Luiz Fernando Puttow Southier

Explorando Parâmetros na Modelagem, Síntese e Implementação de Controladores para Sistemas a Eventos Discretos

Pato Branco, PR, Brasil

2019

Luiz Fernando Puttow Southier

Explorando Parâmetros na Modelagem, Síntese e Implementação de Controladores para Sistemas a Eventos Discretos

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná – Câmpus Pato Branco como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Marcelo Teixeira

Coorientador: Prof. Dr. Marco Antônio de Castro Barbosa

Pato Branco, PR, Brasil
2019

S726e Southier, Luiz Fernando Puttow.
Explorando parâmetros na modelagem, síntese e implementação de controladores para sistemas a eventos discretos / Luiz Fernando Puttow Southier -- 2019.
62 f. : il. ; 30 cm

Orientador: Prof. Dr. Marcelo Teixeira
Coorientador: Marco Antonio de Castro Barbosa
Dissertação (Mestrado) - Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Engenharia Elétrica. Pato Branco, PR, 2019.
Bibliografia: f. 60-62

1. Automação. 2. Teoria dos autômatos 3. Sistemas de controle digital. I. Teixeira, Marcelo, orient. II. Barbosa, Marco Antonio de Castro, coorient. III. Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Engenharia Elétrica. IV. Título.

CDD 22. ed. 621.3

Ficha Catalográfica elaborada por
Rosana Silva CRB9/1745
Biblioteca da UTFPR Campus Pato Branco

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Pato Branco
Diretoria de Pesquisa e Pós-Graduação
Programa de Pós-Graduação em Engenharia Elétrica

TERMO DE APROVAÇÃO

Título da Dissertação n.° 072

"Explorando Parâmetros na Modelagem, Síntese e Implementação de Controladores para Sistemas a Eventos Discretos"

por

Luiz Fernando Puttow Southier

Dissertação apresentada às nove horas, do dia cinco de setembro de dois mil e dezenove, como requisito parcial para obtenção do título de MESTRE EM ENGENHARIA ELÉTRICA, do Programa de Pós-Graduação em Engenharia Elétrica – Universidade Tecnológica Federal do Paraná, *Campus* Pato Branco. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho APROVADO.

Banca examinadora:

Prof. Dr. Marcelo Teixeira
UTFPR/PB
(orientador)

Prof. Dr. Dalcimar Casanova
UTFPR/PR

Prof. Dr. César Rafael Claure Torrico
UTFPR/PB

Prof. Dr. Edson Emilio Scalabrin
(participação à distância)
PUC/CT

Homologado por:

Prof. Dr. Gustavo Weber Denardin
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica - PPGEE/UTFPR

A versão deste termo, devidamente assinada, encontra-se em arquivo na Coordenação do PPGEE - UTFPR – *Campus* Pato Branco.

Resumo

A *Teoria de Controle Supervisório* (TCS) estrutura formalmente a obtenção de controladores para *Sistemas a Eventos Discretos* (SEDs) com base na teoria dos *Autômatos Finitos Determinísticos* (AFDs) e *Linguagens*. Dentre os aspectos que dificultam a aplicação dessa teoria estão problemas de modelagem, síntese e implementação, pois AFDs são em geral limitados ao expressar características avançadas de SEDs, como reconhecimento e mudança de contexto. Nos últimos anos, inúmeras abordagens vêm sendo propostas na literatura como forma de lidar com esses problemas. Dentre elas, o uso de AFDs com parâmetros, ou seja, estruturas formais que partem de um modelo original, inicial, e incorporam a ele detalhes que permitem identificar e alterar contextos. Dispor de um modelo que reconhece e gerencia contexto pode agregar vantagens a diversas etapas do projeto de controladores para SEDs, dependendo do mecanismo utilizado para parametrização. Em geral, parâmetros podem estar associados tanto a estados quanto a transições de um AFD, mas cada abordagem é estruturada sob um formalismo específico, de forma tal que a comparação e integração entre elas não é direta. Assim, cada abordagem pode levar a diferentes soluções de controle, modeladas, computadas e implementadas usando estratégias distintas. Isso impossibilita que possíveis vantagens de cada uma possam ser combinadas em diferentes etapas do projeto de um controlador. Esse trabalho mostra como combinar vantagens do uso de parâmetros na modelagem, síntese e implementação de controladores para SEDs. Parte-se do pressuposto de que tarefas de modelagem são mais naturalmente conduzidas usando estados parametrizados, pois, nessa abordagem, fórmulas e variáveis facilitam implementar intuitivamente o mecanismo de identificação e gerenciamento dos contextos de um SED. No entanto, a parametrização de estados por variáveis não explora adequadamente aspectos de modularização, uma vez que, o domínio de uma variável é atômico e, quando combinado com outros domínios, de outras variáveis, pode levar à explosão do espaço de estados, inviabilizando o tratamento algorítmico e impedindo que vantagens de modelagem se propaguem para as etapas de síntese e implementação. Nesse sentido, apresenta-se um método de conversão que possibilita migrar do domínio de estados para o de eventos parametrizados, preservando essencialmente o mesmo comportamento. O resultado é um conjunto de AFDs modulares que desmembram a noção atômica de uma variável no modelo de um SED. Então, são discutidos os benefícios do uso de modelos com eventos parametrizados em estratégias eficientes de síntese e em estruturas descentralizadas de implementação presentes na literatura.

Palavras-chave: Sistema a Eventos Discretos. Teoria de Controle Supervisório. Parametrização. Autômatos Finitos Estendidos.

Abstract

The Supervisory Control Theory (SCT) formally structures the design of controllers for Discrete Event Systems (DESs) based on the Theory of Finite Deterministic Automata (FDA) and Languages. Among the aspects that limit the application of this theory are problems of modeling, synthesis and implementation, since FDAs are usually limited in expressing advanced features of DESs, such as recognition and change of contexts. In recent years, several approaches have been proposed in the literature as a way to deal with these problems. Among them, the use of FDA with parameters, that is, formal structures that depart from an original, initial model, and incorporate to it details that allow identifying and altering contexts. Having a model that recognizes and manages context can add advantages to several stages of the design of controllers for DESs, depending on the mechanism used for parameterization. In general, parameters may be associated to states or transitions of an FDA, but each approach is structured under a specific formalism, so that the comparison and integration among them is not straightforward. Thus, each approach can lead to different control solutions, modeled, computed and implemented using different strategies. This makes it impossible for potential advantages of each to be combined at different stages of controller design. This work shows how to combine the advantages of using parameters in the modeling, synthesis and implementation of controllers for DESs. It is assumed that modeling tasks are more naturally conducted using parameterized states because in this approach, formulas and variables facilitate the implementation of the mechanism for identifying and managing the contexts of a DES. However, the parameterization of states by variables does not adequately explore aspects of modularization, since the domain of a variable is atomic and, when combined with other domains, of other variables, can lead to the explosion of state space, therefore preventing modeling advantages from spreading to the synthesis and implementation steps. In this sense, a conversion method is presented that allows migrating from the domain of parameterized states to the domain of parameterized events, preserving essentially the same behavior. The result is a set of modular FDA that dismember the atomic notion of a variable in the model of a DES. Then, the benefits of using models with parameterized events in efficient synthesis strategies and decentralized structure are discussed.

Keywords:Discrete Event Systems. Supervisory Control Theory. Parametrization. Extended Finite-state Automata.

