Lucas Vinícius Ribeiro Alves

Aplicação de Autômatos Sincronizáveis na Segurança de Sistemas a Eventos Discretos

Belo Horizonte, Brasil

2020

Lucas Vinícius Ribeiro Alves

Aplicação de Autômatos Sincronizáveis na Segurança de Sistemas a Eventos Discretos

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Doutor em Engenharia Elétrica.

Universidade Federal de Minas Gerais – UFMG

Escola de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica

Orientadora: Patrícia Nascimento Pena

Belo Horizonte, Brasil

2020

Alves, Lucas Vinícius Ribeiro.
A474a Aplicação de autômatos sincronizáveis na segurança de sistemas a eventos discretos [recurso eletrônico] / Lucas Vinícius Ribeiro Alves. - 2020.
1 recurso online (91 f. : il., color.) : pdf.

Orientadora: Patrícia Nascimento Pena.

Tese (doutorado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Bibliografia: f.87-91.

Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia elétrica - Teses. 2. Teoria do controle - Teses. 3. Sistemas de tempo discreto – Teses. I. Pena, Patrícia Nascimento. II. Universidade Federal de Minas Gerais. Escola de Engenharia. III. Título.

CDU: 621.3(043)

Ficha catalográfica: Biblioteca Prof. Mário Werneck, Escola de Engenharia da UFMG

UFMG
UNIVERSIDADE FEDERAL DE MINAS GERAIS

UNIVERSIDADE FEDERAL DE MINAS GERAIS
Escola de Engenharia
Programa de Pós-Graduação em Engenharia Elétrica

"Aplicação de Autômatos Sincronizáveis na Segurança de Sistemas a Eventos Discretos"

Lucas Vinicius Ribeiro Alves

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Doutor em Engenharia Elétrica.

Aprovada em 17 de dezembro de 2020.

Por:

Prof. Dr. Patrícia Nascimento Pena
DELT (UFMG)

Prof. Dr. José Eduardo Ribeiro Cury
DAS (UFSC)

Prof. Dr. João Carlos dos Santos Basílio
DEE (UFRJ)

Prof. Dr. Bruno Vilhena Adorno
DEE (UFMG)

Prof. Dr. Ricardo Hiroshi Caldeira Takahashi
DMAT (UFMG)

Prof. Dr. Vinícius Mariano Gonçalves
DEE (UFMG)

Este trabalho é dedicado à minha avó,
Maria Lúcia (in memoriam).

Agradecimentos

Os agradecimentos principais são direcionados aos meus pais, Onofre e Flávia, ao meu irmão Victor e ao meu amigo Diego pelo apoio durante todos esses anos de estudo. Agradeço também aos meus demais familiares e amigos que de forma direta ou indireta contribuíram para o desenvolvimento desse trabalho.

Agradecimentos especiais são direcionados ao Laboratório de Análise e Controle de Sistemas a Eventos Discretos da Universidade Federal de Minas Gerais (LACSED), bem como à minha orientadora, professora Patrícia, pelos conselhos que foram essenciais ao desenvolvimento desta tese.

"O tempo vai te rolar do alto do morro
E sinto as quinas vão machucar
Se prepara robozin".
(Robozin, Supercombo)

Resumo

Sistemas de manufatura estão cada vez mais sujeitos a ataques externos e falhas, seja por defeitos em sensores e atuadores ou seja por problemas de comunicação. Nesse contexto, a segurança desses sistemas é um problema cada vez mais estudado, inclusive na área de Sistemas a Eventos Discretos (SEDs) utilizando a Teoria de Controle Supervisório (TCS). Neste trabalho propomos tratar problemas de segurança em SEDs utilizando a Teoria de Autômatos Sincronizáveis, que nos permite utilizar uma palavra de sincronização para sincronizar os supervisores e plantas no caso onde, devido a uma falha, o estado que o supervisor estima que a planta se encontra não corresponde ao estado físico da planta. Os resultados desse trabalho permitem modelar sistemas e estender a TCS para lidar com modelos baseados em autômatos sincronizáveis, tanto pela síntese monolítica quando aplicando controle modular local, tratando também de técnicas complementares, como a redução e localização de supervisores. Além disso, propõe-se uma maneira de modelar sistemas que não são originalmente sincronizáveis como autômatos sincronizáveis pela adição de eventos de recuperação.

Palavras-chaves: Sistemas a Eventos Discretos; Teoria de Controle Supervisório; Autômatos Sincronizáveis.

Abstract

Manufacturing systems are increasingly subject to external attacks and failures, whether by defects in sensors and actuators or by communication problems. In this context, the security of these systems is an increasingly studied problem, including in the field of Discrete Event Systems (DES) using Supervisory Control Theory (SCT). In this work we propose to deal with security problems in DES using the Synchronizing Automata Theory, which allows us to use a synchronizing word to synchronize the supervisors and plants in case the state that the supervisor estimates the plant does not correspond to the physical state of the plant. The results of this work allow us to model systems and to extend the SCT to deal with models based in synchronizing automata, using the monolithic synthesis or applying local modular control, and to deal with complimentary techniques such as the supervisor reduction and supervisor localization. In addition, we propose a way to model systems that are not originally synchronizing as synchronizing automata inserting recovery events.

Key-words: Discrete Event Systems; Synchronizing Automata; Supervisory Control Theory.

Lista de ilustrações

Lista de abreviaturas e siglas

SED	Sistemas a Eventos Discretos
TCS	Teoria de Controle Supervisório
AFD	Autômato Finito Determinístico

Lista de símbolos

G	Um autômato qualquer
S	Um supervisor qualquer
$\in$	Pertence a
$\notin$	Não pertence a
$\mid$	Tal que
$\exists$	Existe
$\forall$	Para todo
$\bigcap$	Operador interseção de conjuntos
$\bigcup$	Operador união de conjuntos
Σ	Alfabeto ou conjunto de eventos
Σ_c	Conjunto de eventos controláveis
Σ_{nc}	Conjunto de eventos não controláveis
ϵ	Palavra nula
Σ^*	Conjunto de todas as palavras que podem ser formadas com os eventos de Σ incluindo a palavra nula ϵ
Q	Conjunto de estados
Q_m	Conjuntos de estados marcados
K, L	Linguagens quaisquer
$\mathcal{L}(G)$	Linguagem Gerada do autômato G
$\mathcal{L}_m(G)$	Linguagem Marcada do autômato G
$Syn(G)$	Linguagem Composta pelas Palavras de Sincronização do autômato G
$Syn_{q_0}(G)$	Linguagem Composta pelas Palavras de Sincronização do autômato G que levam ao estado inicial

Sumário

1 Introdução

Grande parte dos sistemas atuais têm sua dinâmica regida pela ocorrência de eventos assíncronos no tempo, podendo ser classificados como Sistemas a Eventos Discretos (SED). Uma das abordagens existentes para tratar o problema de controle de Sistemas a Eventos Discretos é o chamado Modelo RW (WONHAM, 2014), onde os sistemas são modelados por Autômatos Finitos Determinísticos e sobre esses modelos é aplicado uma metodologia chamada Teoria de Controle Supervisório (TCS) que permite a construção de uma entidade chamada *supervisor* que garante o comportamento seguro do sistema de maneira minimamente restritiva.

Na Teoria de Controle Supervisório, a planta e o supervisor (controlador) são tratados como entidades distintas e o supervisor tem como papel observar os eventos executados na planta e proibir a ocorrência de certos eventos, chamados eventos controláveis. A TCS vem sendo desenvolvida desde a década de 80, e inúmeras contribuições à literatura visando tratar o problema modularmente (WONHAM; RAMADGE, 1988; QUEIROZ; CURY, 2000), de maneira descentralizada (RUDIE; WONHAM, 1992), hierarquicamente (WONG; WONHAM, 1996), incrementalmente (BRANDIN; MALIK; MALIK, 2004) estão disponíveis.

Mais recentemente, a partir dos anos 2000, com o aumento da troca de informações em ambientes digitais, cresce a preocupação com a segurança de sistemas computacionais (SABOORI; HADJICOSTIS, 2007). No caso da planta e do supervisor, o sistema é passível de sofrer ataques de agentes externos a fim de roubar informações ou atuar maliciosamente no sistema, como mostrado na Figura 1.

Os ataques têm se tornando cada vez mais sofisticados, tendo como principais objetivos furtar informação, sabotar sistemas e extorquir (TANKARD, 2011; BEUHRING; SALOUS, 2014). Um dos principais tipos de ameaça são os chamados APT - *Advanced Persistent Threat*, usualmente desenvolvidos para atacar um alvo específico (VIRVILIS;

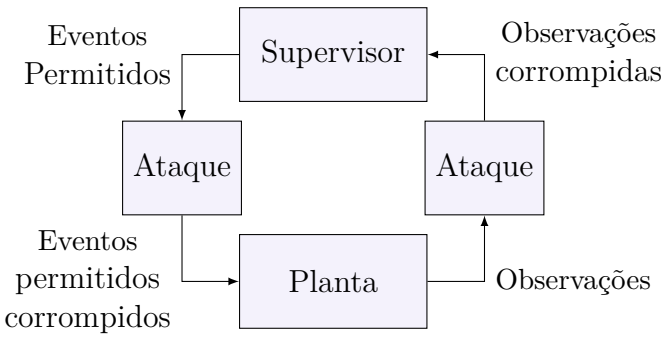


Figura 1 – Sistema sob ataque ou falha.

GRITZALIS; APOSTOLOPOULOS, 2013) e permanecer nesses sistemas sem ser notado por longos períodos de tempo.

Recuperação de falhas é, atualmente, parte essencial de um sistema de manufatura moderno. A maior parte da informação que trafega nas chamadas *Plantas Inteligentes* é acessada por meio de redes de comunicação em tempo real (CHRISTOFIDES et al., 2007), de forma que além de problemas relacionados a sensores e atuadores, precisamos levar em consideração ataques de agentes maliciosos ao sistema. Em sistemas computacionais muitos problemas podem ser resolvidos reiniciando o *software*, contudo em sistemas industriais, devido a restrições de segurança e confiabilidade, esse reinício não é trivial (ABAD et al., 2016).

Do ponto de vista de proteção contra ataques, quando o agente externo interfere no processo, existem poucas abordagens na literatura que tratam o problema (WAKAIKI; TABUADA; HESPANHA, 2017). Em geral existem extensões da Teoria de Controle Supervisório (TCS) relacionadas ao controle robusto ao controle resistente a falhas, mas que não tratam diretamente alterações relacionadas a um agente malicioso. Além disso, existe a propriedade da Opacidade em Sistemas a Eventos Discretos, que busca garantir segurança quanto ao acesso de invasores às informações da planta.

Em um sistema controlado pela TCS, o supervisor restringe a dinâmica do sistema inibindo a execução de eventos controláveis com o objetivo de garantir uma operação segura do sistema. Como observado na Figura 1, o supervisor observa eventos da planta e, dessa forma, estima seu estado atual. Essa observação, contudo, é suscetível a problemas de comunicação, falha de sensores ou mesmo da atuação de agentes maliciosos (SU, 2017).

É evidente que um agente malicioso que atua no sistema impedindo que o supervisor observe certos eventos que ocorrem na planta pode gerar uma dessincronização entre o supervisor e a planta. Quando esse agente malicioso é removido, o sistema pode estar em uma situação onde o estado físico da planta é diferente daquele estimado pelo supervisor, levando o sistema a estados não seguros ou a bloqueios.

O problema de recuperação de Sistemas a Eventos Discretos pode ser dividido em três sub-problemas (LOBORG, 1994):

1. Detecção: Consiste em detectar discrepâncias entre o estado do sistema e as especificações/supervisores (CARVALHO et al., 2018).

2. Diagnóstico: Consiste em detectar a falha que gerou a discrepância. Em Sistemas a Eventos Discretos, esse problema é tratado utilizando técnicas de diagnóstico (LAFORTUNE; LIN; HADJICOSTIS, 2018).

3. Recuperação: Após a eliminação da causa da falha, o agente malicioso ou as peças com defeito, a recuperação consiste em tornar o estado da planta e o estado do

supervisor consistente.

Podemos também acrescentar um sub-problema, anterior à *Detecção* que é *Robustez* do sistema, ou seja, a capacidade do sistema em resistir a ataques ou erros.

A *Detecção* e o *Diagnóstico* são ações correlatas e, muitas vezes, realizadas em um mesmo procedimento. Técnicas de Diagnóstico de Sistemas a Eventos Discretos, em geral, além de permitirem a identificação da ocorrência de uma falha, também permitem analisar qual dispositivo ocasionou a falha. Existem também técnicas de controle robusto como em Alves e coautores (ALVES et al., 2019), onde o supervisor é robusto à perda intermitente de observação de eventos não controláveis. Apesar de tais técnicas serem interessantes do ponto de vista de manter o sistema em funcionamento, essas técnicas são limitadas visto ser necessário conhecer, em tempo de projeto, quais eventos são propensos a falhas, além de não serem aplicáveis em muitos casos visto que a incerteza quanto ao estado da planta limita as ações de controle possíveis.

Shu (SHU, 2014) lida com o problema de recuperação em sistemas de manufatura acionando *eventos de recuperação* quando uma determinada cadeia de eventos leva o sistema a um estado de falha. Esses *eventos de recuperação* não podem ser desabilitados pelo supervisor e são usados por ele a fim de recuperar o sistema. Por outro lado, Andersson e coautores (ANDERSSON; LENNARTSON; FABIAN, 2009; ANDERSSON; LENNARTSON; FABIAN, 2010; ANDERSSON et al., 2011), Bergagard e coautores (BERGAGåRD; FABIAN, 2013; BERGAGåRD; FALKMAN; FABIAN, 2015) apresentam um método de recuperação de sistemas de manufatura, modelados por operações e coordenação de operações (*coordination of operations* – COP), após a ocorrência de falhas não previstas usando o conceito de *estados de reinicialização* permitindo que o processo de recuperação ressincronize o estado físico da planta com o estado do COP.

Neste trabalho propomos a utilização de Autômatos Sincronizáveis para lidar com o problema de recuperação em Sistemas a Eventos Discretos. Partimos do pressuposto que uma falha já foi detectada e resolvida, restando apenas levar o sistema a uma situação na qual a produção possa continuar.

Os Autômatos Sincronizáveis se mostram extremamente robustos quanto à ocorrência de falhas (DELYON; MALER, 1994), visto sua capacidade de voltar ao comportamento desejado por meio da sincronização. Uma situação similar ocorre quando o estado ativo da planta não corresponde ao estado ativo do supervisor e a sincronização permite levar o sistema a um estado conhecido e recuperar o sincronismo entre planta e supervisor.

No cotidiano, a palavra sincronização é utilizada nos mais variados sentidos, normalmente nos pontos de vista temporal e informacional.

Do ponto de vista temporal dizemos que dois sistemas idênticos estão sincronizados quando se comportam da mesma forma ao serem observados como, por exemplo, no nado

sincronizado. Por outro lado, dizemos que dois sistemas diferentes estão sincronizados quando mudanças nesses sistemas ocorrem ao mesmo tempo como, por exemplo, as imagens e sons em um vídeo.

Do ponto de vista informacional, dizemos que dois sistemas estão sincronizados quando compartilham a mesma informação, como quando os contatos de um telefone celular são replicados em um computador e os que estavam no computador são replicados no telefone. Nesse caso, dizemos que o telefone está sincronizado com o computador.

No sentido científico, sincronização guarda semelhanças com seu significado cotidiano, mas é um pouco mais geral. A sincronização começou a ser estudada pela física no século XVII, quando Huygens observou que dois relógios de pêndulo, fracamente acoplados, se tornam sincronizados (PIKOVSKY; ROSENBLUM; KURTHS, 2001).

No contexto de sistemas dinâmicos, o estudo da sincronização tem se desenvolvido muito na área de sistemas caóticos onde dois ou mais sistemas podem ser considerados sincronizados quando ajustam alguma propriedade a um comportamento comum devido ao acoplamento ou a uma força (BOCCALETTI et al., 2002). Dessa definição ampla, surgiram diversos tipos diferentes de sincronização como, por exemplo, sincronização idêntica, de fase, de atraso, e sincronização generalizada, entre diversas outras.

Apesar de, em geral, estar associada ao tempo, o conceito de sincronização também pode ser aplicado a sistemas orientados à ocorrência de eventos. No contexto dos Autômatos Finitos Determinísticos (AFD), dois autômatos idênticos estão sincronizados quando estão no mesmo estado. Nesse contexto, o maior desafio está em determinar uma *Palavra de Sincronização* para esse autômato, ou seja, uma palavra tal que, ao ser executada, sempre leva dois autômatos idênticos ao mesmo estado, independentemente dos estados em que se encontram. Essa palavra nem sempre existe, de forma que a sua existência nos permite classificar autômatos como sincronizáveis ou não sincronizáveis.

A sincronização de autômatos apresenta diversas aplicações práticas, como por exemplo na recuperação de falhas. Como grande parte dos sistemas pode ser modelado como um sistema discreto quando observado em alto nível e, em geral, é difícil estimar o estado atual em um sistema, a existência de uma palavra de sincronização pode ter um papel fundamental nesse contexto.

1.1 Objetivos

O objetivo desse trabalho é desenvolver uma metodologia de recuperação de sincronização entre o supervisor e a planta utilizando o conceito de palavra de sincronização. Como objetivos principais, podemos enumerar:

1. Modelar problemas de controle supervisório utilizando autômatos sincronizáveis.

2. Analisar sob quais circunstâncias as operações relacionadas à TCS, quando aplicadas a autômatos sincronizáveis, resultam em autômatos sincronizáveis.

3. Analisar a existência de sincronismo em técnicas correlatas à TCS como:

   a) Controle Modular Local

   b) Redução de Supervisores

   c) Localização de Supervisores

4. Utilizar eventos de recuperação para tornar autômatos em autômatos sincronizáveis

Como objetivos secundários, podemos citar:

1. Adicionar novos eventos ao sistema de forma a torná-lo sincronizável e assim recuperar sistemas legados.

2. Utilizar propriedades de autômatos sincronizáveis para redirecionar o sistema a determinados estados.

3. Desenvolver heurísticas que auxiliam na busca por palavras de sincronização.

1.2 Organização do Trabalho

O Capítulo 2 trata dos conceitos básicos necessários no desenvolvimento e entendimento deste trabalho. No Capítulo 3 apresentamos os resultados teóricos desenvolvidos para extensão da TCS para lidar com autômatos sincronizáveis com relação ao estado inicial. No Capítulo 4 apresentamos a extensão da TCS para problemas com autômatos que podem ser transformados em autômatos sincronizados utilizando eventos de recuperação e no Capítulo 5 apresentamos algoritmos desenvolvidos para lidar com sistemas que não são originalmente modelados por autômatos sincronizáveis. No Capítulo 6 apresentamos um estudo de caso e por fim, no Capítulo 7 são apresentadas as conclusões.

2 Conceitos Preliminares

Sistemas a eventos discretos são sistemas dinâmicos que interagem com entidades externas por meio de estímulos, chamados eventos. Esses eventos podem ser o início ou fim de uma tarefa, o acionamento de um sensor, além de eventos internos, como o fim de uma temporização, por exemplo. Em geral, define-se estado como a configuração do sistema entre a ocorrência de dois eventos consecutivos, ou seja, a ocorrência de um evento acarreta em uma transição de estado no sistema (CASSANDRAS; LAFORTUNE, 2009), mesmo quando a transição leva ao mesmo estado de origem. Neste trabalho foi utilizada a teoria de autômatos e linguagens para modelar Sistemas a Eventos Discretos.

2.1 Linguagens

O conjunto de eventos de um SED pode ser visto como o conjunto de símbolos de uma linguagem e as sequências de eventos possíveis em determinado sistema podem ser tratadas como palavras.

Definição 2.1.1. Seja Σ um conjunto finito de símbolos, usualmente referido como alfabeto. O conjunto Σ^+ contém todas as sequências finitas de símbolos, na forma $\sigma_1\sigma_2...\sigma_k$ onde $k > 0$ e $\sigma_i \in \Sigma$. O conjunto $\Sigma^* = \{\epsilon\} \cup \Sigma^+$ contém todas as sequências finitas de símbolos, incluindo a sequência vazia, com nenhum símbolo, representada por ϵ.

Uma sequência $s \in \Sigma^*$ é, usualmente, chamada de palavra ou cadeia sobre o alfabeto Σ. O comprimento ou cardinalidade de s é denotado por $|s|$ e definido como:

$$|\epsilon| = 0$$
$$|\sigma_1\sigma_2...\sigma_k| = k.$$

Definição 2.1.2. Sejam $s, u \in \Sigma^*$, uma sequência $t = su$ é formada pela concatenação de s e u. Neste caso, diz-se que s é um prefixo de t e u é um sufixo de t.

Usualmente é de interesse um subconjunto das sequências formadas por símbolos de um alfabeto, sendo esse subconjunto chamado de linguagem.

Definição 2.1.3. Seja $L \subseteq \Sigma^*$ um conjunto de sequências sobre o alfabeto Σ. O conjunto L é denominado uma linguagem.

É possível definir diversas operações em linguagens, como união, complemento, interseção e diferença, além da concatenação.

Neste trabalho, a fim de simplificar a notação, uma linguagem composta por apenas uma sequência $\{u\}$ será representada apenas por u e a linguagem vazia é representada por $\emptyset$.

Definição 2.1.4. Sejam $L_1, L_2 \subseteq \Sigma^*$ duas linguagens quaisquer sobre o alfabeto Σ. A concatenação de L_1 e L_2 é dada por:

$$L_1 L_2 = \{u_1 u_2 \,|\, u_1 \in L_1 \wedge u_2 \in L_2\}.$$

A concatenação é uma operação associativa que admite como identidade a linguagem composta pela sequência vazia ϵ, ou seja, $L\epsilon = \epsilon L$ e, além disso, é uma operação distributiva sobre a união, logo, $L(L_1 \cup L_2) = LL_1 \cup LL_2$.

A potência de uma linguagem é definida como $L^0 = \{\epsilon\}$, $L^1 = L$ e, por indução, $L^n = L^{n-1}L$, para $n \geq 2$.

Definição 2.1.5. Seja L uma linguagem, o fechamento de Kleene de L, denotada por L^*, é a união de todas as potências de L:

$$L^* = \bigcup_{n \geq 0} L^n.$$

O prefixo fechamento de uma linguagem L, denotado por $\overline{L}$ é uma linguagem que contém todos os prefixos de sequências pertencentes a L, incluindo a sequência vazia ϵ:

$$\overline{L} = \{s \in \Sigma^* \,|\, u \in \Sigma^* \wedge su \in L\}.$$

Outra operação comum, aplicada a sequências e linguagens, é a projeção natural, partindo de um alfabeto Σ_1 para um alfabeto menor Σ_2, tal que $\Sigma_2 \subseteq \Sigma_1$.

