeções natural e inversa são realizadas para o conjunto de eventos observáveis Σ_o, elas são denominadas, respectivamente, por P_o e P_o^{-1}, sendo $P_o : \Sigma^* \to \Sigma_o^*$ e $P_o^{-1} : \Sigma_o^* \to 2^{\Sigma^*}$.

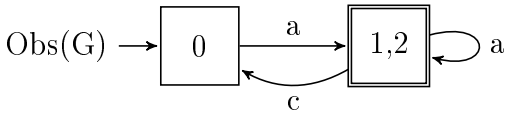
Quando se tratando de um autômato determinístico, mas com eventos não observáveis, pode ser utilizado o observador. Para análise do algoritmo para construir o observador de um autômato com eventos não observáveis, consultar a página 8 em Basílio, Carvalho e Moreira (2010). Para um autômato G com eventos não observáveis, os estados do observador são os estados em que G pode estar após a observação de uma sequência de eventos observáveis.

O observador de um autômato G para um conjunto de eventos observáveis Σ_o é dado por $G_{obs} = Obs(G) = (X_{obs}, \Sigma_o, \delta_{obs}, \Gamma_{obs}, x_{0,obs}, X_{m,obs})$, em que $X_{obs} \in 2^X$, $X_{m,obs} = \{B \in X_{obs} : B \cap X_m \neq \emptyset\}$, $\delta_{obs}(x_{obs}, \sigma) = \bigcup_{(x \in x_{obs}) \wedge \delta(x,\sigma)}$ é definida $UR(\delta(x, \sigma))$, $\Gamma_{obs}(x_{obs}) = \bigcup_{x \in x_{obs}} \Gamma(x) \cap \Sigma_o, x_{0,obs} = UR(x_0)$. O alcance não observável é definido por $UR(x) = \{x' \in X : (\exists s \in \Sigma_{uo}^*)[\delta(x, s) = x']\}$. Se $Y \in 2^X$ então $UR(Y) = \bigcup_{x \in Y} UR(x)$. Note que, $L(G_{obs}) = P_o[L(G)]$.

Para exemplificar o resultado da obtenção do autômato observador, a Figura 3 mostra o observador do autômato G do Exemplo 1.

Para sistemas parcialmente observáveis, o objetivo do controle é projetar contro-

Figura 3 – Autômato observador de G para o Exemplo 1



Fonte: Elaborado pelo autor (2021).

ladores supervisórios que garantam que o comportamento do sistema em malha fechada obedeça as especificações, considerando também eventos não controláveis e não observáveis. A controlabilidade é uma propriedade definida para sistema que são possíveis de controlar mesmo na presença de eventos não controláveis, enquanto a observabilidade é uma propriedade definida para sistemas possíveis de controlar na presença de eventos não observáveis (LAFORTUNE, 2007).

Segundo Cassandras e Lafortune (2009), no controle sob observação parcial, a ação de controle somente pode mudar após a ocorrência de um evento observável e, quando um evento observável ocorre, a ação de controle é atualizada instantaneamente. Observabilidade é uma propriedade que garante que dados suficientes sobre o comportamento da planta estão disponíveis. Além disso, não existe uma relação específica entre a controlabilidade e a observabilidade, sendo que um evento não observável pode ser controlável e um evento não controlável pode ser observável.

Para o controle de um sistema com observação reduzida, o supervisor sob observação parcial ($\Sigma_{uo} \neq \emptyset$) decide quais eventos são habilitados/desabilitados baseado na sua projeção natural de eventos da sequência gerada pela planta. Quando a planta executa uma sequência s, o supervisor observa a projeção $P_o(s)$, em que $P_o : \Sigma^* \rightarrow \Sigma_o^*$. Se duas sequências s_1 e s_2 tiverem a mesma projeção ($P_i(s_1) = P_i(s_2)$), o supervisor não tem como diferenciar qual sequência ocorreu e precisa tomar uma mesma decisão para ambas. Sendo assim, a decisão do supervisor não é tomada com base em $L(G)$, e sim com base em $P_o[L(G)]$.

A subseção a seguir formaliza as noções de observabilidade e normalidade, bem como apresenta uma forma de obter um supervisor sob observação parcial que imponha um comportamento para o sistema para garantir a normalidade.

2.3.2 Observabilidade e normalidade

Este trabalho considera a necessidade de controle sob observação parcial devido à ocorrência de falhas no sistema, sendo as falhas eventos não observáveis. Ao considerar falhas no sistema, o modelo da planta G sob influência de uma falha f é denotado como G^f, com $\Sigma_f = \{f\}$, sendo f não observável e não controlável (formalmente, $f \in \Sigma_{uo} \cap \Sigma_{uc}$).

O comportamento desejado de um sistema sem influência da falha é denominado *comportamento nominal*. No comportamento nominal a falha ainda não ocorreu e o sistema se comporta da forma normal como foi planejado. Quando se considera o funcionamento de um sistema sujeito à ação de falhas, o comportamento é dividido em comportamento nominal e *comportamento com falha*. No comportamento com falha a falha ocorreu e o sistema se comporta de forma diferente de suas especificações.

Quando em um sistema existem eventos não observáveis $\Sigma_{uo} \subset \Sigma$, o supervisor S não é capaz de observar todos os eventos que ocorrerem em G. Nesse sentido, é necessário que uma decisão seja baseada na projeção $P_o : \Sigma^* \to \Sigma_o^*$, pois o supervisor não é capaz de distinguir entre duas sequências s_1 e s_2 se elas tiverem a mesma projeção, ou seja, se $P_o(s_1) = P_o(s_2)$. Sendo assim, podem existir sequências diferentes e que exigem ações diferentes para o controle, mas com as mesmas projeções. Nesse sentido é preciso verificar que isso não ocorra, sendo isso definido utilizando o conceito da observabilidade, conforme a Definição 2.

Definição 2 *(Observabilidade).* Seja $G = (X, \Sigma, \delta, x_0, \Gamma, X_m)$, $\Sigma_c, \Sigma_o \subseteq \Sigma$ e suponha $M = \overline{M} = L(G)$. Seja a especificação de linguagem $K \subseteq M$, em que $K \neq \emptyset$, então K é observável em relação a M, Σ_o e Σ_c, se $\forall s \in \overline{K}$ e $\forall \sigma \in \Sigma_c$, $(s\sigma \notin \overline{K})$ e $(s\sigma \in M) \Rightarrow P_o^{-1}[P_o(s)]\sigma \cap \overline{K} = \emptyset$.

Em palavras, o conceito de observabilidade diz que, se não for possível diferenciar entre duas sequências, então a ação de controle a ser aplicada após a ocorrência de qualquer uma delas deve ser a mesma.

O comportamento K é observável se, e somente se, $\overline{K}$ é observável. Se K não for observável, então K contém s_1 e s_2, sendo $P_o(s_1) = P_o(s_2)$, com $s_1\sigma \notin \overline{K}$, enquanto $s_2\sigma \in \overline{K}$. Nesse sentido, não é possível obter um supervisor capaz de impor a linguagem K, já que ele não é capaz de diferenciar as sequências s_1 e s_2 que exigem ações de controle

distintas para o evento σ. Além disso, a observabilidade não é preservada sob a união, ou seja, se K_1 e K_2 são observáveis, então $K_1 \cup K_2$ não é necessariamente observável.

Quando um comportamento K não é controlável e observável, uma alternativa é a propriedade da normalidade. A normalidade é uma condição mais forte do que a observabilidade (*i.e.*, normalidade implica observabilidade), além de ser preservada em relação à união. Entretanto, a normalidade não garante que a solução seja maximamente permissiva. A Definição 3 formaliza a noção de normalidade.

Definição 3 *(Normalidade).* Seja $M = \overline{M} \subseteq \Sigma^*$, $K \subseteq M$ e $P_o : \Sigma^* \to \Sigma_o^*$. A linguagem K é normal e.r.a. M e P_o se $\overline{K} = P_o^{-1}[P_o(\overline{K})] \cap M$.

Em palavras, para a normalidade não podem existir duas sequências com a mesma projeção, em que uma faz parte da especificação e a outra não. Com isso, uma linguagem é dita normal se a informação de ocorrência de eventos não observáveis não é necessária para alcançar o comportamento desejado.

Se K não é normal, a suprema sublinguagem normal para K pode ser calculada, sendo a notação $K^{\uparrow N}$ usada para denotar essa linguagem. A fórmula para calcular a suprema sublinguagem normal é apresentada no Teorema 2.

Teorema 2 *(Suprema sublinguagem normal (KUMAR; GARG; MARCUS, 1991)).* Considere $K, L \subseteq \Sigma^*$, e $P_o : \Sigma^* \to \Sigma_o$. Então $K^{\uparrow N}$, a suprema sublinguagem normal para K e.r.a. L e P_o é dada por $K^{\uparrow N} = K - P_o^{-1}[P_o(M \setminus K)]$.

Quando imposto que $\Sigma_c \subseteq \Sigma_o$, a normalidade torna-se equivalente à observabilidade e a controlabilidade, como assegurado pelo Teorema 3.

Teorema 3 *(Equivalência entre observabilidade e normalidade (CASSANDRAS; LAFORTUNE, 2009)).* Suponha que $\Sigma_c \subseteq \Sigma_o$. Se K é controlável (e.r.a. L e Σ_{uc}) e observável (e.r.a. L, Σ_o e Σ_c), então K é normal (e.r.a. L e Σ_o).

Uma vez que uma falha é considerada um evento não observável, sua presença no sistema pode resultar em problemas de observabilidade, o que impacta diretamente na

controlabilidade segura do sistema. Nesse sentido é importante considerar aspectos de diagnosticabilidade e do controle tolerante a falhas. Para Lafortune (2007), um importante elemento no projeto de estratégias de controle integradas para sistemas complexos é a consideração das especificações lógicas que devem ser aplicadas por um controle supervisório seguro, vivo, diagnosticável, modular, reconfigurável e tolerante a falhas.

No próximo capítulo são apresentadas noções de diagnosticabilidade e formas de realizar o diagnóstico de falhas, com destaque para a noção de diagnosticabilidade segura.

3 DIAGNÓSTICO DE FALHAS

Neste capítulo são apresentados fundamentos e definições sobre o diagnóstico de falhas em Sistemas a Eventos Discretos (SEDs) modelados por autômatos. O problema da diagnose de falhas em SEDs foi introduzido por Lin (1994) e Sampath et al. (1995). Os autores apresentaram condições necessárias e suficientes para a diagnose de falhas de SEDs com a construção de um autômato determinístico denominado diagnosticador. Diagnóstico baseado em modelos no contexto de SEDs consiste em detectar eventos não observáveis significativos, como falhas, tomando como base sequências de eventos observáveis do sistema (WANG; YOO; LAFORTUNE, 2007). Para verificar se uma linguagem gerada por G é diagnosticável, pode-se utilizar o autômato diagnosticador.

O diagnóstico de falhas busca detectar, isolar e identificar falhas em um sistema. Detecção indica se um sistema está operando em condições normais, ou se uma falha ocorreu. Se uma falha ocorreu, o isolamento é responsável por localizar o componente que causou a falha. A identificação da falha reconhece a natureza específica da falha, como sua magnitude. Para Zaytoon e Lafortune (2013), detecção requer um modelo do comportamento normal do sistema, enquanto o isolamento e a identificação requerem um modelo do comportamento do sistema submetido às falhas.

Para este trabalho, as falhas consideradas são permanentes. Sendo assim, uma vez que o diagnosticador tiver certeza da ocorrência da falha, ele não voltará atrás e todos os estados seguintes continuarão indicando falha, embora seja possível mudar de um estado normal para um incerto ou certo de falha. Segundo Júnior et al. (2016), falhas permanentes podem surgir abruptamente e mudar o comportamento do sistema de forma repentina (como a ruptura de uma correia em uma esteira), ou podem ocorrer de forma progressiva, com pequenos desvios do comportamento do sistema (como uma pequena fuga de corrente que compromete a performance de um motor).

A diagnose é vista neste trabalho como o processo de detectar online (*on-the-fly*) a ocorrência de falhas usando diagnosticadores que observam sequências de eventos e que são construídos a partir do modelo do sistema.

Para Zaytoon e Lafortune (2013), três principais arquiteturas são usadas para calcular um diagnosticador: centralizada, descentralizada e distribuída.

Na arquitetura centralizada proposta por Sampath et al. (1995), existe um diagnosticador único e esse possui acesso a todos os eventos observáveis do sistema. Essa arquitetura é muitas vezes adequada para sistemas monolíticos, embora para sistemas modularizados, onde existem módulos com acesso apenas de suas informações locais, protocolos para diagnóstico descentralizado e distribuído são necessários.

A arquitetura descentralizada para diagnóstico de falhas em SEDs (também chamada de codiagnose) foi introduzida por Debouk, Lafortune e Teneketzis (2000) e busca detectar as falhas na planta global, mas considera que a informação não é capturada de maneira centralizada. Contant, Lafortune e Teneketzis (2006) introduziram o conceito de diagnosticabilidade modular, no qual sistemas podem ser modelados pela composição síncrona dos modelos de componentes locais.

Uma vez que a estrutura de controle apresentada nesta tese é proposta considerando a arquitetura centralizada, essa é a arquitetura abordada neste capítulo.

3.1 DIAGNOSTICABILIDADE E DIAGNOSTICADOR

Na estrutura centralizada a diagnose é feita usando um diagnosticador monolítico que é construído usando o modelo global do sistema. Uma linguagem é diagnosticável se for possível detectar a ocorrência de eventos de falha $f_i \in \Sigma_f$, com um número finito de eventos depois da ocorrência de f_i e utilizando somente sequências de eventos observáveis, ou seja, $\Sigma_f \subseteq \Sigma_{uo}$. Segundo Carvalho (2011), incorporar eventos não observáveis em um modelo, torna possível considerar o comportamento normal do sistema e também seu comportamento associado à falhas.

O conjunto de eventos associados às falhas é particionado em subconjuntos $\Sigma_{f_i}, i = 1, 2, ..., n$, não necessariamente unitários, formados por modelos que modelam as falhas e que de alguma forma são correlacionados, sendo que $\prod_f = \{\Sigma_{f_1}, \Sigma_{f_2}, ..., \Sigma_{f_n}\}$ denota essa partição. Cada vez que uma falha f_i correr, deve ser entendido que algum evento do conjunto Σ_{f_i} ocorreu. Por simplicidade, o trabalho considera nas definições somente uma única falha, *i.e.*, $\prod_f = \{\Sigma_f\}$, em que $\Sigma_f = \{f\}$.

Formalmente, para determinar se uma linguagem é diagnosticável, utiliza-se a Definição 4, de Sampath et al. (1995):

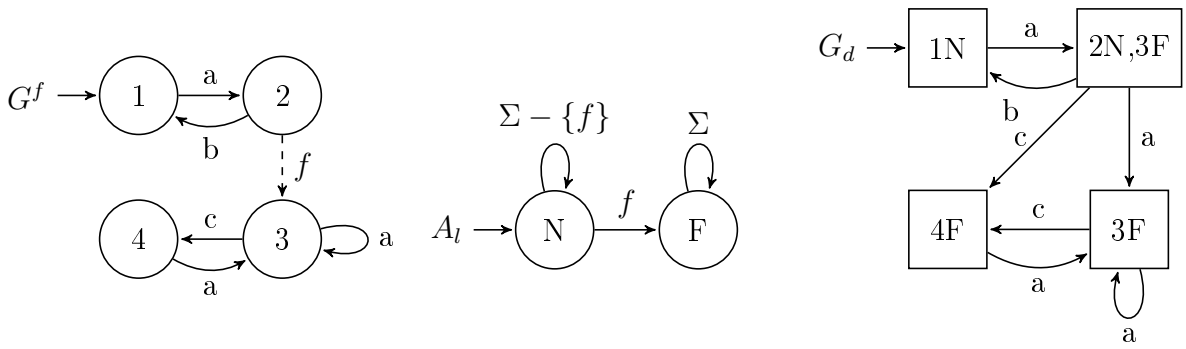
Definição 4 *(Diagnosticabilidade (SAMPATH et al., 1995)).* Uma linguagem prefixo-

fechada e viva L, que não contém ciclos de eventos não-observáveis, é diagnosticável com limite n em relação à projeção P_o e o evento de falha f, se a condição a seguir for verificada: $(\exists n \in \mathbb{N})(\forall s \in \Psi_L(f))(\forall t \in L/s)((\|t\| \geq n \Rightarrow \mathscr{D}))$, na qual $\mathscr{D}$ representa a condição de diagnosticabilidade e é expressa por: $\omega \in P_o^{-1}[P_o(st)] \cap L \Rightarrow f \in \omega$.

Sendo assim, a diagnosticabilidade requer que um evento de falha seja acompanhado de uma sequência distinta de eventos e suficientemente longa para identificar a ocorrência da falha. A propriedade da linguagem ser viva é realizada por uma questão de simplicidade, enquanto a não existência de qualquer ciclo de eventos não observáveis é utilizada para garantir que observações são feitas com certa regularidade.

Para realizar a diagnose de falhas, a partir da observação do comportamento do sistema durante a sua operação (online), pode-se utilizar um autômato diagnosticador. O diagnosticador centralizado G_d é um autômato cujo conjunto de eventos é igual ao conjunto dos eventos observáveis de G^f e com estados rotulados para identificar se uma falha f aconteceu ou não. G_d é construído obtendo a composição paralela de G^f com um autômato rotulador A_l (formalmente $G^f \| A_l$), e depois calculando o observador dessa composição paralela. Ou seja, $G_d = Obs(G^f \| A_l)$. A Figura 4 mostra o modelo de uma planta G^f, o autômato rotulador A_l para uma falha f e o autômato diagnosticador G_d.

Figura 4 – Modelo da planta com falha G^f, autômato rotulador A_l e o diagnosticador para a falha G_d



Fonte: Elaborado pelo autor (2021).

Uma vez que no diagnosticador os estados não são marcados, G_d é definido a partir da quádrupla[1] $G_d = (Q_d, \Sigma_o, \delta_d, q_{d,0})$, sendo que Q_d denota o conjunto de estados

[1] Quando a informação sobre marcação dos estados não for relevante para o autômato, será utilizada a representação na forma de uma quádrupla.

do diagnosticador, Σ_o denota o conjunto de eventos observáveis, $\delta_d : Q_d \times \Sigma_o \rightarrow Q_d$ é a função de transição de estados do diagnosticador e $q_{d,0} \in Q_d$ é o estado inicial do diagnosticador. O estado $q_d \in Q_d$ tem a forma $q_d = \{(q_1, l_1), ..., (q_m, l_m)\}$, sendo $q_i \in Q_d$ e $l_i \in \{N, F\}$ para $i = 1, ..., m$. Seja $q'_d \in Q_d$ um estado de G_d alcançado a partir de q_d por $\sigma \in \Sigma_o$. Seja $q_d = \{(q_1, l_1), ..., (q_m, l_m)\}$ e $q'_d = \{(q'_1, l'_1), ..., (q'_n, l'_n)\}$. Para todo $i \in \{1, ..., m\}$, existe $j \in \{1, ..., n\}$ tal que $q'_i = \delta_d(q_j, s)$, e

$$l'_i = \begin{cases} \text{F, se } l_j = F \text{ ou } (f \in s) \\ \text{N, se } l_j = N \text{ e } (f \notin s) \end{cases} \tag{3.1}$$

Pela formalização do diagnosticador, dependendo se a falha f está ou não na sequência de eventos que alcança um estado de G_d, os estados terão a forma (x_d, F) ou (x_d, N). Para simplificar a notação, esses estados são representados no autômato por q_dF e q_dN, respectivamente.

Um estado $q_d = \{(q_1, l_1), ..., (q_m, l_m)\} \in Q_d$ para $m \in \mathbb{N}$ é: normal, se $l_j = N$ para todo $j = 1, ..., n$; certo de falha se $l_i = F$ para todo $i = 1, ...n$; e incerto de falha se existe $l_j = N$ e $l_i = F$ para algum $i, j \in \{1, ..., m\}$. Um estado é denominado *certo de falha* se em seu rótulo aparece apenas a letra F referente ao autômato rotulador A_l, indicando a certeza de a falha ocorreu. Um estado é denominado *normal* se em seu rótulo aparece apenas a letra N, indicando a certeza de que não ocorreu uma falha. Um estado é denominado *incerto* quando em seu rótulo aparecerem tanto a letra N, quanto a letra F, indicando que não existe a certeza se a falha ocorreu ou não. No diagnosticador G_d da Figura 4 é possível observar um estado normal *1N*, dois estados certos de falha *3F* e *4F* e um estado incerto *2N,3F*.

Uma vez construído o diagnosticador é possível testar a diagnosticabilidade da linguagem. Uma linguagem L gerada por um autômato G é diagnosticável em relação à projeção P_o e ao evento de falha f se, e somente se, seu diagnosticador G_d não tiver ciclos indeterminados. Um ciclo indeterminado é um ciclo de estados incertos que na planta corresponde a dois ciclos, um antes e outro depois da falha. No diagnosticador, um ciclo indeterminado ocasiona um ciclo de estados incertos, que é formado por transições entre estados incertos de falha em G_d, de forma cíclica. É importante ressaltar que um ciclo de estados incertos no diagnosticador não necessariamente implica na existência de um ciclo indeterminado na planta, conforme exemplificação em Cassandras e Lafortune

(2009, p. 114). Em Basílio, Carvalho e Moreira (2010) foi apresentada uma formalização (definição 5 do artigo) para quando um ciclo de estados incertos no diagnosticador implicar em um ciclo indeterminado da planta.

A diagnosticabilidade de uma falha também pode ser verificada a partir de um autômato verificador calculado com espaço de estados polinomial em relação a cardinalidade do espaço de estados de G, conforme propuseram trabalhos como Jiang et al. (2001), Yoo e Lafortune (2002).

3.2 OPÇÕES QUANDO UMA LINGUAGEM NÃO É DIAGNOSTICÁVEL

Quando a linguagem do sistema não satisfaz as condições de diagnosticabilidade, existem algumas alternativas a serem adotadas. Basílio, Carvalho e Moreira (2010) apresentam três dessas possibilidades: (1) introduzir novos sensores; (2) introduzir sensores virtuais; e (3) uso de ações de controle.

O uso de ações de controle foi introduzido na diagnose ativa, apresentada por Sampath, Lafortune e Teneketzis (1998) a partir da definição do Problema da Diagnose Ativa (ADP, do inglês *Active Diagnosis Problem*). Para o ADP, ações de controle são aplicadas para tornar uma linguagem diagnosticável. O termo "ativo" é usado para diferenciar esse método da diagnose passiva, onde o diagnosticador apenas observa o comportamento do sistema e infere sobre potenciais falhas. Sendo assim, a diagnose ativa combina observação e controle e é utilizada para determinar a suprema sublinguagem controlável, observável e diagnosticável.

O objetivo da ADP foi formulado para calcular um supervisor que garanta a diagnosticabilidade de uma linguagem não diagnosticável. Então, seja uma linguagem L gerada por G que não é diagnosticável, é calculado um supervisor sob observação parcial S_P para $K \subseteq G$ tal que $L(S_P/G) = L^{diag}$ onde: $L^{diag} \subseteq K$; L^{diag} é diagnosticável; e L^{diag} é a mais permissiva possível. O detalhamento de como resolver o ADP é encontrado em Sampath, Lafortune e Teneketzis (1998).

Quanto à inclusão de novos sensores (reais ou virtuais) ao sistema, essa inclusão aumenta a quantidade de informação disponível e que pode ser utilizada para permitir a diagnose da falha. Uma possibilidade de uso desses sensores foi apresentada por Sampath et al. (1996) e denominada mapeamento de sensores. O mapeamento de sensores foi introduzido para o problema do desenvolvimento de modelos SEDs para a diagnose de

falhas. Nele os autores apresentaram um procedimento sistemático que incorpora no modelo do sistema informações provenientes de sensores.

Outra possibilidade para permitir a diagnosticabilidade de uma linguagem envolve utilizar informações de outras naturezas para a diagnosticabilidade do sistema, como informações probabilísticas, temporais e nebulosas (fuzzy), como realizado, respectivamente, por Thorsley e Teneketzis (2005), Derbel et al. (2006), Liu e Qiu (2009).

Mesmo a adição de informações ao processo de diagnose pode não resultar em uma linguagem diagnosticável. Quanto as restrições de controle impostas pela diagnose ativa, elas pode restringir demais o comportamento do sistema, de forma a tornar o comportamento restante não mais considerado suficiente para o sistema. Nesse sentido é importante considerar outras propriedades com relação à diagnosticabilidade. Algumas dessas propriedades de diagnosticabilidade são apresentadas na seção a seguir.

3.3 OUTRAS PROPRIEDADES DE DIAGNOSTICABILIDADE

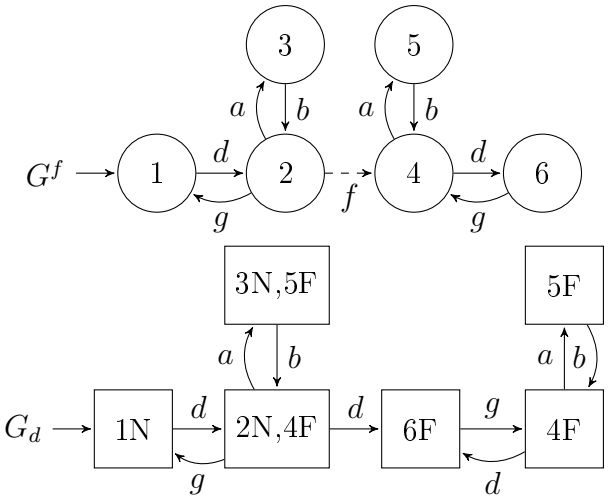
Na literatura, encontram-se trabalhos que apresentam variações ao conceitos de diagnosticabilidade, dentre os quais se destacam os trabalhos de Paoli e Lafortune (2005), Qiu e Kumar (2006), Thorsley e Teneketzis (2005), Zhao, Liu e Liu (2017), Lin et al. (2020). Algumas variações são condições mais fortes de diagnosticabilidade, como a diagnosticabilidade segura, outras apresentam condições mais fracas, como a A-diagnosticabilidade.

A estrutura de controle apresentada nesta tese tem especial interesse na noção de diagnosticabilidade segura (que é abordada na próxima seção) e também interesse na noção de A-diagnosticabilidade.

Para exemplificar a propriedade de A-diagnosticabilidade, considera-se a linguagem representada pelo autômato G^f da Figura 5, sendo $\Sigma_o = \{a, b, d, g\}$ e $\Sigma_f = \Sigma_{uo} = \{f\}$. O diagnosticador G_d também é mostrado na Figura 5.

A linguagem $L(G^f)$ não é diagnosticável, uma vez que existe um ciclo indeterminado em G^f, envolvendo os estados 2 e 3 e os estado 4 e 5. Entretanto, apesar da existência do ciclo indeterminado, é possível confirmar a ocorrência da falha quando for alcançado o estado $6F$ do diagnosticador. Sendo assim, embora a linguagem $L(G^f)$ não seja diagnosticável, ela é A-diagnosticável, conforme noção apresentada na Definição 5.

Figura 5 – Modelo de planta G^f e diagnosticador obtido para G^f, cuja falha f não diagnosticável, mas é A-diagnosticável



Fonte: Elaborado pelo autor (2021).

Definição 5 *(A-diagnosticabilidade).* Uma linguagem L prefixo-fechada, viva e que não contém ciclos de eventos não observáveis é dita A-diagnosticável e.r.a. P_o e f se $(\forall s \in \Psi(f)(\forall t \in L/s)), (\exists t' \in L/st$ tal que a condição $\mathscr{D}$ é atendida para $sst')$, sendo a condição $\mathscr{D}$ expressa como $\forall v \in P_o^{-1}[P_o(stt')] \cap L \Rightarrow f \in v$.

Em palavras, a definição de A-diagnosticabilidade estabelece que, após qualquer ocorrência da falha e para toda continuação pós-falha, sempre seja possível diagnosticar a falha. A A-diagnosticabilidade flexibiliza/relaxa a condição de diagnosticabilidade, permitindo a presença de ciclos indeterminados na planta. A A-diagnosticabilidade foi originalmente proposta por Thorsley e Teneketzis (2005) para SEDs estocásticos, sendo em Chen e Kumar (2013) apresentado que a A-diagnosticabilidade não depende do valor da função de probabilidade. A noção de A-diagnosticabilidade também é apresentada como diagnosticabilidade fraca (LIN et al., 2020), diagnosticabilidade relativa (ZHAO; LIU; LIU, 2017), entre outras formas (ROZÉ; CORDIER, 2002). O termo A-diagnosticabilidade é utilizado na tese por ser o primeiro identificado a apresentar essa noção.

Neste trabalho não se apresentam condições para a A-diagnosticabilidade uma vez que não se usa diretamente esse conceito. Entretanto, o conceito foi apresentado pelo fato de ele servir de base para a introdução do conceito de A-diagnosticabilidade segura, que será tratado no Capítulo 5. Uma discussão sobre isso é feita na Subseção 5.1.3.

3.4 DIAGNOSTICABILIDADE SEGURA DE FALHAS

A diagnose segura em SEDs foi introduzida por Paoli e Lafortune (2005) e está relacionada com a diagnose de falhas associadas com violações críticas de segurança e que podem não ser detectadas e identificadas em tempo suficiente para evitar a ocorrência de uma ação indesejada. A diagnose segura é uma exigência adicional à detecção da falha, sendo que a diagnose deve ser realizada antes de uma determinada cadeia de eventos ocorrer.

Uma planta G é representada como a interconexão de um conjunto de componentes que interagem e cujo modelos são denotados como G_i $(i = 1, ..., n)$, sendo G calculado a partir da composição paralela desses modelos. O comportamento $L(G)$ é restringido para atender um conjunto de especificações e para isso é projetado um supervisor S que, conectado com G, obtém sistema controlado S/G. Posteriormente, as potenciais falhas dos componentes são incluídas no modelo. A notação G_i^f é utilizada para representar o modelo do componente com a inclusão do seu comportamento com falha. Sendo assim, S/G^f contém o comportamento nominal e o comportamento com falha (também chamado de faltoso) da planta sob ação do supervisor nominal. Uma vez que S foi construído para um comportamento nominal, sequências indesejadas de ações podem surgir nas partes com falha. Para obter um controle tolerante a falhas é necessário que, além de ser capaz de detectar a ocorrência de falhas, devem ser prevenidas sequências indesejadas de ações na parte com falhas. Em síntese, Paoli e Lafortune (2005) apresentaram o problema como ter uma planta com o modelo de falha G^f supervisionada por um supervisor nominal S.

Em SEDs, uma falha pode ser modelada como um evento que deve ser identificado pelo sistema de diagnóstico após a ocorrência de alguns eventos no sistema. A quantidade dessas ocorrências é um indicativo do atraso para o diagnóstico e pode ser utilizado para determinar a eficiência do método de diagnóstico (YOO; GARCIA, 2003; VIANA; MOREIRA; BASíLIO, 2019). No caso do diagnóstico seguro, esse atraso está relacionado com a necessidade de identificar a falha antes que um conjunto de subcadeias ocorram.

Sendo assim, para a diagnose segura, considera-se uma linguagem L viva e que não possui ciclos de eventos não observáveis. A linguagem L é diagnosticável se é possível detectar a ocorrência de uma falha com um atraso finito de eventos observáveis. O conjunto $\Phi \subseteq \Sigma^*$ é utilizado para representar o conjunto de cadeias que devem ser impedidas após a ocorrência de uma falha. Uma linguagem é diagnosticável segura se a continuação mais

curta que assegura a detecção da falha não contém nenhum elemento de Φ como subcadeia, ou seja, se é possível fazer a diagnose da falha sem incluir uma cadeia de eventos $\xi \in \Phi$ não desejada.