Lista de ilustrações

Lista de abreviaturas e siglas

SED	Sistema a Eventos Discretos
TCS	Teoria de Controle Supervisório
AFD	Autômato Finito Determinístico
AFE	Autômato Finito Estendido
AFEE	Autômato Finito Estendido na Forma Explícita

Lista de símbolos

Σ	Alfabeto
Q	Conjunto de Estados
q°	Estado Inicial
Q^{ω}	Conjunto de Estados Marcados
Γ	Relação de Transição
Σ^{*}	Conjunto de cadeias possíveis sobre o alfabeto Σ
ϵ	Cadeia composta por nenhum evento
$\mathcal{L}(A)$	Linguagem reconhecida pelo autômato A
$\mathcal{L}^{\omega}(A)$	Linguagem marcada do autômato A
$\|$	Composição Síncrona de AFDs
$A^{\|}$	Composição Síncrona de todos os AFDs no conjunto A
G	Modelo da Planta
E	Modelo das Especificações
K	Modelo da Planta restringido pelas Especificações
Σ_{c}	Conjunto de eventos controláveis
Σ_{u}	Conjunto de eventos não controláveis
S	Supervisor calculado pela síntese com base em K
$\sup\mathcal{C}(K,G)$	Linguagem de S
$\mathfrak{E}$	Abordagem com estados parametrizados
V	Conjunto de variáveis
P_V	Conjunto de fórmulas
$Dom(v)$	Domínio da variável v
v^{o}	Valor inicial da variável v

$\bar{v}$	Valor assumido pela variável v
$'$	Elemento no próximo estado
$\mathcal{V}_p$	Conjunto de variáveis de estado atual utilizadas por p
$\mathcal{V}'_p$	Conjunto de variáveis de próximo estado utilizadas por p
$\hat{v}_p$	Valoração das variáveis em $\mathcal{V}_p$
P_{Vt}	Conjunto de fórmulas de teste
P_{Va}	Conjunto de fórmulas de atualização
$\hat{v}$	Tupla com a valoração de todas as variáveis no conjunto V
$G_{\mathfrak{E}}$	Modelo da Planta com estados parametrizados
$E_{\mathfrak{E}}$	Modelo das Especificações com estados parametrizados
$K_{\mathfrak{E}}$	Modelo de entrada da síntese com estados parametrizados
$S_{\mathfrak{E}}$	Supervisor calculado pela síntese com base em $K_{\mathfrak{E}}$
$A^{\bullet}_{\mathfrak{E}}$	AFE $A_{\mathfrak{E}}$ em sua forma explícita
$\mathfrak{D}$	Modelagem com eventos parametrizados
Δ	Alfabeto parametrizado
Δ^{σ}	Subconjunto do Alfabeto parametrizado contendo as parametrizações do evento σ
Δ^{*}	Conjunto de cadeias possíveis sobre o alfabeto Δ
Π	Mapa mascarador para eventos parametrizados
$G_{\mathfrak{D}}$	Modelo da Planta com eventos parametrizados
$H_{\mathfrak{D}}$	Filtro para modelos com eventos parametrizados
$E_{\mathfrak{D}}$	Modelo das Especificações com eventos parametrizados
$K_{\mathfrak{D}}$	Modelo de entrada da síntese com eventos parametrizados
$S_{\mathfrak{D}}$	Supervisor calculado pela síntese com base em $K_{\mathfrak{D}}$
$\mathbb{V}^{\sigma}$	Conjunto de variáveis envolvidas na representação de contextos de σ
H_V	Conjunto de filtros não modulares que implementam a alternância de contextos das variáveis em V

H	Conjunto de filtros modulares que implementam a alternância de contexto
C	Conjunto de j pares $\{u_{ij}, c_{ij}\}$
u_{ij}	Conjunto unitário representando um estado $q_{u_{ij}}$
c_{ij}	Conjunto complementar de u_{ij} representando um superestado $q_{c_{ij}}$
$G_{\mathfrak{A}}$	Aproximação para $G_{\mathfrak{D}} \| H_{\mathfrak{D}}$
$H_{\mathfrak{A}}$	Parte do filtro selecionada para a síntese eficiente
$G_{\mathfrak{A}}$‘	Aproximação eficiente de $G_{\mathfrak{D}} \| H_{\mathfrak{A}}$
$K_{\mathfrak{A}}$‘	Modelo de entrada da síntese eficiente tal que $K_{\mathfrak{A}}$‘ $= G_{\mathfrak{A}}$‘$\| E_{\mathfrak{D}}$

Sumário

1 Introdução

A demanda por sistemas de automação industrial modernos, avançados e eficientes tem aumentado nos últimos anos, exigindo principalmente equipamentos de controle que sejam confiáveis, possam ser reconfigurados e personalizados conforme a necessidade, e consigam maximizar a produção. De todo modo, a ideia central desses sistemas é transformar matéria-prima em produtos por meio da integração eficiente de pessoas, equipamentos, tecnologia e recursos (ESMAEILIAN; BEHDAD; WANG, 2016). Espera-se que componentes fabris interajam entre si e com o ambiente externo concorrentemente, compartilhando recursos e respeitando restrições operacionais de maneira segura, controlável e maximamente permissiva. Em contrapartida, essas características complexificam a programação de controladores industriais, pois, além dessa ser uma tarefa manual, ela depende de diversos fatores, como do número de componentes físicos a serem programados e coordenados, do tamanho, da capacidade, da complexidade do *workflow* dos processos, etc.

A literatura apresenta opções para conceber tais controladores. Dentre elas, a programação dinâmica e concorrente (CHIEN et al., 2011), o desenvolvimento baseado em interfaces (JHA; NAIR, 2012), o uso de abordagens de modelagem e programação incremental (MOHAJERANI; MALIK; FABIAN, 2017a), de linguagens específicas para a programação de controladores lógicos (como *ladder, grafcet, structured text, etc.*) (QAMSANE; ABDELOUAHED; PHILIPPOT, 2016), linguagens para a coordenação de múltiplos robôs (TSAI, 1991), abordagens bio-inspiradas (CHEN; HWANG; JIANG, 2013), entre outras.

A escolha por uma dessas diferentes abordagens de desenvolvimento passa, sobretudo, pela correta caracterização do sistema que se está tratando. Pode-se observar o sistema como uma entidade que evolui continuamente no tempo, incorporando grandezas físicas como temperatura, velocidade, posição de componentes, por exemplo, ao longo dessa evolução. Um sistema pode também ser observado em relação à sua evolução por sinais (eventos) esporádicos e independentes no tempo.

Nesse último caso, ou seja, quando o sistema evolui pela ocorrência de eventos, ele é definido como um Sistema a Eventos Discretos (SED) (CASSANDRAS; LAFORTUNE, 2009). Se assume que eventos ocorrem de maneira espontânea e esporádica no sistema, e que eles devem ser coordenados externamente por um hardware dedicado e propriamente programado, como por exemplo um microcontrolador. Para esse tipo de sistema, o objetivo de controle é desabilitar certas sequências de eventos possíveis. Sob essa perspectiva, uma abordagem formal que suporta o cálculo automático de controladores, é a *Teoria de Controle Supervisório* (TCS) de (RAMADGE; WONHAM, 1989).

A TCS utiliza *Autômatos Finitos Determinísticos* (AFDs) para modelar o comportamento do sistema, impor restrições e calcular automaticamente o controlador, pois estes modelam a natureza do comportamento que se quer representar: eventos que fazem com que o sistema mude de estado. Apesar de sua relevância teórico-prática, o uso de autômatos ordinários enfrenta limitações ao lidar com sistemas de escala industrial, por exemplo. A maioria dessas limitações está relacionada ao tamanho dos modelos desses sistemas, que implica em aspectos de complexidade. Dentre eles estão a complexidade computacional, relativa ao tempo e à memória necessários para processar modelos grandes nas etapas de síntese e implementação; e a complexidade de engenharia, ou seja, a dificuldade da modelagem envolvendo estruturas extensas, com muitas e complexas combinações de eventos e estados. Juntos, esses aspectos fomentam inúmeras iniciativas na literatura, como a ideia de *parametrizar* modelos, ou seja, associar mais informação aos AFDs com o intuito de tornar a TCS aplicável à resolução de um conjunto maior de problemas reais (WARE; SU, 2017; TEIXEIRA et al., 2015; CURY et al., 2015; SILVA; RIBEIRO; TEIXEIRA, 2017; MOHAJERANI; MALIK; FABIAN, 2017a; MALIK et al., 2017; CLAESSEN et al., 2018).