Definição 2.1.6. Seja $s\sigma \in \Sigma_1^*$, a projeção de $s\sigma$ para um alfabeto $\Sigma_2 \subseteq \Sigma_1$ é definida como:

$$P_{\Sigma_1 \to \Sigma_2}(s\sigma) = P_{\Sigma_1 \to \Sigma_2}(s) P_{\Sigma_1 \to \Sigma_2}(\sigma) \text{ para } s \in \Sigma_1^*, \sigma \in \Sigma_1.$$

onde:

$$\begin{aligned} P_{\Sigma_1 \to \Sigma_2}(\epsilon) &= \epsilon \\ P_{\Sigma_1 \to \Sigma_2}(\sigma) &= \begin{cases} \sigma & \text{se } \sigma \in \Sigma_2 \\ \epsilon & \text{se } \sigma \in \Sigma_1 \setminus \Sigma_2. \end{cases} \end{aligned}$$

A projeção inversa de uma sequência mapeia uma sequência de um alfabeto menor Σ_2 em um alfabeto maior Σ_1, sendo definida como:

$$P^{-1}_{\Sigma_1 \to \Sigma_2}(t) = \{s \in \Sigma_1^* \mid P_{\Sigma_1 \to \Sigma_2}(s) = t\}.$$

É importante observar que a projeção inversa retorna todas as sequências do alfabeto Σ_1 que projetadas formam t, ou seja, $s \in P^{-1}_{\Sigma_1 \to \Sigma_2}(P_{\Sigma_1 \to \Sigma_2}(s))$. Ambas as definições, de projeção e projeção inversa podem ser estendidas para linguagens como mostrado a seguir:

Para $L \subseteq \Sigma_1^*$:

$$P_{\Sigma_1 \to \Sigma_2}(L) = \{t \in \Sigma_2^* \mid (\exists s \in L)\, [P_{\Sigma_1 \to \Sigma_2}(s) = t]\}.$$

Para $L \subseteq \Sigma_2^*$:

$$P^{-1}_{\Sigma_1 \to \Sigma_2}(L) = \{s \in \Sigma_1^* \mid (\exists t \in L)\, [P_{\Sigma_1 \to \Sigma_2}(s) = t]\}. \tag{2.1}$$

Dado um alfabeto Σ, existe um conjunto de linguagens, chamado conjunto de linguagens regulares, $F \in 2^{\Sigma^*}$ que apresenta as seguintes propriedades:

1. F contém as linguagens $\emptyset$ e $\sigma, \forall \sigma \in \Sigma$.
2. F é um conjunto fechado sob as operações de união, concatenação e estrela de Kleene.

Uma característica importante das linguagens regulares é a possibilidade de representá-las como autômatos ou expressões regulares.

Definição 2.1.7. Uma expressão regular é uma expressão formal definida recursivamente como:

1. As linguagens $\emptyset$ e ϵ são expressões regulares.
2. Qualquer símbolo $\sigma \in \Sigma$ é uma expressão regular.

3. Se e_1 e e_2 são expressões regulares, então $(e_1 \cup e_2)$, $(e_1 e_2)$ e e^* também são expressões regulares.

2.2 Autômatos Finitos Determinísticos

Autômatos são grafos direcionados com arestas representando transições, vértices representando estados, o estado inicial é indicado por uma seta e os estados marcados são indicados por círculos concêntricos, como mostrado na Figura 2, contudo uma definição mais formal pode ser apresentada:

Definição 2.2.1. (HOPCROFT; MOTWANI; ULLMAN, 2006) Seja $G = (Q, \Sigma, \delta, q_0, Q_m)$ um autômato finito determinístico definido por uma quíntupla, onde Q é um conjunto finito de estados, Σ é o conjunto finito de símbolos (alfabeto), $\delta : Q \times \Sigma \to Q$ é a função de transição de estados, $q_0 \in Q$ é o estado inicial e $Q_m \subseteq Q$ é o conjunto de estados marcados de Q.

Usualmente δ é uma função de transição parcial e quando ela está definida para um estado q e um evento σ, denota-se por $\delta(q, \sigma)!$. Além disso, a função de transição pode ser estendida para reconhecer cadeias sobre Σ^* como $\delta(q, \sigma s) = q'$ se $\delta(q, \sigma) = x$ e $\delta(x, s) = q'$.

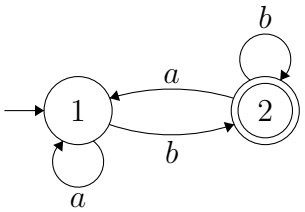


Figura 2 – Exemplo de Autômato Finito Determinístico

Um caminho, em um autômato, é denotado por:

$$c : q_0 \xrightarrow{\sigma_1} q_1 ... q_{n-1} \xrightarrow{\sigma_n} q_n. \tag{2.2}$$

ou, de uma forma compacta, como $c : q_0 \xrightarrow{\sigma_1 ... \sigma_n} q_n$. O conjunto de todas as sequências de símbolos do alfabeto Σ que, partindo do estado inicial, formam um caminho num autômato G compõem a linguagem gerada pelo autômato, representada por $\mathcal{L}(G)$. Além disso, a linguagem marcada de um autômato é definida como um subconjunto da linguagem gerada, $\mathcal{L}_m(G) \subseteq \mathcal{L}(G)$, composto por todas as sequências que terminam em um estado marcado.

Existe também o caminho vazio, representado por $q \xrightarrow{\epsilon} q$, onde ϵ é a palavra vazia, e para qualquer autômato G, $\epsilon \in \mathcal{L}(G)$.

Existem diversas operações que podem ser aplicadas a autômatos, como a de Parte Acessível, Parte Coacessível e Trim.

A parte acessível de um autômato consiste no subautômato composto por todos os estados que podem ser alcançados a partir do estado inicial. Esta operação não modifica a linguagem gerada nem a linguagem marcada do autômato, pois não existe um caminho que, partindo do estado inicial leve aos estados removidos.

A parte coacessível de um autômato consiste no subautômato composto somente pelos estados que podem alcançar um estado marcado. Essa operação modifica a linguagem gerada do autômato, porém não modifica a linguagem marcada.

O *trim* de um autômato é o resultado da aplicação das operações de parte acessível e parte coacessível, em qualquer ordem, sobre o autômato. A operação não modifica a linguagem marcada do autômato.

Existem também operações de composição de múltiplos autômatos, permitindo a representação do comportamento conjunto dos autômatos originais. As principais operações entre dois ou mais autômatos são o produto e a composição paralela (ou produto síncrono).

Definição 2.2.2. Sejam dois autômatos $G_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, Q_{m1})$ e $G_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, Q_{m2})$, o produto de G_1 e G_2, é dado por $G_{1\times2} = (Q_1 \times Q_2, \Sigma_1 \cap \Sigma_2, \delta_{1\times2}, (q_{01} \times q_{02}), Q_{m1} \times Q_{m2})$ onde:

$$\delta_{1\times2}((q_1, q_2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)) \text{ se } \delta_1(q_1, \sigma) \text{ e } \delta_2(q_2, \sigma) \text{ estão definidos} \\ \text{indefinido caso contrário} \end{cases} \tag{2.3}$$

O produto entre dois autômatos gera um novo autômato cuja linguagem marcada é dada por $\mathcal{L}_m(G_{1\times2}) = \mathcal{L}_m(G_1) \cap \mathcal{L}_m(G_2)$ e a linguagem gerada é dada por $\mathcal{L}(G_{1\times2}) = \mathcal{L}(G_1) \cap \mathcal{L}(G_2)$.

Definição 2.2.3. Sejam dois autômatos $G_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, Q_{m1})$ e $G_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, Q_{m2})$, a composição paralela de G_1 e G_2 é dada por $G_1||G_2 = (Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta_{1||2}, (q_{01} \times q_{02}), Q_{m1} \times Q_{m2})$ onde:

$$\delta_{1||2}((q_1, q_2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)) \text{ se } \delta_1(q_1, \sigma) \text{ e } \delta_2(q_2, \sigma) \text{ estão definidos} \\ (\delta_1(q_1, \sigma), q_2) \text{ se } \delta_1(q_1, \sigma) \text{ está definido e } \sigma \notin \Sigma_2 \\ (q_1, \delta_2(q_2, \sigma)) \text{ se } \delta_2(q_2, \sigma) \text{ está definido e } \sigma \notin \Sigma_1 \\ \text{indefinido caso contrário} \end{cases}$$

A composição paralela entre dois autômatos gera um novo autômato cuja linguagem

marcada é dada por $\mathcal{L}_m(G_{1||2}) = P^{-1}_{\Sigma_1\cup\Sigma_2\to\Sigma_1}(\mathcal{L}_m(G_1) \cap P^{-1}_{\Sigma_1\cup\Sigma_2\to\Sigma_2}(\mathcal{L}_m(G_2))$ e a linguagem gerada é dada por $\mathcal{L}(G_{1||2}) = P^{-1}_{\Sigma_1\cup\Sigma_2\to\Sigma_1}(\mathcal{L}(G_1) \cap P^{-1}_{\Sigma_1\cup\Sigma_2\to\Sigma_2}(\mathcal{L}(G_2))$.

2.3 Teoria de Controle Supervisório

A Teoria de Controle Supervisório (TCS) (WONHAM, 2014) é uma abordagem que permite, a partir de plantas e especificações de segurança modeladas por autômatos finitos determinísticos, encontrar um agente de controle, capaz de desabilitar certos eventos a fim de garantir um comportamento seguro e não-bloqueante, em malha fechada.

Na TCS, o conjunto $\Sigma = \Sigma_c \cup \Sigma_{nc}$ é o conjunto de eventos de um sistema, sendo particionado em dois subconjunto, Σ_c, o conjunto de eventos controláveis, e Σ_{nc}, o conjunto de eventos não controláveis, que por sua vez não podem ser desabilitados pelo agente de controle.

O problema de controle supervisório consiste em sintetizar um controlador, chamado *supervisor*, que garanta o comportamento controlável e não bloqueante do sistema de maneira minimamente restritiva. Para que seja possível verificar se existe um supervisor que implementa uma determinada linguagem alvo são necessários os conceitos de controlabilidade de uma linguagem e fechamento de uma linguagem.

Definição 2.3.1. (WONHAM; RAMADGE, 1988) Sejam K e L linguagens, a linguagem K é dita controlável em relação a uma planta G se, e somente se:

$$\overline{K}\Sigma_{nc} \cap L \subseteq \overline{K}.$$

Ou seja, dizemos que K é $\mathcal{L}(G)$-controlável se toda palavra $s\sigma$ onde $s \in \overline{K}$, $\sigma \in \Sigma_{nc}$ e $s\sigma \in \mathcal{L}(G)$ também pertence a $\overline{K}$. Outro fato notável é que quando todos os eventos são controláveis ($\Sigma_{nc} = \emptyset$) chegamos à expressão $\overline{K} \cap \mathcal{L}(G) \subseteq \overline{K}$ que é sempre verdade se $\overline{K}$ e $\mathcal{L}(G)$ possuírem ao menos uma palavra em comum.

Definição 2.3.2. (WONHAM; RAMADGE, 1988) Sejam K e L linguagens, a linguagem K é dita fechada em relação a uma linguagem L se, e somente se:

$$\overline{K} \cap L = K.$$

Ou seja, dizemos que K é $\mathcal{L}_m(G)$-fechada se qualquer prefixo de qualquer palavra de K que pertença também a $\mathcal{L}_m(G)$ for uma palavra de K e vice-versa.

Sendo $K \subseteq \mathcal{L}_m(G)$ com $K \neq \emptyset$ existe um supervisor S não-bloqueante, ou seja, que permita o alcance de um estado marcado, tal que $\mathcal{L}_m(S/G) = K$ se e somente se K é $\mathcal{L}(G)$-controlável e $\mathcal{L}_m(G)$-fechada.

Como a controlabilidade é uma condição necessária para a existência de supervisores, o problema restante é o que fazer quando K não é $\mathcal{L}(G)$-controlável.

A Teoria de Controle Supervisório estabelece que existe uma sublinguagem $K' \subset K$ única que é $\mathcal{L}(G)$-controlável de maneira minimamente restritiva. O problema passa a ser então, dada uma planta G e uma *linguagem alvo* $K \subseteq \Sigma^*$, encontrar um supervisor S tal que $\mathcal{L}(S/G) \subseteq K$.

Definição 2.3.3. (WONHAM; RAMADGE, 1988) Sejam K uma linguagem e G um autômato, $\mathcal{C}(K)$ é o conjunto de todas as sublinguagens de K que são $\mathcal{L}(G)$-controláveis.

$$\mathcal{C}(K) = \{K' | (K' \subseteq K \text{ e } \overline{K'}\Sigma_{nc} \cap \mathcal{L}(G) \subseteq \overline{K'})\}$$

O conjunto $\mathcal{C}(K)$, é não vazio e fechado sob a operação de união, ou seja, a união de quaisquer dois elementos do conjunto também está no conjunto. Dessa maneira, pode-se definir que $sup\mathcal{C}(K) = \bigcup\{K' : K' \in \mathcal{C}(K)\}$, ou seja, a união de todos os elementos de $\mathcal{C}(K)$ é a máxima sublinguagem $\mathcal{L}(G)$-controlável de K.

Para $K_S = sup\mathcal{C}(K \cap \mathcal{L}_m(G))$, se $K_S \neq \emptyset$ existe um supervisor controlável e não bloqueante S tal que $\mathcal{L}_m(S/G) = K_S$ (WONHAM, 2014).

2.3.1 Controle Supervisório Modular Local

A explosão de estados ocorrida durante a síntese de um supervisor monolítico pode ser evitada utilizando técnicas de controle descentralizado (LIN; WONHAM, 1988; RUDIE; WONHAM, 1991) onde é sintetizado um supervisor para cada especificação do sistema. O Controle Supervisório Modular Local (QUEIROZ; CURY, 2002) é uma técnica de controle descentralizado na qual cada supervisor local tem apenas uma visão parcial da planta, o que permite sua aplicação em problemas maiores.

A *planta global* G é composta por n sub-plantas $H_i, i = 1, \ldots, n$, tal que o seu conjunto de eventos Σ_{H_i} é disjunto e $G = ||_{i=1}^{n} H_i$. A *especificação global* E é composta de m sub-especificações $E_j, j = 1 \ldots m$ cujos conjuntos de eventos são representados por Σ_{E_j} e $E = ||_{j=1}^{m} E_j$.

Definição 2.3.4. Sejam $E_j, j \in \{1 \ldots m\}$ as m sub-especificações do sistema e $H_i, i \in \{1 \ldots n\}$ as n sub-plantas do sistema. Uma *planta local* G_j é definida como:

$$G_j = ||_{a \in A_j} H_a$$

onde

$$A_j = \{i \in \{1 \dots n\} | \Sigma_{H_i} \cap \Sigma_{E_j} \neq \emptyset\}.$$

No Controle Modular Local, um *supervisor local*, controlável e não-bloqueante, é definido para cada especificação local:

Definição 2.3.5. Sejam $E_j, j \in \{1 \dots m\}$ as m sub-especificações do sistema e $G_j, j \in \{1 \dots m\}$ as m plantas locais do sistema. Um supervisor local S_j pode ser computado tal que $S_j = sup\mathcal{C}(K_j, G_j)$, onde $K_j = \mathcal{L}_m(G_j \parallel E_j)$.

Cada supervisor é não-bloqueante por construção, mas o comportamento combinado de todos os supervisores locais pode levar a uma situação de bloqueio, dessa forma, um teste de não-conflito (2.4) deve ser aplicado.

Definição 2.3.6. (QUEIROZ; CURY, 2002) Sejam S_j, para $j \in \{1 \dots m\}$, supervisores locais controláveis e não-bloqueantes. O comportamento conjunto desses supervisores é não-conflitante quando:

$$||_{j=1}^{m} \overline{S_j} = \overline{||_{j=1}^{m} S_j}. \tag{2.4}$$

Se (2.4) é satisfeita, a ação conjunta dos supervisores locais é minimamente restritiva.

2.3.2 Redução de Supervisores

Um supervisor monolítico S tem, usualmente, um número grande de estados, o que torna sua aplicação difícil em problemas reais. Esse tipo de supervisor incorpora em si mesmo a dinâmica da planta, de forma que $S = S||G$. Qualquer autômato S_{SIM} tal que $S = S_{SIM}||G$ é uma *equivalência de controle* em relação a S (SU; WONHAM, 2004). Quando S_{SIM} possui menos estados que o supervisor original, ele é chamado de supervisor reduzido.

Os algoritmos de redução de supervisores buscam por um autômato S_{SIM}, menor que S e, no melhor caso encontram um dos autômatos equivalentes em controle a S com o menor número de estados possível (chamado S_{MIN}).

O Problema do Supervisor Mínimo (MSP - *Minimal Supervisor Problem*) é NP-difícil, contudo, usualmente, um supervisor S_{SIM} significativamente menor que S pode ser computado usando algoritmos polinomiais, como os mostrados em (SU; WONHAM, 2004; VAZ; WONHAM, 1986).

Um supervisor reduzido, equivalente em controle a $S = (Q_s, \Sigma, \delta_s, q_{s0}, Q_{sm})$ é construído utilizando uma *cobertura de controle* $\mathcal{C}$.

Definição 2.3.7. Uma relação $\mathcal{R} \subseteq Q_s \times Q_s$ é uma relação de consistência de controle se, $\forall (q, q') \in Q_s$, $(q, q') \in \mathcal{R}$ se e somente se:

$$E(q) \cap D(q') = E(q') \cap D(q) = \emptyset \tag{2.5}$$

$$T(q) = T(q') \Rightarrow M(q) = M(q') \tag{2.6}$$

onde

$$D(q) = \{\sigma \in \Sigma | \neg \delta_s(q, \sigma)! \wedge (\exists s \in \Sigma^*)[\delta_s(q_{s0}, s) = q \wedge \delta(q_0, s\sigma)!]\} \tag{2.7}$$

$$E(q) = \{\sigma \in \Sigma | \delta_s(q, \sigma)!\} \tag{2.8}$$

$$M(q) = \begin{cases} 1 & \text{se } q \in Q_{sm}, \\ 0 & \text{caso contrário} \end{cases} \tag{2.9}$$

$$T(q) = \begin{cases} 1 & \text{se } (\exists s \in \Sigma^*)\delta_s(q_{s0}, s) = q \wedge \delta(q_0, s) \in Q_m, \\ 0 & \text{caso contrário} \end{cases} \tag{2.10}$$

Utilizando a *relação de consistência de controle* $\mathcal{R}$ é possível definir uma cobertura de controle.

Definição 2.3.8. (SU; WONHAM, 2004) Uma cobertura $\mathcal{C} = \{X_j \subset Q | j \in J\}$ de um conjunto Q_s é um conjunto indexado tal que $\mathcal{C} \subseteq (2^{Q_s} \setminus \emptyset)$. Uma cobertura no conjunto de estados de S é chamada *cobertura de controle* quando:

$$(\forall j \in J)[X_j \neq \emptyset \wedge (\forall q, q' \in X_j))(q, q') \in \mathcal{R}] \tag{2.11}$$

$$(\forall j \in J)(\forall \sigma \in \Sigma)(\exists j' \in J)[(\forall q \in X_j)\delta_s(q, \sigma)! \Rightarrow \delta_s(q, \sigma) \in X_{j'}] \tag{2.12}$$

$$\bigcup_{j \in J} X_j = Q_s \tag{2.13}$$

A cobertura de controle consiste em um particionamento do conjunto de estados do autômato de tal forma que as ações de controle executadas nos estados pertencentes a um subconjunto X_j não conflitam entre si. Quando os subconjuntos X_j de uma cobertura de controle $\mathcal{C}$ são disjuntos par a par, esse cobertura de controle é uma *congruência de controle*.

Na Definição 2.3.7, $D(q)$ representa o conjunto de desabilitações no estado q, $E(q)$ é o conjunto de eventos habilitados no estado q, $M(q)$ se refere à marcação do estado no supervisor e $T(q)$ se refere à marcação do estado na planta (SU; WONHAM, 2004). A equação (2.5) estabelece que a interseção entre os eventos habilitados e desabilitados deve ser vazia e a equação (2.6) estabelece que, se um par de estados tem a mesma marcação na planta, eles também devem ter a mesma marcação no supervisor.

Usando uma cobertura de controle $\mathcal{C}$, é possível construir um supervisor $S_{SIM} = (J, \Sigma, \delta_{SIM}, j_0, J_m)$ equivalente em controle a S, onde:

- J é o conjunto de índices de $\mathcal{C}$.
- $J_m = \{j \in J \mid X_j \cap Q_{sm} \neq \emptyset\}$.
- $j_0 \in \{j \in J \mid q_{s0} \in X_j\}$.
- $\delta_{SIM}(j, \sigma) = j'$ é definido se $(\exists q \in X_j)\delta_s(q, \sigma) \in X_{j'} \wedge (\forall q' \in X_j)[\delta_s(q', \sigma)! \Rightarrow \delta_s(q', \sigma) \in X_{j'}]$.

Quanto menor o número de elementos na cobertura de controle $\mathcal{C}$, menor será o supervisor reduzido S_{SIM}. Existem algoritmos na literatura (VAZ; WONHAM, 1986; SU; WONHAM, 2004; SIVOLELLA, 2005) que buscam por uma boa cobertura de controle. Usualmente esses algoritmos buscam por uma congruência de controle a fim de reduzir o espaço de busca do problema. O conceito de normalidade, introduzido em (SU; WONHAM, 2004), garante que o supervisor reduzido não possui informação desnecessária.

Definição 2.3.9. (SU; WONHAM, 2004) Um supervisor $S_{SIM} = (J, \Sigma, \delta_{SIM}, j_0, J_m)$ é *normal* em relação a S se:

1. $(\forall j \in J)(\exists s \in \mathcal{L}(S))[\delta_{SIM}(j_0, s) = j]$
2. $(\forall j, j' \in J)(\forall \sigma \in \Sigma)[\delta_{SIM}(j, \sigma) = j' \Rightarrow (\exists s \in \Sigma^*)[s\sigma \in \mathcal{L}(S) \wedge \delta_{SIM}(j_0, s) = j]]$
3. $(\forall j \in J_m)(\exists s \in \mathcal{L}_m(S))[\delta_{SIM}(j_0, s) = j]$.

O primeiro item especifica que cada estado em S_{SIM} é alcançável por uma cadeia de S. O segundo item define que, a partir de qualquer transição com σ em S_{SIM} existe uma sequência $s\sigma$ na linguagem do supervisor monolítico. O terceiro item especifica que todo o estado marcado em S_{SIM} é alcançado por uma cadeia na linguagem marcada de S. Se um supervisor reduzido não normal é obtido, ele pode ser facilmente transformado em um supervisor normal removendo-se estados e transições. A partir daqui consideramos que o processo de redução de supervisores produz um supervisor normal, ou seja, que o processo de normalização faz parte da redução de supervisores.

2.3.3 Localização de Supervisores

A localização de supervisores é uma abordagem para o *problema de controle distribuído* onde, para cada sub-planta G_k, um supervisor localizado $S_{LOC,k}$ é construído de forma que $S_{LOC} = ||_{k=1}^{n} S_{LOC,k}$ é *equivalente em controle* ao supervisor monolítico S.

O procedimento de localização é similar ao procedimento de redução de supervisores (CAI; WONHAM, 2010), onde, para cada sub-planta $G_k = (Q_k, \Sigma_k, \delta_k, q_{k0}, Q_{km})$, com $\Sigma_k = \Sigma_{ck} \cup \Sigma_{uk}$, uma cobertura de controle $\mathcal{C}_k$ é construída. A principal diferença entre as cobertura de controle $\mathcal{C}_k$ e $\mathcal{C}$ (Definição 2.3.8) é que $\mathcal{C}_k$ utiliza a relação de consistência de controle $\mathcal{R}_k$ no lugar de $\mathcal{R}$.