O conjunto Φ captura as sequências de eventos que são ilegais após a ocorrência de uma falha f. A situação pode ser formalizada definindo a linguagem ilegal $\mathscr{K}_f$ como $\mathscr{K}_f = \{u \in L/s \text{ tal que } (s \in \Phi_L(f))(\exists \xi \in \Phi \text{ tal que } \xi \text{ é uma subcadeia de } u)\}$. Ou seja, $\mathscr{K}_f$ contém todas as possíveis continuações após um evento de falha f e que possuem uma cadeia proibida de Φ como subcadeia. A Definição 6 formaliza o conceito da diagnosticabilidade segura.

Definição 6 *(Diagnosticabilidade Segura (PAOLI; LAFORTUNE, 2005)).* Uma linguagem L prefixo-fechada, viva e sem ciclos de eventos não observáveis é diagnosticável segura e.r.a. projeção P_o, à falha f e à linguagem proibida $\mathscr{K}_f$, se:
(1) L é diagnosticável e.r.a. P_o e f; e
(2) $(\forall s \in \Psi_L(f))(\forall t \in L/s)$, tal que $\|t\| = n$, considerando que $t_c, \|t_c\| = n_{tc}$, seja o mais curto prefixo de t tal que a diagnose seja atendida e então $\overline{t_c} \cap \mathscr{K}_f = \emptyset$.

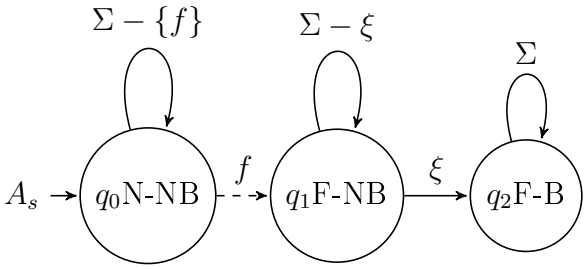
Em palavras, a definição de diagnosticabilidade estabelece que, para uma linguagem ser diagnosticável, é preciso detectar a ocorrência de um evento de falha f com um número finito de eventos depois da ocorrência da falha e utilizando apenas sequências de eventos observáveis. A definição de diagnosticabilidade segura estabelece que a linguagem deve ser diagnosticável e que, após um evento falha f, a continuação mais curta que assegura a detecção da falha não contém uma cadeia ilegal.

Da mesma forma que a diagnose de falhas, a diagnose segura pode ser verificada através de um autômato diagnosticador, denominado diagnosticador seguro, denotado por $G_{ds} = (Q_{ds}, \Sigma_o, \delta_{ds}, q_{ds,0})$. O procedimento completo para obter o diagnosticador seguro é encontrado em Paoli e Lafortune (2005) sendo aqui apresentada a ideia geral. Destaca-se do processo que, o diagnosticador seguro G_{ds}, em adição aos rótulo F e N do diagnosticador, possui outros dois rótulos denominados NB e B (do inglês, *Not Bad State* e *Bad state, respectivamente*). Então, $q_{ds} = \{(q_1, l_1), ..., (q_m, l_m)\}$, onde $q_i \in Q$ e $l_i \in \{(N, NB), (F, NB), (F, B)\}$. Se um estado q_{ds} contém pelo menos um elemento na forma (q, l) com $l = B$ ele é chamado de mau estado, assim como seu estado correspondente

$q \in Q$. Para simplificar a notação, seja $l \in \{(N, NB), (F, NB), (F, B)\}$, os estados de G_{ds} são representados no autômato por $q_dN - NB$, $q_dF - NB$ e $q_dF - B$, respectivamente.

Para obter o diagnosticador seguro é realizada a composição da planta com um autômato rotulador seguro. A Figura 6 ilustra o rotulador seguro A_s quando Φ possui apenas uma cadeia proibida e essa cadeia é constituída de apenas um evento, ou seja, $\xi \in \Phi$ e $\|\Phi\| = 1$. O autômato A_s possui 3 estados que representam: a não ocorrência de uma falha (N-NB); a ocorrência de uma falha, mas a não execução de uma cadeia proibida (F-NB); e a ocorrência da falha com o evento proibido (F-B), sendo esse último chamado de mau estado. O autômato G_{ds} é calculado a partir da composição síncrona do autômato G com o autômato rotulador A_s (formalmente, $G \| A_s$) e posterior cálculo da projeção para os eventos observáveis dessa composição, formalmente $G_{ds} = Obs(G \| A_s)$.

Figura 6 – Autômato rotulador seguro para cadeia com apenas um evento



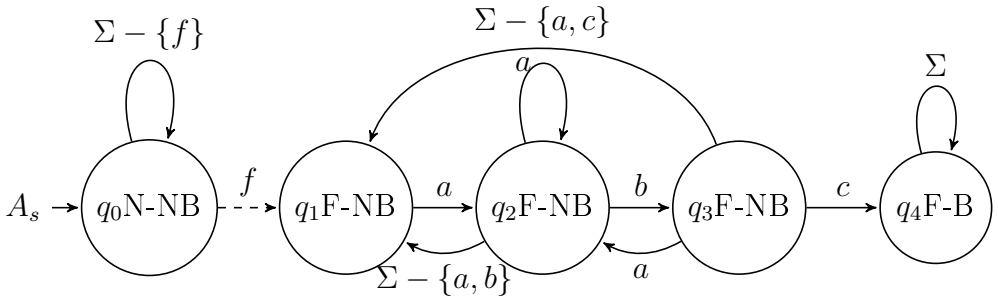
Fonte: Elaborado pelo autor (2021).

Para um sistema com uma falha f e com a cadeia proibida formada por uma sequência de eventos $\Phi = \{abc\}$, o autômato rotulador é o apresentado na Figura 7. Após a falha, uma sequência com o evento a, seguido do evento b e seguidos pelo evento c, conduzirá o sistema para o mau estado.

Em Watanabe et al. (2017a) são introduzidas novas condições para diagnosticabilidade segura, apresentando as condições necessárias e suficientes para garantir a diagnosticabilidade e a diagnosticabilidade segura, empregando diagnosticadores calculados com o alcance de eventos não observáveis e sem o alcance de eventos não observáveis.

Para ilustrar o processo de obtenção do diagnosticador seguro é utilizado o Exemplo 2. O exemplo considera o caso de uma cadeia proibida de comprimento unitário. Nele observa-se que, uma vez que ocorra o evento proibido depois da falha, o diagnosticador alcança os estados rotulados como maus estados (estados com o sufixo "-B").

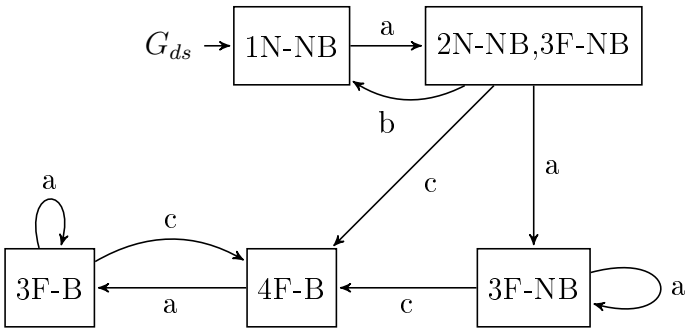
Figura 7 – Autômato rotulador seguro para a cadeia *abc*



Fonte: Elaborado pelo autor (2021).

Exemplo 2 *(Obtenção do diagnosticador seguro).* Considere o modelo de planta G^f apresentada na Figura 4. Para esse sistema se deseja impedir a ocorrência do evento c após a falha f, ou seja, $\Sigma_f = \{f\}$ e $\xi = c \in \Phi$. A Figura 8 mostra o diagnosticador seguro G_{ds} obtido a partir de G^f.

Figura 8 – Diagnosticador seguro G_{ds} para o modelo G^f do Exemplo 2



Fonte: Elaborado pelo autor (2021).

Uma vez construído o diagnosticador seguro é possível testar a diagnosticabilidade segura da linguagem a partir do Teorema 4, apresentado por Watanabe (2019). Para o teorema, considere o diagnosticador seguro calculado para uma falha f e a cadeia proibida $\xi \in \Phi$. São definidos os conjuntos $Q_{ds}^N = \{q_{ds} \in Q_{ds} : q_{ds}$ é normal$\}$, $Q_{ds}^C = \{q_{ds} \in Q_{ds} : q_{ds}$ é certo de falha$\}$, e $Q_{ds}^U = \{q_{ds} \in Q_{ds} : q_{ds}$ é incerto de falha$\}$. Define-se também o conjunto dos primeiros estados certos de falha, dado por $Q_{ds}^{FC} = \{q_{ds} \in Q_{ds}^C : [q_{ds} = \delta_{ds}(q_{ds,0}, v_0)$, com $v_0 \in \Sigma_o^*) \wedge (\nexists q'_{ds} \in Q_{ds}^C : q'_{ds} = \delta_{ds}(q_{ds,0}, v'_0)$ tal que $v'_0 < v_0)]\}$. O conjunto Q_{ds}^B é o conjunto de maus estados e definido formalmente como $Q_{ds}^B = \{q_{ds} \in$

$Q_{ds} : q_{ds}$ é um mau estado }.

Teorema 4 *(Teste da diagnosticabilidade segura).* Considere uma linguagem diagnosticável L. L é diagnosticável segura e.r.a. projeção P_o, a falha f e a linguagem proibida $\mathscr{K}_f$ se, e somente se, no diagnosticador seguro:
(TC1) $\nexists q_{ds} \in Q_{ds}^{U} : q_{ds} \in Q_{ds}^{B}$; e
(TC2) $\nexists q_{ds} \in Q_{ds}^{FC} : q_{ds} \in Q_{ds}^{B}$.

Em palavras, uma linguagem é diagnosticável segura se seu diagnosticador G_{ds} não tiver ciclos indeterminados de falha e o mau estado for um estado certo de falha e seu antecessor não for um estado incerto de falha.

O diagnosticador seguro da Figura 8 mostra que a linguagem não é diagnosticável segura. Embora não exista um ciclo indeterminado de falha, o estado *(4F-B)* pode ser alcançado por um estado incerto de falha (no caso, o estado *(2N-NB,3F-NB)*).

Para as situações onde a linguagem não é diagnosticável segura, Paoli e Lafortune (2005) adotaram uma estratégia baseada na diagnose ativa, calculando um supervisor que, a partir do controle, imponha um comportamento ao sistema que seja diagnosticável seguro. O problema foi formulado como diagnose segura ativa (do inglês, *active safe diagnosis problem* - ASDP).

O objetivo da ASDP é calcular um supervisor que garanta a diagnosticabilidade segura de uma linguagem não diagnosticável segura em relação às cadeias proibidas Φ. Então, seja uma linguagem L, gerada por G, e que não é diagnosticável segura, é calculado um supervisor sob observação parcial S_P para $K \subseteq L$ tal que $L(S_P/G) = L^{safe}$ em que: $L^{safe} \subseteq K$; L^{safe} é diagnosticável segura e.r.a. Φ; e L^{safe} é a mais permissiva possível.

A diagnosticabilidade segura de falhas de Paoli e Lafortune (2005) foi estendida para o caso descentralizado por Qiu, Wen e Kumar (2009) e denominada codiagnosticabilidade segura. Para a codiagnosticabilidade segura é preciso que pelo menos um diagnosticador local detecte a falha com atraso limitado e antes que uma especificação de segurança seja violada. A codiagnosticabilidade segura pode ser vista como uma extensão da noção de codiagnosticabilidade proposta por Qiu e Kumar (2006).

4 CONTROLE TOLERANTE A FALHAS

Sistemas de manufatura são constituídos de diversos subsistemas, como robôs, máquinas e esteiras, sendo que a produção depende do funcionamento correto desses. A economia e o dia a dia das cidades depende do funcionamento de grandes redes de distribuição de energia e sistemas de transporte, podendo a falha em um simples componente ter efeitos maiores na disponibilidade e no desempenho do sistema como um todo (BLANKE et al., 2016). Falhas são a primeira causa de mudanças na estrutura de um sistema ou de seus parâmetros e que levam para um eventual desempenho degradado do sistema ou mesmo a perda de seu funcionamento.

Para evitar a deterioração ou danos aos equipamentos e pessoas, falhas devem ser diagnosticadas tão rápido quanto possível e decisões que interrompem a propagação desses efeitos devem ser tomadas (BLANKE et al., 2016). Para Wen (2009), tolerância a falhas é a habilidade de um sistema trabalhar corretamente, mesmo que algum componente não funcione como desejado, mantendo o sistema confiável e disponível quando necessário.

Controladores tolerantes a falhas são projetados para manter a operação do sistema dentro de limites seguros e mitigar os efeitos de mau funcionamentos do sistema ou de um componente (YU; JIANG, 2015). Um sistema tolerante a falhas significa que ele funciona satisfatoriamente após a ocorrência de uma falha, tendo o controlador a habilidade de reagir a existência da falha, ajustando suas atividades (BLANKE et al., 2016).

Uma falha em um sistema dinâmico é uma deturpação na estrutura do sistema, ou nos parâmetros deste, como, por exemplo, bloqueio de um atuador, perda de um sensor ou desconexão de um componente do sistema (BLANKE et al., 2016). Segundo Yu e Jiang (2015), o projeto de um sistema que tolere falhas envolve complexidade consideravelmente maior e exige um detalhado estudo do sistema físico, incluindo os modos de falha, seu impacto e as ações corretivas necessárias.

Em diversas situações de falha, redundância é um elemento indispensável para o Controle Tolerante a Falhas (CTF). Para Jalote (1994), redundância é definida como componentes do sistema que não são necessários para seu correto funcionamento, estando presentes apenas para quando uma falha ocorrer. Ainda, segundo Wen (2009), a redundância pode ser obtida através do hardware ou software.

Métodos de CTF para SEDs podem ser classificados em duas categorias: passivo (WEN et al., 2008; WITTMANN; RICHTER; MOOR, 2013) e ativo (NKE; LUNZE, 2012; PAOLI; SARTINI; LAFORTUNE, 2011). Segundo Niguez, Amari e Faure (2015), que comparou os dois métodos, embora o método passivo permita um CTF sem um diagnosticador ele não é aplicável para todos os tipos de falhas e exige o modelo das falhas e seus impactos no sistema. O método ativo por sua vez é aplicável para um maior número de falhas, mas exige o uso de um diagnosticador. Esses dois métodos são abordados a seguir.

4.1 MÉTODO PASSIVO DE CTF

Em um CTF passivo, um controlador único é projetado considerando tanto as condições normais de funcionamento, quanto com falhas. Normalmente o controlador integra o sistema de redundância, não sendo necessário um gerenciamento em tempo real dessas redundâncias (YU; JIANG, 2015). O método passivo também é chamado de controle robusto.

Segundo Blanke et al. (2016), controladores robustos somente existem para uma classe restrita de falhas e o funcionamento de um controlador robusto é subótimo para a planta nominal devido à fixação dos parâmetros para um equilíbrio entre desempenho e robustez.

O propósito do método passivo é encontrar um controlador fixo entre todas as soluções admissíveis, que corresponde para cada situação livre de falha. Como vários cenários de falhas e condições normais são incluídas, sacrifícios precisam ser feitos para alcançar as condições normais de operação, mantendo um certo grau de conservadorismo no desempenho do sistema (YU; JIANG, 2015). O CTF passivo usa técnicas de controle robusto para garantir que o sistema em malha fechada permaneça insensível a certas falhas e, com isso, continue a operar com o mesmo controlador e estrutura de sistema (PAOLI; SARTINI; LAFORTUNE, 2011).

4.2 MÉTODO ATIVO DE CTF

No CTF ativo, a informação do monitoramento do sistema é utilizada para fazer a diagnose e os resultados são então utilizados para reconfigurar o controle, bem como gerenciar redundâncias, assegurando que o sistema está seguro (YU; JIANG, 2015).

Na abordagem passiva é necessário incluir no projeto todos os subsistemas que podem ser utilizados em cada situação. Assim, se uma válvula só é usada após a falha, ainda assim essa válvula estará no modelo da planta para o cálculo de um supervisor robusto. Quando uma falha é um evento raro de ocorrer, acrescentar ao modelo da planta uma válvula utilizará recursos para algo que dificilmente será necessário. Na abordagem ativa (chaveada) é possível usar apenas os elementos de planta que são importantes para cada estratégia de controle (antes e depois do diagnóstico da falha).

Uma vez identificada e isolada a falha é preciso reconfigurar a lei de controle para tolerar a falha, garantindo um desempenho pré-especificado e possivelmente degradado, que mantenha o alcance de objetivos por parte do sistema com falha. Essa abordagem é semelhante à ideia utilizada no controle adaptativo, na qual, baseado na estimativa da falha, é realizada uma reconfiguração explícita do controlador na presença da falha (PAOLI; SARTINI; LAFORTUNE, 2011).

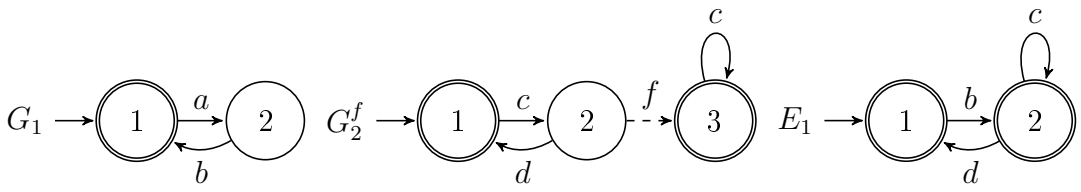
Quando uma falha no sistema ocorrer, for detectada e isolada, um coordenador realiza o chaveamento de um novo controlador para o sistema. Esse chaveamento substitui o supervisor nominal por um contendo ações de contingenciamento da falha, seja restringido o comportamento do sistema ou ativando sistemas redundantes relacionados com a falha. O supervisor chaveado pela ocorrência da falha será denominado supervisor pós-falha (também chamado de supervisor degradado) e será denotado como S_i^{deg}, $i = 1, 2, 3, ..., n$, sendo n a quantidade de ocorrências de uma falha f.

Para ilustrar o CTF ativo e o processo de chaveamento do supervisor nominal para um supervisor pós-falha será utilizado o Exemplo 3. No exemplo, após a falha não é mais possível habilitar o evento d, gerando um bloqueio no sistema.

Exemplo 3 *(CTF ativo).* Considere os modelos da planta e especificação de controle mostrados na Figura 9. O modelo de planta G_2^f é o modelo do componente com a influência de uma falha f. Na Figura 10 são mostrados o supervisor S para o comportamento nominal do sistema e o comportamento alcançado pela planta com falha sob ação do supervisor S. Considera-se para o sistema que $\Sigma_c = \{a, c, d\}$, $\Sigma_{uc} = \{b, f\}$, $\Sigma_o = \{a, b, c, d\}$ e $\Sigma_{uo} = \{f\}$.

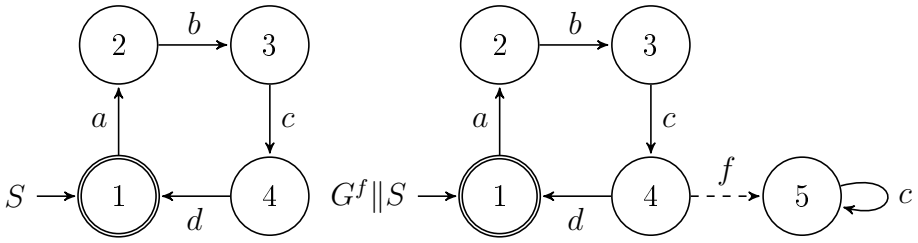
Na Figura 11 são mostrados o diagnosticador G_d da falha e um supervisor pós-falha

Figura 9 – Modelo das plantas e especificação de controle para exemplo do CTF ativo



Fonte: Elaborado pelo autor (2021).

Figura 10 – Supervisor nominal e modelo do sistema com falha sendo controlado pelo supervisor nominal, para exemplo do CTF ativo

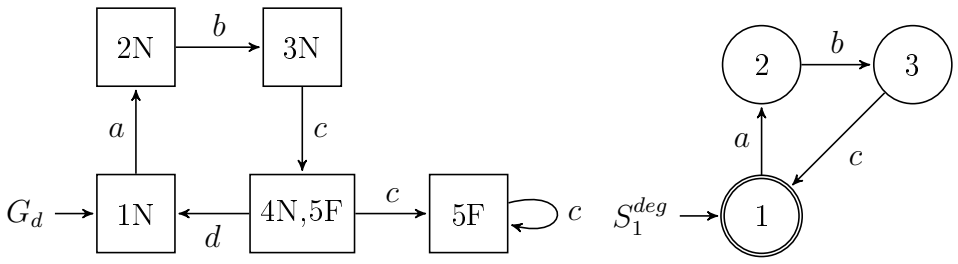


Fonte: Elaborado pelo autor (2021).

S_1^{deg} para ser chaveado após a diagnose da falha. A falha f é diagnosticável e confirmada a partir do estado $5F$ de G_d, de forma que é possível chavear para o supervisor S_1^{deg} nesse estado e manter o funcionamento do sistema, mesmo que de forma degradada.

Sendo assim, após a diagnose da falha, pode ser alterada a política de controle imposta pelo supervisor nominal S, para a política de controle imposta pelo supervisor pós-falha S_1^{deg}, que não incorre em bloqueio [1].

Figura 11 – Diagnosticador e supervisor pós-falha para exemplo do CTF ativo



Fonte: Elaborado pelo autor (2021).

Na proposta de Paoli, Sartini e Lafortune (2011), o CTF ativo utiliza a noção

[1] O supervisor pós-falha poderia também acionar algum componente redundante em sua operação, não apenas alterar algo com relação à especificação de controle, como realizado no exemplo.

de diagnosticabilidade segura vista em Paoli e Lafortune (2005). Os autores introduzem a noção de controlabilidade segura e definem a noção de CTF ativo, com respeito às especificações pós-falhas, como uma propriedade para a operação segura do sistema após a falha. Nesse sentido, Paoli e Lafortune (2005) propuseram o controle tolerante a falhas seguro, apresentado a seguir.

4.3 CTF ATIVO SEGURO

O CTF seguro é uma noção que se preocupa em não conduzir o sistema para uma situação insegura após falhas. O trabalho de Paoli, Sartini e Lafortune (2011) considera que a diagnosticabilidade segura é uma condição necessária para o CTF e que a controlabilidade segura representa a capacidade de, após a ocorrência da falha, evitar zonas proibidas. Uma zona proibida é alcançada a partir de uma sequência de eventos denominada cadeia proibida. Uma cadeia proibida pode ser uma situação que conduz o sistema para um bloqueio ou um problema de segurança. Um exemplo de sequência de eventos indesejada é, após uma falha de comunicação na operação de dois braços robóticos, ambos acessarem um *buffer* ao mesmo tempo.

No CTF seguro a linguagem do sistema G^f é dividida em duas partes: a parte nominal e a parte faltosa (parte após ocorrência da falha). A parte faltosa inclui a linguagem ilegal $\mathscr{K}_f$, com todas as possíveis continuações após a falha f que contém uma cadeia proibida do conjunto Φ como subcadeia.

Para o CTF seguro é utilizada a noção da controlabilidade segura, introduzida por Paoli, Sartini e Lafortune (2011) e formalizada na Definição 7. Essa noção garante que o sistema irá impedir a execução de sequências indesejadas após falhas.

Definição 7 *(Controlabilidade Segura (PAOLI; SARTINI; LAFORTUNE, 2011)).* Uma linguagem L prefixo-fechada, viva e sem ciclos de eventos não observáveis é controlável segura e.r.a. projeção P_o, a falha f e a linguagem proibida $\mathscr{K}_f$, se:
(1) L é diagnosticável segura e.r.a. P_o e f; e
(2) considere uma cadeia qualquer $s \in L$ tal que $f \in s$ e $s = v\sigma$, com $\sigma \in \Sigma_o$. Suponha que a condição de diagnosticabilidade não é atendida para a cadeia v, mas é atendida para a cadeia s, então, $(\forall t \in L/s)$ tal que $t = u\xi$ com $\xi \in \Phi$, $\exists z \in \Sigma_c$ tal que $z \in t$.

Em palavras, uma linguagem é controlável segura se ela for diagnosticável segura e, para qualquer cadeia que contém uma falha f e as cadeias proibidas contidas em Φ, se existir um evento observável que assegura a detecção da falha antes do sistema concluir alguma cadeia proibida e existir um evento controlável após o evento observável que impede a conclusão dessa cadeia proibida.

Apenas limitar a execução de sequências indesejadas após a falha não é suficiente para alcançar um CTF. Para Paoli, Sartini e Lafortune (2011), o objetivo do projeto de sistemas tolerantes a falhas consiste em diagnosticar a ocorrência da falha antes de executar alguma cadeia proibida, forçar o sistema a parar sua evolução antes da execução de uma sequência proibida e guiar o comportamento com falha para uma nova especificação. Nesse sentido, Paoli, Sartini e Lafortune (2011) propuseram o uso da controlabilidade segura no CTF ativo. O CTF ativo é a propriedade de continuar a operação de forma segura após uma falha. O uso da controlabilidade segura no CTF ativo será denominada CTF ativo seguro.

O projeto de um CTF ativo seguro envolve: (a) diagnosticar a ocorrência do evento de falha antes do sistema executar uma cadeia proibida; (b) interromper a evolução do sistema antes da execução dessa cadeia proibida; e (c) conduzir o comportamento faltoso para uma especificação pós-falha e que seja segura.

Paoli, Sartini e Lafortune (2011) mostraram que o problema do CTF ativo seguro, em uma arquitetura centralizada, poderia ser resolvido usando uma arquitetura geral de controle baseada no uso de um tipo especial de controlador chamado *diagnosticador-controlador* (*diagnosing-controller*). O diagnosticador-controlador é utilizado para detectar falhas e alterar a política nominal de controle para alguma política de controle correspondente à falha. Para esse método, os autores definiram que o diagnosticador do sistema deve gerar um sinal de interrupção em estados que antecedem um mau estado, desabilitando o supervisor nominal e habilitando um novo supervisor pós-falha, sem a presença da cadeia proibida.

A estratégia de CTF ativo adotada por Paoli, Sartini e Lafortune (2011) considera o caso em que a falha modifica o sistema de forma que a especificação nominal não pode mais ser obtida. Nesse caso, a estratégia consiste em chavear estratégias de controle. Essas estratégias de controle podem ser especificações mais relaxadas ou especificações que devem conduzir o sistema para um estado seguro. Para essas hipóteses de estra-

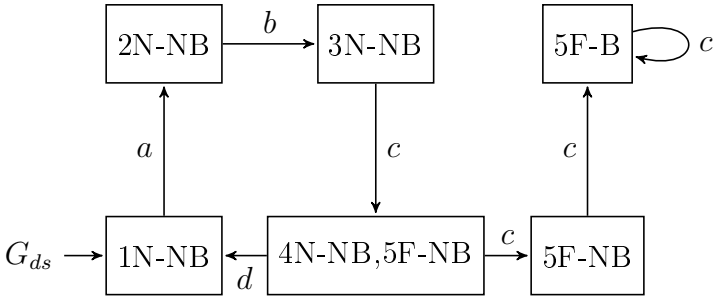
tégia de controle é comum considerar que as especificações pós-falha são diferentes das especificações nominais.

Para obter o CTF ativo seguro, retoma-se o conjunto Q_{ds}^{FC}. Para cada $q_i = \{(x_j, F); (x_k, F), ..., (x_l, F)\} \in Q_{ds}^{FC}$, constrói-se um novo modelo pós-falha S_i^{deg}, utilizando a parte acessível de G^f de todos os estados distintos $x_j, x_k, ..., x_l$ de G^f que aparecem no i-ésimo estado do diagnosticador seguro.

Para ilustrar o CTF ativo seguro e o processo de chaveamento do supervisor nominal para um supervisor pós-falha, é retomado o Exemplo 3. Considera-se agora a necessidade de impedir a cadeia proibida $\xi = cc \in \Phi$, ou seja, não se pode permitir que o evento c ocorra duas vezes seguidas após a falha.

O diagnosticador seguro G_{ds} é mostrado na Figura 12. Pelo diagnosticador seguro é possível constatar que a linguagem é diagnosticável segura, uma vez que o mau estado *5F-B* é um mau estado certo de falha e é alcançado por um estado certo de falha do diagnosticador seguro, e o evento c é controlável (conforme condições da Proposição 7).

Figura 12 – Diagnosticador seguro para exemplo do CTF ativo seguro



Fonte: Elaborado pelo autor (2021).

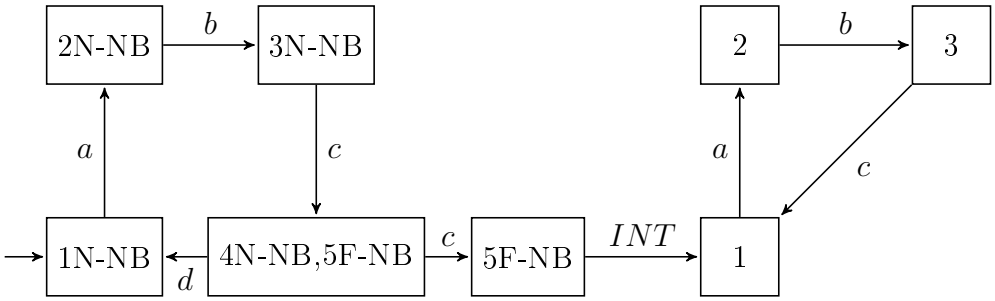
Para o CTF ativo seguro, o diagnosticador seguro G_{ds} deve emitir um sinal de interrupção em seu estado *(5F-NB)* para impedir o supervisor nominal S de atuar e acabar conduzindo o sistema para o mau estado. O sinal de interrupção também altera a política de controle impondo o comportamento do supervisor pós-falha S_i^{deg} apropriado para tratar o problema. Esse processo é o chamado de chaveamento do supervisor.

Na solução do CTF ativo seguro apresentada por Paoli, Sartini e Lafortune (2011), os autores optaram pelo diagnosticador-controlador assumir o controle pós-falha em sua estrutura. Nesse sentido, a interrupção chaveia o supervisor nominal, removendo sua atuação no controle do sistema, e esse controle passa a ser efetuado pelo diagnosticador-

controlador.

Para ilustrar o diagnosticador-controlador, retorna-se ao Exemplo 3. A partir do diagnosticador seguro mostrado na Figura 12 e, uma vez que no conjunto com os primeiros estados certos de falha possui apenas um estado, $Q_{ds}^{FC} = \{(5F - NB)\}$, apenas um supervisor pós-falha é necessário. Esse supervisor pós-falha é conforme o S_1^{deg}, mostrado na Figura 11. Então, no estado *(5F-NB)*, é gerada a interrupção INT, que faz com que não seja mais permitido o evento controlável c e que interrompe o supervisor nominal, passando a atuar com seu modelo pós-falha. O resultado é o diagnosticador-controlador mostrado na Figura 13.

Figura 13 – Diagnosticador-controlador para exemplo do CTF ativo seguro



Fonte: Elaborado pelo autor (2021).

A partir da abordagem de CTF ativo seguro são formuladas as contribuições desta tese. No Capítulo 5 são apresentados resultados preliminares obtidos no contexto desta tese e que são usados como base para a obtenção dos resultados principais, apresentados no Capítulo 6.

5 CONTRIBUIÇÕES AO TEMA DE CONTROLABILIDADE SEGURA DE SEDS

Neste capítulo são apresentadas algumas contribuições relacionadas à controlabilidade segura de SEDs. Essas contribuições são usadas como base para a obtenção dos resultados apresentados no Capítulo 6.