Basicamente, parâmetros são informações que enriquecem modelos de SEDs. Eles podem ser associados, por meio de abordagens distintas, tanto a estados quanto a eventos. Cada abordagem de parametrização apresenta vantagens específicas, dependendo da aplicação e da etapa de projeto do controlador. Por exemplo, parâmetros associados a eventos têm a característica de aumentar o grau de comunicação do sistema de controle, mas reduzir a memória necessária para armazenar estados do sistema. Em contrapartida, parâmetros associados a estados reduzem a necessidade de comunicação entre as unidades de controle, mas tendem a aumentar a memória necessária para implementar a solução.

No entanto, cada uma delas possui seu próprio *framework*, o que resulta em uma estratégia distinta para a modelagem do problema, para a síntese do controlador e para a implementação. Assim, em conjunção, elas não combinam essas vantagens, e a literatura não oferece um mecanismo de conversão entre os modelos para que se possa tirar proveito de suas melhores características nas diferentes fases.

Essa dissertação inicialmente descreve as abordagens de parametrização por estados e por eventos. Então, o uso de cada abordagem é avaliado e comparado em diferentes fases do projeto de controladores para SEDs e suas vantagens são evidenciadas. Com base nessa análise, propõe-se um mecanismo de conversão entre modelos que permite extrair e combinar vantagens das abordagens com parâmetros, no escopo de um mesmo projeto de controlador (SOUTHIER; TEIXEIRA, 2019). O resultado desse método de conversão leva a um conjunto de AFDs modulares, que podem então ser fornecidos como entrada para algoritmos de síntese, para o cálculo do sistema de controle em si e subsequente implementação. Nesse trabalho, a síntese é conduzida usando-se o algoritmo proposto por

Cury et al. (2015) que utiliza abstração de modelos, o que retorna vantagens de síntese, em conjunção com o método de Rosa, Teixeira e Malik (2018) que leva ao resultado ótimo. Ainda, realiza-se a análise dos resultados da síntese com abstrações a partir de modelos convertidos pelo método proposto, e a exploração de como o resultado de síntese pode ser implementado em uma estrutura descentralizada herdada de Rosa et al. (2017). Assim, percorrem-se todas as etapas, desde a modelagem até a implementação de controladores para SEDs, mostrando-se como as vantagens do uso de parâmetros podem ser exploradas em cada fase sem onerar os demais aspectos do projeto.

De modo geral, como síntese dos resultados dessa dissertação pretende-se ampliar o potencial da TCS para ser aplicada em escala industrial. Alguns esforços nesse sentido têm sido reportados na literatura, especialmente no setor automotivo, que vem adotando tecnologia embarcada com técnicas de TCS e suas extensões para o tratamento de aspectos como segurança, tolerância a faltas, verificação, flexibilização, mitigação de erros, performance, etc. (KORSSEN et al., 2018; DAVYDOV; LARIONOV; NAGUL, 2018; CAMPOS; ROSSA; COLOMBO, 2018; CLAESSEN et al., 2018). Entende-se que o método de conversão proposto seja diretamente integrável a essas iniciativas com potencial impactos positivos em termos de redução de complexidade do projeto de controladores, o que o coloca em melhor sintonia com a característica embarcada desses dispositivos.

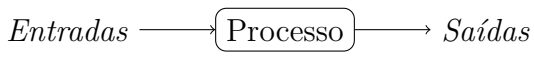
Estruturalmente, esse documento é assim organizado: o Capítulo 2 resgata conceitos de sistemas e controle, bem como da modelagem de SEDs por meio de AFDs e o uso de tais modelos para cálculo do controlador pela TCS; o Capítulo 3 apresenta a ideia de parametrização explorando como se dá o seu uso nas etapas de modelagem e síntese do controlador, tanto para eventos, quanto para estados parametrizados, e as compara com a abordagem não parametrizada; o Capítulo 4 propõe o mecanismo de conversão dos modelos e apresenta resultados das conversões aplicadas a problemas da literatura; o Capítulo 5 descreve como os modelos convertidos se integram nas etapas de síntese e implementação eficiente.

2 Modelagem e controle de sistemas

Um sistema é um conjunto de componentes que trabalham para cumprir um objetivo específico (CASSANDRAS; LAFORTUNE, 2009). Exemplos de sistemas incluem o sistema nervoso de um ser humano, que é composto por partes menores como cérebro, nervos e medula; o Sistema Solar, que inclui planetas, satélites, o Sol e outros corpos; e uma linha de produção em uma fábrica, a qual possui esteiras, braços robóticos, sensores, entre outros.

Analisando o comportamento de um dado sistema, é possível criar um modelo que sintetiza matematicamente as principais informações sobre como o sistema age e interage com componentes externos. Na área de controle, os modelos de sistemas são encarados como um processo que recebe algumas informações, entradas, e fornece outras, saídas (NISE, 2011). O modelo de um sistema por si só, sem nenhuma restrição ou ação de controle, é conhecido como *planta*. A Figura 1 apresenta uma planta genérica.

Figura 1 – Planta



Fonte: Dorf e Bishop (2001)

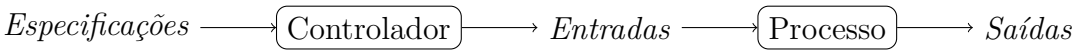
Um controlador é um dispositivo utilizado para alterar as entradas de um processo, agindo indiretamente em suas saídas. Controladores são utilizados para restringir o comportamento da planta de modo que suas saídas estejam de acordo com parâmetros especificados. Dessa maneira *especificações* de controle ou *restrições* de comportamento são utilizados no projeto de controladores. Quando as informações de saída de um processo são fornecidas ao controlador, diz-se que o sistema em questão está em *malha fechada*, caso contrário, diz-se que o que sistema está em *malha aberta*. Dorf e Bishop (2001) apresentam a Figura 2 que ilustra os conceitos apresentados.

Os sistemas, e seus respectivos controladores, podem ser classificados de acordo com a Figura 3. Essas classificações não são mutuamente exclusivas, ainda assim servem ao propósito de descrever o escopo de diferentes aspectos da *Teoria de Sistemas e Controle* (OGATA, 1996; CASSANDRAS; LAFORTUNE, 2009):

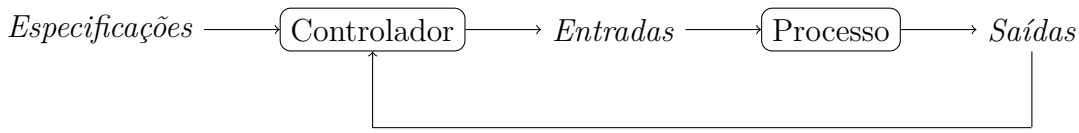
- A saída de sistemas estáticos é independente de valores passados da entrada. A saída de sistemas dinâmicos depende dos valores anteriores da entrada;
- Sistemas invariantes no tempo apresentam sempre a mesma saída para uma entrada específica independente do momento no tempo em que ela ocorre;

Figura 2 – Controle em Malha aberta e em Malha fechada

Malha aberta:

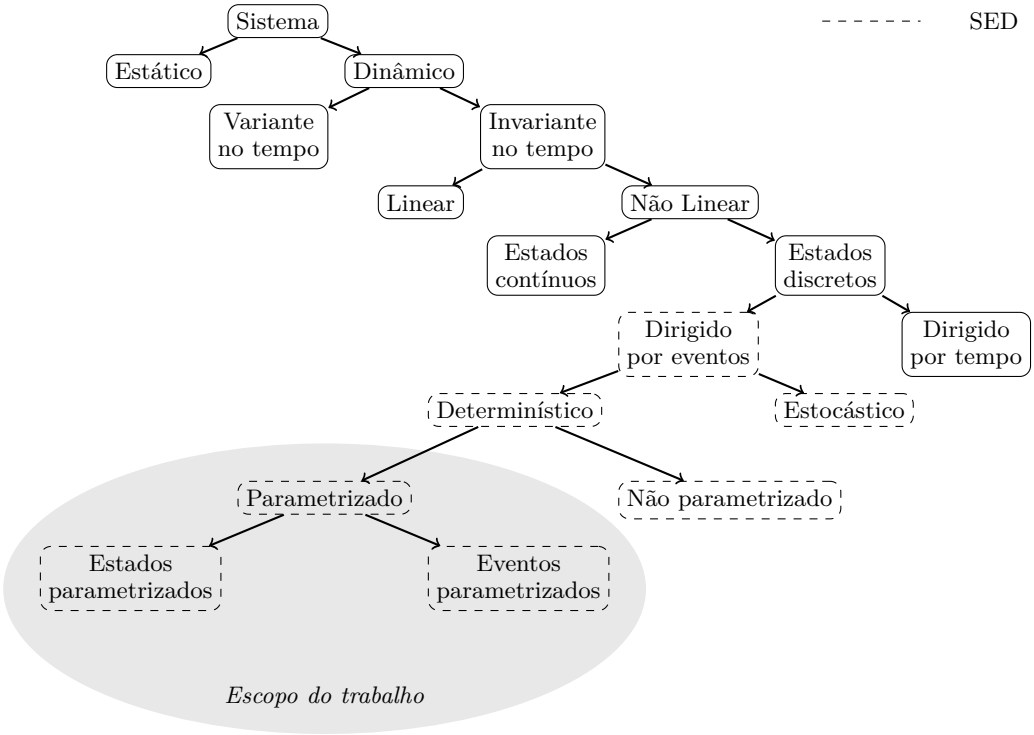


Malha fechada:



Fonte: Dorf e Bishop (2001)

Figura 3 – Classificação de Sistemas e Controladores



Fonte: Adaptado de Cassandras e Lafortune (2009)

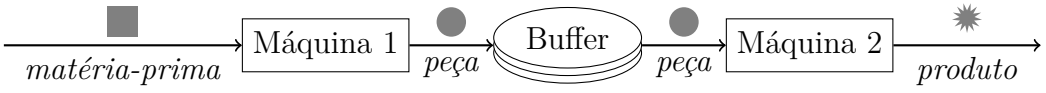
- Considerando entradas quaisquer i_1 e i_2 e suas respectivas saídas o_1 e o_2, um sistema é linear se para uma entrada combinada $i_1 + i_2$ apresentar a saída $o_1 + o_2$;
- As entradas e saídas de sistemas de estado contínuo podem assumir quaisquer valores reais (ou complexos, dependendo do sistema). Em sistemas de estado discreto, entradas e saídas assumem valores em um conjunto discreto, por exemplo, valores inteiros não negativos;

- Em sistemas dirigidos pelo tempo, o estado do sistema se altera continuamente conforme o tempo muda. Em sistemas dirigidos por eventos, é apenas a ocorrência de eventos discretos gerados de forma assíncrona que força transições instantâneas de estado. Entre ocorrências de eventos, o estado permanece inalterado. Esses sistemas são conhecidos como SEDs (RAMADGE; WONHAM, 1987);
- SEDs são estocásticos sempre que uma ou mais de suas variáveis de saída são aleatórias. Nesse caso, o estado do sistema é descrito por um processo estocástico e uma estrutura probabilística é necessária para caracterizar o comportamento do sistema. Nesse trabalho, consideram-se apenas SEDs determinísticos;
- SEDs parametrizados incorporam informações adicionais, parâmetros, aos estados ou eventos modelados e suas características serão exploradas no próximo capítulo.

2.1 Exemplo de SED

O SED mostrado na Figura 4 é composto por duas máquinas interconectadas por um *buffer*. A máquina um (m_1) recebe matéria-prima e produz uma peça de um produto específico. As peças produzidas por ela são armazenadas em um *buffer* (b_1) que possui capacidade máxima de 3 unidades. A máquina dois (m_2) retira uma peça por vez do *buffer* e realiza um processo de acabamento construindo assim um produto.

Figura 4 – Exemplo de SED



Algumas restrições no comportamento desse sistema podem ser impostas:

- m_1 não pode colocar peças no *buffer* caso sua capacidade máxima já tenha sido alcançada, isto é, deve se evitar o *overflow*;
- m_2 não pode retirar peças do *buffer* caso ele esteja vazio, isto é, não se pode ocorrer *underflow*.

Observa-se que, tanto no comportamento intrínseco do sistema, planta, quanto nas restrições apresentadas, não se faz menção ao tempo. Nesse exemplo, é a ordem de ocorrência dos eventos (chegar insumo, colocar peça no *buffer*, retirar peça do *buffer*, finalizar produto) que fazem com que o sistema evolua em seu espaço de estados. Dessa maneira, a modelagem de SEDs precisa ser definida baseada nesse tipo de comportamento dirigido por eventos.

2.2 Modelagem e Controle de SEDs

Enquanto um sistema dirigido pelo tempo pode ser descrito por equações diferenciais, para Cassandras e Lafortune (2009) os SEDs são mais naturalmente modelados por estruturas de diagramação, como AFDs, por exemplo. Hopcroft, Motwani e Ullman (1939) definem um AFD como sendo a quíntupla $A = (\Sigma, Q, q^\circ, Q^\omega, \Gamma)$, em que:

- Σ é um conjunto de símbolos de entrada chamado de *alfabeto*. Cada símbolo representa um evento passível de ocorrer no SED a ser modelado;
- Q é um *conjunto finito de estados*. Cada componente desse conjunto representa um estado que o sistema pode atingir conforme os eventos ocorrem;
- $q^\circ \in Q$ é o *estado inicial*. Todo sistema possui um estado anterior a qualquer ocorrência de evento;
- $Q^\omega \subseteq Q$ é o *conjunto de estados finais ou marcados*. São estados específicos do sistema que representam a finalização de uma dada tarefa;
- $\Gamma \subseteq Q \times \Sigma \times Q$ é uma *relação de transição*.

Um AFD pode ser representado por meio de um grafo direcionado. Os estados do sistema são representados pelos nós desse grafo. As transições com eventos são representadas por arcos entre os nós do grafo. O estado inicial é identificado por uma seta indicativa e os estados finais são ilustrados com borda dupla. Considerando estados q_0 e q_1, e um evento σ, neste trabalho a notação $q_0 \xrightarrow{\sigma} q_1$, indica que Γ possui um evento σ partindo do estado q_0 para o estado q_1, que também pode ser denotado por $\Gamma(q_0, \sigma) = q_1$ (CASSANDRAS; LAFORTUNE, 2009). A notação $q_0 \xrightarrow{\sigma}$ significa $q_0 \xrightarrow{\sigma} q_1$ para algum estado $q_1 \in Q$.

Uma sequência de eventos é chamada de *cadeia*, e o conjunto de todas as cadeias possíveis de serem formadas com os eventos de Σ é denotada por Σ^*. Uma cadeia composta por nenhum evento é representada pelo símbolo ϵ. Duas cadeias $s, t \in \Sigma^*$ podem ser concatenadas como st. A notação de transições pode ser estendida para uma cadeia $s \in \Sigma^*$ por $q_0 \xrightarrow{s} q_1$

A partir de Σ^*, podem ser definidos subconjuntos, denominados *linguagem* ($\mathcal{L}$), que incorporam cadeias reconhecidas em determinado contexto. Ou seja, para um dado alfabeto Σ, $\mathcal{L} \subseteq \Sigma^*$, caracteriza $\mathcal{L}$ como uma linguagem sobre Σ cujas cadeias apresentam algum significado. Um AFD pode ser utilizado para decidir se uma cadeia é aceita ou não, ou seja, se pertence ou não a uma linguagem. Ainda, o prefixo fechamento de uma linguagem $\mathcal{L} \in \Sigma^*$ é definido por $\overline{\mathcal{L}} = \{s \in \Sigma^* | st \in \mathcal{L} \text{ para alguma cadeia } t \in \Sigma^*\}$.