Definição 2.3.10. (CAI; WONHAM, 2010) Uma relação $\mathcal{R}_k \subseteq Q_s \times Q_s$ é uma relação de consistência de controle se, $\forall (q, q') \in Q_s$, $(q, q') \in \mathcal{R}_k$ se e somente se:

$$E(q) \cap D_k(q') = E(q') \cap D_k(q) = \emptyset \tag{2.14}$$

$$T(q) = T(q') \Rightarrow M(q) = M(q') \tag{2.15}$$

onde E, M e T são os mesmos da Definição 2.3.7 e:

$$D_k(q) = \{\sigma \in \Sigma_{ck} | \neg \delta_s(q, \sigma)! \wedge (\exists s \in \Sigma^*)[\delta_s(q_{s0}, s) = q \wedge \delta(q_0, s\sigma)!]\} \tag{2.16}$$

D_k indica os eventos controláveis definidos em G_k que são desabilitados no estado q do supervisor. Após a criação da cobertura de controle $\mathcal{C}_k = \{X_j \in Q_s | j \in J_k\}$, o supervisor localizado $S_{LOC,k} = (J_k, \Sigma_{L,k}, \delta_{L,k}, j_{k0}, J_{km})$ para a sub-planta G_k é definido como:

- J_k é o conjunto de índices de $\mathcal{C}_k$.
- $J_{km} = \{j \in J_k \mid X_j \cap Q_{sm} \neq \emptyset\}$.
- $j_{k0} \in \{j \in J_k \mid q_{s0} \in X_j\}$.
- $\delta'_{L,k}(j, \sigma) = j'$ é definido se $(\exists q \in X_j)\delta_s(q, \sigma) \in X_{j'} \wedge (\forall q' \in X_j)[\delta_s(q', \sigma)! \Rightarrow \delta_s(q', \sigma) \in X_{j'}]$.
- $\Sigma_{L,k} = \Sigma_k \cup \Sigma_{com,k}$, onde $\Sigma_{L,k}$ são eventos de Σ exceto aqueles presentes apenas em auto-laços e:

$$\Sigma_{com,k} = \{\sigma \in (\Sigma \setminus \Sigma_k) | (\exists j, j' \in J_k) j \neq j' \wedge \delta'_{L,k}(j, \sigma) = j'\}. \tag{2.17}$$

- $\delta_{L,k}$ é $\delta'_{L,k}$ restrita apenas a $\Sigma_{L,k}$.

2.4 Sincronização em Autômatos

Dado um autômato finito determinístico (AFD), dizemos que G é sincronizável quando existe uma cadeia, chamada palavra de sincronização, que, quando executada, leva o autômato a um determinado estado independentemente do estado no qual o autômato se encontra.

A existência de uma palavra de sincronização em um autômato tem aplicação em diversas áreas, como em automação industrial, relacionado ao carregamento, montagem e empacotamento de produtos (NATARAJAN, 1986; NATARAJAN, 1989). A partir desses trabalhos originais surgiram desenvolvimentos teóricas na área de sincronização de autômatos no contexto de automação industrial (EPPSTEIN, 1988; GOLDBERG, 1993; CHEN; IERARDI, 1995).

Existem também aplicações na área de biocomputação (BENENSON et al., 2003), onde moléculas de DNA se comportam como autômatos finitos funcionando paralelamente e que, diferentemente de computadores eletrônicos, não podem ser reiniciados de maneira direta. Nesse sentido, surge a necessidade de adicionar à solução moléculas de DNA cujos nucleotídeos codificam uma palavra de sincronização.

A maior parte dos sistemas é composto por um conjunto de subsistemas semi-independentes, como no caso de *softwares* de tempo real, que precisam ser integrados, garantindo que certas restrições temporais sejam respeitadas. Nesse contexto de sistemas hierárquicos de tempo real, são utilizados protocolos de sincronização (BEHNAM et al., 2007; NOLTE et al., 2009; HEUVEL; BRIL; LUKKIEN, 2012).

Problemas de sincronização em sistemas de comunicação foram estudados por Jürgensen (2008), onde é feita uma análise de problemas de comunicação relacionados à sincronização, como a previsibilidade do sistema quando um agente observa o sistema durante um intervalo de tempo, ou a possibilidade de um agente, não sabendo o estado atual, atuar no sistema para levá-lo a um estado de interesse.

Larsen, Laursen e Srba (2014), por sua vez, estudam o problema de sincronização sob observação parcial no contexto de sistemas embarcados, como por exemplo, no caso de um problema de inicialização de um satélite, em que o painel solar fica em uma posição desconhecida. Dados os limites de energia do satélite e considerando que do ponto de vista energético transmitir informação é mais custoso do que receber, o problema pode ser tratado como um problema de sincronização, onde deseja-se colocar o painel solar do satélite em uma posição conhecida e as únicas informações recebidas do satélite são se a posição atual gera ou não energia.

Na área de Teoria dos Jogos, especificamente em jogos com condição de opacidade (MAUBERT, 2009), ou seja, jogos onde um jogador que não possui informação completa quer adquirir completo conhecimento com relação ao estado atual, enquanto o oponente busca fazer com que ele não consiga alcançar esse objetivo, o conceito de sincronização de autômatos tem um papel muito importante, visto que a existência de uma palavra de sincronização permite levar ao conhecimento do estado atual.

Definição 2.4.1. Um autômato finito determinístico $G = (Q, \Sigma, \delta, \_, \_)$ é sincro-

nizável se, e somente se, existe uma palavra $w \in \Sigma^*$ tal que para cada $q, q' \in Q$ $\delta(q, w) = \delta(q', w)$.

Exemplo 2.1. Considere o autômato sincronizável $A = (Q, \Sigma, \_, \_, \_)$ da Figura 3. A palavra $w = ab^3ab^3a$ leva o autômato ao estado 1, independentemente do estado de origem. É importante observar que qualquer palavra sw, $s \in \Sigma^*$ também leva o autômato ao estado 1 e consequentemente também é uma palavra de sincronização.

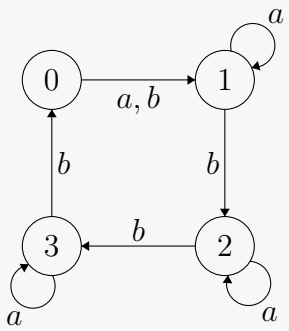


Figura 3 – Exemplo 2.1- Autômato sincronizável (VOLKOV, 2008).

Um exemplo de aplicação da sincronização em processos industriais, apresentado por Ananichev e Volkov (2004), é uma máquina que orienta peças para que sejam empacotadas usando obstáculos. Neste caso, uma determinada sequência de obstáculos define uma palavra de sincronização do autômato mostrado na Figura 4, ou seja, independentemente da orientação na qual o autômato se encontra, após a execução da sequência "$low - HIGH - HIGH - HIGH - low - HIGH - HIGH - HIGH - low$" a peça termina orientada com o detalhe para a esquerda.

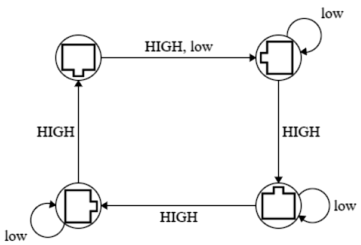


Figura 4 – Autômato que modela a orientação da peça ao passar por obstáculos

É importante notar que, caso o autômato possua uma palavra de sincronização s, qualquer palavra s^n com $n > 0$ também é uma palavra de sincronização. De maneira semelhante, qualquer combinação de duas palavras de sincronização também é uma palavra de sincronização.

Definição 2.4.2. Seja $A = (Q, \Sigma, \_, \_, \_)$ um autômato sincronizável, o conjunto de todas as infinitas palavras de sincronização de A é denotado por $Syn(A)$.

Usualmente, na área de autômatos sincronizáveis, se utiliza a notação $Q_1.w = Q_2$, onde $Q_2 = \{\delta(q, w) | q \in Q_1\}$ para expandir a função de transição para um conjunto de estados.

Definição 2.4.3. (GUSEV; MASLENNIKOVA; PRIBAVKINA, 2014) Uma linguagem L sobre um alfabeto Σ é dita ideal se $L = \Sigma^* L \Sigma^*$. O autômato mínimo A_L, tal que $Syn(A_L) = L$, é um autômato sincronizável.

Quando um autômato A é completo, i.e. $\mathcal{L}(A) = \Sigma^*$, a linguagem $Syn(A)$ é uma linguagem ideal.

Do ponto de vista matemático, o problema de sincronização de autômatos é estudado, principalmente, com relação à Conjectura de Černý (MASTERS, 2012), sendo ainda hoje um dos grandes problemas abertos da matemática.

Conjectura 1. *Seja $A = (Q, \Sigma, \_, \_, \_)$ um autômato sincronizável com conjunto de palavras de sincronização dado por $Syn(A)$, tal que $n = |Q|$. Nesse caso $min\{|w| : w \in Syn(A)\} \leq (n-1)^2$ quando $n > 1$.*

A Černý estabeleceu que, para todo $n > 1$ existe um autômato sincronizável com n estados tal que sua menor palavra de sincronização apresenta tamanho $(n-1)^2$. A conjectura considera esse autômato um pior caso, ou seja, a menor palavra de sincronização de um autômato sincronizável possui no máximo tamanho $(n-1)^2$.

Na área de Sistemas a Eventos Discretos, existe uma série de artigos tratando do problema de sincronização em Redes de Petri (POCCI et al., 2013; POCCI et al., 2014a; POCCI et al., 2014b; POCCI et al., 2016). Tais trabalhos abordam problemas de recuperação de erros, utilizando palavras de sincronização para levar o sistema a uma condição conhecida. Apesar de nossos esforços, não encontramos literatura na área de Teoria de Controle Supervisório que utiliza conceitos da teoria de autômatos sincronizáveis.

3 Controle Supervisório em Autômatos Sincronizáveis

Neste trabalho, buscou-se adaptar a Teoria de Controle Supervisório para lidar com autômatos sincronizáveis, de forma que seja possível usar o conceito de sincronização para recuperar o sistema após a ocorrência de uma falha.

A Seção 3.1 contém as definições básicas apresentadas no capítulo, a Seção 3.2 trata de como o sincronismo se comporta ao aplicarmos algumas operação em autômatos sincronizáveis, a Seção 3.3 trata das condições sob as quais um supervisor, obtido usando a TCS é sincronizável e as Seções 3.4, 3.5 e 3.6, respectivamente, expandem os resultados para a síntese modular local, para a redução de supervisores e para a localização de supervisores.

3.1 Definições Básicas

Quando modelamos plantas e especificações na Teoria de Controle Supervisório, é comum utilizarmos funções de transição (δ) parciais, ou seja, nem todo evento definido no conjunto de eventos do autômato pode ocorrer em qualquer estado. Na literatura, a sincronização, quando observada em autômatos com função de transição parcial é chamada de *Sincronização Cuidadosa* (*Careful Synchronization*) (MARTYUGIN, 2013).

No contexto de Sistemas a Eventos Discretos controlados pela Teoria de Controle Supervisório, é mais prático definir sincronização com relação ao estado inicial.

Definição 3.1.1. Um autômato $G = (Q, \Sigma, \_, q_0, \_)$ é sincronizável com relação ao estado inicial se existe uma palavra $w \in \Sigma^*$, chamada palavra de sincronização, tal que $Q.w = \{q_0\}$.

Na Definição 3.1.1, G é um autômato sincronizável com relação ao estado inicial se, para qualquer estado $q \in Q$, existe uma palavra w tal que $q.w = q_0$. O conjunto de palavras de sincronização com relação ao estado inicial de um autômato G é representado por $Syn_{q_0}(G)$. Por simplificação, nesse trabalho, denotamos $I = Syn_{q_0}(G)$.

Exemplo 3.1. Seja $A = (\_, \Sigma_1, \_, \_, \_)$ um autômato sincronizável com dois estados, mostrado na Figura 5. A palavra $w = c \in \Sigma_1^*$ é uma palavra de sincronização e o autômato A é sincronizável com relação ao estado inicial. O conjunto de todas as palavras de sincronização desse autômato é dado por $I = cc^*$.

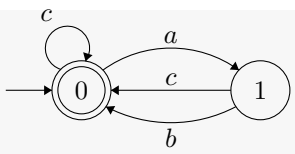


Figura 5 – Exemplo 3.1: Autômato Sincronizável A, $I = cc^*$.

A Proposição 1 apresenta algumas propriedades dos autômatos sincronizáveis com relação ao estado inicial, adaptado de (VOLKOV, 2008).

Proposição 1. *Seja $G = (Q, \Sigma, \delta, q_0, Q_m)$ um autômato sincronizável com relação ao estado inicial e $I \neq \emptyset$. Então:*

a) $\mathcal{L}(G)I\mathcal{L}(G) \subseteq \mathcal{L}(G)$,

b) $\mathcal{L}(G)I\mathcal{L}_m(G) \subseteq \mathcal{L}_m(G)$,

c) $\mathcal{L}_m(G)I\mathcal{L}_m(G) \subseteq \mathcal{L}_m(G)$.

Prova. *Cada palavra $s \in \mathcal{L}(G)$ leva a um estado q ($q_0.s = q$) que, quando seguido por uma palavra $w \in I$, alcança o estado q_0 ($q.w = q_0$), da Definição 3.1.1. Dessa forma, $\forall s \in \mathcal{L}(G)$, $\forall w \in I$ $q_0.sw = q_0$.*

a) Sejam $s \in \mathcal{L}(G)$, e $w \in I$, temos:

$$sw \in \mathcal{L}(G)I \subseteq \mathcal{L}(G) \tag{3.1}$$

e $q_0.sw = q_0$. Sabemos também que, para qualquer autômato G, a linguagem gerada a partir do estado inicial q_0 é $\mathcal{L}(G)$ e, por isso, podemos concatenar $\mathcal{L}(G)$ e obter:

$$sw\mathcal{L}(G) \subseteq \mathcal{L}(G)I\mathcal{L}(G) \subseteq \mathcal{L}(G). \tag{3.2}$$

Equação (3.2) se torna:

$$\mathcal{L}(G)I\mathcal{L}(G) \subseteq \mathcal{L}(G). \tag{3.3}$$

provando a).

b) Sejam $u \in \mathcal{L}(G)$, e $w \in I$, temos:

$$uw \in \mathcal{L}(G)I \tag{3.4}$$

e $q_0.uw = q_0$. Sabemos também que, para qualquer autômato G, $\mathcal{L}(G) \cap \mathcal{L}_m(G) = \mathcal{L}_m(G)$. Então podemos concatenar $\mathcal{L}_m(G)$ em ambos os lados de (3.4):

$$uw\mathcal{L}_m(G) \subseteq \mathcal{L}(G)I\mathcal{L}_m(G) \tag{3.5}$$

Como $\forall s \in \mathcal{L}(G)I$, $q_0.s = q_0$, *temos:*

$$uw\mathcal{L}_m(G) \subseteq \mathcal{L}(G)I\mathcal{L}_m(G) \subseteq \mathcal{L}_m(G). \tag{3.6}$$

provando b).

c) Considerando que

$$\mathcal{L}_m(G) \subseteq \mathcal{L}(G), \tag{3.7}$$

temos

$$\mathcal{L}_m(G)I\mathcal{L}_m(G) \subseteq \mathcal{L}_m(G). \tag{3.8}$$

■

Quando uma linguagem L apresenta a característica $LIL \subseteq L$ ela é chamada *linguagem sincronizável*.

De forma semelhante, é possível mostrar que um autômato mínimo que implementa uma linguagem sincronizável é um autômato sincronizável com relação ao estado inicial, como mostrado na Proposição 2.

Proposição 2. *Seja L uma linguagem regular com $LIL \subseteq L$, para um I não vazio. Um autômato mínimo $G = (Q, \Sigma, \delta, q_0, Q_m)$, tal que $\mathcal{L}_m(G) = L$,é sincronizável com relação ao estado inicial.*

Prova. *Para cada estado $q \in Q_m$ existe ao menos uma cadeia $s \in L$ tal que $\delta(q_0, s) = q$. Como $LIL \subseteq L$, podemos escolher uma cadeia $w \in L$ de forma que $\delta(q_0, sw) = \delta(q, w) = q'$ e a linguagem a partir de q' é L, a mesma linguagem que pode ser executada a partir do estado inicial. Dessa forma, se G é mínimo, $q' = q_0$ ou, caso contrário, os estados q' e q_o poderiam ser unidos visto que a linguagem a partir deles é igual.* ■

É importante notar que a Definição 3.1.1 não apresenta perda de generalidade, visto que todo autômato sincronizável com relação ao estado inicial que também seja acessível é sincronizável com relação a todos os demais estados, visto que é sempre possível alcançar o estado inicial a partir de qualquer outro estado e, por ser acessível, alcançar qualquer estado a partir do estado inicial.

Corolário 1. *Se $G = (Q, \Sigma, \delta, q_0, Q_m)$ é um autômato sincronizável com relação ao estado inicial e cada estado de G é acessível, então:*

a) G é um autômato sincronizável com relação a qualquer estado $q' \in Q$;

b) G é coacessível.

Prova. *Se G é sincronizável com relação ao estado inicial então existe um conjunto $I \neq \emptyset$ tal que, usando a Proposição 1,*

$$\mathcal{L}(G)I\mathcal{L}(G) \subseteq \mathcal{L}(G).$$

Se G é acessível, para cada estado $q \in Q$ existe ao menos uma palavra $u \in \mathcal{L}(G)$ tal que:

$$q_0.u = q \tag{3.9}$$

e como $I \neq \emptyset$, a partir de qualquer estado $q' \in Q$, $q'.w = q_0$, com $w \in I$. De (3.9), sabemos que $q'.wu = q$. Então, $Q.wu = \{q\}$, $wu \in I$ e G é sincronizável com relação ao estado q, demonstrando a proposição a). Se G é acessível, cada estado $q \in Q_m \subseteq Q$ é alcançável a partir do estado inicial, $q_0.u = q$, com $u \in \mathcal{L}_m(G)$. Se G é sincronizável, então existe $w \in I$ tal que $q.w = q_0$ e a partir de q_0 todos os estados são alcançáveis. Então, podemos concluir que G é coacessível, demonstrando o item b). ■

3.2 Operações com Autômatos Sincronizáveis

Considerando que as plantas e especificações de um sistema sobre o qual deseja-se aplicar a Teoria de Controle Supervisório são modelados por autômatos sincronizáveis com relação ao estado inicial, faz-se necessário analisar as operações utilizadas na síntese de supervisores e observar como as palavras de sincronização sobrevivem às operações aplicadas.

Lema 1. *Sejam $L \subseteq \Sigma_1^*$ uma linguagem sincronizável e I o conjunto de todas as palavras de sincronização com relação ao estado inicial do autômato que implementa L. Considerando a projeção natural $P : \Sigma_2^* \to \Sigma_1^*$, $\Sigma_1 \subseteq \Sigma_2$, a linguagem $P^{-1}(L)$ é também um linguagem sincronizável com relação ao estado inicial.*

Prova. *Para mostrar que $P^{-1}(L)$ é uma linguagem sincronizável, é necessário demonstrar que $\exists I_K$, tal que $P^{-1}(L)I_K P^{-1}(L) \subseteq P^{-1}(L)$. Como L é uma linguagem sincronizável, então $LIL \subseteq L$ (Proposição 1). Aplicando a projeção inversa em ambos os lados:*

$$P^{-1}(LIL) \subseteq P^{-1}(L)$$

Podemos decompor o lado esquerdo da equação, resultando em:

$$P^{-1}(L)P^{-1}(I)P^{-1}(L) \subseteq P^{-1}(L)$$

substituindo $P^{-1}(I) = I_K$ e $P^{-1}(L) = B$ temos:

$$BI_KB \subseteq B.$$

Então, $B = P^{-1}(L)$ é uma linguagem sincronizável com relação ao estado inicial.

■

No contexto de autômatos, o Lema 1 pode ser explicado pelo fato de a projeção inversa criar auto-laços em todos os estados para cada evento em $\Sigma_2 \setminus \Sigma_1$. É fácil observar que essa operação não torna o autômato não sincronizável, apenas aumenta o número de palavras de sincronização.

Exemplo 3.2. Seja $A = (\_, \Sigma_1, \_, \_, \_)$ um autômato sincronizável com relação ao estado inicial, apresentado na Figura 5 e sejam $\Sigma_1 = \{a, b, c\}$ e $\Sigma = \{a, b, c, x\}$. Considere a projeção natural $P : \Sigma^* \to \Sigma_1^*$. A Figura 6 mostra o autômato que implementa a linguagem $P^{-1}(\mathcal{L}_m(A))$. É fácil observar que qualquer palavra em $I_P = (\Sigma \setminus \Sigma_1)^* c (\Sigma \setminus \Sigma_1)^* c^* (\Sigma \setminus \Sigma_1)^* = x^* c x^* c^* x^* \in \Sigma^*$ é uma palavra de sincronização ao estado inicial do autômato resultante.

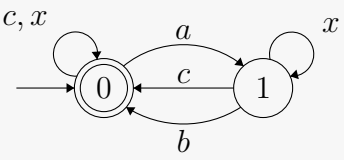


Figura 6 – Exemplo 3.2: Autômato sincronizável com relação ao estado inicial modelando a projeção inversa de A em Σ.

Agora analisamos como a interseção entre linguagens interfere na sincronização

Lema 2. *Sejam $L_1, L_2 \subseteq \Sigma^*$ linguagens sincronizáveis e sejam I_1, I_2, respectivamente, seus conjuntos de palavras de sincronização. Se $I_1 \cap I_2 \neq \emptyset$, então a linguagem não vazia $K = L_1 \cap L_2$ é uma linguagem sincronizável e o autômato definido por ela é sincronizável com relação ao estado inicial.*

Prova. *A partir da Proposição 1, temos:*

$$L_1 I_1 L_1 \subseteq L_1 \tag{3.10}$$

$$L_2 I_2 L_2 \subseteq L_2. \tag{3.11}$$

para todos $s \in L_1 \cap L_2$ e $w \in I_1 \cap I_2$ é simples observar que $sws \in L_1 I_1 L_1$ e $sws \in L_2 I_2 L_2$, então:

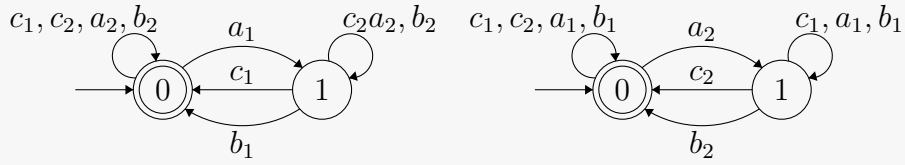
$$sws \in L_1 I_1 L_1 \cap L_2 I_2 L_2. \tag{3.12}$$

(3.12) pode ser reescrita como:

$$sws \in (L_1 \cap L_2)(I_1 \cap I_2)(L_1 \cap L_2) \subseteq L_1 \cap L_2. \tag{3.13}$$

então, $(L_1 \cap L_2)(I_1 \cap I_2)(L_1 \cap L_2) \subseteq L_1 \cap L_2$ e $L_1 \cap L_2$ é uma linguagem sincronizável. ■

Exemplo 3.3. Considere dois autômatos $A_1 = (\_, \Sigma_1, \_, \_, \_)$ e $A_2 = (\_, \Sigma_2, \_, \_, \_)$, tais que $\Sigma_1 \cap \Sigma_2 \neq \emptyset$, apresentados na Figura 7. O autômato que modela a linguagem $\mathcal{L}_m(A_1) \cap \mathcal{L}_m(A_2)$ é sincronizável com relação ao estado inicial, como mostrado na Figura 8.



(a) A_1 (b) A_2

Figura 7 – Exemplo 3.3: Autômatos sincronizáveis A_1 e A_2.