Inicialmente é feito uma revisão na noção de diagnosticabilidade segura e, com base no conceito envolvido, propõe-se um relaxamento das condições para que uma linguagem seja diagnosticável segura. Como resultado, introduz-se o conceito de A-diagnosticabilidade segura. A partir da noção de A-diagnosticabilidade segura introduz-se também a noção de A-controlabilidade segura.

O capítulo também apresenta as noções de diagnosticabilidade segura pelo diagnosticador e controlabilidade segura pelo diagnosticador, noções importantes em relação ao uso do diagnosticador no processo de diagnose de falhas.

Por fim, para quando um sistema não é A-controlável seguro, é introduzida uma forma não condicionada de controlabilidade segura, denominada controlabilidade segura irrestrita.

5.1 RELAXAMENTO DAS CONDIÇÕES PARA A DIAGNOSTICABILIDADE SEGURA E CONTROLABILIDADE SEGURA

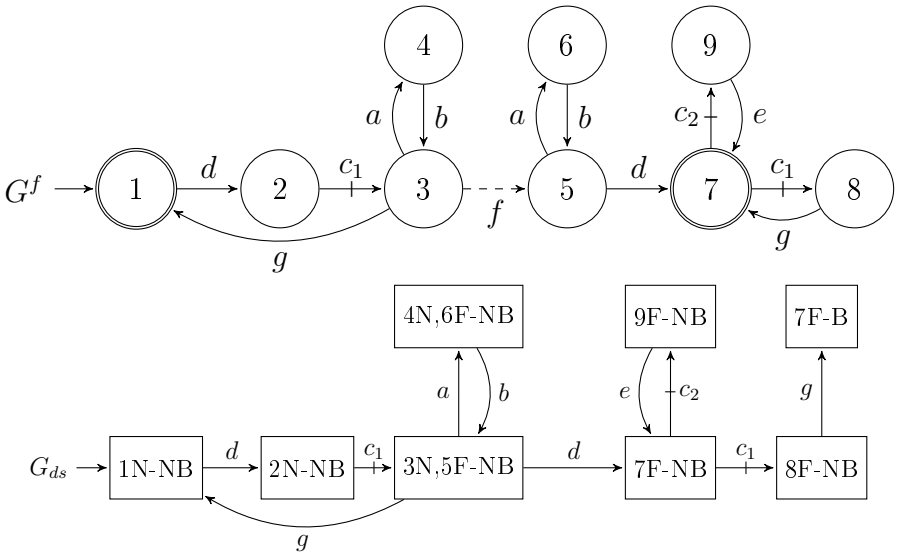
A definição original de controlabilidade segura de uma linguagem apresenta como condição que a linguagem seja diagnosticável segura, que por sua vez apresenta a condição da linguagem ser diagnosticável.

Para introduzir a noção de A-diagnosticabilidade segura, é utilizado um exemplo que ilustra as considerações realizadas para o relaxamento da condição de diagnosticabilidade segura.

Considere para o exemplo que $\Sigma_c = \{c_1, c_2\}$, $\Sigma_{uc} = \{a, b, d, g\}$, $\Sigma_{uo} = \Sigma_f = \{f\}$, $\Sigma_o = \Sigma - \Sigma_{uo}$ e $\Phi = \{g\}$. Planta e diagnosticador seguro são mostrados na Figura 14, sendo que o diagnosticador seguro G_{ds} teve seus estados e transições após o mau estado *7F-B* removidos para facilitar a visualização.

A linguagem $L(G^f)$ não é diagnosticável, uma vez que existe um ciclo indeterminado envolvendo os estados 3 e 4 e os estados 5 e 6, de G^f. Entretanto, tendo em vista

Figura 14 – Modelo de planta com falha G^f e seu diagnosticador seguro G_{ds} para a falha f. O autômato G_{ds} foi truncado e não são mostrados os estados a partir do estado 7F-B



Fonte: Elaborado pelo autor (2021).

que a cadeia ilegal ocorre somente após o estado 7, é possível confirmar a ocorrência da falha antes da execução dessa cadeia. Ou seja, a existência dos ciclos indeterminados, nesse caso, não afeta o diagnóstico seguro. Sendo assim, uma vez que o ciclo não está relacionado com a cadeia proibida (não ocorre uma cadeia proibida enquanto se está no ciclo indeterminado), mesmo a linguagem $L(G^f)$ não sendo diagnosticável, é possível afirmar que, antes da ocorrência da cadeia ilegal, a falha é detectada ao alcançar o estado *7F-NB* do diagnosticador seguro. Além disso, também é possível, ao alcançar o estado 7, impedir o evento c_1 e obter a controlabilidade segura em relação à cadeia ilegal.

A partir da observação realizada utilizando o exemplo, introduz-se as noções de A-diagnosticabilidade segura e da A-controlabilidade segura, sendo ambas as noções contribuições desta tese.

5.1.1 A-Diagnosticabilidade Segura (A-DS)

A noção de A-diagnosticabilidade segura relaxa a condição imposta pela noção de diagnosticabilidade segura, que exige que uma linguagem seja diagnosticável. A Definição 8 formaliza a noção de A-diagnosticabilidade segura.

Definição 8 *(A-diagnosticabilidade segura).* Uma linguagem L prefixo-fechada, viva e que não contém ciclos de eventos não observáveis é dita A-diagnosticável segura e.r.a. P_o, f e $\mathscr{K}_f$ se $(\forall st \in L : s \in \Psi_L(f)$ e $t = u\xi$, com $\xi \in \Phi)$, $(\exists sw < st$ tal que a condição $\mathscr{D}$ é satisfeita para sw e $\overline{w} \cap \mathscr{K}_f = \emptyset)$, sendo que a condição de diagnosticabilidade $\mathscr{D}$ é dada por $\forall v \in P_o^{-1}[P_o(sw)] \cap L \Rightarrow f \in v$.

Em palavras, uma linguagem L é A-diagnosticável segura se as ocorrências de falhas que levam a comportamentos proibidos são diagnosticáveis seguras. Exemplificando a partir da linguagem representada pelo autômato G^f da Figura 14, qualquer cadeia $s \in dc_1(ab + gdc_1)^* f$ é A-diagnosticável segura, uma vez que é possível confirmar a falha antes da ocorrência do cadeia ilegal g após a falha.

A propriedade de A-diagnosticabilidade segura de uma linguagem pode ser verificada a partir de um diagnosticador seguro. A Proposição 1 formaliza a verificação. Para a proposição, reapresenta-se as definições dos conjuntos do diagnosticador seguro introduzidos na Seção 3.4: considere o diagnosticador seguro $G_{ds} = (Q_{ds}, \Sigma_o, \delta_{ds}, q_{ds,0})$ calculado para uma falha f e o conjunto de cadeias proibidas Φ. São definidos os conjuntos $Q_{ds}^N = \{q_{ds} \in Q_{ds} : q_{ds}$ é normal$\}$, $Q_{ds}^C = \{q_{ds} \in Q_{ds} : q_{ds}$ é certo de falha$\}$, e $Q_{ds}^U = \{q_{ds} \in Q_{ds} : q_{ds}$ é incerto de falha$\}$. Define-se também o conjunto dos primeiros estados certos de falha, dado por $Q_{ds}^{FC} = \{q_{ds} \in Q_{ds}^C : [q_{ds} = \delta_{ds}(q_{ds,0}, v_0)$, com $v_0 \in \Sigma_o^*) \wedge (\nexists q'_{ds} \in Q_{ds}^C : q'_{ds} = \delta_{ds}(q_{ds,0}, v'_0)$ tal que $v'_0 < v_0)]\}$. O conjunto Q_{ds}^B é o conjunto de maus estados e definido formalmente como $Q_{ds}^B = \{q_{ds} \in Q_{ds} : q_{ds}$ é um mau estado $\}$.

Proposição 1 *(Condições para a A-diagnosticabilidade segura).* Considere uma linguagem L, não necessariamente diagnosticável, prefixo-fechada, viva e sem ciclos de eventos não observáveis. Seja G um autômato que gera L e um diagnosticador seguro $G_{ds} = (Q_{ds}, \Sigma, \delta_{ds}, q_{ds,0})$ obtido a partir de G. A linguagem L é A-diagnosticável segura e.r.a. P_o, f e $\mathscr{K}_f$ se, e somente se, no diagnosticador seguro:
(i) $\nexists q_{ds} \in Q_{ds}^U : q_{ds} \in Q_{ds}^B$; e
(ii) $\nexists q_{ds} \in Q_{ds}^{FC} : q_{ds} \in Q_{ds}^B$.

Prova. Necessidade e suficiência são provas por contradição.

($\Rightarrow$) Suponha que L não é A-diagnosticável segura, mas que uma das condições impostas no teorema não é atendida. Se a condição (i) não é atendida, então existe um estado no diagnosticador seguro que é incerto de falha e que também é um mau estado. Assuma que esse estado seja alcançado no diagnosticador com a cadeia $u_o = P_o(st)$, com $t = u\xi$, ou seja, $q_{ds} = \delta_{ds}(q_{ds,0}, u_o)$ é tal que $q_{ds} \in Q^U_{ds} \cap Q^B_{ds}$. Então $\nexists sw < st$ tal que a condição $\mathscr{D}$ seja atendida para sw. Se a condição (ii) não é atendida, então o primeiro estado certo de falha alcançado no diagnosticador com uma determinada cadeia é um mau estado. Nesse caso, $\exists q_{ds} \in Q^{FC}_{ds} \cap Q^B_{ds}$ tal que $q_{ds} = \delta_{ds}(q_{ds,0}, P_o(st))$, com $s \in \Psi_L(f)$ e $t = u\xi$, $\xi \in \Phi$. Em ambos os casos, pode-se afirmar que $\exists st \in L : s \in \Psi_L(f)$ e $t = u\xi$, com $\xi \in \Phi$, para a qual $\nexists sw < st$ tal que a condição $\mathscr{D}$ é atendida para sw e $\overline{w} \cap \mathscr{K}_f \neq \emptyset$. Sendo assim, pela Definição 8, pode-se concluir que a linguagem L não é A-diagnosticável segura, contrariando a hipótese inicial.

($\Leftarrow$) Assuma agora que as condições (i) e (ii) são atendidas, mas que a linguagem não é A-diagnosticável segura. Pela Definição 8, se L não é A-diagnosticável segura, então $\exists st \in L : s \in \Psi_L(f)$ e $t = u\xi$, com $\xi \in \Phi$ para a qual $\nexists sw < st$ tal que a condição $\mathscr{D}$ é satisfeita para sw e $\overline{w} \cap \mathscr{K}_f = \emptyset$. Suponha que a condição $\mathscr{D}$ não é atendida para st mas é satisfeita para $st\sigma$, com $\sigma \in \Sigma_o$. Nesse caso, $q_{ds} = \delta_{ds}(q_{ds,0}, P_o(st\sigma))$ é tal que $q_{ds} \in Q^{FC}_{ds}$ e $q'_{ds} = \delta_{ds}(q_{ds,0}, P_o(st)) \in Q^U_{ds}$. Além disso, $q'_{ds} \in Q^B_{ds}$, uma vez que $t = u\xi$, com $\xi \in \Phi$. Sendo assim, $\exists q'_{ds} \in Q^U_{ds} \cap Q^B_{ds}$, o que contraria a hipótese de que a condição (i) era satisfeita. Suponha agora que a condição $\mathscr{D}$ é atendida para $sw = st$. Nesse caso, $\overline{w} \cap \mathscr{K}_f \neq \emptyset$ e então $q_{ds} = \delta_{ds}(q_{ds,0}, P_o(st)) \in Q^{FC}_{ds} \cap Q^B_{ds}$. Assim, $\exists q_{ds} \in Q^{FC}_{ds} \cap Q^B_{ds}$, o que contraria a hipótese de que a condição (ii) era atendida. ⧫

Em palavras, uma linguagem é A-diagnosticável segura se, no diagnosticador seguro, não existir um estado incerto de falha que seja um mau estado e não existir um primeiro estado certo de falha que seja um mau estado. No caso do diagnosticador G_{ds} da Figura 14, tem-se que $Q^B_{ds} = \{(7F - B)\}$, $Q^{FC}_{ds} = \{(7F - NB)\}$ e $Q^C_{ds} = \{(7F - NB), (8F - NB), (9F - NB), (7F - B)\}$. Com isso é possível verificar que a linguagem é A-diagnosticável segura, uma vez que o mau estado *7F-B* é certo de falha e somente alcançado a partir do estado certo de falha *8F-NB*.

A Proposição 1 é similar ao teorema apresentado por Watanabe (2019) para a diagnosticabilidade segura, com a diferença de não exigir que a linguagem L seja diagnos-

ticável.

5.1.2 A-Controlabilidade Segura (A-CS)

Uma vez definida a noção de A-diagnosticabilidade segura (A-DS), é feita a definição da noção de A-controlabilidade segura, que tem como condição necessária a A-DS. A Definição 9 formaliza a noção de A-controlabilidade segura (A-CS).

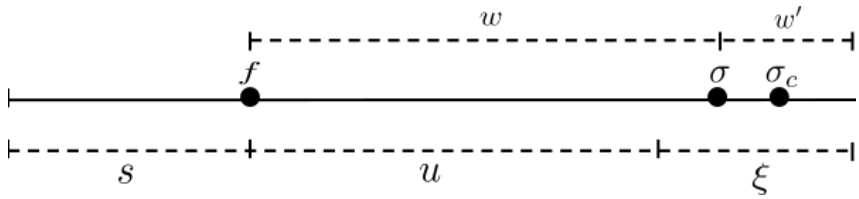
Definição 9 *(A-Controlabilidade segura).* Uma linguagem L prefixo-fechada, viva e que não contém ciclos de eventos não observáveis é dita A-controlável segura e.r.a. P_o, f, $\mathscr{K}_f$ e Σ_c se

(i) L é A-diagnosticável segura; e

(ii) ($\forall su\xi \in L$, com $s \in \Psi_L(f)$ e $u\xi = w\sigma w'$, $\sigma \in \Sigma_o$, tal que a condição $\mathscr{D}$ é atendida para $sw\sigma$ mas não é atendida para sw) $\Rightarrow$ ($\exists \sigma_c \in \Sigma_c : \sigma_c \in w'$).

Em palavras, uma linguagem é A-controlável segura se ela for A-diagnosticável segura e existir um evento controlável após a confirmação da falha e antes de alguma cadeia proibida ser completada. Portanto, o evento controlável σ_c pode estar antes do início da cadeia ilegal (na subcadeia u) ou pode ser um evento da subcadeia ilegal (na subcadeia $\xi \in \Phi$). A definição de A-controlabilidade segura é ilustrada na Figura 15.

Figura 15 – Gráfico para ilustrar a noção de A-controlabilidade segura



Fonte: Elaborado pelo autor (2021).

A A-CS é verificada a partir do diagnosticador seguro. A Proposição 2 apresenta as condições para a A-controlabilidade segura.

Proposição 2 *(Condições para a A-controlabilidade segura).* Considere uma linguagem L, não necessariamente diagnosticável, prefixo-fechada, viva e sem ciclos de eventos não observáveis. Seja G um autômato que gera L e seja $G_{ds} = (Q_{ds}, \Sigma_o, \delta_{ds}, q_{ds,0})$ o diagnosti-

cador seguro obtido a partir de G. Considere que $\Sigma_c \subseteq \Sigma_o$. A linguagem L é A-controlável segura e.r.a. P_o, f, Σ_c e $\mathscr{K}_f$ se, e somente se, as seguintes condições são atendidas:
(i) A-diagnosticabilidade segura: $\nexists q_{ds} \in Q_{ds}^U : q_{ds} \in Q_{ds}^B$ e $\nexists q_{ds} \in Q_{ds}^{FC} : q_{ds} \in Q_{ds}^B$; e
(ii) Controlabilidade segura: $\forall q_{ds} \in Q_{ds}^{FC}$, $\nexists w_o \in \Sigma_{uc}^* : \delta_{ds}(q_{ds}, w_o) = q_{ds,B}$, sendo $q_{ds,B} \in Q_{ds}^B$.

Prova. A prova é feita em duas partes.
($\Rightarrow$) A necessidade é provada por contradição. Suponha que L é A-controlável segura, mas que (i) não é satisfeita. Então $\exists q_{ds} \in Q_{ds}^U : q_{ds} \in Q_{ds}^B$ ou $\exists q_{ds} \in Q_{ds}^{FC} : q_{ds} \in Q_{ds}^B$. Dessa forma, pela Proposição 1, pode-se afirmar que L não é A-diagnosticável segura. Assim, pela Definição 9, L não é A-controlável segura, o que contraria a hipótese inicial. Suponha agora que a condição (i) é atendida, mas que a condição (ii) não é satisfeita. Assim, pela Proposição 1, sabe-se que L é A-diagnosticável segura e que $\nexists q_{ds} \in Q_{ds}^U : q_{ds} \in Q_{ds}^B$ e $\nexists q_{ds} \in Q_{ds}^{FC} : q_{ds} \in Q_{ds}^B$. Se a condição (ii) é violada, então $\exists q_{ds} \in Q_{ds}^{FC}$ para o qual $\exists w_o \in \Sigma_{uc}^*$ tal que $\delta_{ds}(q_{ds,0}, w_o) = q_{ds,B}$, sendo $q_{ds,B} \in Q_{ds}^B$. Sem perda de generalidade, assuma que $q_{ds,B} \in Q_{ds}^B$ é alcançado com um acadeia $s_o z_o = P_o(su\xi)$, com $s \in \Psi_L(f)$ e $\xi \in \Phi$, ou seja, $q_{ds,B} = \delta_{ds}(q_{ds,0}, s_o z_o)$. Assuma ainda que $u\xi = w\sigma w'$, com $\sigma \in \Sigma_o$, e que a condição $\mathscr{D}$ é atendida para $sw\sigma$ mas não é satisfeita para sw. Então $\delta_{ds}(q_{ds,0}, P_o(sw\sigma)) = q_{ds} \in Q_{ds}^{FC}$. Como a condição (ii) não é satisfeita, tem-se que $\exists w_o \in \Sigma_{uc}^*$ tal que $\delta_{ds}(q_{ds}, w_o) = q_{ds,B}$. Dessa forma, $u\xi = w\sigma w'$ é tal que $P_o(w') = w'_o \in \Sigma_{uc}^*$. Tendo em vista que $\Sigma_c \subseteq \Sigma_o$, então $w' \in \Sigma_{uc}^*$, ou seja, $\nexists \sigma_c \in \Sigma_c : \sigma_c \in w'$. Pode-se concluir então que $\exists su\xi \in L$, com $s \in \Psi_L(f)$ e $u\xi = w\sigma w'$, $\sigma \in \Sigma_o$, tal que a condição $\mathscr{D}$ é atendida para $sw\sigma$ mas não é atendida para sw, de modo que $\nexists \sigma_c \in \Sigma_c : \sigma_c \in w'$, o que viola a condição (ii) da Definição 9. Com isso, pode-se afirmar que L não é A-controlável segura, contrariando a hipótese inicial.
($\Leftarrow$) A suficiência também é provada por contradição. Suponha que L não é A-controlável segura, mas que as condições (i) e (ii) são atendidas. Sabe-se que, se L não é A-controlável segura, pelo menos uma das condições da Definição 9 não é válida. Se a condição (i) da Definição 9 não é satisfeita, então, pela Proposição 1, sabe-se que $\exists q_{ds} \in Q_{ds}^U$ ou $\exists q_{ds} \in Q_{ds}^{FC}$ tal que $q_{ds} \in Q_{ds}^B$, o que viola a condição (i), contrariando a hipótese inicial. Considere agora que a condição (i) da Definição 9 é satisfeita, mas que a condição (ii)

dessa mesma definição não é atendida. Considere uma cadeia $su\xi \in L$, com $s \in \Psi_L(f)$ e $u\xi = w\sigma w'$, $\sigma \in \Sigma_o$, tal que a condição $\mathscr{D}$ é atendida para $sw\sigma$ mas não é atendida para sw. Como a condição (ii) da Definição 9 não é satisfeita, então $\nexists \sigma_c \in \Sigma_c$ tal que $\sigma_c \in w'$ e, portanto, $w' \in \Sigma_{uc}^*$. Dessa forma $P_o(w') = w'_o \in \Sigma_{uc}^*$. Sabe-se que $\delta_{ds}(q_{ds,0}, P_o(sw\sigma)) = q_{ds} \in Q_{ds}^{FC}$ e que $\delta_{ds}(q_{ds,0}, P_o(su\xi)) = q_{ds,B} \in Q_{ds}^{B}$. Além disso, para este estado $q_{ds} \in Q_{ds}^{FC}$ tem-se que $\delta_{ds}(q_{ds}, w'_o) = q_{ds,B} \in Q_{ds}^{B}$. Dessa forma, $\exists q_{ds} \in Q_{ds}^{FC}$ para o qual $\exists w'_o \in \Sigma_{uc}^*$ tal que $\delta_{ds}(q_{ds}, w'_o) = q_{ds,B}$, sendo $q_{ds,B} \in Q_{ds}^{B}$, violando a condição (ii) e contrariando a hipótese inicial. ♦

Em palavras, uma linguagem é controlável segura se todos seus maus estados são certos de falha, sem serem os primeiros estados a detectar a falha, e se existir um evento controlável entre o primeiro estado que detecta a falha em cada cadeia pós-falha e qualquer mau estado subsequente a ele. No caso do diagnosticador G_{ds} da Figura 14, tem-se que $Q_{ds}^{B} = \{(7F - B)\}$, $Q_{ds}^{FC} = \{(7F - NB)\}$ e $Q_{ds}^{C} = \{(7F - NB), (8F - NB), (9F - NB), (7F - B)\}$. Com isso é possível verificar que a linguagem é controlável segura, uma vez que o mau estado *7F-B* é certo de falha e somente alcançado a partir do estado certo de falha *8F-NB*, e que entre o primeiro estado que confirma a falha *7F-NB* e o mau estado, existe o evento controlável c_1 que pode ser desabilitado para impedir alcançar o mau estado.

A diferença da Proposição 2 em relação ao apresentado por Watanabe (2019) para a controlabilidade segura pela diagnose, reside no relaxamento da condição (i), que altera a necessidade de que L seja diagnosticável segura, para que ela passe a exigir que L seja A-diagnosticável segura.

A subseção a seguir apresenta uma discussão sobre a definição da A-DS com relação à noção de A-diagnosticabilidade.

5.1.3 Discussão sobre a definição da A-diagnosticabilidade segura

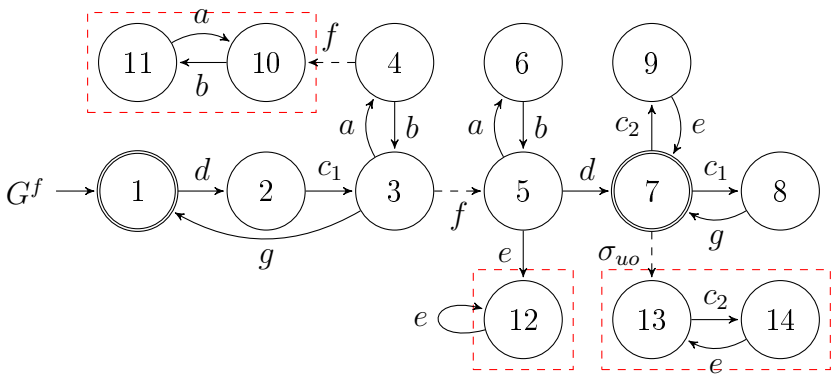
Embora neste trabalho se tenha optado pela denominação A-diagnosticabilidade segura, a A-diagnosticabilidade da linguagem não é considerada como condição necessária para A-DS. Esta subseção é utilizada para explicar o motivo.

O autômato mostrado na Figura 16 é uma alteração do autômato mostrado anteriormente, na Figura 14. Nele foram adicionados novos estados e transições (demarcados

utilizando retângulos tracejados na cor vermelha) para ilustrar as discussões em relação à definição da A-diagnosticabilidade segura.

A linguagem $L(G^f)$ do autômato da Figura 16 não é diagnosticável e também não é A-diagnosticável. A A-diagnosticabilidade relaxa a condição de diagnosticabilidade, exigindo ainda que a falha seja diagnosticada, mas permitindo a presença de ciclos indeterminados na planta. Mesmo assim, em sua definição é exigido que todas as cadeias com a falha tenham sempre continuações que levem à diagnose da falha. É possível verificar em G^f que uma cadeia com o evento de falha que alcança os estados 10 e 11 não possui uma continuação que permita detectar a falha f. Mesmo assim, isso não representa um problema para a A-diagnosticabilidade segura, uma vez que não é possível ocorrer uma cadeia proibida na sequência. Em relação à uma cadeia que alcança o estado 12, nesse caso é possível detectar a falha, mas não existe interesse nessa cadeia, uma vez que, em sua continuação, também não ocorre uma cadeia proibida. O mesmo é válido para uma cadeia que alcança os estados 13 e 14[1].

Figura 16 – Autômato modificado da Figura 14, para discussão sobre a definição da A-diagnosticabilidade segura



Fonte: Elaborado pelo autor (2021).

Sendo assim, a propriedade da A-diagnosticabilidade requer a A-diagnosticabilidade apenas das cadeias que possuem uma continuação que leve a um comportamento proibido (cadeia ilegal). As demais cadeias não precisam ser A-diagnosticáveis.

Além disso, embora a diagnose de falhas considere a garantia de diagnosticar uma falha com um número finito de eventos após a falha, considera-se que, para o controle

[1]Para melhor entendimento de como verificar a A-diagnosticabilidade de uma linguagem, é recomendada a leitura de Lin et al. (2020).

seguro, a garantia da diagnose da falha com esse atraso finito não é importante, desde que não comprometa as condições de segurança. Ou seja, mesmo que nunca seja possível diagnosticar a falha, se essa impossibilidade não resultar na execução de uma cadeia proibida, então o sistema estará seguro.

Na seção a seguir são apresentadas noções importantes para a estrutura de controle introduzida no próximo capítulo. Estas noções envolvem a utilização do diagnosticador para auxílio do sistema no processo de diagnóstico de falhas.

5.2 DIAGNOSTICABILIDADE SEGURA E CONTROLABILIDADE SEGURA PELO DIAGNOSTICADOR

Conforme apresentado na fundamentação teórica, quando os modelos do sistema não satisfazem as condições de diagnosticabilidade, existem alternativas que podem ser adotadas, entre elas a introdução de novos sensores, reais ou virtuais. Essas possibilidades introduzem informações adicionais ao sistema, as quais podem ser usadas para garantir a diagnose de falhas. Além disso, o sistema de controle pode já estar implementado e pode ser inviável alterá-lo (por exemplo, para supervisores com implementação em hardware), ou se deseja apenas adicionar ao sistema um componente de tolerância a falhas (controle seguro).

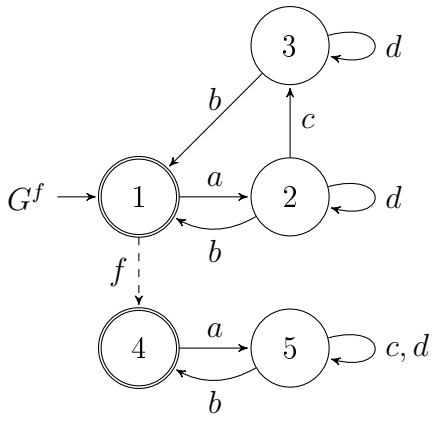
A seguir é introduzida uma noção relacionada com a A-diagnosticabilidade segura, sendo denominada de *A-diagnosticabilidade segura pelo diagnosticador* (A-DSD). Essa noção é apresentada para exemplificar a obtenção da A-diagnosticabilidade segura de um sistema que originalmente não é A-diagnosticável seguro, mas que com o uso de recursos disponíveis no diagnosticador seguro (por exemplo, o uso de mapeamento de sensores) se obtém um diagnosticador seguro capaz de permitir a A-diagnosticabilidade segura do sistema. Assim, para chamar a atenção ao fato deste diagnosticador reconhecer uma linguagem diferente da linguagem da planta, adota-se o termo *pelo diagnosticador*. Essa noção é importante no contexto da proposta do Diagnosticador Controlador Ativo que é introduzido no Capítulo 6.

Para ilustrar a noção de A-DSD é utilizado o Exemplo 4, que apresenta uma situação em que a linguagem da planta não é A-diagnosticável segura.

Exemplo 4 *(linguagem não A-diagnosticável segura).* Considere o modelo de uma planta

G^f mostrado na Figura 17, sendo $\Sigma_o = \{a, b, c, d\}$, $\Sigma_{uo} = \Sigma_f = \{f\}$, $\Sigma_c = \{d\}$ e $\Sigma_{uc} = \Sigma - \Sigma_c$. Considere que haja uma única cadeia proibida, dada por $\xi = d$, i.e., $\Phi = \{d\}$. A linguagem $L(G^f)$ não é A-diagnosticável segura e.r.a. P_o, a falha f e o conjunto Φ, pois nem sempre é possível diagnosticar a falha antes da ocorrência do evento d após a falha.

Figura 17 – Planta G^f para o Exemplo 4, com uma linguagem que não é A-DS

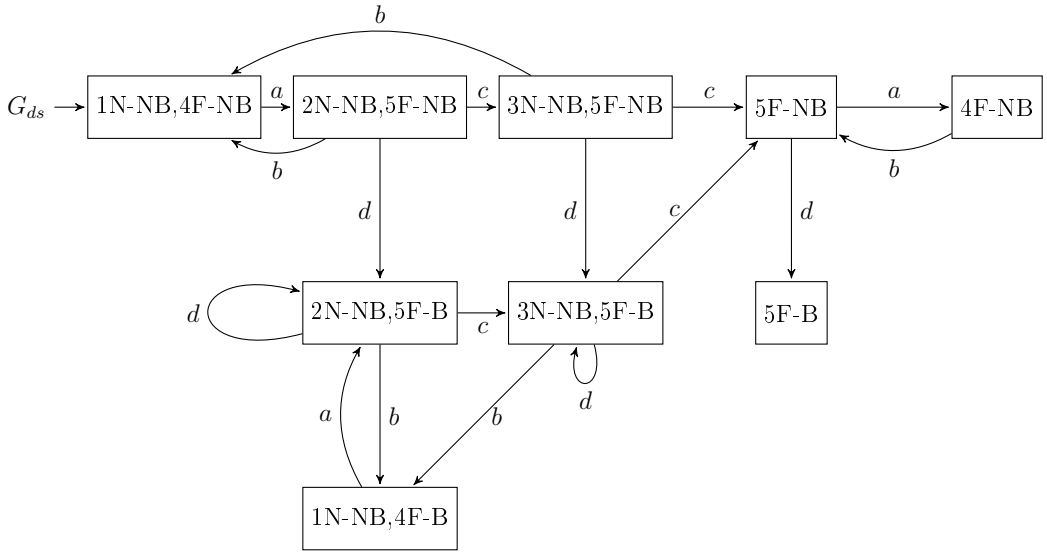


Fonte: Elaborado pelo autor (2021).

A Figura 18 mostra o diagnosticador seguro G_{ds} calculado para a falha f de G^f (foram omitidos os estados e transições após o mau estado para facilitar a visualização). O diagnosticador seguro para a falha mostra a existência de ciclos de estados incertos, mas existe um caminho que leva para a confirmação da falha no estado *5F-NB*. Mesmo assim, a falha não é A-diagnosticável segura, uma vez que a cadeia ilegal d pode ocorrer antes da confirmação da falha, por exemplo, no estado *(2N-NB,5F-B)*. Essa conclusão pode ser facilmente obtida usando as condições apresentadas na Proposição 1. Tendo em vista que existe ao menos um estado incerto com o rótulo B, a condição (i) da referida proposição não é satisfeita, o que leva à conclusão de que a linguagem $L(G^f)$ não é A-DS.