A linguagem de um AFD qualquer A é o conjunto de todas as cadeias reconhecidas por ele e é denotada por $\mathcal{L}(A)$ (HOPCROFT; MOTWANI; ULLMAN, 1939). Na prática,

linguagens reconhecidas por autômatos são adequadas para processamento computacional, e são chamadas de *linguagens regulares* (CASSANDRAS; LAFORTUNE, 2009). O conjunto de cadeias reconhecidas por um autômato A, e que acabam em um estado marcado, é chamado *Linguagem Marcada* e é denotada $\mathcal{L}^{\omega}(A)$. Neste trabalho, uma cadeia s reconhecida por um autômato A é denotada por $A \xrightarrow{s}$ significando $q^{\circ} \xrightarrow{s} q$ para algum estado $q \in Q$. Ainda, $A \rightarrow q$ significa $q^{\circ} \xrightarrow{s} q$ para alguma cadeia $s \in \Sigma^{*}$.

O comportamento de dois AFDs quaisquer, A_1 e A_2 por exemplo, pode ser combinado por meio da operação de *Composição Síncrona* ($\|$) que gera um novo autômato $A = A_1 \,\|\, A_2$. Com base no estado corrente de A_1 e A_2, se existirem transições com o mesmo evento em ambos os autômatos, estas são realizadas ao mesmo tempo. Caso contrário, o autômato que não possui a transição fica no mesmo estado, enquanto o que possui, executa-a. Caso um autômato possua um evento bloqueado em determinado estado, esse bloqueio será estendido ao segundo autômato.

Formalmente, Cassandras e Lafortune (2009) definem a composição síncrona entre os autômatos $A_1 = (\Sigma_1, Q_1, q_1^{\circ}, Q_1^{\omega}, \Gamma_1)$ e $A_2 = (\Sigma_2, Q_2, q_2^{\circ}, Q_2^{\omega}, \Gamma_2)$ como:

$$A_1 \| A_2 = \left(\Sigma_1 \cup \Sigma_2, Q_1 \times Q_2, (q_1^{\circ}, q_2^{\circ}), Q_1^{\omega} \times Q_2^{\omega}, \Gamma_{1\|2}\right), \text{ tal que}$$

$$\Gamma_{1\|2}((q_1, q_2), \sigma) = \begin{cases} (\Gamma_1(q_1, \sigma), \Gamma_2(q_2, \sigma)) & \text{se } \exists\ \Gamma_1(q_1, \sigma) \text{ e } \exists\ \Gamma_2(q_2, \sigma) \\ (\Gamma_1(q_1, \sigma), q_2) & \text{se } \exists\ \Gamma_1(q_1, \sigma) \text{ e } \nexists\ \Gamma_2(q_2, \sigma) \\ (q_1, \Gamma_2(q_2, \sigma)) & \text{se } \nexists\ \Gamma_1(q_1, \sigma) \text{ e } \exists\ \Gamma_2(q_2, \sigma) \\ \text{indefinido} & \text{se } \nexists\ \Gamma_1(q_1, \sigma) \text{ e } \nexists\ \Gamma_2(q_2, \sigma) \end{cases} \tag{2.1}$$

A notação $A^{\|}$ indica a composição de um conjunto $A = \{A_1, ...A_n\}$ de AFDs, isto é, $A^{\|} = A_1 \| ... \| A_n$.

Na modelagem de SEDs, se utiliza AFDs para modelar o comportamento do sistema sem restrições, planta G, por meio de um conjunto de modelos, tal que $G = G_1 \| G_2 \| \ldots \| G_n$. Cada modelo G_i nesse grupo modela um componente da planta. A fim de que sobre a planta sejam impostas restrições é necessário que se modele um conjunto de especificações, tal que $E = E_1 \| \ldots \| E_m$. Por fim, denota-se por $K = G \,\|\, E$ o modelo da planta restringido pelas especificações. Ainda, no contexto de modelagem de SEDs, $\mathcal{L}$ (G) modela todas as possíveis cadeias a serem executadas sobre a planta, ou seja as partes de tarefas que podem ser executadas pela planta, e $\mathcal{L}^{\omega}$ (G) está associada a ideia de tarefas que foram completadas.

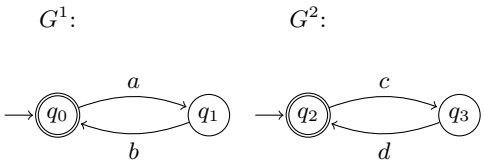
2.2.1 Exemplo de modelagem e controle de um SED

Considerando o exemplo apresentado na Figura 4, é possível construir AFDs que modelem o comportamento dos componentes da planta do sistema, bem como das restrições propostas. Para o ambiente de fábrica, os eventos são $\Sigma = \{a, b, c, d\}$, tal que a representa

a chegada de um insumo para m_1, b a saída de uma peça de m_1 e colocação desta no *buffer*, c retirada de uma peça do *buffer* e colocação em m_2, e d a saída de um produto de m_2.

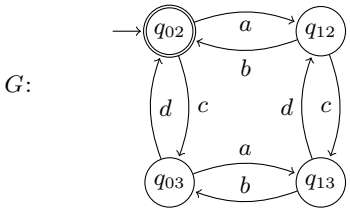
Os estados iniciais q_0 e q_2 representam respectivamente as máquinas m_1 e m_2 em ociosidade, e os estados q_1 e q_3, as máquinas em funcionamento. O comportamento dessas máquinas é modelado pelos AFDs G_1 e G_2 na Figura 5.

Figura 5 – Componentes da Planta - Modelo não parametrizado



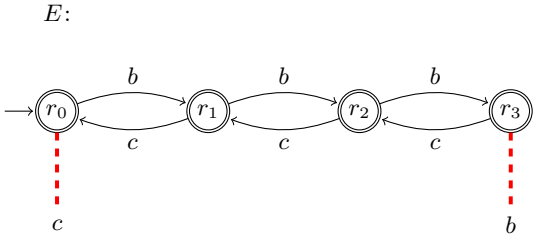
Os modelos de cada máquina, G_1 e G_2, podem ser sincronizados para obtenção do modelo da planta do sistema $G = G_1 \parallel G_2$, que é apresentado na Figura 6.

Figura 6 – Planta - Modelo não parametrizado



Considerando apenas os modelos das Figuras 5 ou 6, nota-se que seria possível ocorrer um evento c e depois um evento a, por exemplo. Na modelagem dos eventos, retirada de uma peça do *buffer* (c), não pode ocorrer antes da chegada de um insumo para m_1 (a). Para que as restrições possam ser impostas, um modelo adicional precisa ser criado. Tal modelo precisa incorporar a ordem correta de funcionamento das máquinas, bem como evitar *underflow* e *overflow* do *buffer*. A Figura 7 apresenta esse modelo. Observa-se que nos estados r_0 e r_3 os eventos c e b não podem ocorrer respectivamente (representação gráfica por meio do tracejado).