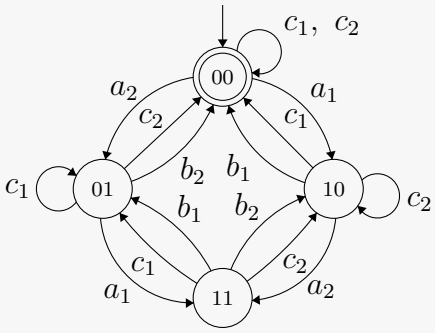


Figura 8 – Exemplo 3.3: Autômato sincronizável com relação ao estado inicial que modela o comportamento $A_1 \cap A_2$. As palavras $w_1 = c_1c_2$ e $w_2 = c_2c_1$ são as menores palavras de sincronização desse autômato.

Usando o Lema 1 e o Lema 2, é possível definir condições sob as quais a composição paralela mantém a sincronicidade dos autômatos sincronizáveis originais. Esse resultado é apresentado na Proposição 3.

Proposição 3. *Sejam $G_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, Q_{m1})$ e $G_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, Q_{m2})$ autômatos sincronizáveis com relação ao estado inicial e seja $\Sigma = \Sigma_1 \cup \Sigma_2$. O autômato mínimo resultante $G = G_1||G_2$ é sincronizável com relação ao estado inicial se $P^{-1}_{\Sigma\to\Sigma_1}(I_1) \cap P^{-1}_{\Sigma\to\Sigma_2}(I_2) \neq \emptyset$, com $P_{\Sigma\to\Sigma_i} : \Sigma^* \to \Sigma_i^*$, $i = 1, 2$.*

Prova. *Como G_1 e G_2 são sincronizáveis com relação ao estado inicial, sabemos, usando Lema 1, que as linguagens $P^{-1}_{\Sigma\to\Sigma_1}(\mathcal{L}(G_1))$, $P^{-1}_{\Sigma\to\Sigma_1}(\mathcal{L}_m(G_1))$, $P^{-1}_{\Sigma\to\Sigma_2}(\mathcal{L}(G_2))$ e $P^{-1}_{\Sigma\to\Sigma_2}(\mathcal{L}_m(G_2))$ são também sincronizáveis com relação ao estado inicial.*

Considerando que $P^{-1}_{\Sigma\to\Sigma_1}(I_1) \cap P^{-1}_{\Sigma\to\Sigma_2}(I_2) \neq \emptyset$ e também que:

$$\mathcal{L}(G_1||G_2) = P^{-1}_{\Sigma\to\Sigma_1}(\mathcal{L}(G_1) \cap P^{-1}_{\Sigma\to\Sigma_2}(\mathcal{L}(G_2))$$

temos, usando Lema 2, que a linguagem $\mathcal{L}(G_1||G_2)$ é uma linguagem sincronizável e que $G_1||G_2$ é uma autômato sincronizável com relação ao estado inicial. ■

Exemplo 3.4. Propulsores de satélites do tipo RIT (*Radiofrequency Ion Thrusters*) operam utilizando um campo de radiofrequência para acelerar partículas de xenônio. Podemos simplificar o processo de propulsão em duas etapas sequenciais: (1) Ionização do xenônio; (2) aceleração do xenônio. Podemos considerar que, para inicializar e finalizar cada processo existem dois eventos, um interno (iniciado por I) que está implementado na lógica de controle do satélite e um externo (iniciado por E) que é recebido pelo satélite de uma central de comando na Terra.

(Ia_1, Ea_1) Eventos controláveis que inicializam o processo de ionização,

(Ib_1, Eb_1) Eventos não-controláveis que finalizam o processo de ionização,

(Ia_2, Ea_2) Eventos controláveis que inicializam o processo de aceleração,

(Ib_2, Eb_2) Eventos não-controláveis que finalizam o processo de aceleração.

É importante notar que os eventos iniciados por E devem estar habilitados em todos os estados, visto que constem em mensagens recebidas da central de comando. Dessa forma, como visto na Figura 9, o autômato resultante de cada etapa é um autômato sincronizável com relação ao estado inicial, com menor palavra de sincronização $w = Eb_i$.

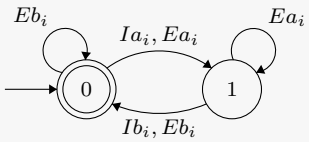


Figura 9 – Exemplo 3.4: Autômato sincronizável que modela as etapas T_i, $i \in \{1, 2\}$ de um propulsor RIT.

A composição de ambos os autômatos $T = T_1||T_2$ também é um autômato sincronizável com relação ao estado inicial, como visto na Figura 10, com menores palavras de sincronização ao estado inicial dadas por $w_1 = Eb_1Eb_2$ e $w_2 = Eb_2Eb_1$.

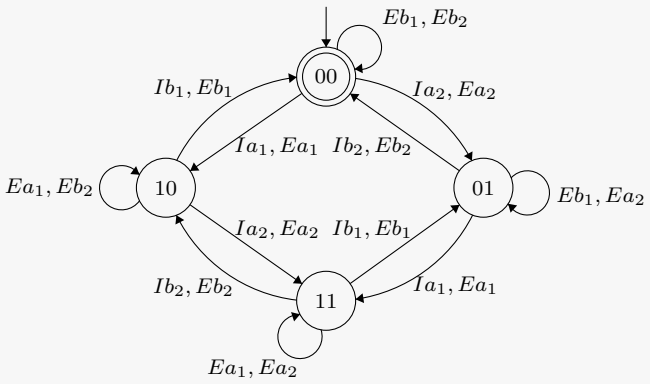


Figura 10 – Exemplo 3.4: Autômato sincronizável que modela a dinâmica da planta $T = T_1||T_2$ de um propulsor RIT.

3.3 Teoria de Controle Supervisório em Autômatos Sincronizáveis

Após definir como os autômatos sincronizáveis com relação ao estado inicial se comportam sob operações, falta estabelecer sob quais condições adicionais temos supervisores sincronizáveis quando plantas e especificações são sincronizáveis com relação ao estado inicial.

Teorema 1. *Sejam G a planta e E a especificação, ambos modelados como autômatos sincronizáveis com relação ao estado inicial. Um supervisor (não-vazio) controlável e não-bloqueante S, obtido a partir de G e E é, também, um autômato sincronizável com relação ao estado inicial se $\Sigma_{nc}^* \cap I \neq \emptyset$, com $I = Syn_{q_0}(G \parallel E)$.*

Prova. *Sejam $K = G \parallel E = (Q, \Sigma, \delta, q_0, Q_m)$ e $S = (Q_s, \Sigma, \delta_s, q_0, Q_{ms})$, onde $Q_s \subseteq Q$ e $Q_{ms} \subseteq Q_m$. Com relação à controlabilidade, cada estado $q_f \in Q \setminus Q_s$ é um mau estado, porque falha o princípio da controlabilidade.*

Como $\Sigma_{nc}^* \cap I \neq \emptyset$, existe ao menos um palavra $w = \sigma_1\sigma_2 \ldots \sigma_n \in \Sigma_{nc}^* \cap I$. Há duas possibilidades a serem consideradas.

a) a palavra w executada a partir de um estado qualquer $q \in Q$ não visita um mau estado; Se esse é o caso, como todos os estados de K que são visitados são bons estados, eles serão mantidos em S. Então, $w \in I$ será uma palavra de sincronização de S.

b) a palavra w executada a partir de um estado qualquer $q \in Q$ visita um mau estado; Ao obter S, estados do autômato que implementa K são removidos, quando são maus estados. Se existe uma palavra $w = \sigma_1\sigma_2 \ldots \sigma_n \in \Sigma_{nc}^* \cap I$, onde $I = Syn_{q_0}(K)$, então cada $q' \in Q$ onde $q' \xrightarrow{\sigma_1 \ldots \sigma_p} q_f \xrightarrow{\sigma_{p+1} \ldots \sigma_n} q_0$ também é um mau estado e, dessa forma, não está em Q_s, assim cada estado q que leva a um mau estado, por meio de eventos não controláveis, também é removido, logo w é completamente removida, indicando que $S = \emptyset$ e o supervisor é vazio.

Após a remoção dos maus estados, a parte acessível do autômato resultante é sempre coacessível (Corolário 1) e, dessa forma, o supervisor é não-bloqueante. ■

Exemplo 3.5. Retornando ao problema do propulsor RIT (Exemplo 3.4), podemos definir uma especificação para o sistema que consiste em garantir que, em modo automático, só seja possível inicializar a aceleração do xenônio após finalizar a ionização,

ou seja, Ia_2 somente pode ocorrer após Ib_1, contudo, é importante impedir que Ia_2 ocorra quando Eb_1 ou Eb_2 ocorrerem, visto que, nesse caso, os operadores estão controlando o sistema remotamente. O autômato sincronizável com relação ao estado inicial que modela a especificação está apresentado na Figura 11.

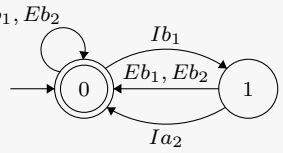


Figura 11 – Exemplo 3.5: Autômato sincronizável que modela a especificação de segurança E de um propulsor RIT.

A partir da planta T a da especificação E, é possível obter um supervisor controlável e não bloqueante para o sistema, apresentado na Figura 12. Como pode ser visto, o sistema atende à especificação de possuir ao menos um palavra de sincronização completamente composta por eventos não controláveis de forma que o supervisor resultante também é sincronizável com relação ao estado inicial, apresentando como menores palavras de sincronização ao estado inicial $w_1 = Eb_1Eb_2$ e $w_2 = Eb_2Eb_1$, assim como a planta T.

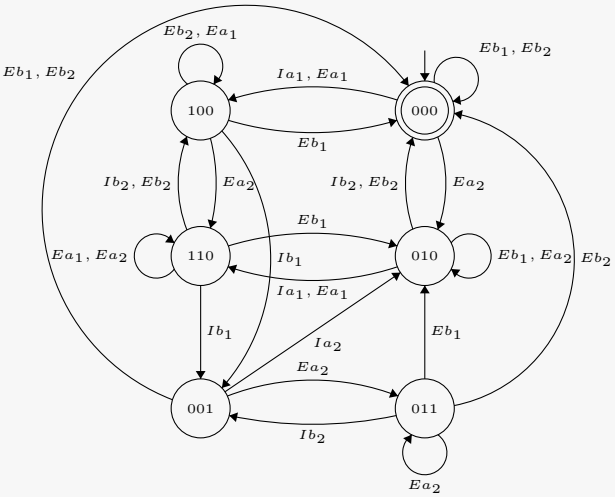


Figura 12 – Exemplo 3.5: Autômato sincronizável do supervisor S obtido para o sistema $K = T||E$ do propulsor RIT.

Supondo que, devido a uma falha de comunicação, a central perca informação quanto ao estado do propulsor do satélite. É possível levar o propulsor a um estado conhecido aplicando a palavra de sincronização w_1. Caso deseje-se forçar a ligação de um propulsor, utiliza-se a cadeia de eventos $s = Ea_1Eb_1Ea_2Eb_2$. Contudo não é possível executar s de qualquer estado, como por exemplo no estado 011, de forma que é necessário executar a palavra w_1s para que a ação seja executada corretamente independentemente do estado no qual o sistema se encontra.

Os resultados desenvolvidos até nessa seção foram apresentados no Congresso

Brasileiro de Automática de 2018 (ALVES; PENA, 2018).

3.4 Controle Supervisório Modular Local

A extensão dos resultados obtidos na Seção 3.3 para o Controle Modular Local (QUEIROZ; CURY, 2002) é feita a seguir.

Corolário 2. *Sejam E_j as especificações locais do sistema e $G_j = (\_, \Sigma_j, \_, \_, \_)$ as plantas locais, com $\Sigma_{uj} \subseteq \Sigma_j$ como o conjunto de eventos não controláveis de G_j e $j \in \{1 \dots m\}$. Se G_j e E_j são autômatos sincronizáveis com relação ao estado inicial, $\Sigma^*_{uj} \cap I_j \neq \emptyset$, $I_j = Syn_{q_0}(G_j \parallel E_j)$ então os supervisores locais S_j também são autômatos sincronizáveis com relação ao estado inicial.*

Prova. *Esse resultado segue da aplicação direta do Teorema 1 nas especificações e plantas locais.* ■

Caso os supervisores locais sejam sincronizáveis com relação ao estado inicial, sabe-se pelo Corolário 1 que são não-bloqueantes. No Corolário 3 demonstramos que o comportamento conjunto dos supervisores é sempre não-conflitante.

Corolário 3. *Sejam S_j com $i \in \{1 \dots m\}$ os supervisores locais, definidos como autômatos sincronizáveis com relação ao estado inicial, então esses supervisores são não-conflitantes.*

Prova. *Se S_j é sincronizável com relação ao estado inicial, pelo Corolário 1, sabemos que S_j é coacessível e, dessa forma:*

$$\overline{\mathcal{L}_m(S_j)} = \mathcal{L}(S_j). \tag{3.14}$$

Pelo Teorema 3, sabemos que $S = ||_{j=1}^{m} S_j$ é um autômato sincronizável com relação ao estado inicial e, portanto, coacessível, tal que:

$$\overline{\mathcal{L}_m(S)} = \mathcal{L}(S)$$

substituindo S por $||_{j=1}^{m} S_j$ em ambos os lados da equação:

$$\overline{\mathcal{L}_m(||_{j=1}^{m} S_j)} = ||_{j=1}^{m} \mathcal{L}(S_j).$$

Usando a equação (3.14) temos:

$$\overline{\mathcal{L}_m(||_{j=1}^{m} S_j)} = ||_{j=1}^{m} \overline{\mathcal{L}_m(S_j)}.$$

Dessa forma, os supervisores S_j são não-conflitantes. ■

3.5 Redução de Supervisores

Nesta seção estabelecemos as condições sob as quais um supervisor reduzido S_{SIM}, obtido a partir de um supervisor monolítico sincronizável com relação ao estado inicial também é sincronizável.

Os resultados nas Seções 3.5 e 3.6 foram compilados em um artigo e submetidos para o *American Control Conference* 2021 (ACC 2021) (ALVES; PENA, 2021).

Teorema 2. *Seja $S = (\_, \Sigma, \delta_s, q_{s0}, \_)$ um supervisor monolítico sincronizável com relação ao estado inicial e $S_{SIM} = (J, \Sigma, \delta_{SIM}, j_0, \_)$ um supervisor reduzido, equivalente em controle a S definido por uma cobertura de controle $\mathcal{C} = \{X_j \in Q_s | j \in J\}$ como na Definição 2.3.8. Se S_{SIM} é normal e q_{s0} é definido apenas no conjunto X_{j_0} então S_{SIM} é sincronizável com relação ao estado inicial.*

Prova. *Como S_{SIM} é normal, cada estado $j \in J$ pode ser alcançado por um caminho $s \in \mathcal{L}(S)$, Definição 2.3.9-1). Dado que S é sincronizável com relação ao estado inicial, existe um conjunto não vazio I tal que $\mathcal{L}(S)\, I\, \mathcal{L}(S) \subseteq \mathcal{L}(S)$, Proposição 1. Pela construção de S_{SIM}, $\mathcal{L}(S)\, I\, \mathcal{L}(S) \subseteq \mathcal{L}(S_{SIM})$ e $(\forall s \in \mathcal{L}(S))\delta_s(q_0, s) = q \Rightarrow \delta_{SIM}(j_0, s) = j \wedge q \in X_j$. Dessa forma, para cada palavra $w \in I$ e cada estado $j \in J$ existe uma palavra $s \in \mathcal{L}(S)$ tal que $\delta_{SIM}(j_0, s) = j$, $\delta_{SIM}(j, w) = j'$ e $q_{s0} \in X_{j'}$. Como q_{s0} é definido apenas no conjunto X_{j_0}, $j' = j_0$.*

Dados que $(\forall j \in J)(\forall w \in I)\delta_{SIM}(j, w) = j_0$, então $\mathcal{L}(S_{SIM})\ I\ \mathcal{L}(S_{SIM}) \subseteq \mathcal{L}(S_{SIM})$ e S_{SIM} é sincronizável com relação ao estado inicial. ■

Nos algoritmos de redução de supervisores, cada estado do supervisor reduzido S_{SIM} corresponde a um conjunto de estados do supervisor S, tal que, um sequência de eventos s que leva S a um estado q deve levar S_{SIM} a um estado que contém q. Seguindo esse raciocínio, uma palavra de sincronização w que leva S ao seu estado inicial q_{s0} deve levar S_{SIM} a um estado que contém q_{s0} e, quando existe apenas um estado que contém q_{s0}, esse estado é o estado inicial de S_{SIM}, indicando que S_{SIM} é sincronizável com relação ao estado inicial.

3.6 Localização de Supervisores

Considerando que a ideia de localização de supervisores é uma extensão direta da redução de supervisores, as premissas para que os supervisores localizados sejam sincronizáveis com relação ao estado inicial é similar. Primeiro, definimos o conceito de supervisor localizado normal, de maneira similar à Definição 2.3.9, o que auxilia na obtenção da prova.

Definição 3.6.1. Um supervisor $S_{LOC,k} = (J_k, \Sigma_{L,k}, \delta_{L,k}, j_{k0}, J_{km})$ é *normal* em relação a S se:

1. $(\forall j \in J_k)(\exists s \in P_{\Sigma\to\Sigma_{L,k}}(\mathcal{L}(S)))[\delta_{L,k}(j_{k0}, s) = j]$;
2. $(\forall j, j' \in J_k)(\forall \sigma \in \Sigma_k)[\delta_{L,k}(j,\sigma) = j' \Rightarrow (\exists s \in \Sigma_k^*)[s\sigma \in P_{\Sigma\to\Sigma_{L,k}}(\mathcal{L}(S)) \wedge \delta_{L,k}(j_0, s) = j]]$;
3. $(\forall j \in J_{km})(\exists s \in P_{\Sigma\to\Sigma_{L,k}}(\mathcal{L}_m(S)))[\delta_{L,k}(j_{k0}, s) = j]$.

O primeiro item especifica que cada estado de $S_{LOC,k}$ é alcançado por uma cadeia em S projetada para Σ_k. O segundo item especifica que para toda transição rotulada por σ em $S_{LOC,k}$ existe uma sequência $s\sigma$ na linguagem do supervisor, projetada para Σ_k. O terceiro item especifica que todo estado marcado em $S_{LOC,k}$ é alcançado por uma sequência na linguagem marcada de S projetada para Σ_k.

O principal resultado dessa seção está mostrado no Teorema 3, onde apresentamos as condições nas quais um supervisor localizado é sincronizável com relação ao estado inicial, dado que o supervisor monolítico é sincronizável.

Teorema 3. *Seja $S = (Q_s, \Sigma, \delta_s, q_{s0}, Q_{sm})$ um supervisor monolítico sincronizável com relação ao estado inicial e $S_{LOC,k} = (I_k, \Sigma_{L,k}, \delta_{L,k}, i_{k0}, I_{km})$ um supervisor localizado para a sub-planta G_k, definido pela cobertura de controle $\mathcal{C}_k = \{X_j \in Q_s | j \in J\}$, como na Definição 2.3.8, e $||_{k=1}^n S_{LOC,k}$ é equivalente em controle a S. Se $S_{LOC,k}$ é normal, seguindo a Definição 3.6.1, e q_{s0} é definido apenas no conjunto $X_{j_{k0}}$ então $S_{LOC,k}$ é sincronizável com relação ao estado inicial.*

Prova. *Essa prova segue os mesmos passos da prova do Teorema 2. Como $S_{LOC,k}$ é normal, cada estado $j \in J$ pode ser alcançado por um caminho $s \in P_{\Sigma\to\Sigma_{L,k}}(\mathcal{L}(S))$, Definição 3.6.1-1). Como S é sincronizável com relação ao estado inicial, existe um conjunto não vazio I tal que $\mathcal{L}(S)\, I\, \mathcal{L}(S) \subseteq \mathcal{L}(S)$, Proposição 1. Pela construção de $S_{LOC,k}$, $P_{\Sigma\to\Sigma_{L,k}}(\mathcal{L}(S))\, P_{\Sigma\to\Sigma_{L,k}}(I)\, P_{\Sigma\to\Sigma_{L,k}}(\mathcal{L}(S)) \subseteq \mathcal{L}(S_{LOC,k})$ e $(\forall s \in \mathcal{L}(S))\delta_s(q_0, s) = q \to \delta_{L,k}(j_{k0}, P_{\Sigma\to\Sigma_{L,k}}(s)) = j \wedge q \in X_j$.*

Dessa forma, pra qualquer palavra $w \in P_{\Sigma\to\Sigma_{L,k}}(I)$ e todo estado $j \in J$ existe uma palavra $s \in P_{\Sigma\to\Sigma_{L,k}}(\mathcal{L}(S))$ tal que $\delta_{L,k}(j_{k0}, s) = j$, $\delta_{L,k}(j, w) = j'$ e $q_{s0} \in X_{j'}$. Como q_{s0} é definido apenas no conjunto X_{j_0}, $j' = j_{k0}$.

Dado que $(\forall j \in J)(\forall w \in P_{\Sigma\to\Sigma_{L,k}}(I))\delta_{L,k}(j, w) = j_{k0}$, então $\mathcal{L}(S_{L,k})\, P_{\Sigma\to\Sigma_{L,k}}(I)\, \mathcal{L}(S_{L,k}) \subseteq \mathcal{L}(S_{L,k})$ e $S_{L,k}$ é sincronizável com relação ao estado inicial. ■

Cada estado de um supervisor localizado $S_{LOC,k}$ corresponde a um conjunto de estados do supervisor monolítico S, tal que, uma sequência de eventos s que leva S a um

estado q deve levar $S_{LOC,k}$, quando considerados apenas os eventos definidos em $\Sigma_{L,k}$, a um estado que contém q. Uma palavra de sincronização w que leva S a seu estado inicial q_{s0} deve levar $S_{LOC,k}$, quando projetada em $\Sigma_{L,k}$, a um estado que contém q_{s0}. Se existe apenas um único estado em $S_{LOC,k}$ que contém q_{s0}, esse estado é o estado inicial de $S_{LOC,k}$, indicando que o supervisor localizado também é sincronizável com relação ao estado inicial.

Em ambos os Teoremas 2 e 3, a principal restrição é que o estado inicial do supervisor monolítico deve ser definido apenas em um único elemento da cobertura de controle e que o supervisor resultante seja normal. Considerando que os principais algoritmos de redução e localização de supervisores constroem congruências de controle e cada estado do supervisor está representado apenas uma vez nesse tipo de cobertura, as restrições já são usualmente atendidas.

4 Sincronização Utilizando Eventos de Recuperação

Para integrar o conceito de sincronização com a Teoria de Controle Supervisório, mesmo em problemas que, originalmente, não são modelados por autômatos sincronizáveis, propomos a criação de eventos de recuperação, similares aos definidos em (SHU, 2014), contudo distintos para cada planta e especificação do sistema, posicionados de modo a transformar esses autômatos em autômatos sincronizáveis com relação ao estado inicial. Em (SHU, 2014) os eventos de recuperação são usados num contexto diferente, sendo definidos para conectar um modelo do sistema em falha com um modelo do sistema em funcionamento correto.