Considerando-se agora a inclusão de informação no diagnosticador seguro a partir do uso da técnica de mapeamento de sensores. Essa informação adicional torna possível a A-diagnosticabilidade segura do sistema. Para o exemplo, considere o diagnosticador seguro G_{ds} construído a partir do mapeamento de um sensor (com um sensor hipotético que assume os valores x e y), mostrado na Figura 19. O mapeamento de sensor considera que, ao ocorrer o evento a antes da falha, o valor do sensor é x e, depois da falha, o valor do sensor é y quando ocorre o evento a. O valor do sensor na ocorrência dos

Figura 18 – Diagnosticador seguro G_{ds} para falha f para o Exemplo 4. O autômato foi truncado e não são mostrados os estados e transições a partir do mau estado certo de falha *5F-B*



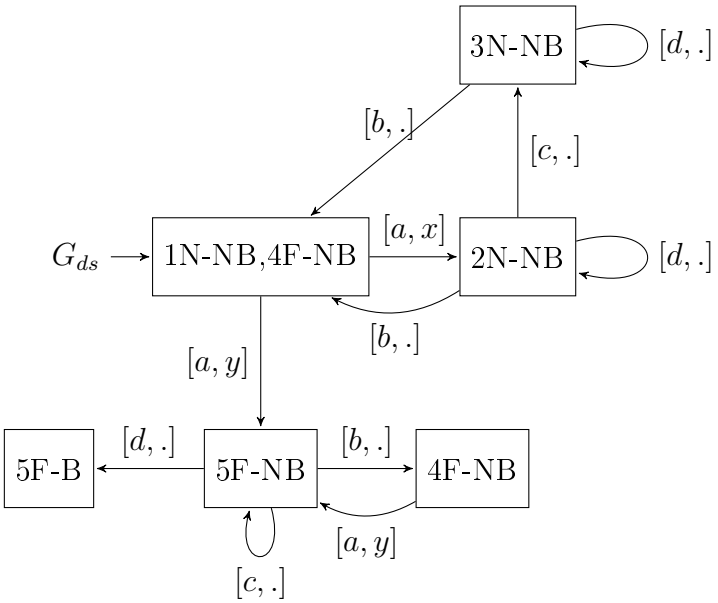
Fonte: Elaborado pelo autor (2021).

demais eventos não é relevante, podendo assumir qualquer um dos valores. As transições com o mapeamento de sensor serão mostradas na forma $[\sigma, sensor]$, sendo $\sigma \in \Sigma_o$ e $sensor \in \{x, y\}$. Para facilitar a visualização, o autômato G_{ds} apresenta o mapeamento de sensores apenas associado com o evento onde ele permite a diferenciação da ocorrência da falha (os demais resultados de mapeamento foram substituídos por ".").

O diagnosticador seguro mostra que, a partir do mapeamento de sensores utilizado, é possível ter informação de quando a falha ocorreu e antes da ocorrência da cadeia ilegal. Sendo assim, embora a linguagem da planta não seja A-diagnosticável segura, com o acréscimo da informação do sensor no diagnosticador seguro, é possível detectar a falha de forma segura.

No exemplo apresentado, planta e diagnosticador possuem observações similares, embora exista a diferença perceptiva do sistema ocasionada pelo mapeamento de sensores. Essa diferença entre a observação da planta e percepção do diagnosticador é denominada *mapeamento perceptivo* e é denotada por P_o^+. No caso de um diagnosticador com mapeamento de sensores, uma cadeia $s \in L(G^f)$ terá o mapeamento perceptivo P_o^+ para os eventos observáveis da planta, acrescido do mapeamento de sensor utilizado no cálculo do diagnosticador. Exemplificando a partir da planta e do diagnosticador seguro do

Figura 19 – Diagnosticador seguro com mapeamento de sensor que permite a A-diagnosticabilidade segura do sistema. O autômato foi truncado e não são mostrados os estados e transições a partir do estado 5F-B



Fonte: Elaborado pelo autor (2021).

Exemplo 4, a cadeia $ab \in L(G^f)$ terá $P_o^+(ab) = [a,x][b,.]$ e a cadeia $fab \in L(G^f)$ terá $P_o^+(fab) = [a,y][b,.]$.

Sendo assim, é utilizado o termo *A-diagnosticabilidade segura pelo diagnosticador* para se referir aos sistemas em que um diagnosticador seguro permite a A-diagnosticabilidade segura de uma falha. Ou seja, mesmo a linguagem de uma planta não atendendo a condição de A-diagnosticabilidade segura, pode existir um diagnosticador seguro que permita ao sistema alcançar essa condição. É importante comentar que, o uso de um diagnosticador para permitir a diagnosticabilidade do sistema quando a linguagem da planta não é diagnosticável, não é uma noção nova. Vale destacar que, quando se acrescentam informações (via mapeamento de sensores), para a obtenção de um diagnosticador, a linguagem em análise passa a não ser exatamente a linguagem da planta, mas sim uma linguagem aumentada com essas informações adicionais. Assim, nos parece interessante tratar de forma diferente essa situação, motivo pelo qual optou-se por adotar o termo para facilitar a apresentação de conceitos que virão em seguida.

É fácil perceber que, quando a linguagem de uma planta é A-diagnosticável segura, ela também é A-DSD e com isso o diagnosticador seguro pode ter a mesma informação que a linguagem observável da planta ($P_o^+ = P_o$). Com isso, é utilizada a A-DSD para se

referir de forma mais geral em relação à diagnosticabilidade segura do sistema, seja pela linguagem da planta ser A-diagnosticável segura, seja porque a A-diagnosticabilidade segura foi alcançada pelo diagnosticador seguro.

De maneira similar à A-DSD, é possível obter a *A-controlabilidade segura pelo diagnosticador* (A-CSD) se for possível diagnosticar a falha a tempo de impedir algum evento controlável que permita o posterior alcance de um mau estado. Mesmo a linguagem de uma planta não atendendo a condição de A-controlabilidade segura, pode existir um diagnosticador que permita diagnosticar a falha de forma segura e a tempo de ser desabilitado algum evento controlável que alcança um mau estado do sistema. Sendo assim, a A-CSD assume que a linguagem é controlável segura se existir um diagnosticador seguro que permite diagnosticar a falha antes da planta completar as cadeias proibidas depois da ocorrência da falha e a tempo de permitir a desabilitação de algum evento controlável que impeça o sistema de alcançar um mau estado.

5.2.1 Considerações sobre a diagnosticabilidade segura e controlabilidade segura pelo diagnosticador

Em diversos sistemas industriais, o projeto de subsistemas de diagnóstico é feito depois do projeto inicial do sistema. Nesse sentido, as condições de diagnosticabilidade podem não ter sido consideradas no projeto inicial do sistema. Por exemplo, em Derbel et al. (2006) é apresentado um sistema no qual se misturam dois produtos e, com a mistura adequada, é atingida uma concentração ideal dentro de um tempo especificado. Quando ocorre a falha, essa concentração não é atingida dentro desse tempo e, com isso, é diagnosticada a falha. Ou seja, essa questão temporal (da concentração ideal dentro de um determinado tempo) pode não ter sido levada em consideração na obtenção da lógica de controle, mas ainda assim pode ser considerada na obtenção do diagnosticador.

Sendo assim, em alguns projetos as falhas são identificadas apenas durante a utilização do sistema e posteriormente são elaboradas formas de diagnosticar a falha. Assim como pode ocorrer com a diagnosticabilidade de uma falha, o mesmo pode ocorrer com as condições de controlabilidade segura, ou seja, a cadeias proibidas podem ser identificadas quando o sistema já está em operação.

Para a situação onde um diagnosticador possui mais informação do que a disponibilizada para obtenção do supervisor, foi apresentado o mapeamento perceptivo para

expressar a relação de uma cadeia observada da planta e seu correspondente percebido em um diagnosticador. Foi ilustrado o mapeamento perceptivo associado ao uso de mapeamento de sensores no diagnosticador.

O mapeamento perceptivo é útil para abstrair a relação de percepção também quando o diagnosticador possui outras naturezas de informação adicional. Por exemplo, o mapeamento perceptivo pode ser utilizado quando a diagnosticabilidade de uma falha é possível a partir de um autômato temporizado ou um autômato estocástico. Diagnosticadores temporizados permitem a detecção de uma falha não somente observando uma sequência de eventos, mas também incluindo alguma informação do tempo da ocorrência dos eventos. Diagnosticadores que utilizam autômatos estocásticos adicionam uma estrutura probabilística para estimar a ocorrência de uma falha.

O mapeamento perceptivo foi utilizado para abstrair a ação síncrona do supervisor e do diagnosticador a partir de uma sequência de eventos observável da planta. Essa relação não é simples de representar. Com relação ao mapeamento de sensores, uma alternativa nesse sentido seria utilizar o mapeamento de sensores baseado na especificação de linguagens (MS-BEL), proposto por Cruz, Carvalho e Basílio (2020). O MS-BEL, altera o mapeamento de sensores utilizando regras usualmente adotadas na especificação da linguagem admissível de SEDs e obtém autômatos que modelam o comportamento dos sensores do sistema.

Mesmo o acréscimo de mais informações em um diagnosticador pode não resultar na A-CSD. Nessa situação, para manter a controlabilidade segura do sistema é preciso realizar ações de controle mais rígidas com relação à falha. Nesse sentido é introduzida a noção de controlabilidade segura irrestrita.

5.3 CONTROLABILIDADE SEGURA IRRESTRITA

Quando um sistema não é A-CS e não se conseguir obter um sistema A-CSD, não é possível ter certeza sobre a ocorrência de alguma de suas falhas antes de poder atuar e desabilitar um evento controlável que impede alguma de suas cadeias proibidas. Observa-se que, quando o sistema não é A-CS, mesmo assim algumas falhas e cadeias proibidas podem permitir a A-DS e a A-CS.

Para fins do CTF, é importante que todas as cadeias proibidas possam ser impedidas de ocorrer. Assim, para o caso de um SED que não é controlável seguro, apresenta-se

a seguir uma uma nova condição de controlabilidade segura, denominada controlabilidade segura irrestrita (CSI), conforme a Definição 10.

Definição 10 *(Controlabilidade segura irrestrita).* Uma linguagem L prefixo-fechada é dita controlável segura irrestrita e.r.a. Σ_c, f e Φ, se $(\forall st \in L : s \in \Psi_L(f)$ e $t = u\xi$, com $\xi \in \Phi)$, $\exists \sigma_c \in \Sigma_c : \sigma_c \in st$.

Em palavras, a noção de linguagem controlável segura irrestrita exige que, para qualquer cadeia com falha seguida de alguma cadeia proibida, exista um evento controlável nesse caminho que possa ser desabilitado para impedir a execução da cadeia proibida. Uma vez que Definição 10 não faz referência ao momento de ocorrência desse evento controlável, ele pode ocorrer após a falha, sendo inclusive um prefixo da cadeia proibida, ou até mesmo pode ocorrer antes da falha.

No caso de haver mais de um evento controlável que possa ser desabilitado para evitar a ocorrência de uma cadeia proibida, pode ser interessante desabilitar o último evento dessa cadeia para que o controle seja minimamente restritivo. Ou seja, para uma cadeia s com uma cadeia proibida após uma falha deve-se desabilitar o evento σ_c tal que σ_c é o evento final da cadeia $r\sigma_c = \Omega(s)$, onde $\Omega(s) := r\sigma_c \in \overline{s} : (\nexists r'\sigma'_c \in \overline{s}$ tal que $r\sigma_c < r'\sigma'_c$, com $r, r' \in \Sigma^*$ e $\sigma_c, \sigma'_c \in \Sigma_c)$.

Assim sendo, como essa definição não leva em consideração aspectos de diagnosticabilidade da linguagem, essa desabilitação pode ocorrer em situações nas quais não se tem certeza sobre a ocorrência de falhas. Nesses casos, o comportamento não faltoso do sistema pode ser restrito de forma protecionista. Logicamente que, se uma linguagem é controlável segura, as desabilitações ocorrerão apenas quando se tem certeza da ocorrência da falha.

A propriedade de CSI pode ser verificada a partir do diagnosticador seguro, conforme apresentado na Proposição 3. Para a proposição, considera-se um autômato $\widehat{G}_{ds} = (\widehat{Q}, \Sigma, \widehat{\delta}, \widehat{q}_0)$ tal que $L(\widehat{G}_{ds}) = P_c[L(G_{ds})]$, sendo $P_c : \Sigma^* \to \Sigma_c^*$. O autômato $\widehat{G}_{ds}$ é obtido da seguinte forma:

Passo 1: Atribuir a Σ_o o conjunto de eventos controláveis, ou seja, todos os eventos controláveis, e somente estes, serão considerados observáveis;

Passo 2: Calcular $\widehat{G}_{ds} = Obs(G_{ds})$ tendo como base Σ_o obtido no passo 1.

Assim como no caso do diagnosticador seguro, um estado $\widehat{q}_{ds}$ pertence a Q_{ds}^{B} se existe ao menos um rótulo B em um dos estados que fazem parte do mesmo.

Proposição 3 *(Condições para a controlabilidade segura irrestrita).* Considere uma linguagem prefixo-fechada L. Seja G um autômato que gera L e seja $G_{ds} = (Q_{ds}, \Sigma_o, \delta_{ds}, q_{ds,0})$ o diagnosticador seguro obtido a partir de G. Seja $\widehat{G}_{ds} = (\widehat{Q}, \Sigma, \widehat{\delta}, \widehat{q}_0)$ tal que $L(\widehat{G}_{ds}) = P_c[L(G_{ds})]$, sendo $P_c : \Sigma^* \rightarrow \Sigma_c^*$. Seja também $\Sigma_c \subseteq \Sigma_o$. A linguagem L é dita controlável segura irrestrita e.r.a. Σ_c, f e Φ se, e somente se, para $\widehat{G}_{ds}$, $\widehat{q}_{ds,0} \notin \widehat{Q}_{ds}^{B}$.

Prova. As condições necessária e suficiente são demonstradas por contradição.
($\Rightarrow$) Suponha que L é controlável segura irrestrita, mas que $\widehat{q}_{ds,0} \in \widehat{Q}_{ds}^{B}$. Como $\widehat{q}_{ds,0} \in \widehat{Q}_{ds}^{B}$, sabe-se que $\exists q_{ds} \in Q_{ds}$ tal que $(q_{ds}, FB) \in \widehat{q}_{ds,0}$, sendo que $q_{ds} = \delta_{ds}(q_{ds,0}, P_o(st))$, com $s \in \Psi_L(f)$ e $t = u\xi$, com $\xi \in \Phi$. Pode-se afirmar ainda que $P_c[P_o(st)] = \varepsilon$, ou seja, a cadeia st não possui nenhum evento controlável que seja observável. Como $\Sigma_c \subseteq \Sigma_o$, conclui-se que $\nexists \sigma_c \in \Sigma_c$ tal que $\sigma_c \in st$. Sendo assim, $\exists st \in L : s \in \Psi_L(f)$ e $t = u\xi$, com $\xi \in \Phi$, para a qual $\nexists \sigma_c \in \Sigma_c : \sigma_c \in st$ e, portanto, L não é controlável segura irrestrita, o que viola a hipótese inicial.
($\Leftarrow$) Suponha agora que $\widehat{q}_{ds,0} \notin \widehat{Q}_{ds}^{B}$, mas que L não é controlável segura irrestrita. Então, pela Definição 10, $\exists st \in L : s \in \Psi_L(f)$ e $t = u\xi$, com $\xi \in \Phi$, para a qual $\nexists \sigma_c \in \Sigma_c$ tal que $\sigma_c \in st$. Como $t = u\xi$, sabe-se que $\delta_{ds}(q_{ds,0}, P_o(st)) = q_{ds} \in Q_{ds}^{B}$. Além disso, como $\nexists \sigma_c \in \Sigma_c$ tal que $\sigma_c \in st$, tem-se que $P_c[P_o(st)] = \varepsilon$ e, dessa forma, $(q_{ds}, FB) \in \widehat{q}_{ds,0}$. Conclui-se então que $\widehat{q}_{ds,0} \in \widehat{Q}_{ds}^{B}$, violando a hipótese inicial. ♦

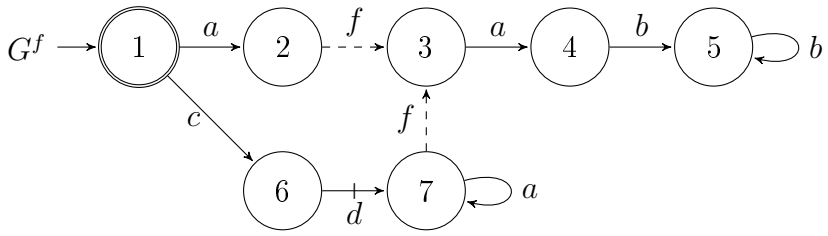
Em palavras, uma linguagem é CSI se, e somente se, o estado inicial de $\widehat{G}_{ds}$ não é um mau estado. Ou seja, a noção de linguagem CSI considera que, para qualquer cadeia com falha seguida de alguma cadeia proibida, a cadeia deve ter algum evento controlável, para que, com a desabilitação desse evento, seja possível impedir o alcance de um mau estado.

Caso exista algum evento controlável não observável, a condição imposta na Proposição 3 deixa de ser necessária, mas continua sendo suficiente. Ou seja, mesmo que o estado inicial de $\widehat{G}_{ds}$ seja um mau estado, ainda assim L pode ser CSI. Por outro lado, se

$\widehat{q}_{ds,0}$ não é um mau estado, então L é CSI.

Para exemplificar o processo de verificação da CSI, uma linguagem não CSI é representada no modelo da Figura 20. Considera-se para o exemplo que, $\Sigma_c = \{d\}$, $\Sigma_{uc} = \{a, b, c, f\}$, $\Sigma_{uo} = \Sigma_f = \{f\}$ e $\Sigma_o = \Sigma - \Sigma_{uo}$.

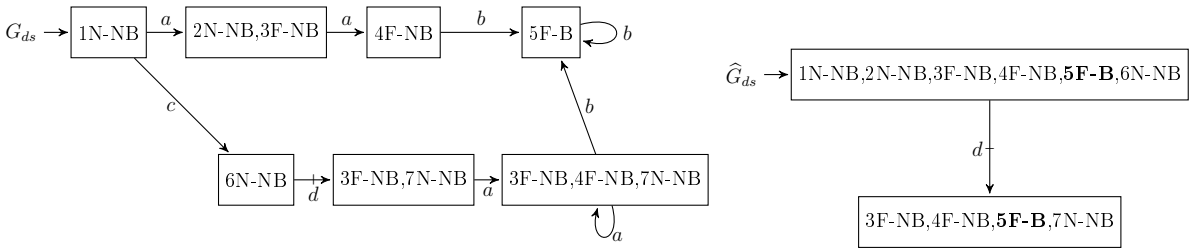
Figura 20 – Modelo G^f para uma linguagem não controlável segura irrestrita



Fonte: Elaborado pelo autor (2021).

Para a controlabilidade segura, considere uma cadeia proibida $\xi = ab \in \Phi$. A Figura 21 mostra o diagnosticador G_{ds} para o modelo G^f da Figura 20. A figura também mostra $\widehat{G}_{ds}$ obtido para o alfabeto dos eventos controláveis Σ_c.

Figura 21 – Diagnosticadores seguro G_{ds} e $\widehat{G}_{ds}$ para verificar a CSI



Fonte: Elaborado pelo autor (2021).

Considerando-se a cadeia $s = afab$, embora seja possível diagnosticar a falha de forma segura, não existe um evento controlável que possa ser usado para impedir o alcance do mau estado e com isso, $\widehat{\delta}(\widehat{q}_0, P_c(s)) = \widehat{q}_0$. Por outro lado, a cadeia $s' = cdfab$ mostra que L não é A-CS (a cadeia $P_o(s')$ alcança o estado *(5F-B)* de G_{ds}, que é um primeiro estado certo de falha e também um mau estado), embora seja possível desabilitar o evento d e impedir alcançar o mau estado antes disso.

Para garantir uma ação de controle que ao mesmo tempo em que é maximamente permissiva ela também garante a não ocorrência de cadeias proibidas, deve-se explorar a combinação das propriedades de A-CSD e CSI, dando prioridade para o uso da A-CSD

sempre que possível. Nesse sentido, o capítulo a seguir define a estrutura de controle seguro pelo Diagnosticador Controlador Ativo.

Em tempo, a controlabilidade segura pode ser considerada um caso especial da controlabilidade segura irrestrita. Essa relação é formalizada a partir da Proposição 4.

Proposição 4 *(Relação entre controlabilidade segura e a controlabilidade segura irrestrita).* Se uma linguagem L é controlável segura, então L é controlável segura irrestrita.

Prova. A prova é uma decorrência imediata da Definição 9 e da Definição 10. ♦

6 PROPOSTA DE UM DIAGNOSTICADOR CONTROLADOR ATIVO PARA O CONTROLE TOLERANTE A FALHAS

A arquitetura para controle tolerante a falhas seguro apresentada por Paoli, Sartini e Lafortune (2011) é baseada em um tipo especial de diagnosticador chamado *diagnosticador-controlador (diagnosing-controller)*, que é composto de um diagnosticador e um conjunto de supervisores pós-falha. Ao alcançar determinados estados certos de falha, uma interrupção é gerada para interromper a operação do supervisor nominal e passar a utilizar a lógica de controle imposta pelo diagnosticador-controlador. Para que essa abordagem possa ser utilizada, é necessária uma linguagem controlável segura.

Conforme apresentado no capítulo anterior, o controle seguro pode ser feito a partir do relaxamento da condição de controlabilidade segura, no caso, a partir da noção de A-diagnosticabilidade segura (A-DS) e A-controlabilidade segura (A-CS).

Quando um sistema não é A-DS e, consequentemente, não é A-CS, as ações para o controle seguro precisam ser tomadas mesmo na incerteza de ocorrência da falha, o que pode provocar restrições de controle sobre o funcionamento nominal do sistema. Para restringir as ações de controle de forma segura na incerteza da falha, é proposta uma nova estrutura de CTF ativo, que faz uso de um novo diagnosticador, denominado *Diagnosticador Controlador Ativo* (DCA). Bloqueios ocasionados pelas restrições impostas são relevados nesta tese, sendo sugerida essa questão como parte dos trabalhos futuros.

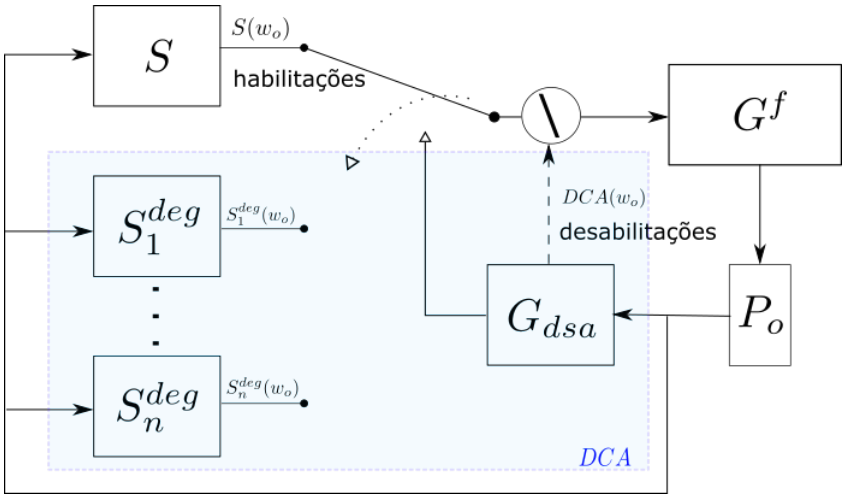
Neste capítulo, é introduzido o DCA e é proposta uma adaptação na estrutura de controle tolerante a falhas ativo apresentada por Paoli, Sartini e Lafortune (2011), a fim de obter uma estrutura de controle tolerante a falhas baseada no uso do DCA. O DCA objetiva garantir a controlabilidade segura do sistema e permitir o chaveamento de supervisores pós-falha após a detecção de falhas, aproveitando a arquitetura de CTF ativa.

O DCA utiliza em sua arquitetura um diagnosticador seguro ativo G_{dsa}, que é responsável por realizar o diagnóstico seguro de falhas e identificar eventos que, com sua desabilitação, garantem o controle seguro. O DCA, por sua vez, incorpora o G_{dsa} e os supervisores pós-falha que são utilizados após interrupção do supervisor nominal.

A Figura 22 ilustra a arquitetura proposta, na qual o supervisor S e o DCA observam eventos da planta. O DCA é a estrutura composta por todos elementos do

bloco destacado em azul, envolvendo G_{dsa} e os supervisores pós-falha S_i^{deg}. Para cada cadeia observada, o supervisor responde com um conjunto de eventos habilitados e o DCA com um conjunto de eventos que devem ser desabilitados para impedir a ocorrência de cadeias proibidas. É realizada então a subtração do conjunto de eventos habilitados pelo supervisor, dos eventos desabilitados pelo DCA. O resultado dessa subtração consiste no conjunto de eventos habilitados, que podem ser executados pela planta, com a garantia da não ocorrência de uma cadeia proibida após uma falha. Sendo assim, o DCA não é somente responsável por diagnosticar a ocorrência de uma falha f (possibilitando o chaveamento para um supervisor pós-falha), mas também é responsável por desabilitar eventos para prevenir cadeias proibidas de ocorrerem. Enquanto o DCA está incerto sobre a ocorrência da falha, ele mantém o supervisor nominal em atuação, mas possivelmente removendo eventos do conjunto de eventos habilitados pelo supervisor. Contudo, quando a falha é diagnosticada, o DCA chaveia/troca o supervisor nominal S por um supervisor pós-falha apropriado (S_i^{deg}). Esse chaveamento pode ocorrer conforme proposto por Paoli, Sartini e Lafortune (2011) para o diagnosticador-controlador.

Figura 22 – Arquitetura de controle tolerante a falhas ativo usando o DCA



Fonte: Elaborado pelo autor (2021).

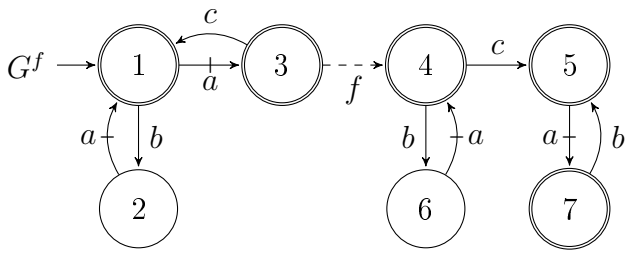
Não faz parte do escopo desta tese apresentar como são obtidos os supervisores pós-falha, nem aspectos associados com a implementação do chaveamento desses supervisores. A solução é apresentada para a situação na qual existe uma falha e um conjunto de cadeias proibidas após essa falha, podendo ou não ser utilizados componentes redundantes após a falha. Quanto ao uso de componentes redundantes, é considerado que eles estão presentes

na planta, podendo ser considerados no modelo da planta para obtenção do supervisor nominal, ou apenas no modelo da planta para obtenção dos supervisores pós-falha.

Para exemplificar o problema da controlabilidade segura quando não existe a A-diagnosticabilidade segura, é utilizado o Exemplo 5.

Exemplo 5 *(exemplo motivacional).* Considere o sistema cujo modelo é apresentado na Figura 23, sendo $\Sigma_o = \{a, b, c\}$, $\Sigma_f = \Sigma_{uo} = \{f\}$, $\Sigma_c = \{a\}$, $\Sigma_{uc} = \Sigma - \Sigma_c$. Para o controle seguro é necessário impedir a cadeia proibida $\Phi = \{a\}$.

Figura 23 – Modelo de planta com falha G^f para o Exemplo 5



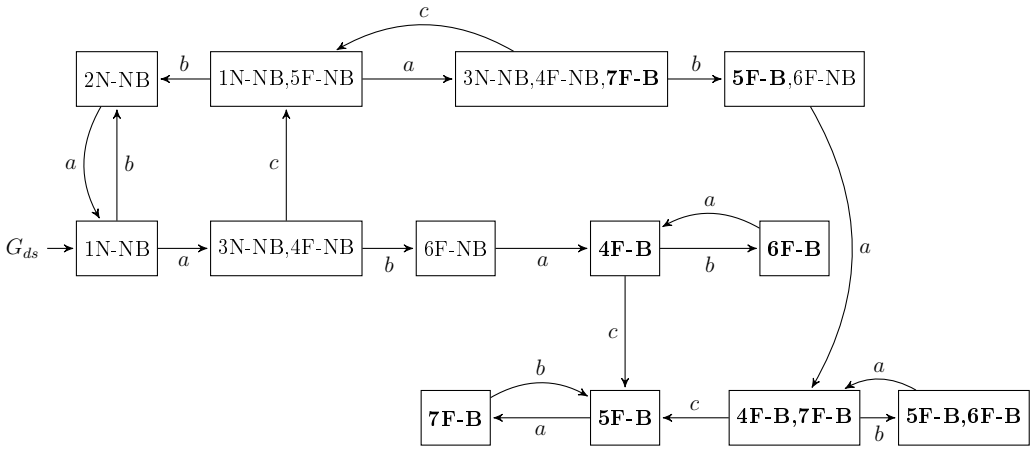
Fonte: Elaborado pelo autor (2021).

O diagnosticador seguro G_{ds} calculado a partir de G^f é mostrado na Figura 24. Com base na definição de controlabilidade segura de Paoli, Sartini e Lafortune (2011), o sistema não é controlável seguro. Conforme a Proposição 1, conclui-se que o sistema não é A-diagnosticável seguro, uma vez que o estado *(3N-NB,4F-NB,7F-B)* é incerto de falha e é um mau estado. Consequentemente, uma vez que o sistema não é A-DS, também não é A-CS uma vez que a condição (i) da Proposição 2 não é atendida. Ou seja, mesmo com o relaxamento da condição de diagnosticabilidade segura, com a noção de A-DS, o sistema não é A-controlável seguro.

Embora a linguagem representada por G^f não seja A-DS, o sistema atende a condição para a CSI, conforme Definição 10. Nesse sentido, a estrutura de controle utilizando o DCA garante que as ações de controle não permitam alcançar os maus estados *(3N-NB,4F-NB,7F-B)* e *(4F-B)* e permite o chaveamento de um supervisor pós-falha no estado *(6F-NB)*. Essas ações de controle, para que não sejam alcançados maus estados, são identificadas a partir de um autômato G_{dsa}, denominado Diagnosticador Seguro Ativo.

A Figura 25 apresenta a sequência de etapas para obtenção do autômato G_{DCA},

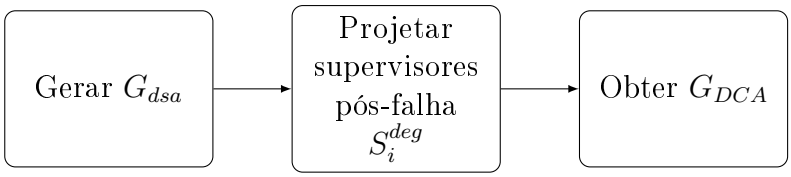
Figura 24 – Diagnosticador G_{ds} calculado para o Exemplo 5



Fonte: Elaborado pelo autor (2021).

que representa o DCA. De maneira sintetizada, as etapas consistem em (i) gerar o diagnosticador seguro ativo G_{dsa}; (ii) calcular os supervisores pós-falha; e (iii) obter G_{DCA}.

Figura 25 – Etapas para obtenção do DCA



Fonte: Elaborado pelo autor (2021).

A seguir, na Seção 6.1, é formalizado o diagnosticador seguro ativo. Na Seção 6.2 é discutido sobre a obtenção dos supervisores pós-falha e do DCA. Na Seção 6.3 são apresentadas considerações sobre o CTF ativo usando o DCA.