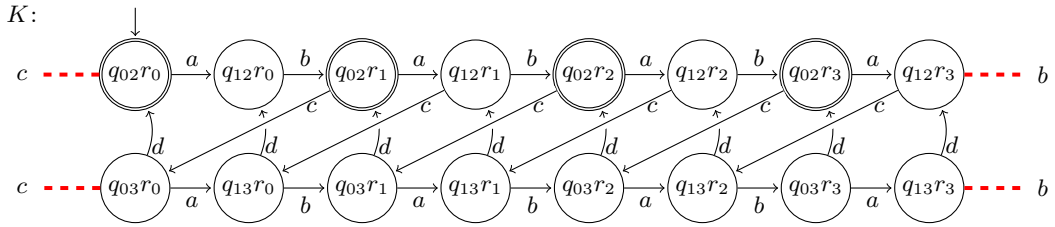
Figura 7 – Especificações - Modelo não parametrizado



Os estados de E, de r_0 até r_3, representam respectivamente a quantidade de peças no *buffer*, de zero a 3. O comportamento combinado $K = E \parallel G_1 \parallel G_2$ modela o sistema para que este aja de acordo com as especificações propostas. O modelo K é apresentado

na Figura 8. Observa-se que agora em K não é mais possível ocorrer um evento c e depois um evento a, por exemplo, pois bloqueio do evento c no estado inicial do modelo E foi sincronizado com o comportamento da planta em G, no modelo K.

Figura 8 – Modelo da planta restringido pelas especificações



Considerando o modelo K obtido, assume-se que todos os eventos possíveis no sistema podem ser controlados por uma entidade externa (um controlador automático). Porém, uma questão importante no controle de SEDs é relacionada à habilidade para lidar com a impossibilidade de desabilitar certos eventos do sistema, ou seja, eventos que, na prática, apresentam a característica de não poderem ser controlados diretamente. Não considerar essa impossibilidade é, em geral, associado à ideia de inconsistência de controle, ou seja, o controle automático comanda uma ação que efetivamente não pode ser imposta sobre o sistema físico.

No exemplo proposto, os eventos b e d que acontecem quando as máquinas finalizam seus respectivos processos são eventos não controláveis, e não podem ser desabilitados diretamente. Uma abordagem formal que viabiliza o tratamento correto e seguro de eventos não controláveis é a TCS, apresentada na sequência.

2.3 Teoria de Controle Supervisório

A *Teoria de Controle Supervisório* (TCS) define um método para síntese do controlador com base no modelo da planta G e das especificações E de um SED. Para isso, particiona-se o conjunto de eventos $\Sigma = \Sigma_c \cup \Sigma_u$, tal que Σ_c é o conjunto de eventos controláveis, isto é, aqueles que podem ser desabilitados e normalmente são os sinais que saem do controlador, e Σ_u é o conjunto de eventos não controláveis, que não podem ser desabilitados diretamente e correspondem aos sinais de sensores e informações de entrada ao controlador (RAMADGE; WONHAM, 1987; RAMADGE; WONHAM, 1989).

Com base nesse particionamento, pode-se calcular o controlador utilizando a TCS. Esse cálculo utiliza os conceitos de linguagem *controlável* e *não bloqueante*:

Definição 1 *Uma linguagem qualquer $\mathcal{L} \subseteq \Sigma^*$ é controlável em relação a uma linguagem L se $\overline{\mathcal{L}}\Sigma_u \cap L \subseteq \overline{\mathcal{L}}$ (RAMADGE; WONHAM, 1987; RAMADGE; WONHAM, 1989).*

Ou seja, após qualquer prefixo de $\mathcal{L}$, se um evento não controlável é possível em $\mathcal{L}$ e também em L, a cadeia resultante continua em $\overline{\mathcal{L}}$.

Definição 2 *Uma linguagem $\mathcal{L}$ é dita não bloqueante se $\mathcal{L} = \overline{\mathcal{L}^{\omega}}$ (RAMADGE; WONHAM, 1989).*

Ou seja, uma linguagem $\mathcal{L}(A)$ reconhecida por um autômato A qualquer é não bloqueante se todos os estados desse autômato são alcançáveis partindo do estado inicial e, a partir de qualquer estado, é possível se alcançar um estado marcado.

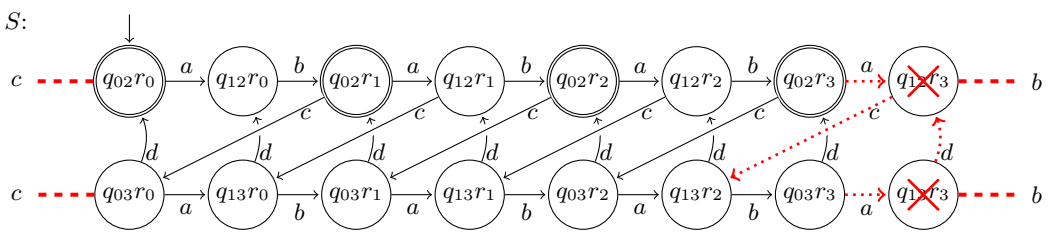
Com base no particionamento do conjunto de eventos, Ramadge e Wonham (1987) definem um algoritmo para síntese do controlador que tem como entrada o autômato K, tal que $K = G \| E$, e como saída o autômato S. O autômato S é não bloqueante e a linguagem reconhecida por ele, $\mathcal{L}(S)$, é controlável em relação à linguagem da planta $\mathcal{L}(G)$. Assim garante-se que nenhum evento não controlável é desabilitado diretamente pelo autômato S.

Ainda, o algoritmo de síntese calcula o autômato S cuja linguagem, denotada $\sup\mathcal{C}(K, G)$, é *maximamente permissiva*, ou seja, o comportamento de $\sup\mathcal{C}(K, G)$ é o mais próximo possível do comportamento de K, além de ser controlável e de não permitir o bloqueio. Isso significa que o algoritmo de síntese remove apenas os estados e transições de K necessários para garantir as condições de controlabilidade e não bloqueio, mantendo o máximo de estados e transições possível (RAMADGE; WONHAM, 1987; RAMADGE; WONHAM, 1989). $\sup\mathcal{C}(K, G)$ é dita a solução ótima de controle.

2.3.1 Exemplo de síntese do controlador

Considerando o exemplo apresentado ao longo do capítulo e o modelo K (Figura 8) de entrada do processo de síntese, é possível calcular S. No modelo K, se observa que o evento não controlável b é desabilitado em dois estados, desabilitação essa que foi herdada da especificação E. Como o controlador não possuí a capacidade de desabilitar b por sua característica não controlável, a operação de síntese remove os estados $q_{12}r_3$ e $q_{13}r_3$ e suas respectivas transições (representadas em tracejado) no modelo S da Figura 9.

Figura 9 – Modelo do controlador

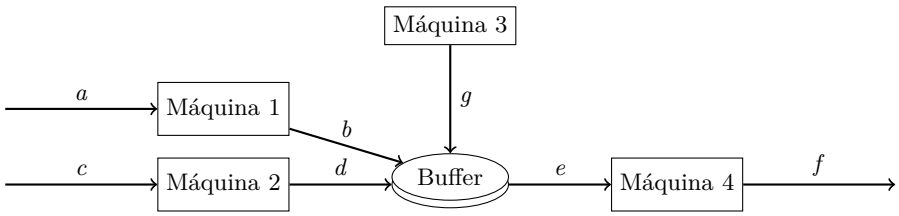


A operação de síntese retorna um modelo controlável e, como apenas os estados necessários foram removidos, então ele também é maximamente permissivo para esse conjunto de especificações. Pode-se mostrar que esse resultado é também não bloqueante, então S modelando sup$\mathcal{C}(K, G)$ é um resultado ótimo de controle. Os próximos capítulos desse trabalho exploram outras características e extensões associadas à modelagem e síntese de controladores conforme a TCS não parametrizada, aqui apresentada. A seção seguinte explora um exemplo adicional que possui características particulares a serem exploradas nos métodos de conversão nos próximos capítulos.