4.1 Modelagem

Considere um sistema composto por plantas $M_i' = (Q_i, \Sigma_i', \delta_i', q_{0i}, \_)$, $i \in \{1 \dots m\}$, e especificações $B_j' = (Q_j, \Sigma_j', \delta_j', q_{0j}, \_)$, $j \in \{1 \dots n\}$. O procedimento para transformar esses autômatos em autômatos sincronizáveis com relação ao estado inicial:

a) Para cada planta M_i' a redefinimos como $M_i = (Q_i, \Sigma_i, \delta_i, q_{0i}, \_)$ onde $\Sigma_i = \Sigma_i' \cup \Sigma_{ri}$, $\Sigma_{ri} = \{r_i\}$, e δ_i como:

$$\delta_i(q, \sigma) = \begin{cases} q_{0i} & \text{if } \sigma = r_i \\ \delta_i'(q, \sigma) & \text{se } \sigma \neq r_i. \end{cases}$$

b) Para cada especificação B_j' redefinimos como $B_j = (Q_j, \Sigma_j, \delta_j, q_{0j}, \_)$ onde $\Sigma_j = \Sigma_j' \cup \{r_{B_j}\}$ e δ_j como:

$$\delta_j(q, \sigma) = \begin{cases} q_{0j} & \text{se } \sigma = r_{B_j} \\ \delta_j'(q, \sigma) & \text{se } \sigma \neq r_{B_j}. \end{cases}$$

Os eventos de recuperação são um elemento de uma nova partição do conjunto de eventos do autômato e possuem uma participação distinta dos eventos controláveis e não controláveis na síntese de supervisores (SHU, 2014). Quando essas plantas e especificações modelam, respectivamente, máquinas e *buffers*, o efeito físico da ocorrência de um evento de recuperação consiste em reiniciar a máquina e esvaziar o *buffer*.

4.2 Síntese de Supervisores com Eventos de Recuperação

Ao criarmos uma nova partição, é necessário redefinir as propriedades de controlabilidade e bloqueio, necessárias para a síntese de supervisores. Ao analisarmos a controlabilidade do sistema, os eventos de recuperação devem ser tratados como eventos não controláveis, visto que não podem ter sua ocorrência inibida pelo supervisor. Por outro lado, na análise de bloqueio os eventos de recuperação devem ser desconsiderados visto que não faz sentido que um evento de recuperação ocorra em condições normais de operação do sistema.

Nesse sentido, se torna necessário reapresentar algumas definições da Teoria de Controle Supervisório adequando-as à introdução da nova partição do conjunto de eventos.

Definição 4.2.1. Seja $G = (\_, \Sigma, \_, \_, \_)$ um autômato finito determinístico, sincronizável com relação ao estado inicial, e seja $\Sigma = \Sigma_c \cup \Sigma_{nc} \cup \Sigma_r$ o conjunto de eventos, com Σ_c como o conjunto de eventos controláveis, Σ_{nc} o conjunto de eventos não controláveis e Σ_r o conjunto de eventos de recuperação. Qualquer palavra de tamanho m formada por um arranjo, sem repetição, de elementos de Σ_r, com $m = |\Sigma_r|$ é uma palavra de sincronização de G.

Exemplo 4.1. Esse exemplo é um versão estendida da Pequena Fábrica (WONHAM; RAMADGE, 1988), composta por três máquinas e dois *buffers* unitários, Figura 13. Originalmente, cada máquina é modelada por um autômato $M_i = (\_, \Sigma_i, \_, \_, \_)$, $i \in \{1, 2, 3\}$ com dois estados (ocioso ou trabalhando) e duas transições (iniciar o trabalho e finalizar o trabalho). Os *buffers* unitários também são modelados por autômatos B_j, $j \in \{1, 2\}$ compostos de dois estados e duas transições (Figure 14).

Primeiramente, os autômatos da Figura 14 são transformados em autômatos sincronizáveis com relação ao estado inicial pela adição aos eventos de recuperação r_i e r_{B_j} ao modelo de forma que cada autômato seja levado ao estado inicial quando seu correspondente evento de recuperação é executado, conforme discutido na Seção 4.1.

Na Figura 15, o modelo de cada componente do sistema e sua respectiva menor palavra de sincronização são mostrados. A menor palavra de sincronização é a sequência $r_i \in \Sigma^*_{ri}$ para cada planta e $r_{B_j} \in \Sigma^*_{B_j}$ para cada especificação.

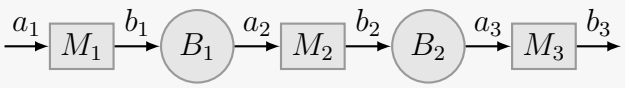
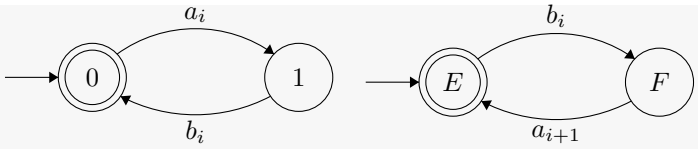
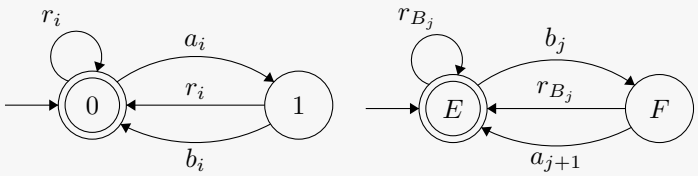


Figura 13 – Exemplo 4.1: Diagrama da Pequena Fábrica Estendida.



(a) M_i, $i \in \{1,2,3\}$ (b) B_j, $j \in \{1,2\}$

Figura 14 – Exemplo 4.1: Autômato original que modela o comportamento da Pequena Fábrica Estendida.



(a) M_i (b) B_j

Figura 15 – Exemplo 4.1: Autômato sincronizável M_i, $i \in \{1,2,3\}$, com $w_i = r_i \in I_1$, e B_j, $j \in \{1,2\}$, $w_{B_j} = r_{B_j}$.

Como demonstrado na Proposição 3, a composição paralela de dois autômatos sincronizáveis com relação ao estado inicial, também é sincronizável com relação ao estado inicial quando há interseção entre seus conjuntos de palavras de sincronização inversamente projetados para um mesmo alfabeto. Quando adicionamos eventos de recuperação como proposto na Seção 4.1, essa interseção é sempre não vazia.

Corolário 4. *Sejam* $G_1 = (\_, \Sigma_1, \_, \_, \_)$ *e* $G_2 = (\_, \Sigma_2, \_, \_, \_)$ *autômatos sincronizáveis com relação ao estado inicial e* $\Sigma = \Sigma_1 \cup \Sigma_2$ *e* $\Sigma_r = \Sigma_{r1} \cup \Sigma_{r2}$*, onde* $\Sigma_r \subset \Sigma$*,* $\Sigma_{r1} \subset \Sigma_1$ *e* $\Sigma_{r2} \subset \Sigma_2$*. Seja também* I_1 *e* I_2 *os conjuntos de palavras de sincronização de* G_1 *e* G_2*, respectivamente. O autômato resultante* $G = G_1||G_2$ *é sincronizável com relação ao estado inicial.*

Prova. *Para mostrar que o autômato* $G = G_1||G_2$ *é sincronizável com relação ao estado inicial é suficiente mostrar que* $P^{-1}_{\Sigma\to\Sigma_1}(I_1) \cap P^{-1}_{\Sigma\to\Sigma_2}(I_2) \neq \emptyset$*, (Proposição 3),* $P_{\Sigma\to\Sigma_i} : \Sigma^* \to \Sigma_i^*$*,* $i \in \{1,2\}$*. Seja* $perm(\Sigma_a, \Sigma_b) = \{s : s \in \Sigma_a^* \wedge \forall\sigma \in \Sigma_b, |P_{\Sigma_a\to\{\sigma\}}(s)| = 1\}$ *uma partição de* Σ_a^* *onde cada evento em* Σ_b *ocorre apenas uma vez. Quando* $\Sigma_a = \Sigma_b$ *a linguagem resultante contém as palavras formadas pelas permutações de eventos em* Σ_a*.*

Por definição,

$$perm(\Sigma_{r1}, \Sigma_{r1}) \subseteq I_1$$

$$perm(\Sigma_{r2}, \Sigma_{r2}) \subseteq I_2$$

e

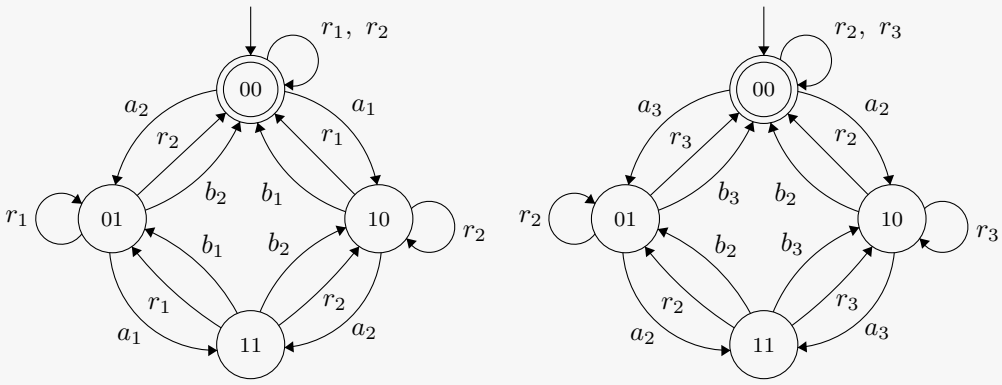
$$perm(\Sigma_r, \Sigma_r) \subseteq perm(\Sigma_r, \Sigma_{r1}) \subseteq P^{-1}_{\Sigma\to\Sigma_1}(perm(\Sigma_{r1}, \Sigma_{r1})) \quad (4.1)$$

$$perm(\Sigma_r, \Sigma_r) \subseteq perm(\Sigma_r, \Sigma_{r2}) \subseteq P^{-1}_{\Sigma\to\Sigma_2}(perm(\Sigma_{r2}, \Sigma_{r2})). \quad (4.2)$$

Se $\Sigma_r \neq \emptyset$, temos que $perm(\Sigma_r, \Sigma_r) \neq \emptyset$. Considerando (4.1) e (4.2), $perm(\Sigma_r, \Sigma_r) \subseteq P^{-1}_{\Sigma\to\Sigma_1}(I_1) \cap P^{-1}_{\Sigma\to\Sigma_2}(I_2)$. Dessa forma, podemos dizer que $G = G_1||G_2$, modelado com eventos de recuperação, é sincronizável com relação ao estado inicial.

■

Exemplo 4.2. Cada planta local $G_j = M_j||M_{j+1}$, $j = \{1, 2\}$, também é um autômato sincronizável com relação ao estado inicial, $G_j = (\_, \Sigma_j \cup \Sigma_{j+1} \cup \{r_j, r_{j+1}\}, \_, \_, \_)$, com menor palavra de sincronização dada por $w_j \in \{r_j r_{j+1},\ r_{j+1} r_j\}$. Para G_1 (Figura 16(a)), $w_1 \in \{r_1 r_2,\ r_2 r_1\}$ e para G_2 (Figura 16(b)), $w_2 \in \{r_2 r_3,\ r_3 r_2\}$.



(a) $G_1 = M_1||M_2$ (b) $G_2 = M_2||M_3$

Figura 16 – Exemplo 4.2: Plantas G_1 e G_2 da Pequena Fábrica Estendida da Figura 13.

É necessário, agora, redefinir a análise de bloqueio e controlabilidade levando em consideração os eventos de recuperação (Σ_r).

A definição de bloqueio para sistemas com eventos de recuperação é dada na Definição 4.2.2

Definição 4.2.2. Seja $G = (\_, \Sigma, \_, \_, \_)$ um autômato finito determinístico, sincronizável com relação ao estado inicial, com $\Sigma_r \subseteq \Sigma$. G é não-bloqueante se:

$$\mathcal{L}(G) \cap (\Sigma \setminus \Sigma_r)^* = \overline{\mathcal{L}_m(G) \cap (\Sigma \setminus \Sigma_r)^*} \quad (4.3)$$

A redefinição de controlabilidade é apresentada na Definição 4.2.3.

Definição 4.2.3. Seja $G = (\_, \Sigma, \_, \_, \_)$ um autômato finito determinístico, sincronizável com relação ao estado inicial, tal que seu conjunto de eventos possa ser

particionado em $\Sigma = \Sigma_c \cup \Sigma_{nc} \cup \Sigma_r$. Uma linguagem $K \subseteq \mathcal{L}_m(G)$ é controlável se:

$$\overline{K}(\Sigma_{nc} \cup \Sigma_r) \cap \mathcal{L}(G) \subseteq \overline{K}.$$

Se K satisfaz a condição, então é controlável, caso contrário, a *máxima sublinguagem controlável e não-bloqueante* pode ser obtida. Teorema 1 é reformulado no Corolário 5.

Corolário 5. *Sejam $G = ||_{i=1}^{m} M_i$ e $E = ||_{j=1}^{n} B_j$, modelado como um autômato sincronizável com relação ao estado inicial, como na Seção 4.1. Seja I o conjunto de palavras de sincronização de $K = G||E$, se $(\Sigma_{nc} \cup \Sigma_r)^* \cap I \neq \emptyset$ então um supervisor, não vazio, controlável e não-bloqueante S, obtido a partir de G e E é, também, um autômato sincronizável com relação ao estado inicial.*

Prova. *Usando o procedimento de modelagem apresentado na Seção 4.1, eventos de recuperação são adicionados nos subsistemas e especificações tal que:*

$$I \cap \Sigma_r^* \neq \emptyset. \tag{4.4}$$

Da Definição 4.2.3 sabemos que os eventos de recuperação não podem ser desativados pelo supervisor (caso aconteça, o teste de controlabilidade falha). Dessa forma, o Teorema 1 é válido substituindo Σ_{nc} por $(\Sigma_{nc} \cup \Sigma_r)$ na definição do teorema e na prova.

Nesse caso, a condição para a validade do Teorema é mudada para $(\Sigma_{nc} \cup \Sigma_r)^* \cap I \neq \emptyset$. Por (4.4) sabemos que $(\Sigma_{nc} \cup \Sigma_r)^* \cap I \neq \emptyset$, tal que a condição é cumprida e o supervisor é sincronizável com relação ao estado inicial. ■

Dessa forma, um supervisor controlável e não-bloqueante é sempre sincronizável com relação ao estado inicial.

Exemplo 4.3. O supervisor monolítico S, do problema da Pequena Fábrica Estendida (Figura 13), sintetizado a partir da planta $G = M_1||M_2||M_3$ e da especificação $E = E_1||E_2$, também é um autômato sincronizável com relação ao estado inicial. Ao menos uma palavra de sincronização do supervisor, $w = r_1 r_2 r_3 r_{B1} r_{B2}$, leva tanto o próprio supervisor quanto a planta ao estado inicial.

O autômato que implementa o supervisor S foi omitido pelo fato de possuir um número grande de estados e transições, contudo o supervisor possui as seguintes características:

- Supervisor de $(M_1||M_2||M_3||E_1||E_3)$
  - Estados: 18
  - Transições: 122

– Transições por eventos de Recuperação: 90

O mesmo procedimento pode ser aplicado ao Controle Modular Local. Como apresentado no Corolário 2, cada supervisor local é controlável e não-bloqueante, contudo, mesmo se o sistema original (sem eventos de recuperação) é não-conflitante, é necessário executar o teste de não-conflito.

É importante notar que a técnica de recuperação de sistemas utilizando eventos de recuperação é muito mais efetiva quando aplicada em conjunto com técnicas de controle descentralizado, visto que nesse caso não é necessário reiniciar o sistema globalmente no caso de um falha.

4.3 Controle Modular Local com Eventos de Recuperação

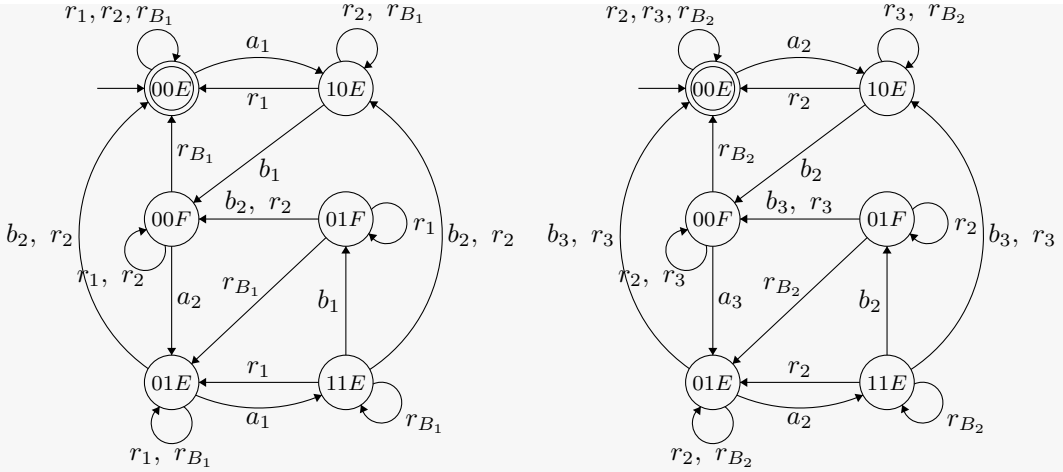
Como os eventos de recuperação são desconsiderados ao analisar bloqueios, o teste de não-conflito (2.4), deve ser adaptado para ignorar os eventos de recuperação (Definição 4.3.1).

Definição 4.3.1. Sejam S_j os supervisores locais do sistema, definidos como autômatos sincronizáveis com relação ao estado inicial e conjunto de eventos $\Sigma_j = \Sigma_{cj} \cup \Sigma_{uj} \cup \Sigma_{rj}$. Esses supervisores são não-conflitantes quando:

$$||_{j=1}^{m} \overline{\mathcal{L}_m(S_j) \cap (\Sigma_j \setminus \Sigma_{rj})^*} = \overline{\mathcal{L}_m(||_{j=1}^{m} S_j) \cap (\Sigma \setminus \Sigma_r)^*}.$$

Em sistemas de manufatura, os eventos de recuperação, usualmente, compartilham transições com eventos não controláveis, ou estão em auto-laços. Nesses casos, os supervisores modulares do sistema original (sem eventos de recuperação) são iguais ao supervisores modulares com eventos de recuperação quando os eventos de recuperação são removidos. Caso isso não aconteça, o supervisores podem ser mais restritivos, visto que na síntese os eventos de recuperação são tratados como não controláveis durante a análise da controlabilidade.

Exemplo 4.4. Cada supervisor local S_j, $j = \{1, 2\}$, do problema da Pequena Fábrica Estendida (Figura 13), também é um autômato sincronizável com relação ao estado inicial, como mostrado na Figura 17. É importante notar que a execução de uma palavra de sincronização do supervisor retorna tanto a planta local quanto a especificação local a um estado conhecido e equivalente ao do supervisor.



(a) S_1 (b) S_2

Figura 17 – Autômatos que modelam os supervisores locais S_1 e S_2 da Pequena Fábrica Estendida.

Suponha que o supervisor S_2 esteja no estado $11E$ e o evento b_3 ocorra, mas por uma falha não seja detectado pelo supervisor. Nesse caso o sistema entra em uma situação de dessincronia, visto que a planta local se encontra no estado 10 e o supervisor local continua no estado $11E$. Detectando e neutralizando a falha, é possível trazer o sistema a uma situação de sincronia executando uma palavra de sincronização do supervisor S_2, como $w = r_2 r_3 r_{B2}$, que reinicia S_2 e sua respectiva planta local, mas não reinicia completamente o supervisor S_1.

Os resultados deste capítulo foram apresentados no artigo publicação na revista *IEEE Transactions on Automation Science and Engineering* (IEEE T-ASE) (ALVES; PENA, 2019).

5 Sincronização em Sistemas Legados

Neste capítulo apresentamos alguns resultados algorítmicos que podem ser aplicados a sistemas legados, ou seja, que foram desenvolvidos sem levar em consideração o conceito de sincronismo ou eventos de recuperação. Nesse caso, os autômatos com função de transição parcial são sincronizados sem utilizar a *sincronização cuidadosa*, ou seja, eventos que não são permitidos em determinados estados podem ser executados na palavra de sincronização. Nesse caso, cabe ao sistema de controle tratar esses eventos, ignorando ou permitindo sua ocorrência quando necessário.

5.1 Sincronização em Sistemas Existentes

A ideia principal dessa seção consiste em adaptar a teoria de autômatos sincronizáveis para a situação onde existe um conjunto autômatos distintos que precisa ser sincronizado, ou seja, esse autômatos precisam estar em estados correspondentes. A fim de tratar esse problema, ao invés de buscar por uma palavra de sincronização que leva o sistema para um único estado, nós buscamos o que chamamos de *palavra de ressincronização* que leva o sistema a um estado qualquer onde ele está sincronizado, ou seja, um estado global no qual os estados dos respectivos componentes combinam.

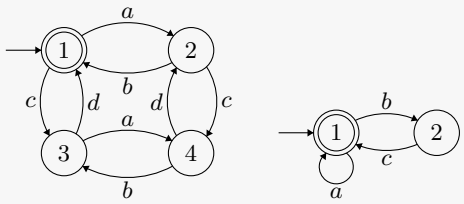
É comum que os sistemas já existentes na indústria tenham função de transição parcial, ou seja, existem eventos não permitidos em determinados estados. Nesse caso, completamos a função de transição dos autômatos com eventos falsos criando auto-laços com os eventos proibidos em cada estado. A adição desses eventos aumenta a probabilidade da existência de uma palavra de sincronização e, na prática, esses eventos são ignorados pelo sistema real.

Definição 5.1.1. Seja $G_i = (Q_i, \Sigma_i, \delta_i, q_{0i}, Q_{mi})$ o autômato que modela o i-ésimo componente do sistema e δ_i seja a função de transição. Podemos definir um novo autômato completo $G_{ci} = (Q_i, \Sigma_{G_c}, \delta_{ci}, q_{0i}, Q_{mi})$, com $\Sigma_i \subseteq \Sigma_{G_c}$, onde:

$$\delta_{ci}(q,\sigma) = \begin{cases} q, & \delta_i(q,\sigma) \text{ não é definido} \\ \delta_i(q,\sigma), & \text{caso contrário} \end{cases} \tag{5.1}$$

Após essa operação temos $\mathcal{L}(G_{ci}) = \Sigma_{G_c}^*$. Usualmente, $\Sigma_{G_c} = \bigcup_{i=1}^{n} \Sigma_i$ é o conjunto de eventos global, ou seja, todos os componentes passam a ter o mesmo alfabeto.

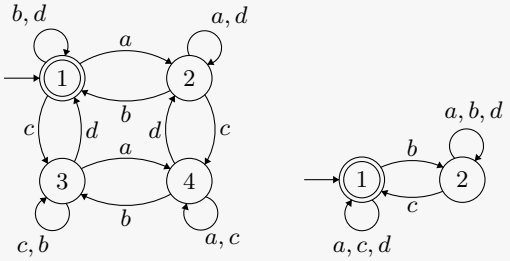
Exemplo 5.1. Considere um sistema composto por dois autômatos, G_1 e G_2 mostrados na Figura 18. Ambos os autômatos não são completos e o conjunto de eventos é $\Sigma = \{a, b, c, d\}$.



(a) G_1 Original (b) G_2 Original

Figura 18 – Componentes do Sistema.

Os autômatos mostrados na Figura 18 não são completos e possuem conjuntos de eventos diferentes. O procedimento de tornar os autômatos completos (Figura 19) consiste em adicionar auto-laços com os eventos faltantes em cada estado. Esses auto-laços indicam que esses eventos são ignorados pelos componentes do sistema e a ocorrência deles não altera o estado do componente.



(a) Autômato G_{c1} (b) Autômato G_{c2}

Figura 19 – Componentes do sistema com função de transição completa.