6.1 DIAGNOSTICADOR SEGURO ATIVO

Seja $G = (X, \Sigma, \delta, x_0, \Gamma, X_m)$, o diagnosticador seguro ativo para G é um autômato de estados finitos dado por $G_{dsa} = (Q_{dsa}, \Sigma_o, \delta_{dsa}, q_{dsa,0})$, sendo $q_{dsa} = \{(x_1, l_1), (x_2, l_2), ..., (x_n, l_n), CI\}$, com $x \in X$, $l \in \{(N, NB), (F, NB), (F, B)\}$. $CI \in 2^{\Sigma_c}$ é chamado de *Condição de Impedimento*. O eventos existentes na condição de impedimento de cada estado correspondem aos eventos que devem ser desabilitados pelo DCA para evitar a execução de cadeias proibidas naquele estado.

Os rótulos F e N indicam, respectivamente, se evento de falha f ocorreu ou não, enquanto os rótulos B e NB (do inglês, *Bad state* e *Not Bad state*) indicam, respectivamente, se uma cadeia proibida ocorreu ou não depois da falha.

Com relação à rotulação de falha, se um estado $q_{dsa} \in Q_{dsa}$ contém todos seus estados x com rótulo F, ele é chamado de estado certo de falha e indica que a falha já ocorreu em G após observar uma cadeia $u_o \in \Sigma_o^*$ tal que $\delta_{dsa}(q_{dsa,0}, u_o) = q_{dsa}$. Se um estado $q_{dsa} \in Q_{dsa}$ contém todos seus estados x com rótulo N, ele é chamado de estado normal e indica que não há falha em G após observar uma cadeia $u_o \in \Sigma_o^*$ tal que $\delta_{dsa}(q_{dsa,0}, u_o) = q_{dsa}$. Se um estado $q_{dsa} \in Q_{dsa}$ contém ao menos um estado x com rótulo N e ao menos um estado x com rótulo F, ele é chamado estado incerto de falha e indica que ele não tem certeza se a falha já ocorreu em G após observar uma cadeia $u_o \in \Sigma_o^*$ tal que $\delta_{dsa}(q_{dsa,0}, u_o) = q_{dsa}$.

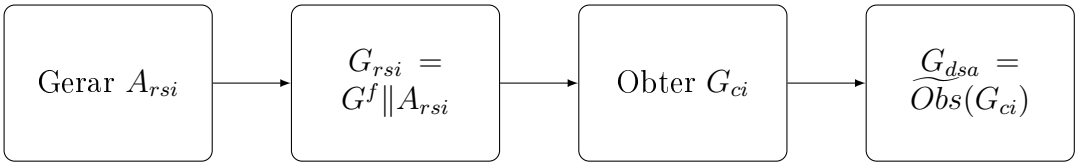
Com relação à propriedade de mau estado, se um estado $q_{dsa} \in Q_{dsa}$ contém algum de seus estados x com um rótulo B, ele é chamado de mau estado e indica que, se o estado x foi alcançado, então uma cadeia proibida ocorreu depois da falha. Se todos os estados x de um estado $q_{dsa} \in Q_{dsa}$ possuem apenas o rótulo NB, então este estado q_{dsa} não é considerado um mau estado e garante que não ocorreu uma cadeia proibida após a falha após observar uma cadeia $u_o \in \Sigma_o^*$ tal que $\delta_{dsa}(q_{dsa,0}, u_o) = q_{dsa}$.

Com relação à condição de impedimento, se um estado $q_{dsa} \in Q_{dsa}$ contém $CI = \emptyset$, significa que q_{dsa} não indica a necessidade de impedir a ocorrência de nenhum evento. Caso contrário, q_{dsa} é chamado de *estado impedidor* para os eventos controláveis que compõem o conjunto CI, o que significa que o estado indica a necessidade de desabilitação desses eventos.

A Figura 26 apresenta uma sequência de etapas para obtenção do autômato G_{dsa}. De maneira sintetizada, as etapas consistem em (i) gerar o autômato rotulador A_{rsi}; (ii) realizar a composição paralela do rotulador A_{rsi} com o modelo da planta com falha, gerando um autômato denominado G_{rsi}; (iii) determinar quais estados possuem evento que, com sua desabilitação, é possível impedir o alcance de um mau estado, obtendo G_{ci}; (iv) calcular um observador de G_{ci} e obter as condições de impedimento para G_{dsa}. Cada uma das etapas é detalhada no decorrer dessa seção.

A seguir é descrita a primeira etapa, que envolve a geração do rotulador para as cadeias proibidas de Φ. Esse autômato rotulador é denominado Reconhecedor Seguro

Figura 26 – Etapas para obtenção de G_{dsa}



Fonte: Elaborado pelo autor (2021).

Interrompido.

6.1.1 Reconhecedor Seguro Interrompido

O primeiro passo para calcular o G_{dsa} é gerar um autômato rotulador A_{rsi}, denominado Reconhecedor Seguro Interrompido (RSI). O RSI identifica a ocorrência das cadeias proibidas após uma falha e rotula os estados da planta, identificando maus estados e estados de falha. O RSI também é responsável por identificar estados que precisam de atenção com relação aos eventos que, com sua desabilitação, podem impedir a execução das cadeias proibidas.

Com relação ao rotulador seguro utilizado por Paoli e Lafortune (2005), o RSI é um rotulador que se diferencia principalmente por não incluir transições a partir do mau estado (o termo *interrompido* denota esse truncamento do autômato). Sua proposta atende a necessidade da controlabilidade segura, que não tem interesse na operação do sistema em maus estados.

Associado aos estados de A_{rsi}, são adicionados os rótulos *N-NB*, *F-NB* e *F-B*, os quais representam, respectivamente, o estado normal e que não é um mau estado, um estado certo de falha e que não é um mau estado e o estado certo de falha que é o mau estado.

Seja $\|\xi\|$ o comprimento de uma cadeia proibida, com $\xi \in \Phi$, o autômato A_{rsi} tem $n = \|\xi\| + 2$ estados. Os estados são conforme segue: o primeiro estado é o estado inicial e nomeado $(q_0 - N - NB)$; o segundo estado é alcançado pelo evento de falha e é nomeado de $(q_1 - F - NB)$; o estado alcançado com a ocorrência de uma cadeia proibida após a falha (quando a cadeia é completada), nomeado com $(q_n - F - B)$ e chamado de *mau estado*. Os outros possíveis estados $(q_i - F - NB)$, com $i = 2, ..., n-2$, são estados intermediários para a completa sequência da cadeia proibida.

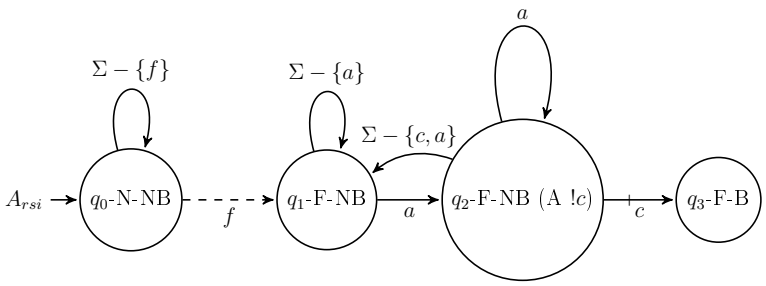
Para a continuidade do processo, é definida a operação $\Omega(s)$ para extrair o mais

longo prefixo da cadeia s cujo último evento da cadeia seja controlável. Seja $s \in \Sigma^*$, $\Omega(s) := r\sigma_c \in \overline{s} : (\nexists r'\sigma'_c \in \overline{s}$ tal que $r\sigma_c < r'\sigma'_c$, com $r, r' \in \Sigma^*$ e $\sigma_c, \sigma'_c \in \Sigma_c)$.

Caso algum evento da cadeia proibida seja controlável, um estado pós-falha do autômato é nomeado como $(q_i - F - NB(A\ !\sigma_c))$, chamado de *estado de atenção*, e que corresponde ao último estado que possui uma transição de partida com o último evento controlável da cadeia proibida, sendo esse evento denotado por $!\sigma_c$ (com $\sigma_c \in \Sigma_c$). Ou seja, o estado de atenção de A_{rsi}, para um evento $\sigma_c \in \Sigma_c$, é o estado alcançado a partir do estado $q_1 - F - NB$ com a cadeia $r \in \Sigma^*$, com $r\sigma_c = \Omega(\xi)$. Caso a cadeia proibida não tenha um evento controlável, então o estado de atenção é o estado inicial de A_{rsi} e nenhum evento controlável é associado a ele.

A Figura 27 mostra o autômato A_{rsi} para um alfabeto $\Sigma = \{a, b, c, f\}$, sendo $\Sigma_o = \{a, b, c\}$, $\Sigma_{uo} = \Sigma_f = \{f\}$, $\Sigma_c = \{b, c\}$ e $\xi = ac \in \Phi$. A partir do mau estado não são adicionadas transições, considerando que, se o mau estado for alcançado, então o controle falhou em impedir a cadeia proibida.

Figura 27 – Autômato A_{rsi} para uma cadeia proibida $\xi = ac \in \Phi$



Fonte: Elaborado pelo autor (2021).

O estado q_2-*F (A !c)* é responsável por indicar, posteriormente, que pode ser necessário impedir o evento c em estados que tenham essa indicação. Estados rotulados com a condição $(A\ !\sigma_c)$ são facilitadores para identificar os estados que precisam de posterior avaliação para calcular G_{ci}.

Se uma cadeia proibida não contém eventos controláveis, A_{rsi} terá o estado inicial como o estado de atenção. Nesse caso, uma vez que não se tem indicação de qual evento deve ser impedido (considerando apenas a própria cadeia proibida, não se sabe qual evento controlável poderá impedir a ocorrência dessa cadeia), a etapa posterior terá que computar qual evento deve ser impedido. Em tal situação, um evento pode ser impedido antes

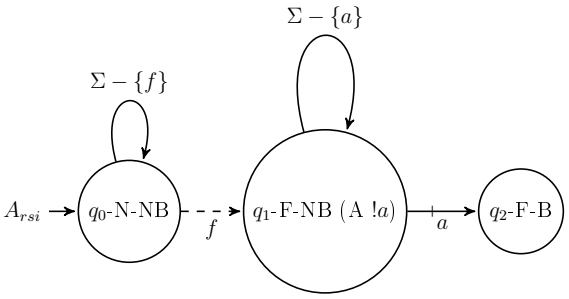
mesmo da falha ocorrer.

No caso de haver mais de uma cadeia proibida, i.e., $\Phi = \{\xi_1, \xi_2, ...\xi_n\}$, obtém-se um $A_{rsi,j}$ para cada $\xi_j \in \Phi$, com $j \in \{1, ..., n\}$, sendo $n = \|\Phi\|$. Nesse caso, para obter o RSI para todas as cadeias de Φ, é preciso realizar a composição paralela de cada A_{rsi} gerado para $\xi \in \Phi$. Ou seja, $A_{rsi} = A_{rsi,1}\|...\|A_{rsi,n}$, sendo $n = \|\Phi\|$. Por simplicidade, o trabalho considerará, em sua maioria, situações com apenas uma cadeia proibida no conjunto Φ.

Além disso, é importante destacar que, cada falha $f_i \in \Sigma_f$ pode ter seu conjunto de cadeias proibidas Φ_i que devem ser impedidas após a falha. Com isso, para cada falha f_i, cujo $\Phi_i \neq \emptyset$, é preciso gerar um rotulador A_{rsi} para a falha. Por simplicidade, considera-se a existência de um único tipo de falha, ou seja, $\Sigma_f = \{f\}$.

Exemplificando a obtenção de A_{rsi} para o Exemplo 5, uma vez que $\xi = a \in \Phi$ e que $a \in \Sigma_c$, o resultado é o RSI mostrado na Figura 28.

Figura 28 – Rotulador A_{rsi} para a cadeia proibida $\xi = a \in \Phi$



Fonte: Elaborado pelo autor (2021).

Após gerar o rotulador para as falhas e para os maus estados, o próximo passo consiste em realizar a composição da planta com o rotulador, etapa mostrada na subseção a seguir.

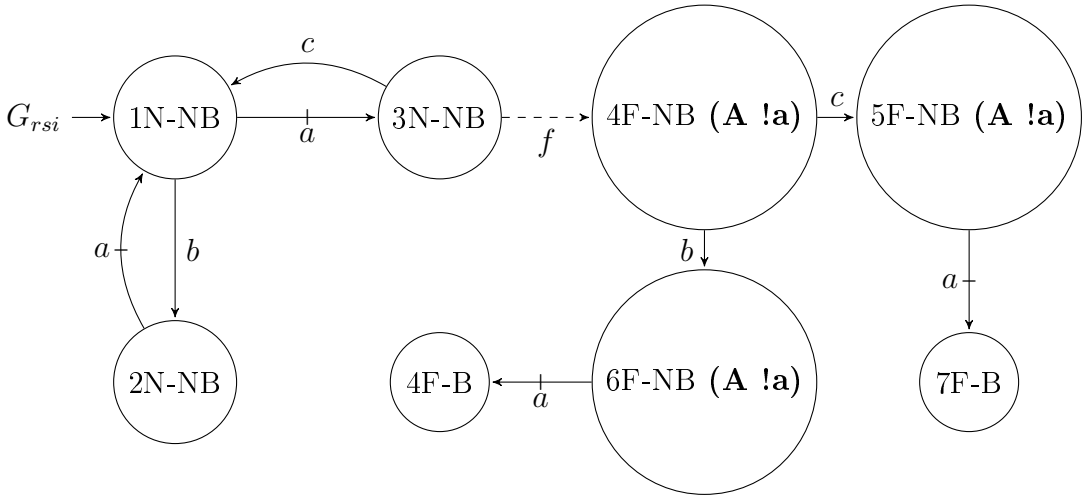
6.1.2 Obtenção de G_{rsi}

O autômato $G_{rsi} = G^f\|A_{rsi}$ é computado a partir da composição paralela do modelo da planta com falha G^f e A_{rsi}, em que A_{rsi} é o RSI gerado para o conjunto de cadeias proibidas Φ.

Na Figura 29 é mostrado o G_{rsi} obtido para o Exemplo 5. O G_{rsi} apresenta, em cada um de seus estado, sua propriedade com relação à falha e identificação se o estado é

um mau estado ou não. Também são rotulados os estados que precisam de atenção.

Figura 29 – $G_{rsi} = G^f \| A_{rsi}$ obtido para o Exemplo 5



Fonte: Elaborado pelo autor (2021).

Com relação aos estados de atenção, é possível identificar três estados de atenção, sendo esses os estados *(4F-NB (A !a))*, *(5F-NB (A !a))* e *(6F-NB (A !a))*. Esses três estados são estados de atenção para o evento $a \in \Sigma_c$.

A seguir é apresentada como é realizada a atribuição de condições de impedimento em G_{ci}, a partir de G_{rsi}.

6.1.3 Obtenção de G_{ci}

O G_{ci} é o resultado da avaliação dos estados de atenção em G_{rsi}, identificando efetivamente que eventos controláveis devem ser impedidos para que o sistema não evolua para um mau estado e determinando o conjunto de impedimentos para cada estado de G_{ci}.

Para determinar as condições de impedimento dos estados de G_{ci} é necessário considerar as seguintes possibilidades para cada cadeia proibida do conjunto Φ: (i) a cadeia proibida contém pelo menos um evento controlável; e (ii) a cadeia proibida não contém evento controlável.

Seja uma cadeia $\xi = r\sigma_c w$ com $r \in \Sigma^*$, $\sigma_c \in \Sigma_c$ e $w \in \Sigma_{uc}^*$. Então, $\Omega(\xi) = r\sigma_c$ e $\xi/r = \sigma_c w$. Deve-se identificar se os estados de atenção de G_{rsi} que possuem uma transição de saída com a cadeia $\sigma_c w$, i.e., se $x_{ci,i} \in X_{ci}$ é um estado de atenção com o evento σ_c para a cadeia proibida ξ e $\sigma_c w \in L(G_{ci}, x_{ci,i})$, então σ_c deve ser adicionado ao conjunto de

impedimentos CI do estado $x_{ci,i}$. Vale destacar que, se $\sigma_c w \in L(G_{ci}, x_{ci,i})$, então o estado alcançado com essa transição é certamente um mau estado, i.e., $\delta_{ci}(x_{ci,i}, \sigma_c w) = x_{ci,j} \in X_{ci}$, onde $x_{ci,j}$ é um mau estado.

Para a situação na qual a cadeia proibida não tem eventos controláveis, a computação do conjunto de impedimentos é mais custosa, uma vez que todos os estados certos e incertos de falha de G_{rsi} serão estados de atenção. Nessa situação, é necessário identificar quais eventos controláveis podem impedir o alcance do mau estado e em que estados esse impedimento é necessário. Isso pode ser feito percorrendo as transições de estado pra trás, a partir do mau estado em direção ao estado inicial, até encontrar uma transição com um evento controlável que, se desabilitado, impedirá o alcance do mau estado. Sendo assim, para toda cadeia $sv\xi$ que alcança um mau estado, deve ser impedido seu último evento controlável $\sigma_c \in \Sigma_c$, ou seja, para toda cadeia $sv\xi \in L(G^f)$ tal que $\delta(x_{rsi,0}, sv\xi) = x_{rsi,j} \in X_{rsi}$, em que $x_{rsi,j}$ é um mau estado, deve ser encontrado um evento σ_c, tal que $r\sigma_c \in \Omega(sv\xi)$, e acrescentado o evento σ_c no conjunto de impedimentos do estado alcançado por $\delta(x_{rsi,0}, r)$.

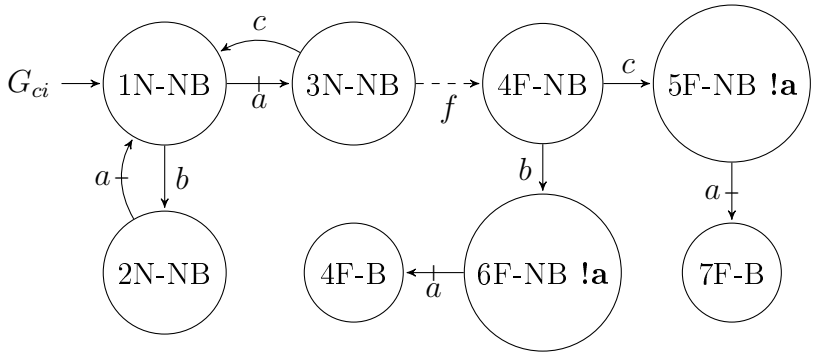
Retoma-se o Exemplo 5 para ilustrar a atribuição dos conjuntos de impedimentos. Para o exemplo, $\xi = a$ e G_{rsi}, mostrado na Figura 29, existem três estados de atenção. Iniciando a avaliação a partir do estado $x_{rsi,i}$ = *(4F-NB)*, uma vez que $\delta_{rsi}(x_{rsi,i}, \xi)$ não é definida, isto é, o estado *(4F-NB)* não possui função de transição de saída com a cadeia $a \in \xi$, então esse estado não terá a condição de impedimento com esse evento. Considere agora o estado $x_{rsi,j}$ = *(5F-NB)*. Uma vez que, δ_{rsi}(*(5F-NB)*, a) é definida (alcançando um mau estado), o estado *(5F-NB)* recebe o evento a como seu conjunto de impedimentos. Sendo assim, o estado *(5F-NB (A !a))* de G_{rsi}, que é um estado de atenção, é avaliado e torna-se o estado *(5F-NB !a)* de G_{ci}. De maneira similar, ao estado *(6F-NB)* também é atribuído o evento a ao seu conjunto de impedimentos. A Figura 30 mostra o resultado dessa operação.

A partir da obtenção de G_{ci} e do conjunto de impedimentos de cada estado, a última etapa é responsável por obter G_{dsa}.

6.1.4 Obtenção de G_{dsa}

A obtenção do diagnosticador seguro ativo é finalizada com a operação $G_{dsa} = \widetilde{Obs}(G_{ci})$. Essa operação envolve calcular o observador de G_{ci} em relação ao conjunto Σ_o,

Figura 30 – G_{ci} obtido para o Exemplo 5.



Fonte: Elaborado pelo autor (2021).

e determinar as condições de impedimento para os estados resultantes em G_{dsa}. Ou seja, a operação $\widetilde{Obs}(G_{ci})$, após o cálculo do observador, reorganiza as informações do estado, com a união dos conjuntos de impedimento que fazem parte de cada estado interno de um mesmo estado q_{dsa}.

Duas possibilidades se evidenciam a partir do cálculo do observador: (i) o estado resultante representa apenas um único estado de G_{ci} e, consequentemente, da planta; e (ii) o estado resultante representa mais de um estado de G_{ci}.

Para a primeira possibilidade, $q_{dca} = x_{ci}$ e, portanto, q_{dsa} terá seu conjunto de impedimentos igual ao conjunto de impedimentos de x_{ci}.

Para a segunda possibilidade, com o cálculo do observador, um estado $q_{dsa} \in Q_{dsa}$ pode reunir dois ou mais estados $x_{ci} \in X_{ci}$. Considerando o agrupamento de dois estados, na forma $\{(x_i, l_i, CI_i), (x_j, l_j, CI_j)\}$, eles serão reorganizados para $q_{dsa,k} = \{(x_i, l_i), (x_j, l_j), CI_k\}$, com $CI_k = CI_i \cup CI_j$.

A partir do conjunto de impedimentos de cada estado de G_{dsa} é definido o conjunto de eventos $\Sigma_{cond} \subseteq \Sigma_c$, que representa os eventos contidos nessas condições de impedimento. Ou seja, Σ_{cond} possui os eventos controláveis que os estados de G_{dsa} podem solicitar a desabilitação. Formalmente Σ_{cond} é obtido por: $\forall q_{dsa,k} \in Q_{dsa}$, $q_{dsa,k}$ na forma (X_{ci}, CI_k), $\bigcup_{k=1}^{m} CI_k$, onde $m = \|Q_{dca}\|$.

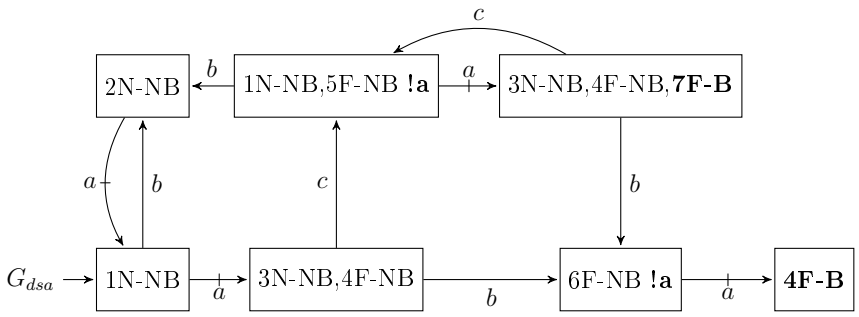
Os estados $q_{dsa} \in Q_{dsa}$ que tiverem algum evento em seu conjunto de impedimentos são denominados *estados impedidores* dos eventos contidos nesse conjunto. Desta forma, são definidos os conjuntos $Q_{dsa}^{ci,\sigma_c} = \{q_{dsa} \in Q_{dsa} : q_{dsa} = \{(x_i, l_i), (x_j, l_j)..., CI\}, \sigma_c \in CI\}$, ou seja, composto dos estados que possuem a condição de impedimento com evento σ_c.

São definidos conjuntos Q_{dsa}^{ci,σ_c} para cada evento $\sigma_c \in \Sigma_{cond}$, podendo um estado estar em mais de um desses conjuntos.

Na seção 5.2 foi apresentada a noção de diagnosticabilidade segura pelo diagnosticador, que considera informações adicionais que o diagnosticador seguro possui, para permitir as condições de A-diagnosticabilidade segura e A-controlabilidade segura. Uma vez que G_{dsa} possui as mesmas capacidades do diagnosticador seguro, no sentido de identificar falhas e maus estados, as mesmas considerações de uso de G_{ds} se aplicam ao G_{dsa}. Assim, de maneira similar ao definido para o diagnosticador seguro, são definidos os conjuntos Q_{dsa}^{B}, Q_{dsa}^{C}, Q_{dsa}^{U} e Q_{dsa}^{N} para representar, respectivamente, o conjunto de estados que são maus estados, o conjunto de estados certos de falha, o conjunto de estados incertos de falha e o conjunto de estados normais.

Continuando com a exemplificação das etapas para obtenção de G_{dsa}, na Figura 31 mostra-se o resultado alcançado com a finalização do processo de obtenção de G_{dsa} para o Exemplo 5.

Figura 31 – Resultado de $G_{dsa} = \widetilde{Obs}(G_{rsi})$ para o Exemplo 5



Fonte: Elaborado pelo autor (2021).

A partir do G_{dsa} da Figura 31, é possível definir os conjuntos $Q_{dsa}^{B} = \{$*(4F-B),(3N-NB,4F-NB,7F-B)*$\}$, $Q_{dsa}^{FC} = \{$*(6F-NB !a)*$\}$ e $Q_{dsa}^{U} = \{$*(1N-NB,5F-NB !a),(3N-NB,4F-NB,7F-B),(3N-NB,4F-NB)*$\}$. A partir desses conjuntos e da Proposição 1, é possível concluir que a linguagem de G^f da Figura 23 não é A-diagnosticável segura, uma vez que existe um mau estado que também é um estado incerto de falha, ou seja, $Q_{dsa}^{B} \cap Q_{dsa}^{U} \neq \emptyset$.

Uma vez que um dos objetivos do DCA é impedir o alcance de maus estados, ou seja, interromper a evolução do sistema a partir da desabilitação de eventos controláveis indicados nos estados impedidores do evento, as transições com esses eventos podem ser removidas do autômato G_{dsa}. Essa etapa é considerada opcional, mas se for utilizada é

importante observar que não mais existirão maus estados em G_{dsa} (todos os estados serão rotulados como NB). Sendo assim, essa informação sobre os maus estados não é mais útil e pode ser retirada de G_{dsa}. Além disso, a remoção das transições pode ocasionar a existência de estados não acessíveis, então é preciso calcular a componente acessível resultante. Também é importante observar que a remoção dos maus estados do modelo prejudica a análise das condições de diagnosticabilidade segura. Portanto, a remoção das transições é considerada apenas para fins de implementação do modelo, embora também seja utilizada para simplificar os exemplos apresentados.

No caso de G_{dsa} mostrado na Figura 31, se for considerada a remoção da transição de partida com o evento que está na condição de impedimento do estado, sua aplicação remove os estados *(3N-NB,4F-NB,7F-B)* e *(4F-B)* e suas respectivas transições de partida e chegada. O cálculo da componente acessível, embora não necessário nesse exemplo, garante que apenas estados acessíveis permanecem no modelo.

Ainda com relação ao exemplo, $\Sigma_{cond} = \{a\}$ e $Q_{dca}^{ci,a} = \{$*(1N,5F-NB !a)*, *(6F-NB !a)*$\}$. A partir dos estados desse conjunto de impedimento, o DCA pode desabilitar eventos que conduzem o sistema para um comportamento inseguro, obtendo a controlabilidade segura.

A seguir são apresentados exemplos que complementam aspectos relacionados com o obtenção de G_{dsa}.

6.1.5 Exemplificação do processo de obtenção de G_{dsa}

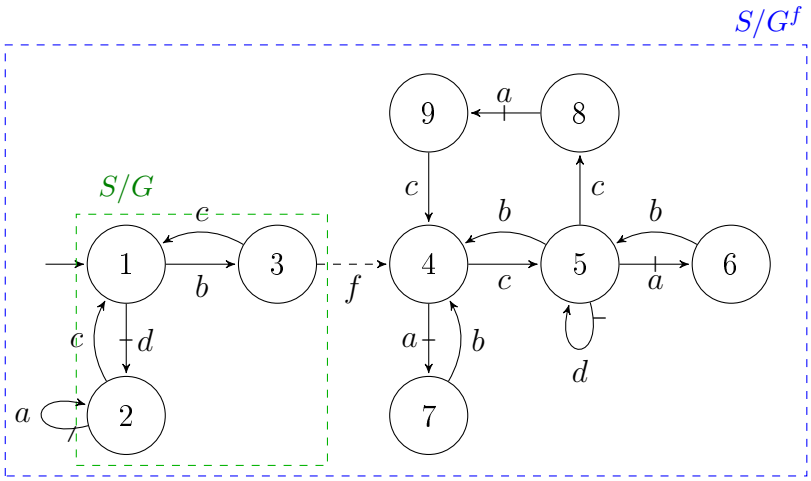
A seguir são apresentados três exemplos. Inicialmente, no Exemplo 6, é exemplificada a obtenção de um G_{dsa} com duas cadeias proibidas.

Exemplo 6 *(duas cadeias proibidas).* Para o exemplo, considere o modelo mostrado na Figura 32, sendo $\Sigma_o = \{a, b, c, d\}$, $\Sigma_f = \Sigma_{uo} = \{f\}$, $\Sigma_c = \{a, d\}$, $\Sigma_{uc} = \Sigma - \Sigma_c$.

Destacado na cor verde está o comportamento nominal do sistema S/G (o comportamento nominal do sistema sob a ação de um supervisor nominal), enquanto o modelo completo mostra o sistema com falha sendo controlado pelo supervisor nominal S/G^f.

Para o controle seguro do sistema é necessário impedir o conjunto de cadeias proibidas formado por $\Phi = \{ab, d\}$. Desta forma são gerados dois rotuladores seguros interrom-

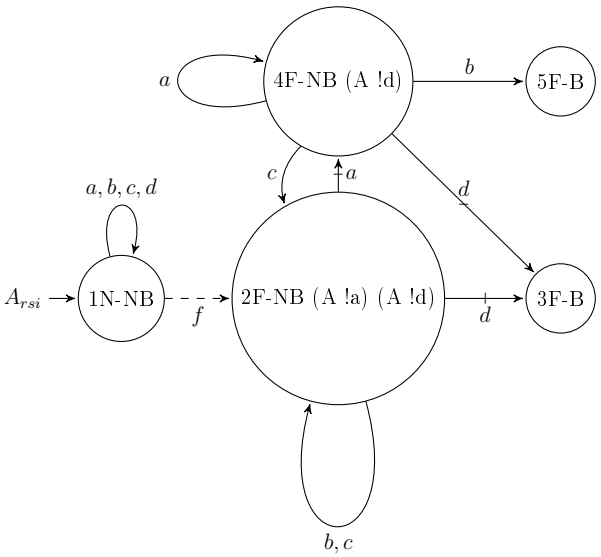
Figura 32 – Comportamento do sistema S/G^f e S/G para o Exemplo 6



Fonte: Elaborado pelo autor (2021).

pidos $A_{rsi,1}$ e $A_{rsi,2}$, um para cada cadeia proibida, sendo $A_{rsi} = A_{rsi,1} \| A_{rsi,2}$ (mostrado na Figura 33).