2.4 Exemplo de sistema com pareamento de peças

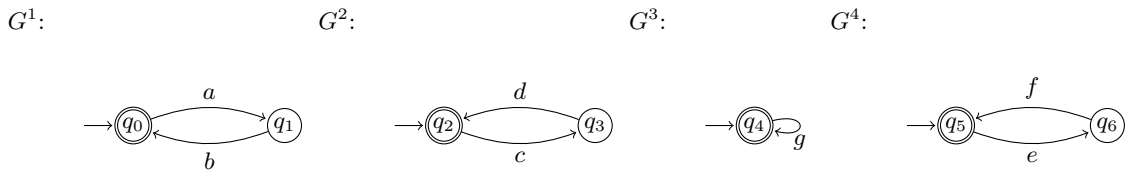
Considere um SED composto por quatro máquinas interconectadas por um *buffer* intermediário ilustradas na Figura 10. As máquinas 1 e 2 recebem um insumo de produção (eventos a e c), manufaturam peças do tipo B e do tipo D, e colocam essas peças em um *buffer* (eventos b e d) em ordem arbitrária. A máquina 3 tem a função de emparelhar conjuntos de duas peças do tipo D no *buffer* (evento g), e é desabilitada para outras combinações de tipos de peças. Por fim, a máquina 4 retira todas as peças do *buffer* (evento e), as embala e as remove do sistema (evento f).

Figura 10 – Sistema de manufatura com *buffer* intermediário e pareamento de peças



As máquinas 1, 2, 3 e 4 podem ser respectivamente modeladas pelos autômatos G^1, G^2, G^3 e G^4 apresentados na Figura 11, assim o modelo da planta do sistema é dado pela composição $G = G^1 \parallel G^2 \parallel G^3 \parallel G^4$.

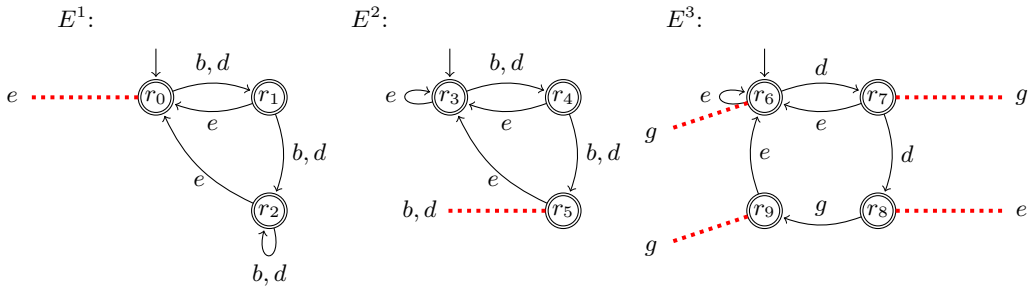
Figura 11 – Plantas - Sistema com pareamento de peças



É assumido que o *buffer* possui capacidade máxima de duas peças e que se quer evitar o seu *underflow* e *overflow*. Ainda, deseja-se que a máquina 4 possa retirar um par de peças do tipo D do *buffer* somente após elas terem sido emparelhadas pela máquina 3. Se o *buffer* não estiver cheio, ou contiver um par de peças do tipo B, ou uma peça do tipo

B e uma peça do tipo D, a máquina 4 deve estar habilitada e a máquina 3 desabilitada. As *especificações* $E = E^1 \,\|E^2\, \|E^3$, mostradas na Figura 12, podem ser criadas para garantir esses requisitos.

Figura 12 – Especificações - Sistema com pareamento de peças



O modelo E^1 desabilita o evento e (retirada do *buffer*) quando o *buffer* está vazio (estado r_0). Ainda, os eventos b e d (inserção de peças no *buffer*) são desabilitados por E^2 quando o *buffer* está cheio (estado r_5). E^3 desabilita e e habilita g quando o *buffer* possui exatamente um par de peças do tipo D (estado r_8), forçando neste caso a ocorrência de g antes da retirada do par de peças do *buffer* (evento e). A composição $K = E \,\|\, G$ modela o comportamento desejado de G sobre as regras impostas por E. O modelo K pode então ser utilizado como entrada para o algoritmo de síntese que tem como resultado modelo controlável, minimamente restritivo e não bloqueante S. A Tabela 1 mostra a quantidade de estados e transições (entre parênteses) destes modelos.

Tabela 1 – Quantidade de estados (e transições) para o exemplo adicional.

G	E	$K = E \,\|\, G$	S
8 (32)	7 (12)	56 (136)	28 (57)

Observa-se que, as especificações apresentadas no exemplo têm sua quantidade de estados vinculada à capacidade do *buffer* e, para um *buffer* com capacidade muito grande, exigiria-se a criação de um modelo com equivalente quantidade de estados. Essa característica é comum em AFDs, que utilizam a própria estrutura de transições e estados para memorizar informações e contextos. A próxima seção enfatiza essa limitação.

2.4.1 A limitação da TCS clássica

Apesar do papel reconhecido dos AFDs na modelagem de SEDs, eles enfrentam limitações quando aplicados a problemas reais de escala industrial. A literatura (MOHAJERANI; MALIK; FABIAN, 2017b; GOHARI; WONHAM, 2000; CURY et al., 2015; TEIXEIRA et al., 2015; ROSA; TEIXEIRA; MALIK, 2018; TEIXEIRA; CURY; QUEIROZ, 2018) mostra que processos comumente encontrados na indústria, como retrabalho, armazenamento, memória e produção paralela, podem requerer centenas de milhares de estados para representar todas as possibilidades com AFDs.

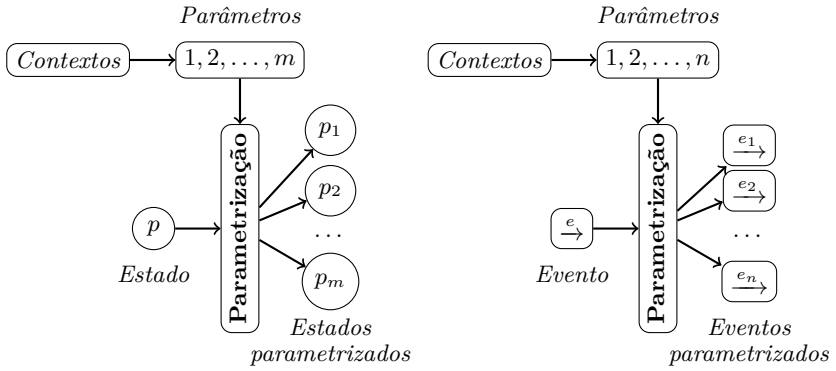
No exemplo da Figura 10, a ativação da máquina 3 (evento g) depende da memorização da quantidade de peças do tipo D colocadas no *buffer* (evento d). Para um *buffer* com capacidade n, e emparelhamento de n peças (evento g) o modelo exigiria $n + 2$ estados. Além disso, de maneira geral, dependendo das diferentes combinações a serem rastreadas no processo industrial, a especificação pode se tornar complexa de ser criada, ou muitas vezes, inviável.

Com base nessa característica limitante, a literatura (TEIXEIRA et al., 2015; CURY et al., 2015) propõe a utilização de modelos com mecanismos que capturam e carregam informações extras. Essas informações podem ser utilizadas para memorizar, identificar e isolar certos contextos. Os contextos são utilizados então para facilitar o processo de modelagem e síntese dos controladores. O próximo capítulo aborda o formalismo envolvido em modelagem e controle de SEDs com reconhecimento de contextos.