É importante notar que, para aplicar essa transformação, o sistema deve ser capaz de lidar com a ocorrência desses eventos proibidos, ou seja é necessário que a ocorrência de um evento proibido em determinado componente não leve esse componente a uma situação de falha ou quebra, esse evento deve ser apenas ignorado, não gerando mudança de estado no componente no qual ele é proibido.

Definição 5.1.2. Sejam $G_{ci} = (Q_i, \Sigma, \delta_i, \_, \_)$, $i \in \{1 \dots m\}$, os componentes do sistema, com conjunto de eventos (Σ) e completos ($\mathcal{L}(G_{ci}) = \Sigma^*$). O comportamento completo do sistema pode ser modelado por $T = ||_{i=1}^{m} G_{ci} = (Q_T, \Sigma, \delta_T, \_, \_)$.

Cada estado de T é uma tupla de estados $(q_1, q_2, \dots, q_m)$, tal que $q_i \in Q_i$. Se T é um autômato sincronizável, é possível encontrar uma palavra de sincronização que leva o sistema a um estado onde todos os componentes do sistema estão em sincronia. Nesse caso, não importa qual estado de T o sistema alcança, apenas se o estado de destino é seguro,

dessa forma, ao invés de buscar por uma sequência que leva a apenas um estado, esperamos encontrar uma sequência que leva o sistema a qualquer estado em que os componentes do sistema estejam sincronizados.

Definição 5.1.3. Seja $T = (Q_T, \Sigma, \delta_T, \_, \_)$ um autômato que modela o comportamento completo do sistema, um conjunto de estados $Q_s \subset Q_T$ é chamado *conjunto de estados sincronizados* quando, para todos os estados em Q_s, os estados correspondentes dos componentes do sistema estão em sincronia. Por outro lado, um conjunto de estados $Q_f = Q_T \setminus Q_s$ é chamado conjunto de estados de falha quando os estados dos componentes não estão em sincronia.

Cada estado em Q_T é uma tupla de estados dos n componentes do sistema, $q_T = \delta_T(q_{0T}, s) = (\delta_1(q_{01}, s), \delta_2(q_{02}, s), \ldots, \delta_n(q_{0n}, s)) = (q_1, q_2, \ldots, q_n)$.

Exemplo 5.2. O autômato T para o sistema apresentado no Exemplo 5.1 é mostrado na Figura 20. Os estados em cinza são os estados do conjunto de estados sincronizados (Q_s). Quando em operação normal, o sistema nunca deveria atingir estados fora de Q_s. Contudo, quando uma falha ocorre, os dois autômatos podem perder a sincronização e um estado em Q_f é alcançado. Essa falha normalmente consiste na ocorrência de um evento que está proibido por ao menos um dos componentes do sistema, como um sensor que é acionado em um estado no qual deveria estar desabilitado.

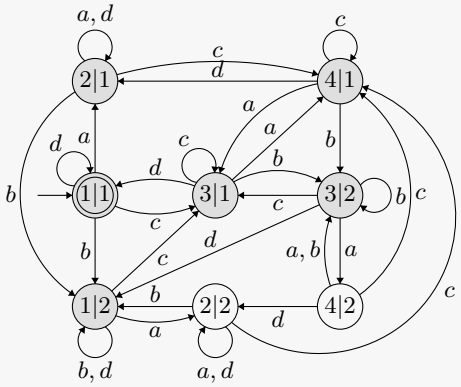


Figura 20 – Autômato T para o sistema apresentado em Exemplo 5.1.

Nesse caso, como os componentes do sistema são os mostrados na Figura 18, definimos Q_s como o conjunto de estados da composição dos dois autômatos originais (sem os auto-laços). Os outros estados em Q_T não são alcançados em condições normais de operação.

Nosso objetivo então é encontrar uma *palavra de ressincronização* w_r no autômato T, tal que $Q_T.w_r \subseteq Q_s$. Para isso, apresentamos um algoritmo heurístico que busca por ao menos uma palavra de ressincronização. Não existe garantia de que o algoritmo encontre

essa palavra com sucesso, visto que a existência dessa palavra depende da topologia do sistema.

5.1.1 Algoritmo

Considerando o autômato T que modela o comportamento completo do sistema, é necessário encontrar uma sequência w_r que sempre leva a um estado em Q_s independentemente do estado do sistema. Para isso, representamos o sistema por meio de um *autômato powerset*:

Definição 5.1.4. Seja $T = (Q_T, \Sigma, \delta_T, q_0, Q_{mT})$ uma autômato completo. Podemos definir um novo autômato completo $P = (Q_P, \Sigma, \delta_P, q_{0P}, Q_{mP})$ onde:

$$
\begin{aligned}
&q_0 = Q_T, \\
&Q_P = 2^{Q_T} \setminus \emptyset, \\
&Q_{mP} = 2^{Q_s} \setminus \emptyset, \\
&\delta_P(q_P, \sigma) = \{\delta_T(q, \sigma) \mid q \in q_P \wedge q_P \in Q_P\}.
\end{aligned}
$$

Cada estado do autômato *powerset* é um subconjunto do conjunto de estados do autômato T. A função de transição de P aplica o evento a cada estado de T que forma um estado em P gerando um novo conjunto de estados e, consequentemente, um novo estado de P. Os estados marcados são os subconjuntos com apenas um elemento, ou seja, que representam apenas um dos estados do autômato original T.

Exemplo 5.3. O *autômato powerset* P para o autômato T, mostrado no Exemplo 5.2, é um autômato com 18 estados, sendo 9 deles marcados. Alcançar um estado marcado leva o sistema à ressincronização. O autômato P está mostrado na Figura 21, os estados marcados estão mostrados em cinza.

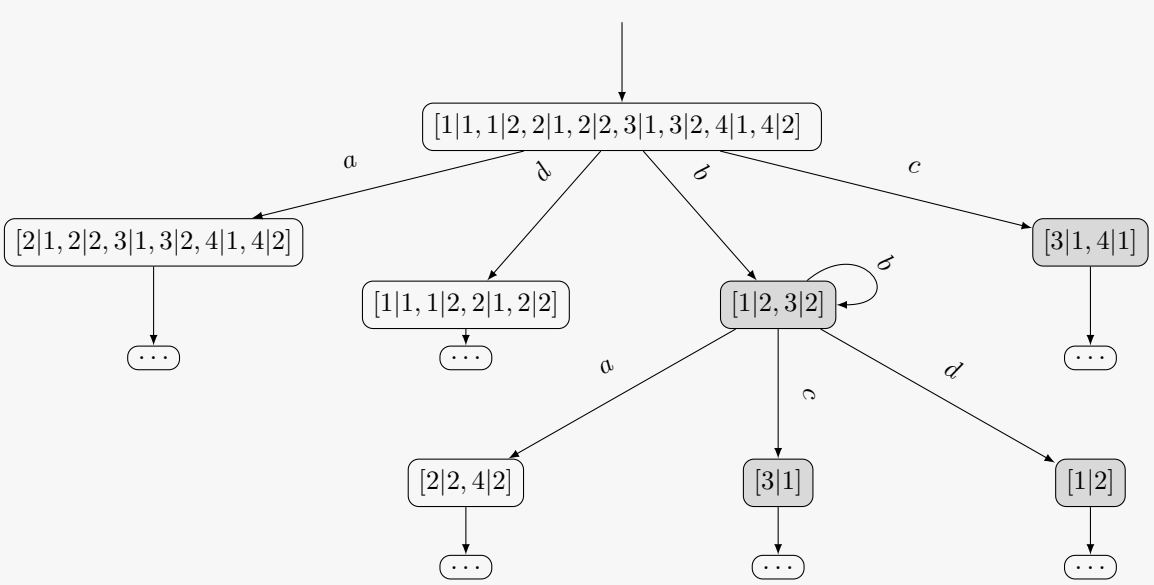


Figura 21 – Parte do autômato P para o sistema mostrado no Exemplo 5.1.

Como o autômato T tem 8 estados, o autômato P teria, no pior caso, $2^8 - 1 = 255$ estados, contudo o algoritmo verifica apenas 18 deles, dado que apenas a parte acessível de P é visitada.

Nesse caso, qualquer $w_r \in \mathcal{L}_m(P)$ leva o sistema a um estado seguro porque leva a um estado marcado de P, desta forma o algoritmo apenas busca por uma palavra que leva a um estado marcado em P.

O algoritmo consiste em uma busca em largura, começando do estado inicial de P e visitando cada estado até que um estado marcado seja alcançado. O conjunto *visited* armazena os estados já visitados pelo algoritmo, evitando assim que um mesmo estado seja avaliado múltiplas vezes, já que usualmente os autômatos são cíclicos. A estrutura *path* armazena a sequência que leva a cada estado visitado.

A primeira fronteira é o estado inicial, composto por todos os estado de T e o *path* para o estado inicial é ϵ (a sequência vazia). Após a inicialização, a cada iteração, o algoritmo constrói uma nova fronteira composta pelos estados adjacentes não visitados.

Para cada estado, o algoritmo avalia seu valor *match*, uma medida percentual de quantos estados de T estão em Q_s no estado atual de P. O valor *match* varia de 0% quando nenhum dos estados pertence a Q_s até 100% quando todos os estado pertencem a Q_s.

O passo heurístico do algoritmo consiste em apenas adicionar um novo estado na fronteira quando seu valor *match* for superior ou igual ao estado da fronteira original ao qual ele é adjacente. A heurística parte do pressuposto que o valor *match* é a grandeza que buscamos maximizar logo não faz sentido escolher estados que diminuem seu valor, contudo é importante frisar que isso não é um garantia, pode ser possível ter que diminuir o valor *match* para que ele aumente posteriormente. Se nenhuma heurística for usada,

Algoritmo 1: Busca pela Palavra de Ressincronização

Data: $P = (Q_P, \Sigma, \delta_P, q_{0P}, Q_{mP})$
Result: w_r

```
1  frontier ← {q_0P}
2  path[q_0P] ← ϵ
3  visited ← ∅
4
5  while |frontier| > 0 do
6
7      visited ← visited ∪ frontier
8      new_frontier ← ∅
9
10     foreach q_o ∈ frontier do
11         match_o ← |q_o ∩ Q_s|/|q_o|
12         foreach σ ∈ Σ do
13             q_d ← δ_P(q_o, σ)
14             match_d ← |q_d ∩ Q_s|/|q_d|
15
16             if q_d ∈ visited OR match_d < match_o then continue
17
18             path[q_d] ← path[q_o] σ
19
20             if q_d ∈ Q_mP then
21                 w_r ← path[q_d]
22                 return
23
24             new_frontier ← new_frontier ∪{q_d}
25
26     frontier ← new_frontier
```

usualmente o algoritmo demora mais para convergir, contudo, se existe uma palavra de ressincronização ela será encontrada.

O algoritmo termina quando encontra um estado com valor *match* 100%, retornando a sequência w_r encontrada.

Exemplo 5.4. Aplicando o algoritmo ao autômato P apresentado na Figura 21, o algoritmo inicia no estado $[1|1, 1|2, 2|1, 2|2, 3|1, 3|2, 4|1, 4|2]$ e para na primeira iteração na qual um dos estados marcados é encontrado $[1|2, 3|2]$ ou $[3|1, 4|1]$. A sequência $w_r = b$ é uma das sequências que pode ser obtida pelo algoritmo e, quando executada no autômato T (Figura 20), independentemente do estado no qual se encontra leva ao estado $1|2$ ou ao estado $3|2$, ambos em Q_s e, dessa forma, os componentes do sistema estão sincronizados. O valor *match* para cada estado apresentado na Figura 21 está mostrado na Tabela 1.

Tabela 1 – Valor *match* calculado para os estados da Figura 21

Estado	Valor Match
$[1\|1, 1\|2, 2\|1, 2\|2, 3\|1, 3\|2, 4\|1, 4\|2]$	75%
$[2\|1, 2\|2, 3\|1, 3\|2, 4\|1, 4\|2]$	63%
$[1\|2, 3\|2]$	100%
$[3\|1, 4\|1]$	100%
$[1\|1, 1\|2, 2\|1, 2\|2]$	75%
$[2\|2, 4\|2]$	0%
$[3\|1]$	100%
$[1\|2]$	100%

5.1.2 Detalhes de Implementação

No pior caso, quando a heurística não tem efeito em reduzir o *branching factor*, a complexidade do algoritmo é $\mathcal{O}(|Q_P| + |\Sigma|)$, contudo é importante notar que, como Q_P é usualmente muito grande, é importante ser cuidadoso na implementação do algoritmo.

O autômato P não deve ser pré-computado, mas sim construído durante a execução do algoritmo de forma que não seja necessário armazenar todos os estados de Q_P na memória ao mesmo tempo. Outra simplificação possível é limitar o tamanho da fronteira, de forma que, após alcançar um determinado número de estados, os demais estados adjacentes são ignorados na iteração.

Os resultados apresentados nessa seção foram apresentados no IFAC World Congress 2020 (ALVES; PENA, 2020b).

5.2 Reconfiguração de Sistemas usando Sincronismo

Nesta seção tratamos do procedimento de reconfiguração de sistemas, levando seus componentes a um estado específico, independentemente do estado no qual o sistema se encontra. Esse processo pode ser dividido em dois passos, primeiro o autômato que representa o sistema em malha fechada é transformado em um autômato completo com a adição de *eventos falsos*, que são observados apenas pelo supervisor. O segundo passo consiste em um algoritmo heurístico que busca por palavras de sincronização nesse autômato resultante.

Partindo dos autômatos que representam o supervisor e a planta, respectivamente S e G, aplica-se a composição paralela tal que $K = S||G$. É importante notar que como $K = K||G$, ele equivalente em controle a S e também é um supervisor para esse sistema.

Exemplo 5.5. O sistema flexível de manufatura consiste em duas máquinas e um *buffer* unitário entre elas, como mostrado na Figura 22.

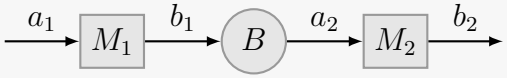
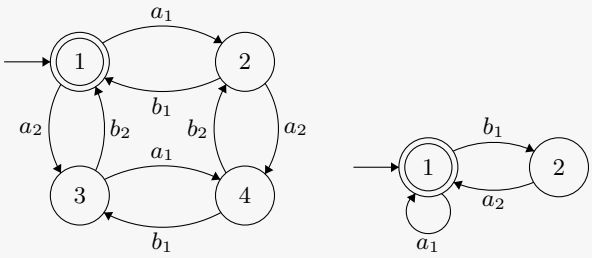


Figura 22 – Diagrama de Pequena Fábrica.

Os autômatos do supervisor e da planta da pequena fábrica estão mostrados na Figura 23. O comportamento em malha fechada do sistema, K, é mostrado na Figura 24.



(a) Autômato de G (b) Autômato de S

Figura 23 – Componentes do Sistema.

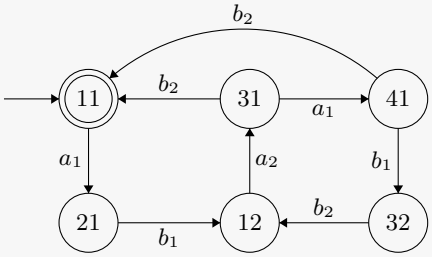


Figura 24 – Autômato K que implementa o comportamento em malha fechada da Pequena Fábrica.

O autômato resultante do sistema em malha fechada tem função de transição parcial, ou seja, nem todo o evento pode ocorrer em cada estado, o que torna menos provável a existência de uma palavra de sincronização. Dessa forma, para tornar mais provável a existência de uma palavra de sincronização, completamos o autômato com auto-laços com *eventos falsos*.

Definição 5.2.1. Seja $K = (Q, \Sigma, \delta, \_, \_)$, obtemos um autômato completo $K_C = (Q, \Sigma, \delta_C, \_, \_)$ onde

$$\delta_C(q, \sigma) = \begin{cases} \delta(q, \sigma), & \text{se } \sigma \in \Gamma(q) \\ \text{q}, & \text{caso contrário.} \end{cases}$$

Usando o mesmo Γ para K e K_C permite que, para cado estado $q \in Q$ definimos um evento σ como um *evento real* quando $\sigma \in \Gamma(q)$ ou um *evento falso* quando $\sigma \notin \Gamma(q)$.

Exemplo 5.6. Aplicando a transformação apresentada na Definição 5.2.1 ao autômato K, apresentado na Figura 24, obtemos o autômato K_C, mostrado na Figura 25.

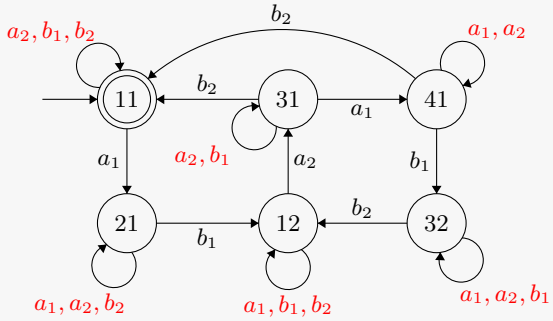


Figura 25 – Autômato K_C representando o sistema em malha fechada com *eventos falsos*.

O sistema implementado continua o mesmo sistema original, contudo ao executar uma cadeia com eventos falsos, conhecendo o estado atual do supervisor e da planta, é possível remover todos os eventos falsos e executar apenas os eventos reais. Nesse caso, conhecer o estado da planta e do supervisor é necessário pois afeta quais eventos são verdadeiros e quais eventos são falsos na palavra de sincronização.

É importante notar que um autômato completo não implica em ser sincronizável. Para verificar se o autômato resultante é sincronizável, o Algoritmo 2 pode ser usado.

5.2.1 Algoritmo

Após gerar o autômato completo, o próximo passo consiste em buscar por uma palavra de sincronização, que leva o sistema a um estado predefinido independentemente do estado de origem. É importante notar que, caso K_C não seja um autômato sincronizável, o algoritmo não converge, visto que não existe palavra de sincronização.

O problema de encontrar a menor palavra de sincronização em um autômato é NP-Difícil (KUDŁACIK; ROMAN; WAGNER, 2012), contudo, aplicando uma transformação não linear no espaço de estados do autômato podemos aplicar um algoritmo polinomial, como a busca em largura.

A abordagem aqui proposta busca obter a menor palavra de sincronização de K_C transformando o problema de encontrar a palavra de sincronização em um problema de busca no autômato powerset de K_C, denotado por P.

Definição 5.2.2. Seja $K_c = (Q, \Sigma, \delta_C, q_0, Q_m)$ um autômato completo. Podemos definir um novo autômato completo $P = (Q_P, \Sigma, \delta_P, q_{0P}, Q_{mP})$ como:

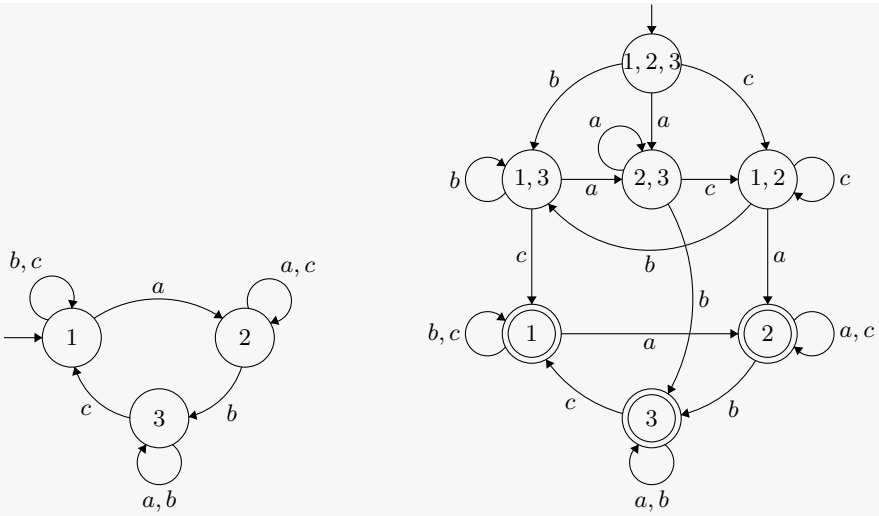
$$
\begin{aligned}
&q_0 = Q,\\
&Q_P = 2^Q \setminus \emptyset,\\
&Q_{mP} = \{q_p \in Q_P : \ |q_p| = 1\},\\
&\delta_P(q_P, \sigma) = \{\delta_C(q, \sigma) \mid q \in q_P \wedge q_P \in Q_P\}.
\end{aligned}
$$

Cada estado do autômato P é um conjunto de estados de K_C, seu estado inicial é o conjunto com todos os estados de K_C e os estados marcados são conjuntos com apenas um estado de K_C.

O autômato K_C é sincronizável se ao menos um estado marcado de P é alcançável. Podemos também encontrar, entre os estados marcados de P aquele que é mais útil para a reconfiguração do sistema, como, por exemplo, o estado inicial.

Exemplo 5.7. A computação de P encontra um autômato composto de 33 estados, sendo 6 deles marcados. Neste problema, o autômato é sincronizável e todos os estados do sistema em malha fechada são alcançáveis independentemente do estado atual do sistema.

Exemplo 5.8. Para ilustrar, considere o autômato A mostrado na Figura 26(a) e o autômato powerset criado a partir de A mostrado na Figura 26(b). Como podemos ver, o estado inicial do autômato powerset é um estado que representa todos os estados do autômato original e os estados marcados são aqueles com apenas um estado. Executando a palavra $w_1 = b\ c$ ($w_2 = a\ b$) a partir de qualquer estado de A leva ao estado 1. Então, w_1 e w_2 são palavras de sincronização.



(a) Autômato A (b) Autômato *Powerset* criado a partir de A

Figura 26 – Exemplo da criação de um autômato powerset.

Nesse sentido, qualquer $w_r \in \mathcal{L}_m(P)$ leva o sistema a um estado marcado de P no qual, por definição, é um único estado em K_C. Então, o algoritmo apresentado apenas tem que encontrar uma palavra que leva o sistema a um estado marcado, partindo do estado inicial de P. Como já discutido na Seção 5.1, P é grande em situações reais e deve ser criado durante a execução do algoritmo.

O algoritmo consiste numa busca em largura com uma heurística de *branch and bound*, partindo do estado inicial de P até que um estado marcado seja encontrado. O conjunto *visited* armazena os estados visitados pelo algoritmo, impedindo que o mesmo estado seja avaliado múltiplas vezes, já que usualmente os autômatos são cíclicos. A estrutura *path* armazena o caminho até cada estado visitado.

A primeira fronteira é o estado inicial, composto por todos os estados de K_C, e o caminho *path* ao estado inicial é ϵ. Após a inicialização (linhas 1 a 4), a cada iteração o algoritmo constrói uma nova fronteira com os estados adjacentes ainda não visitados.

Para cada estado o algoritmo avalia o número de estados de K_C que são representados e a heurística consiste em apenas adicionar novos estados à fronteira esse estado contém o mesmo número de elementos ou menos que o mínimo de elementos atual (linha 15). Essa heurística faz sentido considerando que o objetivo é chegar a um estado composto de apenas um elemento.

O algoritmo para quando um estado de P que contém um único estado de K_C é encontrado, retornando então a primeira sequência w_r encontrada (linhas 21 a 25).