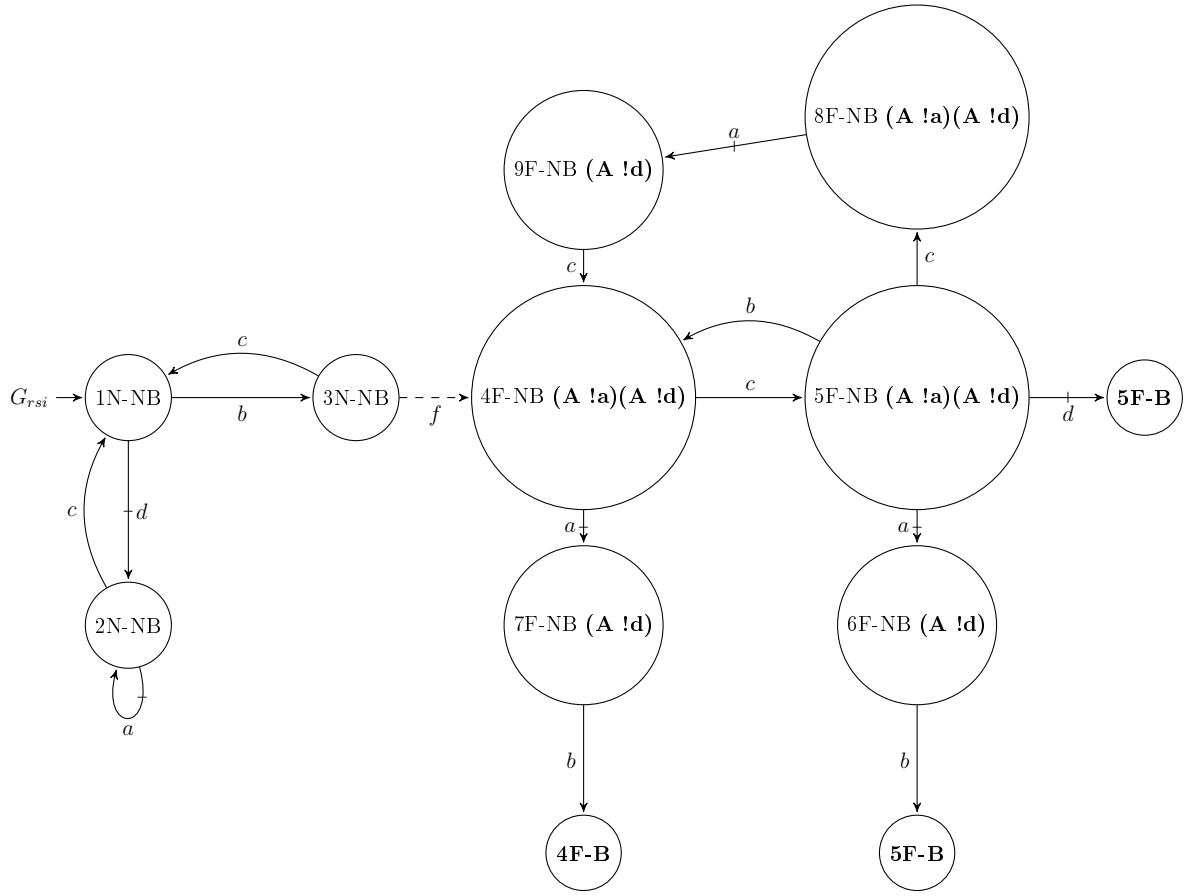
Figura 33 – A_{rsi} para as cadeias proibidas $\Phi = \{ab, d\}$ do Exemplo 6.



Fonte: Elaborado pelo autor (2021).

A Figura 34 mostra o autômato G_{rsi} obtido de $G_{rsi} = G^f \| A_{rsi}$. Nele é possível observar a indicação de seis estados de atenção. Na Figura 35 é mostrado o autômato G_{ci}, com as condição de impedimento definidas a partir da avaliação dos estados de atenção de G_{rsi}.

Figura 34 – Resultado de $G_{rsi} = G^f \| A_{rsi}$ para o Exemplo 6.

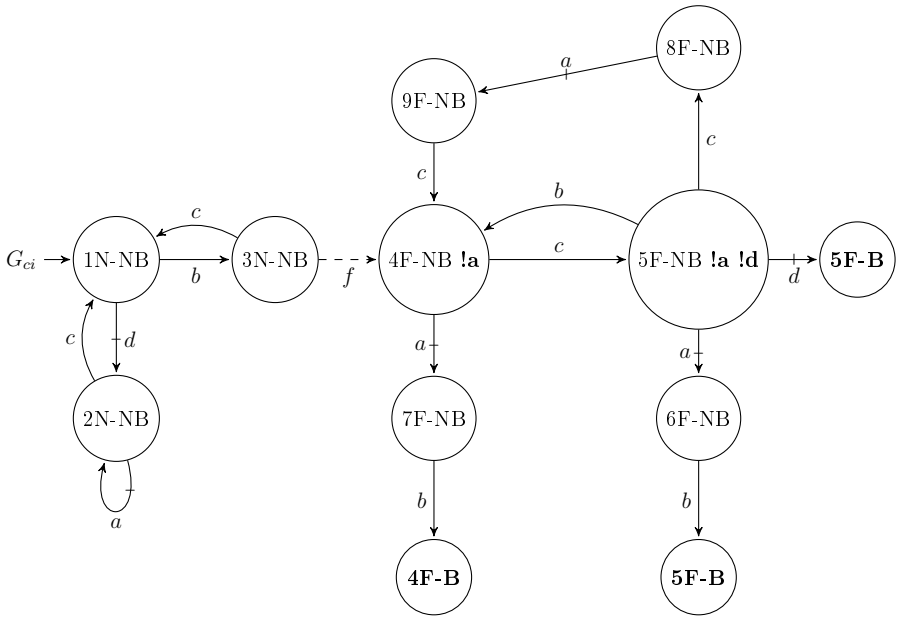


Fonte: Elaborado pelo autor (2021).

Dos seis estados de atenção em G_{rsi}, os estados *(7F-NB (A !d))*, *(8F-NB (A !d))* e *(9F-NB (A !d))* são estado de atenção para o evento d. Como esses estados não possuem função de transição definida com o evento d, seus estados correspondentes em G_{ci} não terão condição de impedimento. Dos três estados de atenção restantes, o estado *(8F-NB (A !a) (A !d))* não possui função de transição estendida definida com a cadeia proibida ab, assim como não possui função de transição definida para a cadeia proibida d. Sendo assim, esse estado não alcança um mau estado imediatamente com essas duas cadeias proibidas. Ou seja, uma vez que $\delta_{rsi}((8F\text{-}NB\ (A\ !a)\ (A\ !d)), ab)$ e $\delta_{rsi}((8F\text{-}NB\ (A\ !a)\ (A\ !d)), ab)$ não são definidas, o estado *(8F-NB (A !a) (A !d))* também não terá condição de impedimento em G_{ci}. O estado *(4F-NB (A !a) (A !d))* não possui função de transição definida para a cadeia proibida d, embora tenha função de transição estendida definida para a cadeia proibida ab, sendo então definida a condição de impedimento do estado *(4F-NB (A !a) (A !d))* em G_{ci} com o evento a. O estado de atenção *(5F-NB (A !a) (A*

!d)) possui função de transição definida para as duas cadeias proibidas, sendo então os eventos de atenção do estado (eventos a e d) adicionados ao conjunto de impedimentos desse estado.

Figura 35 – Resultado de G_{ci} para o Exemplo 6.



Fonte: Elaborado pelo autor (2021).

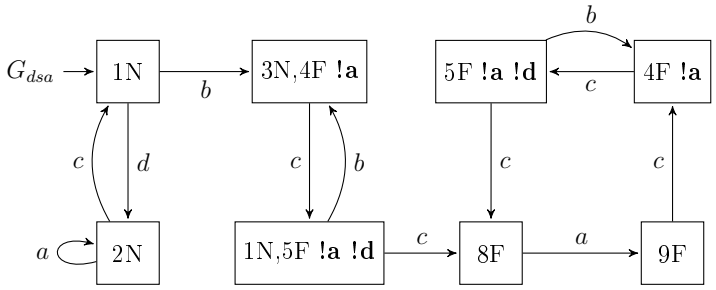
A partir do cálculo $G_{dsa} = \widetilde{Obs}(G_{ci})$ é obtido G_{dsa} (mostrado na Figura 36). Para esse exemplo, optou-se pela remoção das transições de partida com o evento do qual o estado de partida é um estado impedidor do evento. Nesse sentido também foi removido o rótulo que indica que estados são maus estados e quais não são (uma vez que todos os estados resultantes não são maus estados). É importante observar que, a partir da remoção dessas transições, a análise da diagnosticabilidade é prejudicada, não sendo mais possível verificar a A-diagnosticabilidade do sistema.

Por fim, para esse exemplo, $\Sigma_{cond} = \{a, d\}$ e existirão dois conjuntos de estados impedidores, sendo eles $Q_{dsa}^{ci,a} = \{$*(1N,5F !a !d),(3N,4F !a),(5F !a !d),(4F !a)*$\}$ e $Q_{dsa}^{ci,d} =$ $\{$*(1N,5F !a !d),(5F !a !d)*$\}$.

A seguir, é realizada uma modificação no conjunto de eventos observáveis do Exemplo 6 para ilustrar a obtenção de G_{dsa} em uma situação onde um evento $\sigma_c \in \Sigma_{cond}$ não é observável ($\sigma_c \in \Sigma_c$ e $\sigma_c \notin \Sigma_o$).

Exemplo 7 *(evento controlável não observável).* Considere o SED ilustrado na Figura

Figura 36 – $G_{dsa} = \widetilde{Obs}(G_{rsi})$, sendo $\Sigma_o = \{a, b, c, d\}$. As transições com evento em condição de impedimento no estado foram removidas, bem como a informação de que os estados não são maus estados (NB)

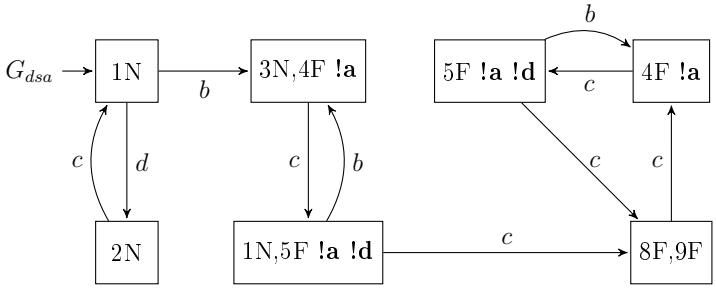


Fonte: Elaborado pelo autor (2021).

32 (mesmo do Exemplo 6), mas assuma que agora o evento a é não observável. Nesse caso, tem-se um evento controlável que é não observável. Têm-se então que $\Sigma_o = \{b, c, d\}$, $\Sigma_f = \{f\}$, $\Sigma_{uo} = \{a, f\}$, $\Sigma_c = \{a, d\}$, $\Sigma_{uc} = \Sigma - \Sigma_c$ e $\Phi = \{ab, d\}$.

O resultado da obtenção de G_{dsa} para um conjunto de eventos observável que não inclui todos os eventos controláveis é mostrado Figura 32.

Figura 37 – $G_{dsa} = \widetilde{Obs}(G_{rsi})$ para o Exemplo 7, sendo $\Sigma_o = \{b, c, d\}$. As transições com evento em condição de impedimento no estado foram removidas, bem como a informação de que os estados não são maus estados (NB)



Fonte: Elaborado pelo autor (2021).

Comparado com G_{dsa} da Figura 32, o diagnosticador seguro ativo da Figura 37 removeu o auto-laço no estado *(2N)* e também agrupou os estados *(8F)* e *(9F)*, em um estado *(8F,9F)*. Neste, o evento controlável a pode ocorrer, uma vez que não consiste em um evento do conjunto de impedimentos de nenhum desses estados. Por outro lado, os estados $q_{dsa} \in Q_{dsa}^{ci,a}$ ainda podem indicar o evento a como evento para ser impedido.

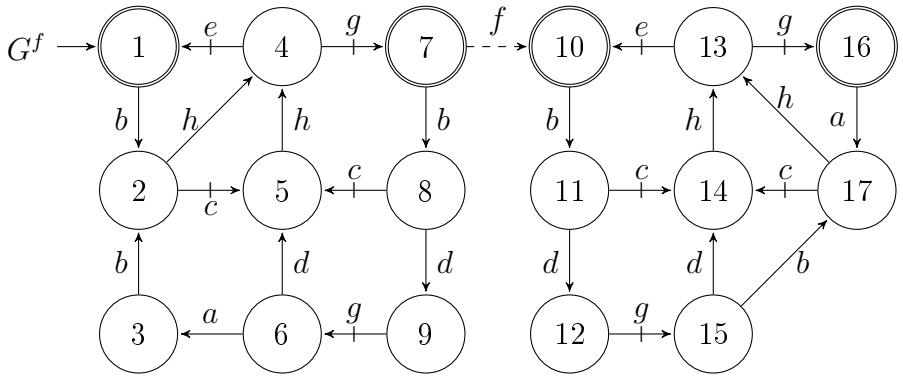
O Exemplo 8 apresenta uma situação na qual um mau estado do diagnosticador

seguro não representa, necessariamente, que a planta possui um mau estado, sendo esse definido pela forma como o estado é alcançado.

Exemplo 8 *(alcance dos maus estados).* Considere o modelo de uma planta com falha G^f mostrado na Figura 38. Considere também que $\Sigma_o = \{a, b, c, d, e, g, h\}$, $\Sigma_f = \Sigma_{uo} = \{f\}$, $\Sigma_c = \{c, e, g\}$, $\Sigma_{uc} = \Sigma - \Sigma_c$. Para a controlabilidade segura é necessário impedir a ocorrência da cadeia $\xi = bc \in \Phi$ após a falha f.

O G_{dsa} para o Exemplo 8 é mostrado na Figura 39, sendo que novamente se optou por remover as transições com eventos que são condição de impedimento no estado de partida e que conduzem o sistema para um mau estado. Também foi removida a informação sobre os estados não serem maus estados, uma vez que, com a remoção das transições, nenhum mau estado aparece no autômato. A partir de G_{dsa} é possível definir que $\Sigma_{cond} = \{c\}$ e o conjunto de estados impedidores do evento c é definido por $Q_{dsa}^{ci,c} = \{$*(2N,11F !c)*, *(11F !c)*, *(17F !c)*$\}$.

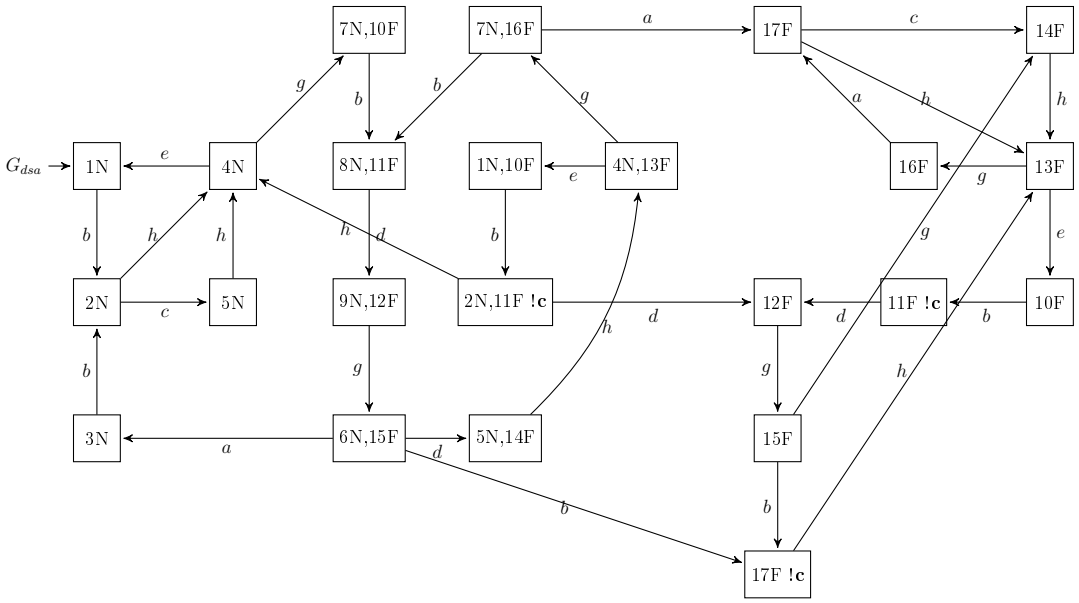
Figura 38 – Modelo da planta com falha G^f para o Exemplo 8



Fonte: Elaborado pelo autor (2021).

Um detalhe a ser observado no G_{dsa} da Figura 39 é a existência dos estados *(17F)* e *(17F !c)*. Embora representem o mesmo estado de G^f da Figura 38, um possui a condição de impedimento do evento c (para interromper a cadeia proibida bc) e outro não. Isso porque o estado pode ser alcançado por uma transição com o evento a (cuja a imediata execução do evento c não seria a execução de uma cadeia proibida), e o outro alcance ao estado é realizado com o evento b (que exige que, em seguida, o sistema não permita o evento c e execute a cadeia proibida).

Figura 39 – $G_{dsa} = \widetilde{Obs}(G_{rsi})$ para o Exemplo 8. As transições com evento em condição de impedimento no estado foram removidas, bem como a informação de que os estados não são maus estados (NB)



Fonte: Elaborado pelo autor (2021).

Uma vez que G_{dsa} permite identificar estados que, com a desabilitação dos eventos em seu conjunto de impedimento, permitem interromper a evolução do sistema para uma situação indesejada, o DCA pode utilizar essa informação para impedir os comportamentos proibidos do sistema enquanto não é possível chavear um supervisor pós-falha. A seção a seguir apresenta como o DCA é obtido.

6.1.6 Algoritmo para obtenção de G_{dsa}

O algoritmo a seguir calcula G_{dsa} utilizando duas funções com o mesmo objetivo (definir quais são os eventos controláveis das condições de impedimento), mas considerando as duas situações possíveis com relação às cadeias proibidas: ter ou não um evento controlável em cada cadeia proibida.

O Algoritmo 1 apresenta os passos para obter G_{dsa} para uma falha f e um conjunto de cadeias proibidas Φ. O resultado do algoritmo é um *array* com um autômato G_{dsa} para cada cadeia proibida $\xi_i \in \Phi$, sendo o tamanho do *array* determinado por $\|\Phi\|$.

Para cada cadeia proibida $\xi \in \Phi$ é extraído o último evento controlável σ_c (se houver) e a subcadeia p_r que ocorre após esse evento (considerando também o evento σ_c). Caso ξ não tenha um evento controlável, a variável σ_c é nula.

Em seguida é gerado o rotulador *RSI* para a cadeia proibida, sendo informados também os eventos controláveis do sistema. Se a cadeia proibida tem um evento controlável, A_{rsi} terá um estado de atenção, caso contrário, não. Dando continuidade, é calculada a composição paralela $G_{rsi} = (G^f \| A_{rsi})$ para se obter a propriedade de falha e identificação dos maus estados.

O passo seguinte executa as funções apropriadas para o restante do cálculo de G_{dsa}, levando em consideração se existe ou não um evento controlável na cadeia proibida. O procedimento é finalizado com a obtenção da componente acessível do autômato G_{ci}, que pode tornar-se inacessível devido à remoção de alguma(s) de suas transições (procedimento opcional das funções).

Algoritmo 1: Algoritmo para cálculo de G_{dsa}

Entrada: G^f, Φ, Σ_c, Σ_o
Saída: DCA[$\|\Phi\|$]

1 $i \leftarrow 0$
2 **para cada** $\xi \in \Phi$ **faça**
3 $\quad r, \sigma_c \leftarrow \Omega(\xi)$
4 $\quad p_r \leftarrow \xi / r$
5 $\quad A_{rsi} \leftarrow$ gerarRSI(ξ, Σ_c)
6 $\quad G_{rsi} \leftarrow G^f \ \| A_{rsi}$
7 $\quad$ **se** σ_c *não é nulo* **então**
8 $\quad\quad G_{ci} \leftarrow$ funcEventoCtrl(G_{rsi}, σ_c, p_r)
9 $\quad$ **senão**
10 $\quad\quad G_{ci} \leftarrow$ funcEventoNaoCtrl(G_{rsi}, Σ_c)
11 $\quad DCA[i] \leftarrow \widetilde{Obs}(G_{ci})$
12 $\quad i \leftarrow i + 1$
13 **return** G_{dsa}

O Algoritmo 2 apresenta os passos para obter G_{ci} no caso da existir um evento controlável na cadeia proibida. Inicialmente é utilizada a função *geraGci()* que consiste de uma função que gera G_{ci} a partir de G_{rsi} (basicamente um cópia, mas lembrando que G_{ci} possui indicação de condições de impedimento). Se, a partir de algum estado de atenção, a função de transição estendida com a cadeia $\sigma_c p_r$ alcança um mau estado, então as seguintes modificações são aplicadas ao estado: (1) o estado recebe a condição de impedimento com o evento σ_c; e (2) a transição de partida do estado com o evento σ_c é removida. Relembrando que o passo 2 é opcional e é considerado para uma etapa de implementação e não para análise das condições.

Algoritmo 2: Função funcEventoCtrl() para cadeia proibida com evento controlável

Entrada: G_{rsi}, σ_c, p_r
Saída: G_{ci}

```
1 G_ci ← geraGci(G_rsi)
2 para cada estado ∈ X_ci faça
3     se estado = "estado de atenção" então
4         estadoAlcançado = δ_ci(estado, p_r)
5         se estadoAlcançado = "mau estado" então
6             adiciona(estado.CI, σ_c)
7             remove δ_ci(estado, σ_c) // opcional
8 return Ac(G_ci) // componente acessível
```

O Algoritmo 3 especifica a função para o cálculo de G_{ci} quando a cadeia proibida não possui um evento controlável. Nesse caso, não se sabe qual evento controlável impede a ocorrência da cadeia proibida. Sendo assim, é preciso considerar se cada evento controlável do sistema pode impedir um estado de alcançar o mau estado a partir de uma cadeia de eventos não controláveis. Na Subseção 6.1.3 foi apresentada a possibilidade de obtenção do conjunto de impedimentos percorrendo as transições de estados pra trás, partindo de um mau estado até encontrar uma transição com um evento controlável que impede alcançar esse mau estado. O Algoritmo 3 utiliza outra estratégia, possivelmente mais simples de entender por não utilizar recursividade. O algoritmo inicia identificando quais são os maus estados e os adiciona em uma lista de impedimentos. Em seguida é verificado que transições alcançam os estados que se deseja impedir (que estão na lista de impedimentos) e, se a transição ocorre com um evento controlável, o estado de partida recebe a condição de impedimento com relação ao evento, senão o próprio estado de partida é adicionado à lista de impedimentos.

Algoritmo 3: Função funcEventoNaoCtrl() para cadeia proibida sem evento controlável

```
   Entrada: G_rsi, Σ_c
   Saída: G_ci
1  G_ci ← geraGci(G_rsi)
   // Determina quais são os maus estados
2  i ← 0
3  para cada estado ∈ X_ci faça
4      se estado = "mau estado" então
5          estadoImpedir[i] ← estado
6
7  para cada estado ∈ estadoImpedir faça
       // t.chegada = δ_ci(t.partida,t.evento)
       // {δ_ci} representa as transições do autômato
8      para cada t ∈ {δ_ci} faça
9          se t.chegada = estado então
10             se t.evento ∈ Σ_c então
11                 adiciona(t.partida.CI, t.evento)
12                 remove δ_ci(t.partida,t.evento) // opcional
13             senão
14                 estadoImpedir[i] = t.partida
15 return Ac(G_ci) // componente acessível opcional
```

Os algoritmos 2 e 3 determinam o conjunto de impedimento dos estados. Essa computação pode ser realizada usando apenas a função funcEventoNaoCtrl(), independente da cadeia proibida ter um evento controlável ou não. A função funcEventoCtrl() se difere por aproveitar uma característica da cadeia proibida.

O resultado do Algoritmo 1 é um conjunto de autômatos, um para cada cadeia proibida do conjunto Φ, que realizam o diagnóstico e indicam quais eventos e o momento que eles podem ser desabilitados para impedir o alcance de maus estados. Optou-se por um conjunto de autômatos devido a possível redução desses autômatos, onde o espaço de estados do autômato resultante será impactado de acordo com o alfabeto das cadeias proibidas.

6.2 DIAGNOSTICADOR CONTROLADOR ATIVO

O projeto de um sistema de controle tolerante a falhas ativo depende de diagnosticar eventos de falha, impedir a ocorrência de comportamentos proibidos e alterar a política de controle de forma a garantir um funcionamento adequado do sistema. No caso do CTF seguro introduzido por Paoli, Sartini e Lafortune (2011), a falha precisa ser diagnosticada antes da execução de sequências proibidas. Além disso, é preciso parar a evolução do sistema antes da execução de sequências proibidas. Para o CTF seguro ativo, o comportamento com falha do sistema deve obedecer especificações de controle especialmente projetadas para lidar com essas sequências proibidas após falhas.

Para a estratégia de CTF ativo, um conjunto de especificações pós-falha são necessários, sendo essas especificações pós-falha expressas na forma da linguagem $\mathscr{K}_i^{deg}$. Segundo Paoli, Sartini e Lafortune (2008), nos casos mais simples, $\mathscr{K}_i^{deg}$ é uma sublinguagem prefixo-fechada do comportamento degradado da planta, removendo os estados ilegais de G^f. Em outros casos, pode ser desejável algum comportamento mínimo para ser satisfeito após o evento de falha. Estratégias para obter os supervisores pós-falha podem ser encontradas em Kumar e Takai (2012).

Nesta tese não se aborda o projeto dos supervisores pós-falha. É considerado que esses supervisores são obtidos para atender às necessidades do sistema e, assim como esperado com relação ao supervisor nominal, são admissíveis para o projeto no contexto do CTF ativo seguro.

As etapas para obtenção do DCA foram apresentadas na Figura 25, sendo na Seção 6.1 mostrado como é gerado o G_{dsa}. Para obter o DCA é utilizado o modelo G_{DCA}, construído a partir de G_{dsa} e dos supervisores pós-falha. Inicialmente, G_{DCA} pode ser considerado uma cópia de G_{dsa}, ou seja $G_{DCA} = G_{dsa}$. Assim, de maneira similar ao definido para o diagnosticador seguro ativo, são definidos os conjuntos Q^B_{DCA}, Q^C_{DCA}, Q^U_{DCA}, Q^N_{DCA} e Q^{ci,σ_c}_{DCA}, com $\sigma_c \in \Sigma_{cond}$, para representar, respectivamente, o conjunto de estados que são maus estados, o conjunto de estados certos de falha, o conjunto de estados incertos de falha, o conjunto de estados normais e o conjunto de estados impedidores.

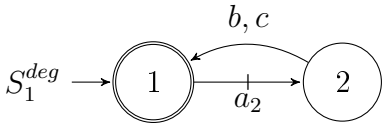
A seguir, devem ser identificados os estados de interesse para realizar o chaveamento dos supervisores pós-falha. Paoli, Sartini e Lafortune (2011) propuseram que esse chaveamento seja realizado em cada um dos primeiros estados certos de falha. Ou seja, para cada estado $q_{DCA} \in Q^{FC}_{DCA}$, é gerado um sinal de interrupção e construído um modelo

pós-falha S_i^{deg} para cada uma dessas interrupções ($i = 1...n$, com $n = \|Q_{DCA}^{FC}\|$). Nesse sentido, para cada $q_{DCA} \in Q_{DCA}^{FC}$, deve ser associada uma interrupção e acoplado, ao respectivo estado, seu modelo do supervisor pós-falha. Com isso, para cada estado de G_{DCA} que contiver uma interrupção, ao alcançar esse estado, o DCA gera a interrupção, interrompendo a estratégia de controle imposta pelo supervisor nominal, e passa a utilizar a política de controle do supervisor pós-falha associado com o estado atual de G_{DCA}.

Retomando-se o Exemplo 5 e o modelo do diagnosticador seguro ativo G_{dsa}, mostrado na Figura 31, tem-se que $Q_{DCA}^{FC} = \{(6F\ !a)\}$, com $n = \|Q_{DCA}^{FC}\| = 1$. Portanto é necessário projetar um único supervisor pós-falha. Esse supervisor pós-falha S_1^{deg} é mostrado na Figura 40. Para esse exemplo é considerado que, após a falha, interrompe-se o uso do componente que possui o evento a e passa-se a utilizar um componente redundante, acionado pelo evento a_2. Essa estratégia em relação ao componente redundante é uma das vantagens do CTF ativo, onde esse componente não precisava fazer parte do modelo nominal do sistema, sendo considerado apenas no modelo pós-falha.

No estado *(6F !a)*, como esse estado é um estado impedidor do evento a, esse evento é desabilitado pelo DCA e é garantida a controlabilidade segura. Além disso, no estado *(6F !a)* é gerada a interrupção (mostrada como uma transição na cor vermelha) para que o controle desabilite o supervisor nominal e passe a atuar com o modelo de G_{DCA}. A Figura 41 mostra o G_{DCA} obtido, onde INT_1 (destacada com a escrita em cor vermelha) indica a interrupção gerada após a diagnose da falha f, no estado *(6F !a)*.

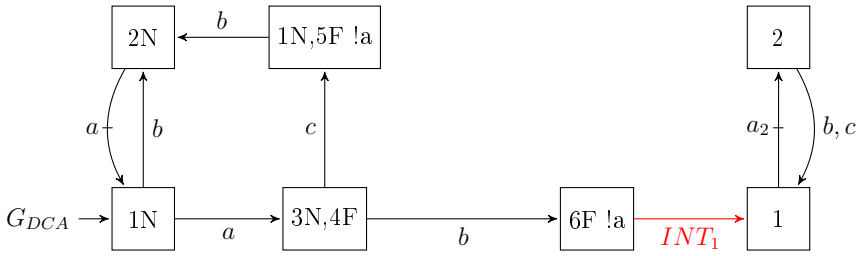
Figura 40 – Supervisor S_1^{deg} para o Exemplo 5



Fonte: Elaborado pelo autor (2021).

Se o sistema for A-controlável seguro, a A-CS garante a diagnose da falha antes da ocorrência de uma cadeia proibida. Caso o sistema não seja A-CS, o DCA atua desabilitando os eventos que estão nas condições de impedimento de cada estado de G_{DCA}, até que um estado certo de falha seja alcançado, permitindo o chaveamento de um supervisor pós-falha.

Figura 41 – G_{DCA} obtido para o Exemplo 5



Fonte: Elaborado pelo autor (2021).

6.3 CTF ATIVO COM O *DCA*

Segundo Moor (2016), o desafio no CTF ativo está no chaveamento entre três modos de operação: (1) sem falhas; (2) a falha ocorreu, mas ainda não está diagnosticada; e (3) a falha ocorreu e foi diagnosticada.

Para esse trabalho, o item (3) é atendido pela estratégia de chaveamento apresentada por Paoli, Sartini e Lafortune (2011). Entre as possibilidades para obtenção dos supervisores pós-falha, é possível fazer uso de elementos da planta que não estavam em uso (por exemplo, sensores e atuadores, redundantes ou não). É importante destacar que, as especificações $\mathscr{K}_i^{deg}$ são consideradas incomparáveis com $L(S/G^f)$ e, por esse motivo, é necessário chavear o supervisor nominal para uma nova política de controle, denotada por S_i^{deg}. Essa habilidade de chaveamento está presente no *DCA*.

Para os itens (1) e (2), a estratégia do *DCA* remete ao CTF passivo, no qual o *DCA* atua para impedir que ocorram cadeias proibidas após falhas. Uma vez que a falha não foi diagnosticada (não se sabe se ela ocorreu ou não), é adotada uma estratégia menos permissiva, não permitindo a execução de uma cadeia proibida em caso de dúvida sobre a ocorrência da falha (a falha pode ter ocorrido, mas ainda não foi diagnosticada), embora seja possível permitir a cadeia, se existir certeza de que a falha não ocorreu.

Sendo assim, o principal objetivo do *DCA* é garantir a controlabilidade segura do sistema, impedindo que o sistema execute algum evento, ou sequência de eventos, que represente algum risco após uma possível ocorrência de falhas. Independentemente de ser um estado certo ou incerto de falha, os estados impedidores asseguram que uma cadeia proibida não ocorrerá depois da falha. Após ter certeza sobre a ocorrência da falha, é possível mudar a lógica de controle para conseguir um comportamento mais adequado do

que o obtido a partir do uso do supervisor nominal e do DCA.