É importante notar que a heurística proposta não garante que o algoritmo encontre uma palavra de sincronização, mesmo que ela exista. Caso o algoritmo não encontre uma palavra de sincronização, um algoritmo exato pode ser utilizado, como os apresentados

Algoritmo 2: Busca pela Palavra de Sincronização

```
Data: P = (Q_P, Σ, δ_P, q_0P, Q_mP)
Result: w_r
frontier ← {q_0P}
path[q_0P] ← ϵ
visited ← ∅
min ← |q_0P|

while |frontier| > 0 do

    visited ← visited ∪ frontier
    new_frontier ← ∅

    foreach q_o ∈ frontier do
        foreach σ ∈ Σ do
            q_d ← δ_P(q_o, σ)

            if q_d ∈ visited OR |q_d| > min then
                continue

            min ← |q_d|
            path[q_d] ← path[q_o] σ

            if q_d ∈ Q_mP then
                w_r ← path[q_d]
                return

            new_frontier ← new_frontier ∪{q_d}

    frontier ← new_frontier
```

em (TRAHTMAN, 2006; KUDŁACIK; ROMAN; WAGNER, 2012) ou outros algoritmos heurísticos como (NATARAJAN, 1986; ROMAN, 2009), contudo com o custo de usar mais memória e tempo de processamento. O algoritmo apresentado foi capaz de encontrar as palavras de sincronização para todos os exemplos testados gerando resultados rapidamente, quando comparado ao tempo de execução sem utilizar a heurística. Como a Conjectura de Cerny (VOLKOV, 2008) estabelece que o limite superior para a menor palavra de sincronização é dada por $(n-1)^2$ onde n é o número de estados do autômato, é possível utilizar essa profundidade como limite de busca e, dessa forma, poupar tempo de processamento.

Exemplo 5.9. Executando o algoritmo para o exemplo da Pequena Fábrica, nos obtemos a palavra de sincronização $w = b_2\ b_1\ a_2\ b_2$ que leva o sistema ao estado inicial independentemente do estado no qual o sistema se encontra. Supondo que o supervisor

esteja no estado 32 e a planta esteja no estado 1 e o supervisor esteja no estado 2, ao aplicar a palavra encontrada no sistema, os dois primeiros eventos de w são ignorados e os dois últimos são executados.

Para o autômato do Exemplo 5.8, apresentado na Figura 26(b), podemos ver, por inspeção, que as menores palavras de sincronização do autômato A são $w_1 = b\,c$ (ao estado inicial), $w_2 = a\,b$ (ao estado 3) e $w_3 = c\,a$ (ao estado 2).

Os resultados desta seção foram apresentados no Workshop on Discrete-Event Systems 2020 (Wodes 2020) (ALVES; PENA, 2020a).

6 Estudo de Caso

Como exemplo de aplicação da técnica de recuperação de sistemas usando eventos de recuperação, será utilizado um Sistema Flexível de Manufatura (SFM) composto por oito dispositivos e oito *buffers* unitários. As plantas são três esteiras (C_1, C_2 e C_3), uma fresadora, um torno, um robô, uma máquina de pintura (MP) e uma máquina de montagem (MM), como mostrado na Figura 27. O SFM é capaz de produzir dois tipos de produto a partir de blocos e tarugos, sendo eles uma base com pino cônico, chamado Produto A, e uma base com um pino cilíndrico pintado, chamado Produto B.

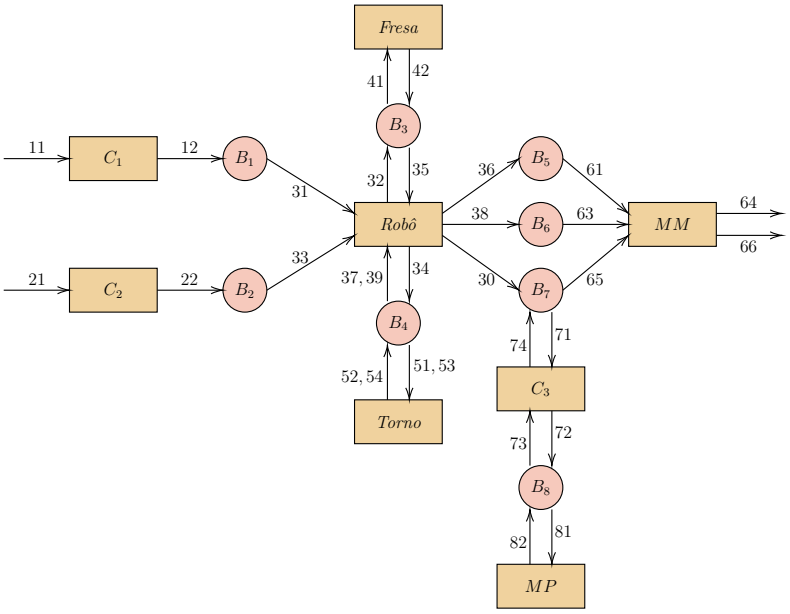


Figura 27 – Diagrama do Sistema Flexível de Manufatura

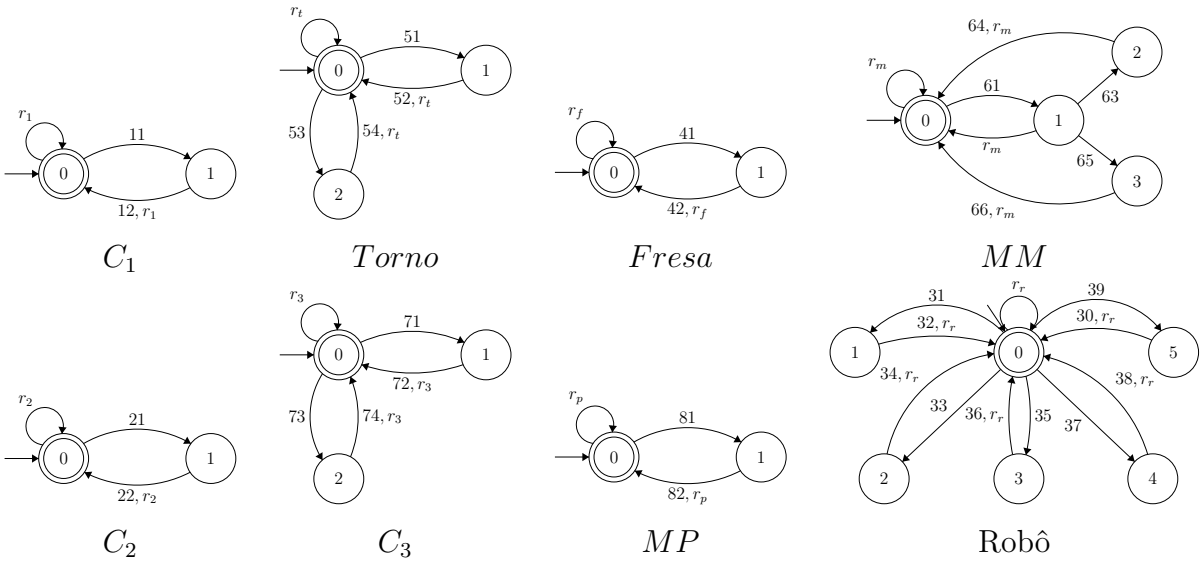


Figura 28 – Autômatos das Plantas que compõem o Sistema Flexível de Manufatura.

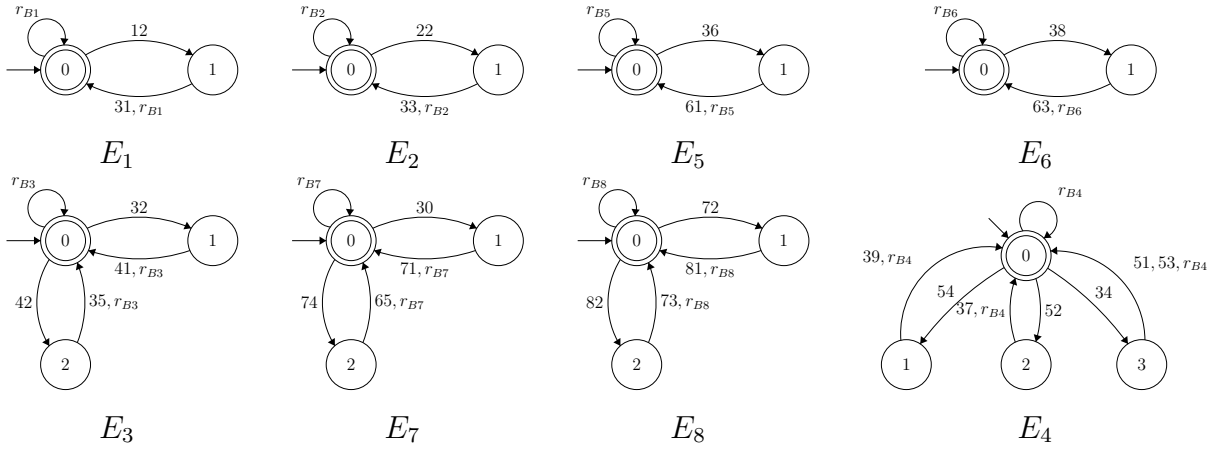


Figura 29 – Autômatos das Especificações de segurança que compõem o Sistema Flexível de Manufatura.

Na Figura 28 estão os autômatos das plantas que formam o Sistema Flexível de Manufatura, modeladas por autômatos sincronizáveis com eventos de recuperação. A Figura 29 apresenta as especificações de segurança do SFM, que garantem que não ocorra *overflow* nem *underflow* nos *buffers* do sistema.

No total, foram criados 15 eventos de recuperação, um evento para cada planta e especificação, contudo, é importante salientar que isso é uma decisão de projeto, caso necessário um mesmo evento de recuperação pode reiniciar mais de autômato.

O supervisor monolítico do sistema possui 70.272 estados e 1.434.804 transições, o que torna difícil sua implementação num sistema real de manufatura. Para tratar esse problema, foi aplicado o Controle Modular Local a fim de obter um conjunto de supervisores controláveis, não-bloqueantes e não-conflitantes entre si. Para isso foi utilizado o *software* UltraDES (MARTINS; ALVES; PENA, 2017), contudo com um algoritmo de síntese adaptado conforme o Corolário 5 e a Definição 4.3.1. Como já sabe-se, *a priori*, que o sistema é conflitante, utilizamos a composição das especificações E_7 e E_8 como uma única especificação local, o que resolve o problema de conflito.

Os supervisores resultantes apresentam as seguintes características:

Supervisor	**Plantas**	**Estados**	**Transições**	**Reset**
S_1	C_1, Robô	18	94	36
S_2	C_2, Robô	18	94	54
S_3	Fresa, Robô	18	90	54
S_4	Torno, Robô	21	105	63
S_5	MM, Robô	44	253	132
S_6	MM, Robô	44	253	132
$S_{7,8}$	C_3, MP, MM, Robô	260	2441	1560

As palavras de sincronização dos supervisores locais reiniciam tanto as suas plantas locais quanto suas especificações locais e essa propriedade pode ser utilizada na recuperação

do sistema.

Suponha, por exemplo, que o evento 51, que liga o torno, tenha sido executado pelo controlador gerando uma mudança de estado no supervisor S_4, porém, devido a uma falha na transmissão do sinal ou a um ataque externo, o torno não liga. Quando isso ocorre, o estado que o supervisor S_4 estima que o torno está é distinto do estado que o dispositivo realmente se encontra, levando o sistema a uma situação de bloqueio, enquanto os demais supervisores continuam executando suas tarefas sem serem diretamente afetados.

Cessando a causa da falha, torna-se necessário levar o sistema a uma situação de sincronia entre o estado do supervisor e da planta. Para isso, executa-se uma palavra de sincronização de S_4, como por exemplo $w_4 = r_{B_4}\, r_r\, r_t$, que recupera o sincronismo entre o supervisor local e a planta local. É importante notar, que a execução de w_4 gera efeitos colaterais nos demais supervisores, porém esses efeitos colaterais não levam ao reinício completo do sistema, somente de determinadas plantas. No caso do exemplo, o robô é reiniciado, o que acarreta em uma mudança de estado em todos os supervisores.

É importante notar que mesmo utilizando o controle modular local podemos obter supervisores locais grandes, como é o caso do supervisor que relacionado às especificações E_7 e E_8 ($S_{7,8}$), composto por 260 estados. Nesse caso as demais técnicas apresentadas possuem um papel relevante, como a redução e localização de supervisores.

Aplicando a redução de supervisores a cada supervisor modular obtemos o seguinte resultado:

Supervisor	**Plantas**	**Estados**	**Transições**	**Reset**
S_{SIM_1}	C_1, Robô	3	9	3
S_{SIM_2}	C_2, Robô	3	9	6
S_{SIM_3}	Fresa, Robô	4	14	8
S_{SIM_4}	Torno, Robô	5	20	10
S_{SIM_5}	MM, Robô	6	35	12
S_{SIM_6}	MM, Robô	29	193	87
$S_{SIM_{7,8}}$	C_3, MP, MM, Robô	64	743	384

Como é possível observar, os supervisores resultantes têm um tamanho menor do que os supervisores originais, de forma que sua implementação em um sistema real é mais fácil e exige menos recursos computacionais. Neste caso, podemos considerar o mesmo problema da análise anterior, onde o evento 51, que liga o torno, tenha sido executado pelo controlador gerando uma mudança de estado no supervisor S_{SIM_4}, mas não na planta. Suponha que o torno (Figura 28) se encontra no estado 0 e o supervisor S_{SIM_4} (Figura 30) se encontra no estado 1, quando o evento 51 ocorre, o torno e o supervisor deveriam transicionar, respectivamente, para os estados 1 e 2, contudo, como o torno, devido a uma falha, não observa o evento, ele permanece no estado 0. Essa situação leva o sistema local a um bloqueio, visto que, ignorando os eventos de recuperação, não existem eventos

habilitados em comum entre o estado 0 do torno e o estado 2 de S_{SIM_4}.

Para resolver esse problema, podemos aplicar uma palavra de sincronização do supervisor S_{SIM_4} $w_{SIM_4} = r_{B_4}\, r_t$, como visto na Figura 30. Quando tratamos do supervisor reduzido, não é necessário reiniciar o robô para reiniciar o supervisor, gerando um efeito colateral muito menor no sistema.

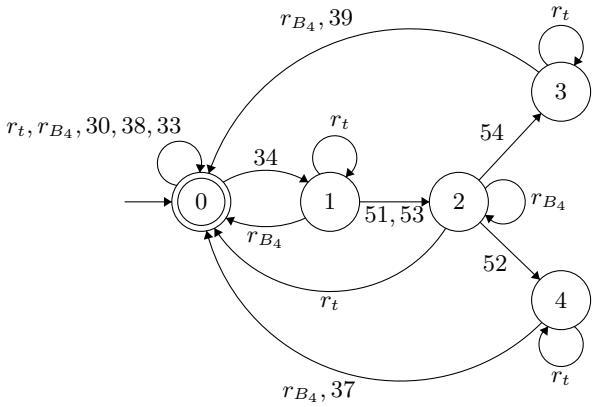


Figura 30 – Supervisor Reduzido S_{SIM_4}.

Outra abordagem para tratar o problema é a criação de supervisores localizados, contudo isso não é viável nesse exemplo pois o Robô é uma planta que interage com quase todos os demais componentes do sistema e seu respectivo supervisor localizado ($S_{L,Robô}$) é muito grande, o que torna a implementação pouco viável.

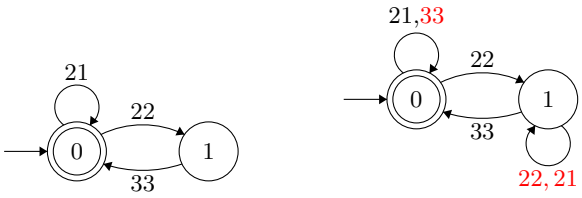
Supervisor	**Plantas**	**Estados**	**Transições**	**Reset**
S_{L,C_1}	C_1	3	39	6
S_{L,C_2}	C_2	3	39	9
$S_{L,Torno}$	Torno	5	45	15
$S_{L,Fresa}$	Fresa	4	42	12
$S_{L,Robô}$	Robô	14.220	326.166	213.300
$S_{L,MM}$	MM	188	2.548	1.504
S_{L,C_3}	C_3	41	549	246
$S_{L,MP}$	MP	41	549	246

É importante notar que alguns cuidados devem ser tomados ao utilizar o supervisor localizado para sincronizar um sistema em falha. Apesar de possuir um supervisor para cada planta, os supervisores localizados podem possuir eventos de comunicação, que não pertencem ao alfabeto da sua respectiva planta. Nesse caso, é necessário reiniciar todos os supervisores localizados que possuam eventos em comum com a planta.

No exemplo apresentado, quando o erro ocorre no evento 51 do torno, além do supervisor $S_{L,Torno}$, os supervisores $S_{L,Robô}$ e $S_{L,MM}$ também seriam reiniciados e como $S_{L,Robô}$ afeta praticamente todo o sistema, isso resultaria em um reinicio global.

6.1 Ressincronização

Considere o mesmo Sistema Flexível de Manufatura apresentado na Figura 27, contudo com todos os eventos de recuperação removidos. Supondo que os autômatos das plantas e supervisores modulares locais reduzidos estejam implementados, podemos aplicar o procedimento desenvolvido na Seção 5.1, a fim de recuperar o sistema de falhas possam ocorrer e levar o sistema a estados indesejados.



a) S_{SIM_2} b) S_{SIM_2} com eventos falsos

Figura 31 – Supervisor local, antes e depois da adição de eventos falsos.

Para exemplificar, aplicamos o método proposto apenas no supervisor S_{SIM_2}, que controla a planta local $C_2||Robô$, mostrado na Figura 31 a, e seu comportamento com a adição de eventos falsos está mostrado na Figura 31 b. Como pode ser visto, são adicionados novos eventos ao supervisor, bem como à planta local, de forma que a composição de ambos apresente tanto os estados seguros, que são alcançados quando o supervisor funciona de maneira adequada, quanto estados de falha, que ocorrem quando o supervisor é incapaz de evitar a ocorrência de um evento proibido. A Figura 32 apresenta a planta local já com os eventos falsos adicionados.

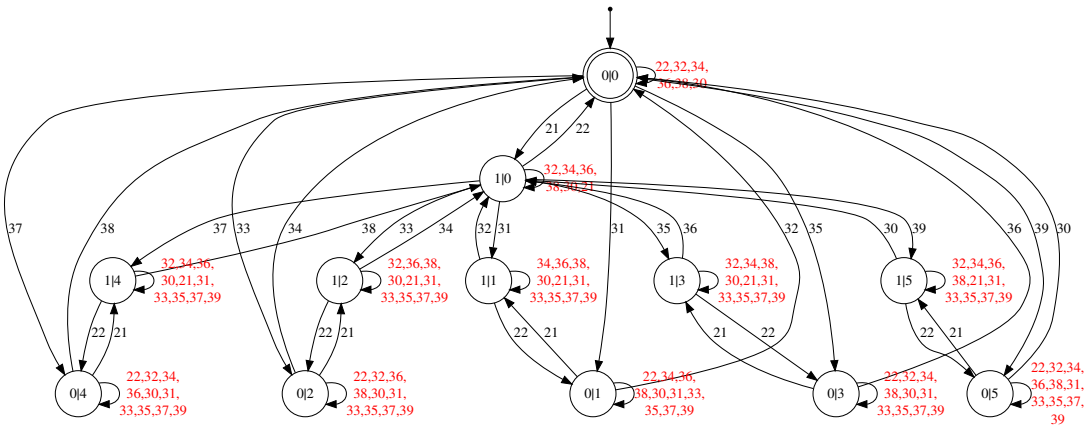


Figura 32 – Planta Local $P_2 = C_2||Robô$ com eventos falsos (gerado pelo GraphViz).

A Figura 33 apresenta o autômato T, que modela a composição do supervisor e da planta com os eventos falsos adicionados. Como é possível ver, dos 28 estados do autômato, 6 deles estão apresentados em vermelho pois são estados de falha, ou seja, estados que não são alcançados quando o sistema está funcionando corretamente. O autômato *powerset* P, construído a partir de T possui no máximo $2^{28} - 1$ estados.

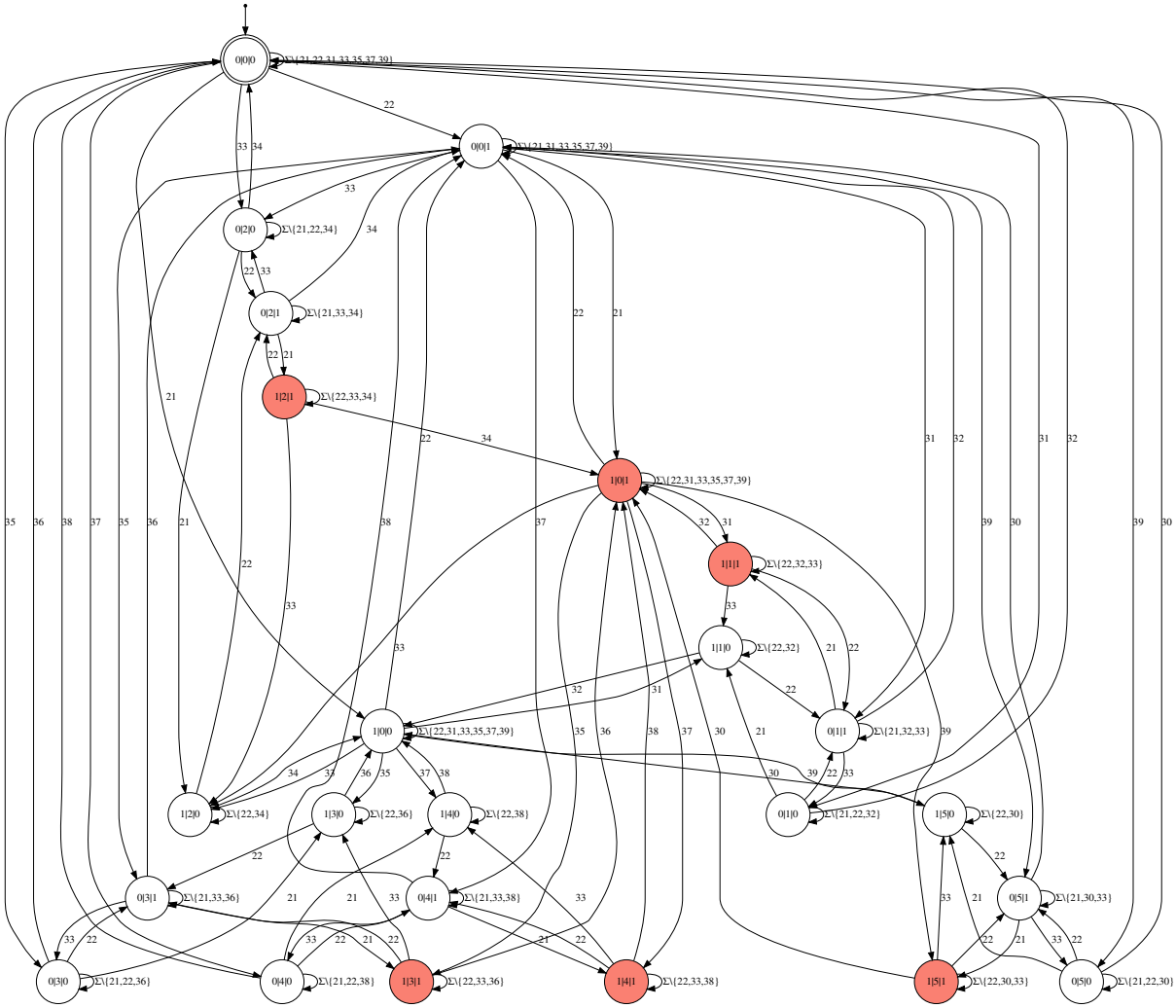


Figura 33 – Autômato T (gerado pelo GraphViz).