Na proposta de Paoli, Sartini e Lafortune (2011), o CTF só é garantido se o sistema for controlável seguro e, nesse caso, os chaveamentos da lógica de controle nominal para lógicas de controle $\mathscr{K}_i^{deg}$ só ocorrem após a diagnose da falha. Com a utilização da estrutura de controle usando o DCA, é garantida a controlabilidade segura do sistema até o chaveamento do supervisor pós-falha. Ou seja, embora os supervisores pós-falha ainda precisem garantir a não ocorrência das cadeias proibidas, eles não precisam ser chaveados imediatamente.

6.4 ASPECTOS DA CONTROLABILIDADE PELO DCA

Além de chavear o supervisor nominal para um supervisor pós-falha, o DCA é responsável garantir o controle seguro durante situações de incerteza com relação à ocorrência da falha, impedindo a execução de cadeias proibidas na planta após uma possível falha. Sendo assim o DCA atua para garantir a controlabilidade segura da planta.

Na arquitetura de controle ilustrada na Figura 22, o supervisor nominal e o DCA agem sobre a planta com objetivos distintos: o supervisor impõe o cumprimento das especificações de controle, enquanto o DCA atua para impedir a ocorrência de cadeias proibidas. A ação conjunta destes sobre a planta é denotada por $S \wedge DCA$ (ação conjunta do supervisor com o DCA).

Para facilitar a análise dos aspectos da controlabilidade pelo DCA, é assumido que o supervisor nominal já foi implementado e atende às especificações de controle do sistema nominal (sem falha). O mesmo é considerado em relação aos supervisores pós-falha, que precisam ser chaveados para atuar após a confirmação da falha.

Sendo assim, o interesse especial está na ação conjunta $S \wedge DCA$ antes dos chaveamentos e, portanto, o foco será dado à essa situação.

A ação conjunta $S \wedge DCA$ é obtida a partir da execução em paralelo do supervisor e do DCA. O supervisor atua determinando o conjunto de eventos habilitados, enquanto o DCA remove das habilitações os eventos cujo estado atual de G_{DCA} é um estado impedidor do evento. Ou seja, após a planta gerar uma cadeia de eventos s, o supervisor determina o conjunto de eventos habilitados após observar $P_o(s)$, enquanto o DCA irá remover habilitações após observar $P_o^+(s)$. Sendo assim, a ação conjunta $S \wedge DCA$ irá impedir que a planta G^f execute ações que conduzam sua execução para um mau estado.

Relembrando que, P_o^+ representa o mapeamento perceptivo de diagnosticador (informações adicionais disponíveis para o diagnosticador). Por simplicidade de entendimento, será considerado $P_o^+ = P_o$ para a análise das condições apresentadas.

Seja DCA uma função que mapeia cadeias s_o em conjuntos de eventos controláveis que devem ser desabilitados, *i.e.*, $DCA : P_o[L(G^f)] \rightarrow 2^{\Sigma_c}$. Sejam $s_o = P_o(s)$, com $s \in L(G^f)$ e $q_{dca,k} = \delta_{dca}(q_{dca,0}, s_o)$ tal que $q_{dca,k} = \{(x_i, l_i), ..., (x_j, l_j), CI_k\}$. Define-se a ação de DCA como $DCA(s_o) = CI_k$. A ação conjunta $S \wedge DCA$, denotada por SD, é um mapa $SD : P_o[L(G^f)] \rightarrow 2^{\Sigma}$. Assim, para uma cadeia $s_o = P_o(s)$ tem-se que $SD(s_o) = S(s_o) \setminus DCA(s_o)$. Dessa forma, $SD(s_o) \cap \Gamma(\delta(x_o, s))$ é o conjunto de eventos habilitados que G^f pode executar no seu estado atual $x = \delta(x_o, s)$, estado esse que foi alcançado a partir de x_o com a cadeia s, *i.e.*, $x = \delta(x_o, s)$.

O Lema 1 é apresentado para formalizar a admissibilidade da ação conjunta $S \wedge DCA$.

Lema 1 *(Admissibilidade de SD).* Seja S um supervisor admissível obtido a partir de G e seja $L(S/G^f)$ o comportamento de G^f sob a ação do supervisor S. A ação conjunta $S \wedge DCA$, denotada por SD, é admissível.

Prova. Considerando que S é admissível, $\forall s \in L(G^f)$, $\Sigma_{uc} \cap \Gamma(\delta(x_o, s)) \subseteq S(s_o)$, sendo $s_o = P_o(s)$. Tendo em vista que $DCA(s_o) \in 2^{\Sigma_c}$, então $\Sigma_{uc} \cap \Gamma(\delta(x_0, s)) \subseteq S(s_o) \setminus DCA(s_o) = SD(s_o)$. Dessa forma, SD é admissível para todo $s_o = P_o(s)$. ♦

Uma vez que SD é admissível, e dada uma planta G^f, o resultado do sistema em malha fechada é denotado por SD/G^f (similar a noção de S/G).

Embora o supervisor S seja próprio (não bloqueante em malha fechada) para G, e com isso $\overline{L_m(S/G)} = L(S/G)$, a ação conjunta SD pode ser bloqueante.

Se um bloqueio ocorrer após a diagnose da falha, então é possível chavear um supervisor pós-falha para tratamento desse bloqueio. Caso o bloqueio ocorra antes da diagnose da falha, o sistema não será considerado tolerante a falhas e precisará de ajustes no projeto. Além de bloqueios, a imposição de ações para impedir as cadeias proibidas pode gerar uma restrição não aceitável para o sistema. Nesse caso, diz-se também que o

sistema não será considerado tolerante a falhas.

O principal objetivo do DCA é garantir a controlabilidade segura do sistema, auxiliando o supervisor no controle do sistema. Para isso, a ação conjunta $S \wedge DCA$ deve garantir a que linguagem do sistema seja segura. Relembrando que, a ação conjunta $S \wedge DCA$ é tratada antes do chaveamento para um supervisor pós-falha e a ação SD/G^f é referente á essa ação antes do chaveamento. A Definição 11 formaliza a linguagem segura.

Definição 11 *(Linguagem segura).* Uma linguagem $L \subseteq L(G^f)$ é dita segura e.r.a K_f se $\forall s \in \Psi_L(f), \nexists t \in L/s : \overline{t} \cap \mathscr{K}_f \neq \emptyset$.

Em palavras, uma linguagem é segura se nela não existem cadeias proibidas de Φ após uma falha f. Na Definição 12 é formalizada a máxima linguagem segura.

Definição 12 *(Máxima linguagem segura).* Seja $L(G^f)$ a linguagem de um sistema G com uma falha f e $\mathscr{K}_f$ a linguagem ilegal que contém todas as possíveis continuações após um evento de falha f e que possuem uma cadeia proibida de Φ como subcadeia. Seja $L(S/G^f)$ o comportamento de G^f sob a ação do supervisor S. Define-se a máxima sublinguagem de $L(S/G^f)$ que é segura e.r.a. $\mathscr{K}_f$ como $L^{safe} := \{w \in L(S/G^f) : \nexists st \in \overline{w}$ na qual $s \in \Psi_{L(S/G^f)}(f)$ e $\overline{t} \cap \mathscr{K}_f \neq \emptyset\}$.

Em palavras, a linguagem $L^{safe} \subseteq L(S/G^f)$ possui todas as cadeias do comportamento nominal e todas as cadeias do comportamento pós-falha que não incluem um elemento de Φ como subcadeia após a falha.

No Teorema 5 são apresentadas as condições para a existência de DCA de forma que a ação conjunta $S \wedge DCA$ permita obter $L(SD/G^f) \subseteq L^{safe}$.

Teorema 5 *(Existência de DCA).* Sejam $G = (X, \Sigma, \delta, x_0, X_m)$ e G^f modelos de SEDs que representam, respectivamente, o comportamento nominal de um sistema e seu comportamento nominal e faltoso. Seja S um supervisor admissível para G. Seja ainda L^{safe} a máxima sublinguagem de $L(S/G^f)$ que é segura e.r.a. K_f. Seja SD a ação conjunta $S \wedge DCA$. Existe $G_{DCA} = (Q_{DCA}, \Sigma_o, \delta_{DCA}, q_{DCA,0})$ tal que $L(SD/G^f) \subseteq L^{safe}$ se, e somente se, $L(S/G^f)$ é controlável segura irrestrita e.r.a. Σ_c, f e Φ.

Prova. A necessidade e a suficiência são provadas por contradição.
($\Rightarrow$) Suponha que $\exists DCA$ tal que $L(SD/G^f) \subseteq L^{safe}$, mas que $L(S/G^f)$ não é controlável segura irrestrita. Então, pela Definição 10, $\exists st \in L(S/G^f)$ tal que $s \in \Psi_{L(S/G^f)}(f)$ e $t = u\xi$, com $\xi \in \Phi$, para a qual $\nexists \sigma_c \in \Sigma_c : \sigma_c \in st$. Tendo em vista que $L^{safe} \subseteq L(S/G^f)$, pode-se afirmar que $\exists w \in L^{safe} : w < st$. Sem perda de generalidade, assuma que $st = w\sigma$, com $\sigma \in \Sigma_{uc}$. Então, $\exists w \in L^{safe}$ para a qual $w\sigma \in L(S/G^f)$ mas $w\sigma \notin L^{safe}$ e, portanto, $L^{safe}\Sigma_{uc} \cap L(S/G^f) \nsubseteq L^{safe}$. Portanto, L^{safe} não é controlável e.r.a. $L(S/G^f)$. Sabe-se ainda que, como $st \in \Sigma_{uc}^*$, e $\forall u \in \overline{st}$, $DCA(u) \in 2^{\Sigma_c}$, então o DCA não acrescenta desabilitações além das impostas pelo supervisor para impedir o completamento da cadeia st. Dessa forma, o DCA é tal que $\exists st \in L(SD/G^f) : st \notin L^{safe}$ e, portanto $L(SD/G^f) \nsubseteq L^{safe}$.
($\Leftarrow$) Suponha que $L(S/G^f)$ é controlável segura irrestrita, mas que $\nexists DCA$ tal que $L(SD/G^f) \subseteq L^{safe}$. Deseja-se provar que $L(S/G^f)$ não é CSI. Sabe-se que se $L(SD/G^f) \nsubseteq L^{safe}$, então $\exists w \in L(SD/G^f)$ tal que $w \notin L^{safe}$. Pela Definição 12, $L^{safe} := \{w \in L(S/G^f) : \nexists st \in \overline{w}$ na qual $s \in \Psi_{L(S/G^f)}(f)$ e $\bar{t} \cap \mathscr{K}_f \neq \emptyset\}$. Assim, pode-se afirmar que w possui uma cadeia de Φ como subcadeia. Assuma que $w = st$, com $s \in \Psi_L(f)$ e $t = u\xi$, com $\xi \in \Phi$. Como $w = st \in L(SD/G^f)$, mesmo com $\bar{t} \cap \mathscr{K}_f \neq \emptyset$, pode-se afirmar que $\forall u \in \overline{st}$, $\nexists \sigma_c \in DCA(P_o(u))$ tal que $\sigma_c \in st$. Assim, $\nexists \sigma_c \in \Sigma_c : \sigma_c \in st$. Dessa forma, pode-se afirmar que $\exists st \in L(S/G^f) : s \in \Psi_L(f)$ e $t = u\xi$, com $\xi \in \Phi$, para o qual $\nexists \sigma_c \in \Sigma_c : \sigma_c \in st$. Portanto, pela Definição 10, $L(S/G^f)$ não é controlável segura irrestrita, o que viola a hipótese inicial. ♦

Uma vez que, pela Proposição 4, a controlabilidade segura implica a CSI, a controlabilidade segura e a A-CS são condições suficientes para existência do DCA, uma vez que em ambas existe a necessidade de um evento controlável após a falha (especificamente após a falha) e que impede alcançar um mau estado. Assim sendo, é considerado que a ação conjunta $S \wedge DCA$ é válida também para linguagens controláveis seguras e A-controláveis seguras.

Embora seja a máxima linguagem segura, L^{safe} não é necessariamente controlável e observável. Na Definição 13 é formalizada a máxima linguagem controlável, normal, prefixo-fechada e segura.

Definição 13 *(Máxima linguagem controlável, normal, prefixo-fechada e segura).* Seja $L(G^f)$ a linguagem de um sistema G com uma falha f. Seja $L(S/G^f)$ o comportamento de G^f sob a ação do supervisor S. Seja L^{safe} a máxima sublinguagem de $L(S/G^f)$ que é segura e.r.a. $\mathscr{K}_f$. Define-se a classe de linguagens de L^{safe} que são controláveis, normais, prefixo-fechadas e seguras e.r.a. K_f como $L^{CNS} := \{K \subseteq L^{safe} : K$ é controlável (e.r.a. $L(S/G^f)$ e $\Sigma_{uc}) \wedge$ normal (e.r.a. $L(S/G^f)$ e $\Sigma_o) \wedge$ prefixo-fechada$\}$.

A implementação para obtenção de L^{CNS}, no contexto do que é apresentado na Definição 13, considera as seguintes expressões:

1. $N^+ = \{N \subseteq L^{safe} : \overline{N}$ é normal e.r.a. $L(S/G^f)$ e $\Sigma_o\}$;
2. $N_o^+ = P_o(N^+)$;
3. $L_o = P_o[L(S/G^f)]$;
4. $K_o^+ = \{K_o \subseteq N_o^+ : K_o$ é controlável e.r.a. L_o e $\Sigma_{uc}\}$; e
5. $L^{CNS} = L(S/G^f) \cap P_o^{-1}(K_o^+)$.

A seguir é mostrado, a partir do Teorema 6, que a ação conjunta $S \wedge DCA$ permite obter a máxima linguagem controlável, normal, prefixo-fechada e segura L^{CNS}.

Teorema 6 *(Controlabilidade segura por SD).* Seja S um supervisor admissível para um SED G, com $\Sigma_c \subseteq \Sigma_o$. Seja G_{DCA} obtido a partir de G^f e Φ. Seja ainda L^{CNS} a máxima sublinguagem de L^{safe} que é controlável, normal, prexo-fechada e segura e.r.a. $L(S/G^f)$, Σ_o e Σ_{uc}. A ação conjunta $S \wedge DCA$, denotada por SD, resulta em $L(SD/G^f) = L^{CNS}$.

Prova. a) Provando inicialmente quando $L(S/G^f)$ não é CSI. Pelo Teorema 5, se $L(S/G^f)$ não é CSI, então $\nexists DCA$ tal que $L(SD/G^f) \subseteq L^{safe}$, o que significa que não existe uma solução para o problema de CTF ativo usando o DCA. Deve-se provar então que $L^{CNS} = \emptyset$. Assuma que $L^{CNS} = K$, sendo $K \subseteq L^{safe} \subseteq L(S/G^f)$. Se $L(S/G^f)$ não é CSI, então, pela Definição 10, $\exists st \in L(S/G^f) : s \in \Psi_{L(S/G^f)}(f)$ e $t = u\xi$, com $\xi \in \Phi$, para a qual $\nexists \sigma_c \in \Sigma_c : \sigma_c \in st$. Tendo em vista que t possui um elemento de Φ como subcadeia, então $st \notin K$. Assim, uma vez que $st \in \Sigma_{uc}^*$, pode-se afirmar que $K\Sigma_{uc} \cap L(S/G^f) \not\subseteq K$,

de modo que K não é controlável e.r.a. $L(S/G^f)$ e, portanto, $L^{CNS} = \emptyset$. Sendo assim, $L(SD/G^f) = L^{CNS} = \emptyset$.

b) Provando agora quando $L(S/G^f)$ é CSI. Pelo Teorema 5, se $L(S/G^f)$ é CSI, então $\exists DCA$ tal que $L(SD/G^f) \subseteq L^{safe}$. Seja então $K \subseteq L(S/G^f)$ tal que $L(SD/G^f) = K \subseteq L^{safe}$.

b.i) Prova-se inicialmente, por contradição, que $L(SD/G^f) \supset L^{CNS}$. Seja $K = L(SD/G^f)$. Assumindo-se que $K \not\supseteq L^{CNS}$, ou seja, que $\exists r \in L^{CNS} : r \notin K$. Se $r \in L^{CNS}$, então $r \in L(S/G^f)$ e como $r \notin K$, pode-se afirmar que $\exists \sigma_c \in \Sigma_c : \sigma_c \in r$. Ainda, como $r \notin K$, pode-se afirmar que para algum $r' \in P_o^{-1}[P_o(r)] \cap L(S/G^f)$, $\exists u' \in L((S/G^f)/r')$ tal que u' possui uma cadeira de Φ como subcadeia e que $\nexists \sigma_c \in \Sigma_c : \sigma_c \in u'$. Sem perda de generalidade, considere que $r' = v\sigma_c$ e $u' = w\xi$, com $f \in v$ ou $f \in w$ e $u' \in \Sigma_{uc}^*$. Assim, $r'u' = v\sigma_c w\xi$ é tal que $r' \notin L^{CNS}$. Tendo em vista que L^{CNS} é normal e.r.a. $L(S/G^f)$, $r' = v\sigma_c \notin L^{CNS} \Rightarrow \nexists r \in P_o^{-1}[P_o(r')] \cap L(S/G^f) : r \in L^{CNS}$ e, portanto, conclui-se que $r \notin L^{CNS}$, o que viola a hipótese inicial.

b.ii) A seguir é provado, por contradição, que $L(SD/G^f) \subset L^{CNS}$. Seja $K = L(SD/G^f)$. Assumindo-se que $K \not\subseteq L^{CNS}$, ou seja, que $\exists r \in K : r \notin L^{CNS}$. Seja $st \in L(S/G^f)$ tal que $s \in \Psi_{L(S/G^f)}(f)$ e $t = u\xi$, com $\xi \in \Phi$. Uma vez que $L(S/G^f)$ é CSI, $r\sigma_c = \Omega(st)$ e $DCA(P_o(r))$ desabilita o evento σ_c após a cadeia r. Com isso é possível afirmar que $r \in K$. Para $r \notin L^{CNS}$ é preciso que, para algum $r' \in P_o^{-1}[P_o(r)] \cap L(S/G^f)$, $\exists u' \in L((S/G^f)/r')$ tal que u' possui uma cadeira de Φ como subcadeia e que $\nexists \sigma_c \in \Sigma_c : \sigma_c \in u'$. Sendo assim, $\exists r'u' \in L(S/G^f)$ tal que $f \in r'u'$, $\sigma_c \in r'$ e u' possui um elemento de Φ como subcadeia após f. Além disso, $\nexists \sigma_c \in \Sigma_c : \sigma_c \in u'$. Nessas condições pode-se concluir que $r' \notin K$. Uma vez que $P_o(r') = P_o(r)$, também é possível afirmar que $r \notin K$, o que viola a hipótese inicial e, portanto, $K \subset L^{CNS}$. ♦

Em palavras, no que se refere ao comportamento de SD/G^f antes de qualquer chaveamento por parte do DCA, a ação de SD sobre G^f impõe o mesmo resultado que seria obtido caso fosse projetado um supervisor com observação parcial para restringir o comportamento do sistema a um comportamento seguro.

Entretanto, conforme comentado anteriormente, quando o DCA possuir mais informações do que o disponível para um supervisor obtido sob observação parcial (conforme apresentado nas noções de diagnosticabilidade segura e controlabilidade segura pelo di-

agnosticador, na Seção 5.2) a ação conjunta $S \wedge DCA$ deve ser menos restritiva que a do referido supervisor, ou seja, $L^{CNS} \subseteq L(SD/G^f) \subseteq L^{safe}$.

É importante destacar que a utilização do DCA, quando o sistema é A-CS, não restringe o comportamento nominal do sistema, atuando apenas após confirmação da falha. Nesse sentido, a Proposição 5 é apresentada para formalizar a ação de controle imposta pela ação conjunta $S \wedge DCA$ atuando na planta nominal do sistema quando a linguagem do sistema é A-CS.

Proposição 5 *(SD/G quando $L(S/G^f)$ é A-CS).* Sejam G e G^f modelos de SEDs que representam, respectivamente, o comportamento nominal de um sistema e seu comportamento completo (nominal e faltoso). Seja S um supervisor admissível para G tal que $L(S/G) = K$. Seja G_{DCA} obtido a partir de G^f e Φ. Seja SD a ação conjunta $S \wedge DCA$. Então $L(SD/G) = K$ se, e somente se, $L(S/G^f)$ é A-controlável segura.

Prova. ($\Rightarrow$) Tendo em vista que $\forall w \in L(G^f)$, $SD(P_o(w)) = S(P_o(w)) \setminus DCA(P_o(w))$, a ação do DCA consiste em acrescentar desabilitações de eventos controláveis às decisões do supervisor. Assim, $L(SD/G) \subseteq L(S/G)$. Resta provar que $L(SD/G) \supseteq L(S/G)$. Se $L(S/G^f)$ é A-CS, pela Definição 9 tem-se que ($\forall su\xi \in L(S/G^f)$, com $s \in \Psi_L(f)$ e $u\xi = w\sigma w'$, $\sigma \in \Sigma_o$, tal que a condição $\mathscr{D}$ é atendida para $sw\sigma$ mas não é atendida para sw) $\Rightarrow$ ($\exists \sigma_c \in \Sigma_c : \sigma_c \in w'$). Seja $s' \in \overline{sw\sigma}$, uma vez que $s' < \Omega(su\xi)$, (i.e., o evento controlável que será usado pelo DCA para impedir a ocorrência da cadeia proibida está após s'), pode-se afirmar que $\nexists \sigma_c \in DCA(P_o(s')) : \sigma_c \in sw\sigma$ com $\sigma_c \in \Sigma_c$. Ou seja, o DCA não inibe a ocorrência da cadeia $sw\sigma$, assim como também não inibe a ocorrência de qualquer cadeia $s'' \in P_o^{-1}[P_o(sw\sigma)] \cap L(S/G^f)$. Dessa forma, $\forall u \in \overline{s''}$ tal que $u \in L(S/G)$, tem-se que $u \in L(SD/G)$. Extrapolando o raciocínio para todas as cadeias de $L(S/G)$ tem-se que $L(S/G) \subseteq L(SD/G)$. Sendo assim, pode-se concluir que $L(SD/G) = L(S/G) = K$.

($\Leftarrow$) Supondo-se que $L(SD/G) = K$, mas que $L(S/G^f)$ não é A-CS. Pelo Teorema 5, se $L(S/G^f)$ é CSI, então ($\forall st \in L(S/G^f) : s \in \Psi_L(f)$ e $t = u\xi$, com $\xi \in \Phi$), $\exists \sigma_c \in \Sigma_c : \sigma_c \in st$. Se $L(S/G^f)$ não é A-CS, então $\exists st$ tal que $r\sigma_c = \Omega(st)$ no qual r não assegura $\mathscr{D}$. Uma vez que r não assegura $\mathscr{D}$ isso significa que a cadeia r é indistinguível quanto

à ocorrência da falha. Supondo-se uma cadeia $u\sigma_c \in L(S/G) : P_o(r) = P_o(u)$. Uma vez que $DCA(P_o(r))$ desabilita o evento σ_c e que $P_o(r) = P_o(u)$, então $u\sigma_c \notin K$. Ou seja, $K \nsubseteq L(SD/G)$, o que contradiz com a hipótese inicial. ♦

Em palavras, a ação de DCA não traz restrições ao comportamento nominal de S/G se, e somente se, $L(S/G^f)$ é A-CS. Em outras palavras, quando a linguagem é A-CS a ação conjunta $S \wedge DCA$ não restringe o comportamento da planta (além do que já é restrito por S) durante o comportamento nominal.

Na subseção a seguir é comparado, a partir de um exemplo, os resultados da ação conjunta $S \wedge DCA$ e de um supervisor obtido sob observação parcial.

6.4.1 Comparação de resultados

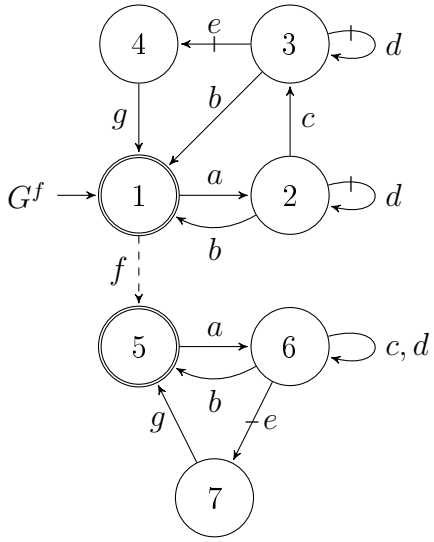
Esta subseção compara o controle alcançado utilizando um supervisor sob observação parcial e o controle com o uso da arquitetura utilizando o DCA. Para isso é utilizado o Exemplo 9, no qual a linguagem $L(G^f)$ não é A-controlável segura. O exemplo também ilustra a aplicação da noção e diagnosticabilidade segura pelo diagnosticador e também ilustra a obtenção do CTF seguro ativo pelo DCA.

Exemplo 9 *(comparação de resultados).* Considere o modelo de uma planta G^f mostrado na Figura 42, sendo $\Sigma_o = \{a, b, c, d, e, g\}$, $\Sigma_{uo} = \Sigma_f = \{f\}$, $\Sigma_c = \{d, e\}$ e $\Sigma_{uc} = \Sigma - \Sigma_c$. Considere a necessidade de impedir duas cadeias proibidas após falha, sendo $\Phi = \{d, e\}$. A linguagem $L(G^f)$ não é A-controlável segura e.r.a. P_o, a falha f, Σ_c e o conjunto Φ.

Uma vez que se deseja impedir a cadeia proibida devido a incerteza com relação à ocorrência da falha, um controle sob observação parcial pode ser utilizado. Para o cálculo do supervisor sob observação parcial é feito uso do rotulador RSI como uma especificação de controle. Essa especificação é denominada E_{rsi}.

Seja G^f o modelo da planta sob influência de uma falha f, E uma especificação de controle para o sistema e E_{rsi}, a especificação para as cadeias proibidas de Φ. O comportamento que se quer controlar é determinado pela linguagem $K = (G^f \| E \| E_{rsi})$. Considere $N = Normal(G^f, K)$ o resultado do cálculo para obter o supervisor para a máxima sublinguagem normal para G^f, K e Σ_c. O supervisor N impede a ocorrência da

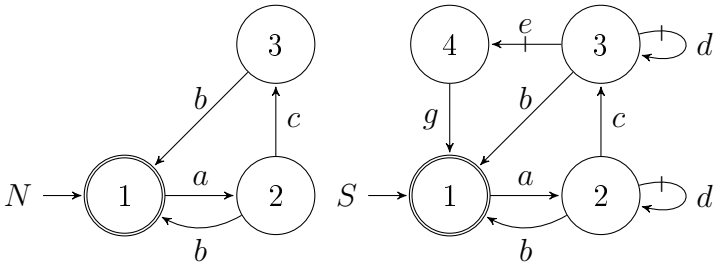
Figura 42 – Planta G^f para o Exemplo 9, com uma linguagem não A-controlável segura



Fonte: Elaborado pelo autor (2021).

cadeia proibida mesmo quando uma falha não é controlável segura. Isso pode ser visto a partir do supervisor N, na Figura 43, obtido para o Exemplo 9. A Figura 43 também mostra um supervisor S, calculado para o comportamento nominal da planta.

Figura 43 – Supervisor N, obtido sob observação parcial. Supervisor S, calculado para o comportamento nominal. Ambos obtidos para o Exemplo 9



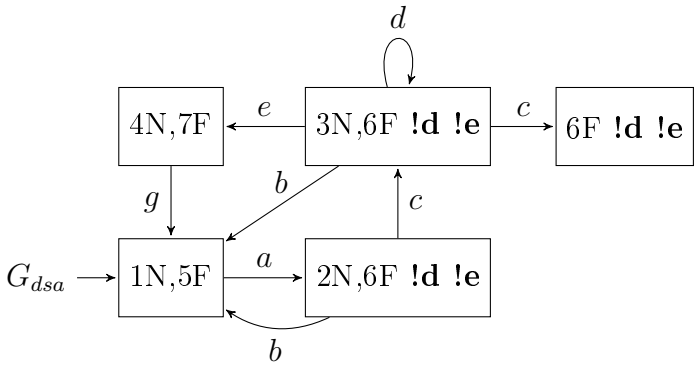
Fonte: Elaborado pelo autor (2021).

A partir do supervisor N é possível perceber que, para impedir os eventos d e e após a falha, uma vez que, para esse exemplo, não se pode diferenciar o comportamento antes e depois da falha, o controle sob observação parcial precisa impedir os eventos em todos os momentos, de forma a obter uma linguagem que apresente normalidade. Embora seja possível chavear um novo supervisor quando a falha for confirmada, o sistema precisa restringir o evento durante toda sua execução sem falhas.

Considere agora o G_{dsa} calculado para o Exemplo 9 e mostrado na Figura 44.

Estados e transições do G_{dsa} foram removidas a partir do estado *(6F !d !e)* para facilitar a visualização. É possível observar em G_{dsa} que a falha não é A-diagnosticável segura, uma vez que as cadeias proibidas $\Phi = \{d, e\}$ podem ocorrer antes da confirmação da falha, a qual ocorre apenas a partir do estado *(6F !d !e)*.

Figura 44 – G_{dsa} obtido para o Exemplo 9. Foram removidos estados e transições a partir do primeiro estado que confirma a falha *(6F !d !e)*



Fonte: Elaborado pelo autor (2021).

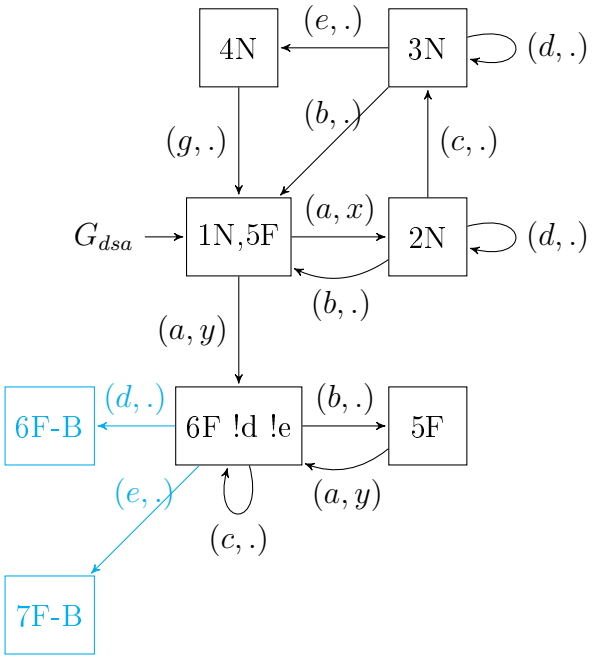
O G_{dsa} apresenta a necessidade de desabilitar os eventos d e e em seus estados impedidores, sendo que existem estados impedidores que são incertos de falha. No caso do Exemplo 9, essas desabilitações impactam também no funcionamento nominal do sistema, restringindo o comportamento nominal de forma idêntica ao que é obtido com a utilização do supervisor sob observação parcial N.

Considere agora G_{dsa} calculado utilizando mapeamento de sensores e que permite que a falha f seja diagnosticável segura pelo diagnosticador e controlável segura pelo diagnosticador. Para isso, considere o uso de um sensor hipotético que assume os valores x e y. As transições com o mapeamento de sensor serão mostradas na forma $[\sigma, sensor]$, sendo $\sigma \in \Sigma_o$ e $sensor \in \{x, y\}$. Para facilitar a visualização, o autômato G_{dsa} apresenta o mapeamento de sensores apenas associado com o evento onde ele permite a diferenciação da ocorrência da falha (os demais resultados de mapeamento foram substituídos por ".").

O diagnosticador seguro ativo G_{dsa} com o mapeamento de sensores é mostrado na Figura 45. O estado impedidor *(6F !d !e)* tem a condição de impedimento para impedir o evento d e o evento e. Para o exemplo, obtou-se por remover as transições com o evento das condições de impedimentos, bem como os rótulos de identificação de maus estados. Mesmo assim, apenas para fins de análise, a Figura 45 mostra os estados e transições que

seriam removidos do modelo (destacados na cor azul).

Figura 45 – G_{dsa} com mapeamento de sensor para o sistema para o Exemplo 9



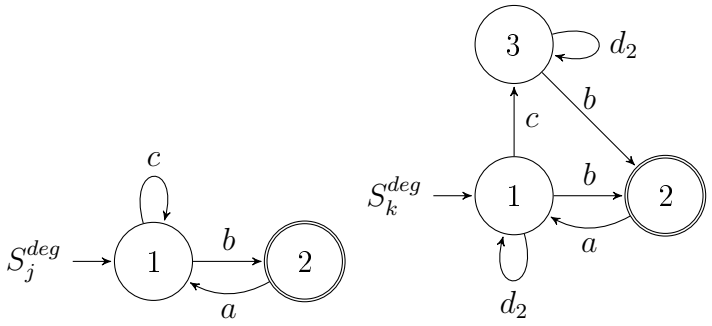
Fonte: Elaborado pelo autor (2021).

Considerando a utilização do G_{dsa} da Figura 45 para obter G_{DCA}, uma vez que o mapeamento de sensores resultou em um sistema A-controlável seguro pelo diagnosticador, a ação conjunta $S \wedge DCA$ permite alcançar o comportamento nominal da planta antes da falha. Após a confirmação da falha, no estado correspondente ao estado *(6F !d !e)* de G_{DCA}, é possível chavear um supervisor pós-falha projetado de forma a não conter as cadeias proibidas. No caso da linguagem permitida por N/G^f, até a confirmação da falha, é $(ab+acb)^*$, enquanto SD/G^f possibilita a linguagem $(ad^*b+ad^*cd^*b+ad^*cd^*eg)^*$, também até a confirmação da falha.

Com relação ao chaveamento do supervisor pós-falha, esse precisa ser projetado para atender as necessidades do sistema e sua restrição adicional de impedir as cadeias proibidas. Considera-se para o Exemplo 9 duas possibilidades para obtenção de supervisores pós-falha. Como primeira possibilidade, o supervisor pós-falha é projetado apenas para não mais permitir as cadeias proibidas após a falha, sendo suficiente ao sistema atuar com um comportamento degradado, e sendo esse comportamento obtido a partir do comportamento faltoso de G^f. Esse primeiro supervisor pós-falha é mostrado em S_j^{deg}, na Figura 46. Como segunda possibilidade, o projeto do supervisor pós-falha aceita uma

parte do comportamento degradado, mas considera a utilização de algum componente redundante. Esse componente redundante introduz ao sistema o evento d_2, em substituição ao evento d que passou a ser impedido por ação do DCA[1]. Esse segundo supervisor-pós falha é mostrado em S_k^{deg}, também na Figura 46.

Figura 46 – Supervisores pós-falha obtidos para o Exemplo 9



Fonte: Elaborado pelo autor (2021).

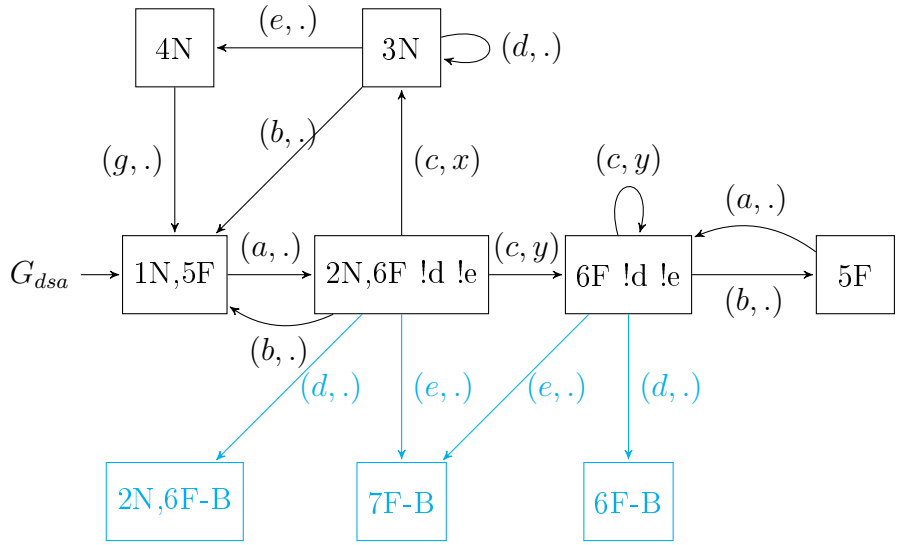
Considerando-se agora um mapeamento de sensores diferente para G_{dsa}, no qual a diferenciação ocorre associada com o evento c. Mesmo com o mapeamento de sensores, a linguagem não é A-diagnosticável segura pelo diagnosticador. Embora o mapeamento de sensores utilizado não seja suficiente para permitir a A-diagnosticabilidade segura do sistema, o mapeamento de sensores diminuiu a incerteza com relação à falha. O resultado desse segundo mapeamento de sensores pode ser observado no G_{dsa} da Figura 47.

A partir desse segundo mapeamento de sensores, a falha não é A-controlável segura, nem A-diagnosticável segura. Embora seja possível confirmar a ocorrência da falha, cadeias proibidas podem ocorrer antes da diagnose, a partir do alcance ao estado incerto de falha *(2N,6F !d !e)*.

Considerando a obtenção do DCA a partir do diagnosticador seguro ativo G_{dsa} da Figura 47 e fazendo uso do supervisor pós-falha S_k^{deg}, o resultado é mostrado em G_{DCA}, na Figura 48. O supervisor pós-falha é chaveado pelo DCA a partir do estado *(6F !d !e)* $\in Q_{dca}^{FC}$, de G_{DCA}. A transição com o evento INT_k foi colorizada em vermelho para destacar que essa transição não é uma transição com evento da planta (o evento não gera algum efeito na planta), e sim uma transição interna da estrutura de controle.

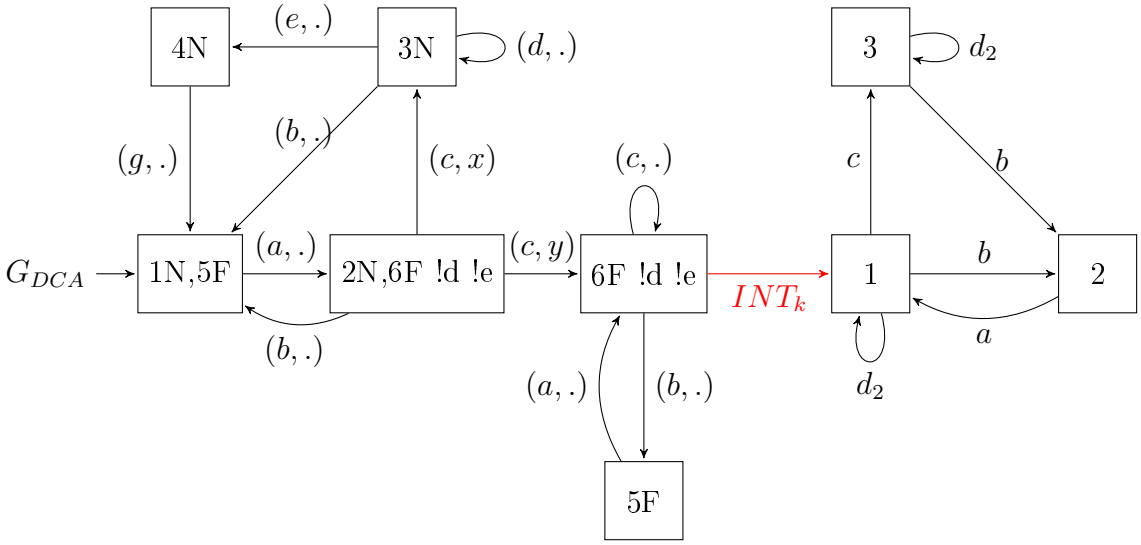
[1] O projeto do supervisor com o componente redundante fez uso de um outro modelo de planta e especificações, que utilizou o componente com o evento d_2, em substituição ao componente com o evento d.

Figura 47 – G_{dsa} com a segunda proposta de mapeamento de sensores. As transições e estados foram removidas a partir do estado *(2N,6F-B)*, para facilitar a visualização



Fonte: Elaborado pelo autor (2021).

Figura 48 – DCA obtido a partir de G_{dsa} e do supervisor pós-falha S_k^{deg}

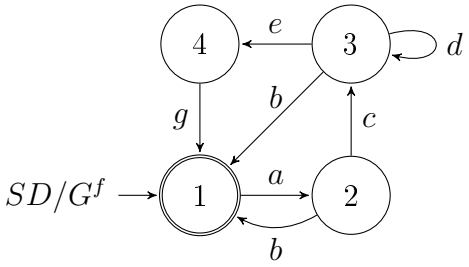


Fonte: Elaborado pelo autor (2021).

A ação conjunta $S \wedge DCA$ resulta no comportamento SD/G^f, até a confirmação da falha, mostrado na Figura 49. Após a confirmação da falha, o comportamento é o representado pelo supervisor pós-falha S_k^{deg}.

Mesmo o DCA sendo obtido com o uso de informações adicionais, desenvolvidas para a diagnose das falhas, o controle alcançado pela ação conjunta $S \wedge DCA$ pode não

Figura 49 – Comportamento em malha fechada SD/G^f para o Exemplo 9



Fonte: Elaborado pelo autor (2021).

resultar em um comportamento suficiente para o sistema, sendo necessário reprojetar o sistema, podendo, por exemplo, adicionar informações adicionais ao DCA e aumentar informações sobre as falhas.

6.4.2 Considerações sobre o uso do DCA

A solução de CTF ativo seguro a partir do diagnosticador-controlador exige uma linguagem controlável segura e esse pode ser um impeditivo na utilização dessa solução, principalmente considerando-se falhas com diversas cadeias proibidas e a existência de várias falhas.

Para aumentar as possibilidades de uso da solução, o próprio exemplo apresentado em Paoli, Sartini e Lafortune (2011) mostra o uso do mapeamento de sensores para permitir as condições de controlabilidade necessárias. Em Watanabe (2019), a condição de controlabilidade segura foi relaxada, possibilitando o uso da prognose e antecipando a possibilidade de ação para o controle seguro. Mesmo com o relaxamento da condição para a controlabilidade segura, com a introdução da A-controlabilidade segura, essa propriedade pode ainda ser muito restritiva em alguns casos.

A utilização do DCA pode permitir a controlabilidade segura mesmo quando o sistema não possui a propriedade da A-CS. As desabilitações impostas pelo DCA podem ser vistas como um controle robusto até o chaveamento do supervisor pós-falha, sendo então o uso do DCA uma estratégia híbrida de CTF. Portando, o DCA não se restringe em impedir eventos que alcançam os maus estados, impondo desabilitações auxiliares ao supervisor, ele também envolve a realização de chaveamentos, impondo o comportamento dos supervisores pós-falha.

Além disso, quando o processo de diagnose necessita de mais informações do que as

utilizadas para o controle, o DCA pode utilizar essas informações adicionais da diagnose para auxiliar no controle seguro do sistema, possibilitando soluções potencialmente menos restritivas.

Mesmo em situações onde o DCA não possui mais informações do que o supervisor, sua utilização do DCA pode ser considerada. Sejam sistemas nos quais a falha e os problemas com relação às cadeias proibidas foram identificados apenas após o sistema já estar em funcionamento, e para quando a utilização da arquitetura de CTF ativo é a mais adequada, não é necessário calcular um supervisor para lidar com a especificação de segurança e um diagnosticador para realizar os chaveamentos, podendo o DCA incorporar as duas tarefas.

Em Moreira e Leal (2020) foi apresentado um exemplo de uso do DCA em um sistema com conotação física, além de apresentar a possibilidade de uso de um DCA reduzido (projeção do DCA para um alfabeto mínimo que permite a diagnose e a controlabilidade segura).

7 CONCLUSÃO E TRABALHOS FUTUROS

Nesta tese, tratou-se do problema de controle tolerante a falhas de SEDs usando a abordagem ativa. Nessa abordagem, a partir de informações online, geralmente provenientes de um diagnosticador, é feito o chaveamento entre a lógica de controle projetada para supervisionar o funcionamento nominal da planta, e uma lógica de controle projetada especialmente para tratar cada uma das situações de falha.

De acordo com os trabalhos encontrados na literatura, a solução desse problema está associada à propriedade da controlabilidade segura, que consiste na capacidade de detectar a ocorrência de falha e, após essa detecção, existir um evento controlável que possa ser desabilitado para impedir a ocorrência de cadeias proibidas, as quais correspondem a comportamentos que são indesejados após a falha. A detecção de falha, antes mencionada, pode corresponder à diagnose ou prognose de falha, mas nesta tese considerou-se apenas a detecção pela diagnose. Assim, a diagnosticabilidade e a diagnosticabilidade segura são apresentadas como condições necessárias para a controlabilidade segura, que, por sua vez, é condição necessária para a solução do problema de controle tolerante a falhas ativo.

Nesta tese, mostrou-se que nem a diagnosticabilidade nem a diagnosticabilidade segura são condições necessárias para a controlabilidade segura. Introduziu-se então condições mais relaxadas, chamadas de A-diagnosticabilidade segura e A-controlabilidade segura, e mostrou-se que elas são necessárias e suficientes para garantir a solução para o problema do controle tolerante a falhas ativo.

Mesmo com essa flexibilização, a solução do problema de controle tolerante a falhas ativo requer que o sistema tenha essa propriedade da A-controlabilidade segura, o que ainda pode ser uma limitação em alguns casos. Para essa situação, nesta tese introduziu-se o Diagnosticador Controlador Ativo (DCA), o qual permite desabilitar eventos controláveis objetivando impedir comportamentos inseguros durante a incerteza da ocorrência de uma falha. Para as cadeias proibidas que permitem a controlabilidade segura, a imposição de desabilitações do DCA não interfere no comportamento nominal do sistema, sendo seu funcionamento similar ao obtido com a atuação do diagnosticador-controlador. Para falhas e cadeias proibidas que não atendem as condições de A-controlabilidade segura, o DCA impõe desabilitações que garantem a controlabilidade segura do sistema. A partir de uma adaptação na estrutura de CTF proposta por Paoli, Sartini e Lafortune (2011),

foi introduzida a estrutura de CTF usando o DCA.

Embora seja possível obter um supervisor com as mesmas informações adicionais disponibilizadas para o diagnosticador e efetuar o controle das cadeias proibidas durante as situações de incerteza da falha, isso ainda não desobriga o uso do diagnosticador para a diagnose da falha na estratégia de CTF ativo.

Quando as limitações impostas pela controlabilidade segura limitam o sistema de forma inaceitável para o projeto, foi proposta a noção de cadeias evitáveis, que relaxam as desabilitações impostas pelo controle seguro. Para seu uso é sugerida uma avaliação da magnitude dos problemas ocasionados pelas sequências indesejadas após a falha.

Em relação a trabalhos futuros, a utilização da arquitetura de CTF ativo de SEDs usando o DCA ainda oferece algumas possibilidades.

O trabalho ilustrou situações nas quais o processo de diagnose demanda uma maior quantidade de informação do que aquela disponível para a obtenção do supervisor. Uma vez que o CTF ativo necessita diagnosticar a falha, aproveitar a mesma estrutura que realiza a diagnose para também garantir a controlabilidade segura é uma otimização dos recursos disponíveis. Entre os possíveis acréscimos de informação ao DCA está o uso da prognose online de falhas, que agrega mais informações sobre a falha. Outra opção que parece interessante está no uso do DCA na abordagem de diagnose ativa não utilizada na tese, em que são aplicados ao sistema sinais para obter a diagnose. Nesse caso o DCA também agiria ativamente para identificar falhas, aumentando sua quantidade de informações.

As falhas consideradas no trabalho são falhas persistentes, ou seja, após a ocorrência da falha f, o sistema continua evoluindo de acordo com o modelo pós-falha, sendo este diferente do modelo nominal. Esse tipo de falha permite o chaveamento de um supervisor pós-falha quando a falha é diagnosticada. Em caso de falhas intermitentes, o chaveamento pode não ser adequado e, se o objetivo for apenas impedir as cadeias proibidas, o DCA pode ser uma estrutura adequada.

Uma vez que DCA permite a desabilitação de eventos controláveis que não são observáveis, a arquitetura usando o DCA pode ser interessante no contexto de uma estrutura descentralizada. Considerando-se a possibilidade de um componente do sistema sujeito à falha e que sua falha impacta em um segundo componente, um DCA associado com o componente faltoso pode solicitar desabilitações desse segundo componente.

Um diagnosticador pode ser reduzido para um subconjunto de eventos que permita manter as condições de diagnosticabilidade necessárias, mesmo com a redução da quantidade de eventos. A redução também pode ser aplicada para alcançar outros objetivos, como para o controle supervisório, no qual o processo de redução é baseado em encontrar conjuntos de estados que se comportam de maneira equivalente com relação as ações de controle. Nesse sentido é possível investigar formas de obter um DCA reduzido de forma a obter um resultado com menor espaço de estados.

Uma vez chaveados os supervisores pós-falha, isso não significa que novas falhas não possam ocorrer. Nesse sentido, uma possibilidade a ser explorada é o chaveamento para outro DCA, calculado para o comportamento degradado, mas considerando novas falhas. O mesmo pode ser feito em relação às situações de incerteza sobre a falha quando a ação conjunta $S \wedge DCA$ ocasionar um bloqueio. Uma estratégia pode ser elaborada para essas situações na qual pode ser chaveado um DCA que lida com a situação específica.

Identificar as situações de bloqueio a partir da ação de SD/G^f é um desafio quando o DCA possui informações diferentes das observações da planta, ou seja, $P_o^+ \neq P_o$. Nesse caso é necessário elaborar uma forma de realizar o mapeamento perceptivo para poder avaliar as situações de bloqueio resultantes da ação conjunta.

130

REFERÊNCIAS

BAILLIEUL, J.; SAMAD, T. **Encyclopedia of systems and control**. [S.l.]: Springer Publishing Company, Incorporated, 2015.

BASíLIO, J. C.; CARVALHO, L. K.; MOREIRA, M. V. Diagnose de falhas em sistemas a eventos discretos modelados por autômatos finitos. **Revista Controle & Automação**, v. 21, n. 5, p. 510–533, 2010.

BLANKE, M.; KINNAERT, M.; LUNZE, J.; STAROSWIECKI, M.; SCHRÖDER, J. **Diagnosis and fault-tolerant control**. [S.l.]: Springer, 2016. v. 3.

BORUTZKY, W. Fault tolerant control. In: **Bond Graph Modelling for Control, Fault Diagnosis and Failure Prognosis**. [S.l.]: Springer, 2021. p. 177–193.

CARVALHO, L. K. **Diagnose robusta de sistemas a eventos discretos**. Tese (Doutorado) — Tese (Doutorado em UFRJ COPPE-PEE Programa de Engenharia Elétrica)-Universidade Federal do Rio de Janeiro, 2011.

CASSANDRAS, C. G.; LAFORTUNE, S. **Introduction to discrete event systems**. [S.l.]: Springer Science & Business Media, 2009.

CHEN, J.; KUMAR, R. Polynomial test for stochastic diagnosability of discrete-event systems. **IEEE Transactions on Automation Science and Engineering**, IEEE, v. 10, n. 4, p. 969–979, 2013.

CIESLAK, R.; DESCLAUX, C.; FAWAZ, A. S.; VARAIYA, P. Supervisory control of discrete-event processes with partial observations. **IEEE Transactions on Automatic Control**, IEEE, v. 33, n. 3, p. 249–260, 1988.

CONTANT, O.; LAFORTUNE, S.; TENEKETZIS, D. Diagnosability of discrete event systems with modular structure. **Discrete Event Dynamic Systems**, Springer, v. 16, n. 1, p. 9–37, 2006.

CRUZ, V. H.; CARVALHO, L. K.; BASíLIO, J. C. Uma nova abordagem do mapeamento de sensores para obtenção de modelos de sistemas eventos discretos sujeitos a falhas. **Anais do XXIII Congresso Brasileiro de Automática**, v. 2, n. 1, 2020.

DEBOUK, R.; LAFORTUNE, S.; TENEKETZIS, D. Coordinated decentralized protocols for failure diagnosis of discrete event systems. **Discrete Event Dynamic Systems**, v. 10, n. 1, p. 33–86, 2000. ISSN 1573-7594. Disponível em: <http://dx.doi.org/10.1023/A:1008335115538>.

DERBEL, H.; YEDDES, M.; HADJ-ALOUANE, N. B.; ALLA, H. Diagnosis of a class of timed discrete event systems. In: **2006 8th International Workshop on Discrete Event Systems**. [S.l.: s.n.], 2006. p. 256–261.

EKANAYAKE, T.; DEWASURENDRA, D.; ABEYRATNE, S.; MA, L.; YARLAGADDA, P. Model-based fault diagnosis and prognosis of dynamic systems: a review. **Procedia Manufacturing**, v. 30, p. 435 – 442, 2019. ISSN 2351-9789. Digital Manufacturing Transforming Industry Towards Sustainable Growth. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2351978919300903>.

FRITZ, R.; ZHANG, P. Overview of fault-tolerant control methods for discrete event systems. **IFAC-PapersOnLine**, v. 51, n. 24, p. 88 – 95, 2018. ISSN 2405-8963. 10th

IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS 2018. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2405896318322249>.

HEIRUNG, T. A. N.; MESBAH, A. Input design for active fault diagnosis. **Annual Reviews in Control**, v. 47, p. 35 – 50, 2019. ISSN 1367-5788. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1367578819300070>.

JALOTE, P. **Fault tolerance in distributed systems**. [S.l.]: Prentice-Hall, Inc., 1994.

JIANG, S.; HUANG, Z.; CHANDRA, V.; KUMAR, R. A polynomial algorithm for testing diagnosability of discrete-event systems. **IEEE Transactions on Automatic Control**, IEEE, v. 46, n. 8, p. 1318–1321, 2001.

JIANG, S.; KUMAR, R. Failure diagnosis of discrete-event systems with linear-time temporal logic specifications. **IEEE Transactions on Automatic Control**, IEEE, v. 49, n. 6, p. 934–945, 2004.

JÚNIOR, W. A. S. et al. Diagnóstico de falhas baseado em autômatos temporizados: aplicação em um sistema modular de manufatura. Universidade Federal de Sergipe, 2016.

KUMAR, R.; GARG, V.; MARCUS, S. I. On controllability and normality of discrete event dynamical systems. **Systems & Control Letters**, Elsevier, v. 17, n. 3, p. 157–168, 1991.

KUMAR, R.; TAKAI, S. A framework for control-reconfiguration following fault-detection in discrete event systems. **IFAC Proceedings Volumes**, v. 45, n. 20, p. 848 – 853, 2012. ISSN 1474-6670. 8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1474667016348613>.

LAFORTUNE, S. On decentralized and distributed control of partially-observed discrete event systems. **Advances in Control Theory and Applications**, Springer, p. 171–184, 2007.

LAFORTUNE, S. Diagnosis of discrete event systems. In: _____. **Encyclopedia of Systems and Control**. London: Springer London, 2015. p. 268–275. ISBN 978-1-4471-5058-9. Disponível em: <https://doi.org/10.1007/978-1-4471-5058-9_56>.

LIN, C.; SHAOLONG, S.; FENG, L.; QIJUN, C.; CHENGJU, L. Weak diagnosability of discrete event systems. In: **15th IFAC Workshop on Discrete Event Systems**. [S.l.: s.n.], 2020.

LIN, F. Robust and adaptive supervisory control of discrete event systems. **IEEE Transactions on Automatic Control**, IEEE, v. 38, n. 12, p. 1848–1852, 1993.

LIN, F. Diagnosability of discrete event systems and its applications. **Discrete Event Dynamic Systems**, Springer, v. 4, n. 2, p. 197–212, 1994.

LIN, F.; WONHAM, W. M. On observability of discrete-event systems. **Information sciences**, Elsevier, v. 44, n. 3, p. 173–198, 1988.

LIU, F. Safe diagnosability of fuzzy discrete-event systems and a polynomial-time verification. **IEEE Transactions on Fuzzy Systems**, v. 23, n. 5, p. 1534–1544, 2015.

LIU, F.; QIU, D. Safe diagnosability of stochastic discrete event systems. **IEEE Transactions on Automatic Control**, v. 53, n. 5, p. 1291–1296, 2008.

LIU, F.; QIU, D. Diagnosability of fuzzy discrete-event systems: A fuzzy approach. **IEEE Transactions on Fuzzy Systems**, IEEE, v. 17, n. 2, p. 372–384, 2009.

LIU, F.; YANG, P.; ZHAO, R.; DZIONG, Z. Verification of safe diagnosability of stochastic discrete-event systems. **International Journal of Control**, Taylor and Francis, v. 0, n. 0, p. 1–8, 2020.

MOOR, T. A discussion of fault-tolerant supervisory control in terms of formal languages. **Annual Reviews in Control**, v. 41, p. 159 – 169, 2016. ISSN 1367-5788. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1367578816300049>.

MOREIRA, B. G.; LEAL, A. B. A proposal for an active diagnoser for safe fault-tolerant control of discrete event systems. In: **Workshop Series on Discrete Event Systems - WODES 2020**. [S.l.: s.n.], 2020.

NIGUEZ, J.; AMARI, S.; FAURE, J.-M. Fault-tolerant control of discrete event systems: Comparison of two approaches on the same case study. In: IEEE. **2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)**. [S.l.], 2015. p. 1–4.

NKE, Y.; LUNZE, J. Control reconfiguration based on unfolding of input/output automata. **IFAC Proceedings Volumes**, Elsevier, v. 45, n. 20, p. 866–873, 2012.

PAOLI, A.; LAFORTUNE, S. Safe diagnosability for fault-tolerant supervision of discrete-event systems. **Automatica**, Elsevier, v. 41, n. 8, p. 1335–1347, 2005.

PAOLI, A.; SARTINI, M.; LAFORTUNE, S. A fault tolerant architecture for supervisory control of discrete event systems. **IFAC Proceedings Volumes**, v. 41, n. 2, p. 6542 – 6547, 2008. ISSN 1474-6670. 17th IFAC World Congress. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1474667016399955>.

PAOLI, A.; SARTINI, M.; LAFORTUNE, S. Active fault tolerant control of discrete event systems using online diagnostics. **Automatica**, Elsevier, v. 47, n. 4, p. 639–649, 2011.

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

QIU, W.; WEN, Q.; KUMAR, R. Decentralized diagnosis of event-driven systems for safely reacting to failures. **IEEE Transactions on Automation Science and Engineering**, IEEE, v. 6, n. 2, p. 362–366, 2009.

RAMADGE, P. J.; WONHAM, W. M. The control of discrete event systems. **Proceedings of the IEEE**, IEEE, v. 77, n. 1, p. 81–98, 1989.

Raman, A.; Sreenivas, R. S. Fault-tolerant control of discrete-event systems with controllability failures. **IEEE Control Systems Letters**, v. 4, n. 3, p. 674–679, 2020.

ROZÉ, L.; CORDIER, M.-O. Diagnosing discrete-event systems: extending the "diagnoser approach" to deal with telecommunication networks. **Discrete Event Dynamic Systems**, Springer, v. 12, n. 1, p. 43–81, 2002.

SAMPATH, M.; LAFORTUNE, S.; TENEKETZIS, D. Active diagnosis of discrete-event systems. **IEEE Transactions on Automatic Control**, IEEE, v. 43, n. 7, p. 908–929, 1998.

SAMPATH, M.; SENGUPTA, R.; LAFORTUNE, S.; SINNAMOHIDEEN, K.; TENEKETZIS, D. C. Diagnosticability of discrete-event models. **IEEE Transactions on Automatic Control**, v. 40, n. 9, p. 1555–1575, september 1995.

SAMPATH, M.; SENGUPTA, R.; LAFORTUNE, S.; SINNAMOHIDEEN, K.; TENEKETZIS, D. C. Failure diagnosis using discrete-event models. **IEEE Transactions on Control Systems Technology**, IEEE, v. 4, n. 2, p. 105–124, 1996.

SÜLEK, A. N.; SCHMIDT, K. W. Computation of fault-tolerant supervisors for discrete event systems. **IFAC Proceedings Volumes**, Elsevier, v. 46, n. 22, p. 115–120, 2013.

THORSLEY, D.; TENEKETZIS, D. Diagnosability of stochastic discrete-event systems. **IEEE Transactions on Automatic Control**, IEEE, v. 50, n. 4, p. 476–492, 2005.

VIANA, G. S.; MOREIRA, M. V.; BASíLIO, J. C. Codiagnosability analysis of discrete-event systems modeled by weighted automata. **IEEE Transactions on Automatic Control**, IEEE, v. 64, n. 10, p. 4361–4368, 2019.

WANG, Y.; YOO, T.-S.; LAFORTUNE, S. Diagnosis of discrete event systems using decentralized architectures. **Discrete Event Dynamic Systems**, Springer, v. 17, n. 2, p. 233–263, 2007.

WATANABE, A.; MOREIRA, B. G.; LEAL, A. B.; CURY, J.; QUEIROZ, M. Hering de. Análise das condições para diagnosticabilidade e prognosticabilidade de falhas. In: **XIII Congresso Brasileiro de Automação Inteligente**. [S.l.: s.n.], 2017.

WATANABE, A. T.; LEAL, A. B.; CURY, J. E.; de Queiroz, M. H. Safe controllability using online prognosis. **IFAC-PapersOnLine**, v. 50, n. 1, p. 12359–12365, 2017. ISSN 2405-8963. 20th IFAC World Congress. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2405896317327106>.

WATANABE, A. T. Y. **Controlabilidade segura de sistemas a eventos discretos utilizando diagnose e prognose online**. Tese (Doutorado) — Universidade do Estados de Santa Catarina, 2019.

WEN, Q. **Fault-tolerant supervisory control of discrete-event systems**. Tese (Doutorado) — Iowa State University, 2009.

WEN, Q.; KUMAR, R.; HUANG, J.; LIU, H. A framework for fault-tolerant control of discrete event systems. **IEEE Transactions on Automatic Control**, IEEE, v. 53, n. 8, p. 1839–1849, 2008.

WITTMANN, T.; RICHTER, J.; MOOR, T. Fault-hiding control reconfiguration for a class of discrete event systems. **IFAC Proceedings Volumes**, Elsevier, v. 46, n. 22, p. 49–54, 2013.

YIN, X.; LAFORTUNE, S. Codiagnosability and coobservability under dynamic observations: Transformation and verification. **Automatica**, v. 61, p. 241–252, 2015. ISSN 0005-1098. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0005109815003490>.

YOO, T.-S.; GARCIA, H. Computation of fault detection delay in discrete-event systems. In: **Proceedings of the 14th International Workshop on Principles of Diagnosis, DX'03**. [S.l.: s.n.], 2003. p. 207–212.

YOO, T.-S.; LAFORTUNE, S. Polynomial-time verification of diagnosability of partially observed discrete-event systems. **IEEE Transactions on automatic control**, IEEE, v. 47, n. 9, p. 1491–1495, 2002.

YU, X.; JIANG, J. A survey of fault-tolerant controllers based on safety-related issues. **Annual Reviews in Control**, Elsevier, v. 39, p. 46–57, 2015.

ZAYTOON, J.; LAFORTUNE, S. Overview of fault diagnosis methods for discrete event systems. **Elsevier Annual Reviews in Control**, p. 308–320, 2013.

ZHAO, R.; LIU, F.; LIU, Z. Relative diagnosability of discrete-event systems and its opacity-based test algorithm. **International Journal of Control, Automation and Systems**, Springer, v. 15, n. 4, p. 1693–1700, 2017.