O Algoritmo 1 foi aplicado ao autômato P e encontra como resultado uma sequência com apenas um evento, $w_r = 22$. Como pode ser observado, a execução do evento 22 sempre leva a um estado seguro, ou seja, um estado no qual a combinação de estados da planta e do supervisor são compatíveis.

É importante notar, contudo, que essa sequência obtida só deve ser utilizada em caso de falha, visto que a palavra de ressincronização obtida pode passar por caminhos que estão no comportamento controlado do sistema, logo seu uso quando o sistema está em funcionamento correto, apesar de levar o sistema a um estados seguro, não necessariamente respeita as especificações de segurança do sistema.

6.2 Reconfiguração

Utilizando novamente o Sistema Flexível de Manufatura (SFM), apresentado na Figura 27, sem os eventos de recuperação, podemos aplicar os resultados da Seção 5.2

para realizar a reconfiguração do sistema, independentemente do estado no qual o sistema se encontra.

O comportamento em malha fechada S do SFM pode ser representado por um autômato com 45.504 estados e 200.124 transições. Após a aplicação do procedimento apresentado na Definição 5.2.1, que torna o autômato completo, o número de estados permanece o mesmo contudo o número de transições aumenta para 1.151.712. No pior cenário, o autômato *powerset* possui $2^{45,504} - 1$ estados.

Para esse problema, o Algoritmo 2 executou em aproximadamente 1 segundo, encontrando uma palavra de sincronização com 59 eventos:

$$\begin{aligned} w_1 = \ & 5\ 14\ 4\ 1\ 9\ 2\ 7\ 26\ 27\ 31\ 12\ 15\ 17 \\ & 19\ 20\ 23\ 22\ 24\ 28\ 29\ 21\ 22\ 23\ 24 \\ & 25\ 26\ 27\ 13\ 2\ 7\ 11\ 23\ 24\ 12\ 15\ 16 \\ & 17\ 19\ 28\ 29\ 21\ 22\ 25\ 26\ 27\ 13\ 16 \\ & 2\ 17\ 19\ 7\ 28\ 29\ 13\ 2\ 7\ 16\ 17\ 19. \end{aligned}$$

Na maior parte dos casos, é mais útil obter uma palavra de sincronização que leva a um estado específico e, para isso, é possível adicionar restrições ao algoritmo para que ele somente pare ao chegar ao estado desejado. Considerando a situação na qual deseja-se alcançar o estado inicial, o tempo de execução do algoritmo aumentou para aproximadamente $1,2$ segundos e a palavra de sincronização obtida possui 68 eventos.

$$\begin{aligned} w_2 = \ & 6\ 2\ 1\ 9\ 8\ 7\ 3\ 31\ 25\ 26\ 19\ 20\ 14\ 27 \\ & 16\ 17\ 21\ 29\ 22\ 23\ 12\ 1\ 9\ 18\ 19\ 20 \\ & 17\ 21\ 24\ 25\ 26\ 11\ 2\ 8\ 13\ 14\ 27\ 12\ 1 \\ & 9\ 18\ 19\ 20\ 22\ 23\ 24\ 30\ 31\ 25\ 26\ 28 \\ & 10\ 29\ 11\ 13\ 12\ 15\ 16\ 18\ 17\ 21\ 19\ 20 \\ & 22\ 23\ 24\ 30\ 31. \end{aligned}$$

Em ambos os casos, o algoritmo foi capaz de encontrar uma palavra de sincronização para um problema de tamanho médio em poucos segundos. Utilizando as sequências obtidas é possível reconfigurar o sistema, ou seja, levá-lo a um determinado estado independentemente do estado no qual ele se encontra. A sequência que leva ao estado inicial, por exemplo, pode ser útil qual deseja-se que todas as máquinas estejam desligadas para realizar uma manutenção.

Como esse sequência possui eventos controláveis e não controláveis, é necessário, ao menos enquanto esse processo de reconfiguração é executado, que a ocorrência dos eventos não controláveis possa ser forçada quando esses eventos correspondem a eventos verdadeiros.

Conclusão

Neste trabalho foram apresentadas diversas definições, lemas, teoremas e corolários que possibilitaram o uso de autômatos sincronizáveis no contexto da Teoria de Controle Supervisório. As principais contribuições desta tese são discutidas a seguir:

a) A extensão da Teoria de Controle Supervisório clássico para a síntese de supervisores com eventos de recuperação. O mesmo resultado também é obtido para o Controle Supervisório Modular Local. Demonstramos também que os algoritmos tradicionais de redução e localização de supervisores produzem, respectivamente, supervisores reduzidos e localizados sincronizáveis quando as plantas e supervisores são sincronizáveis com relação ao estado inicial

b) A definição da classe de autômatos sincronizáveis com relação ao estado inicial. Sabendo que nem sempre é comum encontrar supervisores e plantas sincronizáveis com relação ao estado inicial, foi apresentado um método de modelar sistemas como autômatos sincronizáveis utilizando eventos de recuperação.

c) A análise das condições nas quais a sincronização sobrevive às diversas operações relacionadas à síntese de supervisores, como a projeção inversa, interseção de linguagens e composição paralela de autômatos. Com a adição dos eventos de recuperação foi possível afrouxar as condições para que a sincronização se mantenha.

d) O uso da sincronização como uma forma de recuperar um sistema quando o estado da planta se dessincroniza do estado do supervisor devido a uma falha de sensor ou ataque externo. Sabendo que o supervisor compartilha palavras de sincronização com o comportamento geral do sistema, as diferentes formas de implementação do sistema permitem diferentes graus de desacoplamento dos subsistemas, com diferentes efeitos colaterais no sistema como um todo.

e) A recuperação ou reconfiguração de sistemas legados utilizando palavras de sincronização. Tais abordagens são baseadas em autômatos sincronizáveis utilizando eventos falsos. Nesses casos, é necessário uma atenção maior quanto à implementação, visto que esses eventos falsos devem operar apenas no contexto lógico do sistema, sem afetar diretamente a dinâmica física do sistema.

É importante notar que as proposições e demonstrações apresentadas nesse texto são relacionadas à Teoria de Controle Supervisório tradicional e, dessa forma, se aplicam a diversas outras técnicas baseadas no TCS com poucas ou, muitas vezes, sem nenhuma modificação. De maneira similar, as proposições e demonstrações relacionadas à redução e à localização de supervisores não dependem de um algoritmo específico e se mantém

verdadeiras desde que a implementação cumpra as premissas estabelecidas.

O conteúdo dessa tese gerou os seguintes artigos:

- ALVES, L.; PENA, P. Sincronização em sistemas a eventos discretos. In:XXII Congresso Brasileiro de Automática (CBA). 2018.
- ALVES, L. V. R.; PENA, P. N. Secure recovery procedure for manufacturing systems using synchronizing automata and supervisory control theory.IEEE Transactions on Automation Science and Engineering, IEEE, 2020.
- ALVES L. V. R.; PENA, P. N. Reconfiguration of discrete event systems using synchronizing words. In:Workshop on Discrete-Event Systems 2020 (Wodes 2020). 2020.
- ALVES, L. V. R.; PENA, P. N. Synchronism recovery of discrete event systems. In:IFAC world congress. 2020.
- ALVES, L. V. R.; PENA, P. N. On the reduction and localization of synchronizing supervisors. In:American Control Conference 2021 (ACC 2021). 2021. Submetido.

Diversos trabalhos futuros podem ser elaborados a partir dessa tese, como estudar novas aplicações de autômatos sincronizáveis na área de Sistemas a Eventos Discretos, expandir as definições apresentadas para outras técnicas de controle supervisório, simplificar as premissas dos teoremas propostos, verificar propriedades de opacidade e diagnosticabilidade de autômatos sincronizáveis com relação ao estado inicial.

Referências

ABAD, F. A. T.; MANCUSO, R.; BAK, S.; DANTSKER, O.; CACCAMO, M. Reset-based recovery for real-time cyber-physical systems with temporal safety constraints. In: IEEE. *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. [S.l.], 2016. p. 1–8. Citado na página 22.

ALVES, L.; PENA, P. Sincronização em sistemas a eventos discretos. In: *XXII Congresso Brasileiro de Automática (CBA)*. [S.l.: s.n.], 2018. Citado na página 50.

ALVES, L. V. R.; PENA, P. N. Secure recovery procedure for manufacturing systems using synchronizing automata and supervisory control theory. *IEEE Transactions on Automation Science and Engineering*, IEEE, 2019. No prelo. Citado na página 61.

ALVES, L. V. R.; PENA, P. N. Reconfiguration of discrete event systems using synchronizing words. In: *Workshop on Discrete-Event Systems 2020 (Wodes 2020)*. [S.l.: s.n.], 2020. Citado na página 75.

ALVES, L. V. R.; PENA, P. N. Synchronism recovery of discrete event systems. In: *IFAC world congress*. [S.l.: s.n.], 2020. Citado na página 69.

ALVES, L. V. R.; PENA, P. N. On the reduction and localization of synchronizing supervisors. In: *American Control Conference 2021 (ACC 2021)*. [S.l.: s.n.], 2021. Submetido. Citado na página 51.

ALVES, M. V.; CUNHA, A. E. da; CARVALHO, L. K.; MOREIRA, M. V.; BASILIO, J. C. Robust supervisory control of discrete event systems against intermittent loss of observations. *International Journal of Control*, Taylor & Francis, p. 1–13, 2019. Citado na página 23.

ANANICHEV, D.; VOLKOV, M. Some results on cerny type problems for transformation semigroups. *Semigroups and languages*, p. 23–42, 2004. Citado na página 39.

ANDERSSON, K.; LENNARTSON, B.; FABIAN, M. Synthesis of restart states for manufacturing cell controllers. *IFAC Proceedings Volumes*, Elsevier, v. 42, n. 5, p. 263–268, 2009. Citado na página 23.

ANDERSSON, K.; LENNARTSON, B.; FABIAN, M. Restarting manufacturing systems; restart states and restartability. *IEEE Transactions on Automation Science and Engineering*, v. 7, n. 3, p. 486–499, July 2010. ISSN 1545-5955. Citado na página 23.

ANDERSSON, K.; LENNARTSON, B.; FALKMAN, P.; FABIAN, M. Generation of restart states for manufacturing cell controllers. *Control Engineering Practice*, v. 19, n. 9, p. 1014 – 1022, 2011. ISSN 0967-0661. Special Section: DCDS'09 – The 2nd IFAC Workshop on Dependable Control of Discrete Systems. Citado na página 23.

BEHNAM, M.; SHIN, I.; NOLTE, T.; NOLIN, M. Sirap: a synchronization protocol for hierarchical resource sharingin real-time open systems. In: ACM. *Proceedings of the 7th ACM & IEEE international conference on Embedded software*. [S.l.], 2007. p. 279–288. Citado na página 38.

BENENSON, Y.; ADAR, R.; PAZ-ELIZUR, T.; LIVNEH, Z.; SHAPIRO, E. Dna molecule provides a computing machine with both data and fuel. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 100, n. 5, p. 2191–2196, 2003. Citado na página 38.

BERGAGåRD, P.; FABIAN, M. Calculating restart states for systems modeled by operations using supervisory control theory. *Machines*, v. 1, n. 3, p. 116–141, 2013. ISSN 2075-1702. Citado na página 23.

BERGAGåRD, P.; FALKMAN, P.; FABIAN, M. Modeling and automatic calculation of restart states for an industrial windscreen mounting station. *IFAC-PapersOnLine*, v. 48, n. 3, p. 1030 – 1036, 2015. ISSN 2405-8963. 15th IFAC Symposium on Information Control Problems in Manufacturing. Citado na página 23.

BEUHRING, A.; SALOUS, K. Beyond blacklisting: Cyberdefense in the era of advanced persistent threats. *IEEE Security and Privacy*, v. 12, n. 5, p. 90–93, 2014. ISSN 15407993. Citado na página 21.

BOCCALETTI, S.; KURTHS, J.; OSIPOV, G.; VALLADARES, D.; ZHOU, C. The synchronization of chaotic systems. *Physics reports*, Elsevier, v. 366, n. 1, p. 1–101, 2002. Citado na página 24.

BRANDIN, B. A.; MALIK, R.; MALIK, P. Incremental verification and synthesis of discrete-event systems guided by counter examples. *IEEE Transactions on Control Systems Technology*, IEEE, v. 12, n. 3, p. 387–401, 2004. Citado na página 21.

CAI, K.; WONHAM, W. M. Supervisor localization: a top-down approach to distributed control of discrete-event systems. *IEEE Transactions on Automatic Control*, IEEE, v. 55, n. 3, p. 605–618, 2010. Citado na página 37.

CARVALHO, L. K.; WU, Y.-C.; KWONG, R.; LAFORTUNE, S. Detection and mitigation of classes of attacks in supervisory control systems. *Automatica*, v. 97, p. 121 – 133, 2018. ISSN 0005-1098. Citado na página 22.

CASSANDRAS, C. G.; LAFORTUNE, S. *Introduction to discrete event systems*. [S.l.]: Springer Science & Business Media, 2009. Citado na página 27.

CHEN, Y.-B.; IERARDI, D. The complexity of oblivious plans for orienting and distinguishing polygonal parts. *Algorithmica*, Springer, v. 14, n. 5, p. 367–397, 1995. Citado na página 38.

CHRISTOFIDES, P. D. et al. Smart plant operations: Vision, progress and challenges. *AIChE Journal*, v. 53, n. 11, p. 2734–2741, 2007. Citado na página 22.

DELYON, B.; MALER, O. On the effects of noise and speed on computations. *Theoretical Computer Science*, v. 129, n. 2, p. 279 – 291, 1994. ISSN 0304-3975. Citado na página 23.

EPPSTEIN, D. Reset sequences for finite automata with application to design of parts orienters. In: *Proceedings of the 15th International Colloquium on Automata, Languages and Programming*. London, UK: Springer-Verlag, 1988. (ICALP '88), p. 230–238. ISBN 3-540-19488-6. Citado na página 38.

GOLDBERG, K. Y. Orienting polygonal parts without sensors. *Algorithmica*, v. 10, n. 2, p. 201–225, 1993. ISSN 1432-0541. Citado na página 38.

GUSEV, V. V.; MASLENNIKOVA, M. I.; PRIBAVKINA, E. V. Principal ideal languages and synchronizing automata. *Fundamenta Informaticae*, IOS Press, v. 132, n. 1, p. 95–108, 2014. Citado na página 40.

HEUVEL, M. M. van den; BRIL, R. J.; LUKKIEN, J. J. Transparent synchronization protocols for compositional real-time systems. *IEEE Transactions on Industrial Informatics*, IEEE, v. 8, n. 2, p. 322–336, 2012. Citado na página 38.

HOPCROFT, J. E.; MOTWANI, R.; ULLMAN, J. D. Automata theory, languages, and computation. *International Edition*, v. 24, 2006. Citado na página 30.

JÜRGENSEN, H. Synchronization. *Information and Computation*, Elsevier, v. 206, n. 9-10, p. 1033–1044, 2008. Citado na página 38.

KUDŁACIK, R.; ROMAN, A.; WAGNER, H. Effective synchronizing algorithms. *Expert Systems with Applications*, Elsevier, v. 39, n. 14, p. 11746–11757, 2012. Citado 2 vezes nas páginas 71 e 74.

LAFORTUNE, S.; LIN, F.; HADJICOSTIS, C. N. On the history of diagnosability and opacity in discrete event systems. *Annual Reviews in Control*, v. 45, p. 257–266, 2018. ISSN 13675788. Citado na página 22.

LARSEN, K. G.; LAURSEN, S.; SRBA, J. Synchronizing strategies under partial observability. In: SPRINGER. *International Conference on Concurrency Theory*. [S.l.], 2014. p. 188–202. Citado na página 38.

LIN, F.; WONHAM, W. M. Decentralized supervisory control of discrete-event systems. *Information sciences*, Elsevier, v. 44, n. 3, p. 199–224, 1988. Citado na página 33.

LOBORG, P. Error Recovery in Automation An Overview. *AAAI-94 Spring Symposium on Detecting and . . .*, p. 94–100, 1994. Citado na página 22.

MARTINS, L. R. R.; ALVES, L. V. R.; PENA, P. N. Ultrades-a library for modeling, analysis and control of discrete event systems. *Proceedings of the 20th World Congress of the International Federation of Automatic Control*, Elsevier, v. 50, n. 1, p. 5831–5836, 2017. Citado na página 78.

MARTYUGIN, P. V. Careful synchronization of partial automata with restricted alphabets. In: BULATOV, A. A.; SHUR, A. M. (Ed.). *Computer Science – Theory and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 76–87. ISBN 978-3-642-38536-0. Citado na página 41.

MASTERS, M. D. B. Synchronizing automata and the cerny conjecture. East Carolina University, 2012. Citado na página 40.

MAUBERT, B. *Synchronizing automata and their applications to games with imperfect information*. [S.l.], 2009. Citado na página 38.

NATARAJAN, B. Some paradigms for the automated design of parts feeders. *The International Journal of Robotics Research*, v. 8, n. 6, p. 98–109, 1989. Citado na página 38.

NATARAJAN, B. K. An algorithmic approach to the automated design of parts orienters. In: *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 1986. (SFCS '86), p. 132–142. ISBN 0-8186-0740-8. Citado 2 vezes nas páginas 38 e 74.

NOLTE, T.; SHIN, I.; BEHNAM, M.; SJODIN, M. A synchronization protocol for temporal isolation of software components in vehicular systems. *IEEE Transactions on Industrial Informatics*, IEEE, v. 5, n. 4, p. 375–387, 2009. Citado na página 38.

PIKOVSKY, A.; ROSENBLUM, M.; KURTHS, J. *Synchronization : a universal concept in nonlinear sciences*. Cambridge: Cambridge University Press, 2001. (The Cambridge nonlinear science series). ISBN 0-521-59285-2. Citado na página 24.

POCCI, M.; DEMONGODIN, I.; GIAMBIASI, N.; GIUA, A. A new algorithm to compute synchronizing sequences for synchronized petri nets. In: IEEE. *TENCON 2013-2013 IEEE Region 10 Conference (31194)*. [S.l.], 2013. p. 1–6. Citado na página 40.

POCCI, M.; DEMONGODIN, I.; GIAMBIASI, N.; GIUA, A. Testing experiments on synchronized petri nets. *IEEE Transactions on Automation Science and Engineering*, IEEE, v. 11, n. 1, p. 125–138, 2014. Citado na página 40.

POCCI, M.; DEMONGODIN, I.; GIAMBIASI, N.; GIUA, A. Testing experiments on unbounded systems: synchronizing sequences using petri nets. *IFAC Proceedings Volumes*, Elsevier, v. 47, n. 2, p. 155–161, 2014. Citado na página 40.

POCCI, M.; DEMONGODIN, I.; GIAMBIASI, N.; GIUA, A. Synchronizing sequences on a class of unbounded systems using synchronized petri nets. *Discrete Event Dynamic Systems*, Springer, v. 26, n. 1, p. 85–108, 2016. Citado na página 40.

QUEIROZ, M. H. de; CURY, J. E. R. Modular control of composed systems. In: *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)*. [S.l.: s.n.], 2000. v. 6, p. 4051–4055 vol.6. ISSN 0743-1619. Citado na página 21.

QUEIROZ, M. H. de; CURY, J. E. R. Synthesis and implementation of local modular supervisory control for a manufacturing cell. *Proceedings - 6th International Workshop on Discrete Event Systems, WODES 2002*, p. 377–382, 2002. Citado 3 vezes nas páginas 33, 34 e 50.

ROMAN, A. Synchronizing finite automata with short reset words. *Applied Mathematics and Computation*, Elsevier, v. 209, n. 1, p. 125–136, 2009. Citado na página 74.

RUDIE, K.; WONHAM, W. M. Think globally, act locally: Decentralized supervisory control. In: IEEE. *1991 American Control Conference*. [S.l.], 1991. p. 898–903. Citado na página 33.

RUDIE, K.; WONHAM, W. M. Think globally, act locally: decentralized supervisory control. *IEEE Transactions on Automatic Control*, v. 37, n. 11, p. 1692–1708, Nov 1992. ISSN 0018-9286. Citado na página 21.

SABOORI, A.; HADJICOSTIS, C. N. Notions of Security and Opacity in Discrete Event Systems. *Proceedings of the IEEE Conference on Decision and Control*, p. 5056–5061, 2007. ISSN 01912216. Citado na página 21.

SHU, S. Recoverability of discrete-event systems with faults. *IEEE Transactions on Automation Science and Engineering*, v. 11, n. 3, p. 930–935, July 2014. ISSN 1545-5955. Citado 2 vezes nas páginas 23 e 55.

SIVOLELLA, L. *Contributions for Supervisors Reduction for Discrete Event Systems*. Dissertação (Mestrado) — IME, Rio de Janeiro, Brazil, 2005. Citado na página 36.

SU, R. A Cyber Attack Model with Bounded Sensor Reading Alterations. In: *2017 American Control Conference*. Seattle, USA: [s.n.], 2017. p. 3200–3205. ISBN 9781509059942. Citado na página 22.

SU, R.; WONHAM, W. M. Supervisor reduction for discrete-event systems. *Discrete Event Dynamic Systems*, Springer, v. 14, n. 1, p. 31–53, 2004. Citado 3 vezes nas páginas 34, 35 e 36.

TANKARD, C. Advanced Persistent Threats and How to Monitor and Deter Them. *Network Security*, v. 2011, n. 8, p. 16 – 19, 2011. ISSN 1353-4858. Citado na página 21.

TRAHTMAN, A. An efficient algorithm finds noticeable trends and examples concerning the černy conjecture. In: SPRINGER. *International Symposium on Mathematical Foundations of Computer Science*. [S.l.], 2006. p. 789–800. Citado na página 74.

VAZ, A. F.; WONHAM, W. M. On supervisor reduction in discrete-event systems. *International Journal of Control*, Taylor & Francis, v. 44, n. 2, p. 475–491, 1986. Citado 2 vezes nas páginas 34 e 36.

VIRVILIS, N.; GRITZALIS, D.; APOSTOLOPOULOS, T. Trusted Computing vs. Advanced Persistent Threats: Can a Defender Win this Game? *Proceedings of the IEEE 10th International Conference on Ubiquitous Intelligence and Computing, UIC 2013 and IEEE 10th International Conference on Autonomic and Trusted Computing, ATC 2013*, n. February, p. 396–403, 2013. Citado na página 22.

VOLKOV, M. V. Synchronizing automata and the cerný conjecture. In: _____. *Language and Automata Theory and Applications: Second International Conference, LATA 2008, Tarragona, Spain, March 13-19, 2008. Revised Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 11–27. Citado 4 vezes nas páginas 13, 39, 42 e 74.

WAKAIKI, M.; TABUADA, P.; HESPANHA, J. P. Supervisory control of discrete-event systems under attacks. *CoRR*, abs/1701.00881, 2017. Citado na página 22.

WONG, K. C.; WONHAM, W. M. Hierarchical control of discrete-event systems. *Discrete Event Dynamic Systems*, Springer, v. 6, n. 3, p. 241–273, 1996. Citado na página 21.

WONHAM, W. M. *Supervisory Control of Discrete-Event Systems*. Toronto, Canada: Systems Control Group, Department of Electrical & Computer Engineering, University of Toronto, 2014. Citado 3 vezes nas páginas 21, 32 e 33.

WONHAM, W. M.; RAMADGE, P. J. Modular supervisory control of discrete-event systems. *Mathematics of control, signals and systems*, Springer, v. 1, n. 1, p. 13–30, 1988. Citado 4 vezes nas páginas 21, 32, 33 e 56.