3 Modelagem e Controle com reconhecimento de contexto

Para atenuar os problemas de modelagem, principalmente de especificações complexas, a literatura propõe a utilização de modelos com mecanismos que capturam e carregam contextos (TEIXEIRA et al., 2015; CURY et al., 2015; SILVA; RIBEIRO; TEIXEIRA, 2017). Esses contextos são associados aos modelos por meio de *parâmetros*. Em geral, o processo de *parametrização* consiste em associar um componente do modelo (estado ou evento) em um conjunto de elementos que carregam o contexto desejado. Tal contexto é definido com base em parâmetros que são informados adicionalmente na modelagem. A Figura 13 exemplifica essa ideia e apresenta dois esquemas de parametrização. Na direita, o evento e foi mapeado nos eventos e_1, e_2 e e_n com base nos parâmetros fornecidos. Na esquerda, o mesmo ocorre, mas sobre o estado p que foi parametrizado nos estados p_1, p_2 e p_m.

Figura 13 – Processo ilustrativo de parametrização



As seções seguintes detalham e exemplificam a modelagem e controle com estados e eventos parametrizados.

3.1 Modelagem com estados parametrizados

A abordagem com estados parametrizados, com notação $\mathfrak{E}$, proporciona a criação e a filtragem de diferentes contextos por meio da utilização de *Autômatos Finitos Estendidos* (AFEs). Um AFE é similar a um AFD, mas suas transições incluem *fórmulas* que avaliam ou alteram os valores de *variáveis*. Formalmente, um AFE é definido como uma tupla $A_{\mathfrak{E}} = (\Sigma, V, Q, q^{\circ}, Q^{\omega}, P_V, \Gamma)$ (CHEN; LIN, 2001; CURY et al., 2015) em que:

- Σ é o conjunto de eventos;

- $V = \{v_0, \ldots, v_n\}$ é o conjunto de variáveis;
- Q é o conjunto de estados;
- $q^\circ \in Q$ é o estado inicial;
- $Q^\omega \subseteq Q$ é o subconjunto de estados marcados;
- $P_V = \{p_0, \ldots, p_m\}$ é o conjunto de fórmulas sobre V;
- $\Gamma \subseteq Q \times \Sigma \times P_V \times Q$ é a relação de transição.

A diferença de um AFE para um AFD está na inclusão dos conjuntos de variáveis V e de fórmulas lógicas P_V na tupla e na relação de transição. Em um AFE, a relação de transição avalia os valores correntes das variáveis em V e habilita ou não a ocorrência da transição de um estado para o outro. Essa avaliação se dá pelas fórmulas em P_V. Dessa maneira, além das transições carregarem eventos, carregam também fórmulas. Considerando estados q_0 e $q_1 \in Q$, um evento $\sigma \in \Sigma$, e uma fórmula $p \in P_V$, neste trabalho a notação $q_0 \xrightarrow{\sigma:p} q_1$, indica que Γ possui um evento σ partindo do estado q_0 para o estado q_1 que é habilitado pela fórmula p (CASSANDRAS; LAFORTUNE, 2009). Ainda, neste trabalho a notação $q_0 \xrightarrow{\sigma:p}$ significa $q_0 \xrightarrow{\sigma:p} q_1$ para um estado qualquer $q_1 \in Q$.

Uma variável v tem um domínio discreto e finito $Dom(v)$ e um valor inicial $v^o \in Dom(v)$. O valor assumido por uma variável v é denotado $\bar{v} \in Dom(v)$. $V = \{v_0, \ldots, v_n\}$ é o conjunto de todas as variáveis. Seu domínio é $Dom(V) = Dom(v_0) \times \cdots \times Dom(v_n)$.

Para diferenciar os valores assumidos pelas variáveis antes e depois da transição ocorrer, o conjunto de variáveis de *próximo estado* $V' = \{v'_1, \ldots, v'_n\}$, com $Dom(V) = Dom(V')$, é associado a V. Então, $\bar{v} \in Dom(v)$ e $\bar{v}' \in Dom(v')$ representam o valor assumido por uma variável v respectivamente no estado atual e no próximo estado.

Estruturalmente, cada fórmula $p \in P_V$ é construída usando um conjunto de *variáveis de estado atual*, denotado $\mathcal{V}_p$, e um conjunto de *variáveis de próximo estado*, $\mathcal{V}'_p$, tal que $\mathcal{V}_p \subseteq V$ e $\mathcal{V}'_p \subseteq V'$. A tupla com os valores das variáveis em $\mathcal{V}_p = \{v_n, \cdots, v_m\}$, isto é $\hat{v}_p = (\overline{v_n}, \cdots, \overline{v_m}) \in Dom(\mathcal{V}_p) = Dom(v_n) \times \cdots \times Dom(v_m)$, é chamada de *valoração de estado atual*. Do mesmo modo, a tupla $\hat{v}'_p = (\overline{v_i}', \cdots, \overline{v_j}') \in Dom(\mathcal{V}'_p) = Dom(v'_i) \times \cdots \times Dom(v'_j)$ é chamada de *valoração de próximo estado*, tal que $\mathcal{V}'_p = \{v_i, \cdots, v_j\}$.

Uma fórmula $p \in P_V$ associada a uma transição é avaliada sempre sobre a valoração $\hat{v}_p$ de estado atual, e denota-se $p(\hat{v}_p)$ tal avaliação. Na modelagem de SEDs, particiona-se $P_V = P_{Vt} \cup P_{Va}$ de modo que as fórmulas associadas a modelos de planta somente atualizem valores de variáveis e nunca desabilitem transições. Essas fórmulas são denominadas *fórmulas de atualização*. Diferentemente, as fórmulas associadas a modelos de especificações mantem os valores das variáveis e desabilitam transições conforme testes lógicos. Essas fórmulas são denominadas *fórmulas de teste* ou *de guarda*.

Uma fórmula de atualização $p_a \in P_{Va}$ recebe uma valoração $\hat{v}_{p_a}$ de estado atual e calcula a valoração de próximo estado $\hat{v}'_{p_a}$ com base em uma regra de atualização $\hat{v}'_{p_a} = p_a(\hat{v}_{p_a})$.

Definição 3 *Uma valoração $\hat{v}_p$ de estado atual é dita válida com relação à fórmula de atualização $p \in P_V$ se, e somente se, $\hat{v}'_p = p(\hat{v}_p)$, $\hat{v}_p \in Dom(\mathcal{V}_p)$ e $\hat{v}'_p \in Dom(\mathcal{V}'_p)$.*

Por exemplo, considera-se duas variáveis x e y, tal que $Dom(x) = Dom(y) = \{1, 2, 3\}$ e $V = \{x, y\}$. Então, considera-se $x' = x+y+1$ uma fórmula $p_a \in P_{Va}$ que atualiza o valor de x', com $\mathcal{V}_{p_a} = \{x, y\}$, $\mathcal{V}'_{p_a} = \{x\}$, $Dom(\mathcal{V}_{p_a}) = Dom(V)$, e $Dom(\mathcal{V}'_{p_a}) = Dom(x)$. As valorações de estado atual $\hat{v}_{p_a}$ são da forma $(\bar{x}, \bar{y})$ e as valorações de próximo estado são da forma $(\bar{x})$. Assim, para uma valoração $\hat{v}_{p_a} = (1, 1)$, obtêm-se $p_1(\hat{v}_{p_1}) = (3)$ e então $\hat{v}_{p_1} = (1, 1)$ é válida com relação a p_1. Para $\hat{v}_{p_a} = (2, 2)$, obtêm-se $p_1(\hat{v}_{p_1}) = (5) \notin Dom(\mathcal{V}'_{p_1})$, e $\hat{v}_{p_1} = (2, 2)$ não é válida para p_1.

Uma fórmula lógica de teste $p_t \in P_{Vt}$ recebe uma valoração $\hat{v}_{p_t}$ de estado atual, não altera o valor de variáveis ($\mathcal{V}'_{p_t} = \emptyset$), e habilita ou não a transição com base em uma regra de teste, $p_t(\hat{v}_{p_t}) = true$ ou $p_t(\hat{v}_{p_t}) = false$. Por exemplo, considera-se a fórmula de teste p_t com regra $y > 2$, tendo-se $\mathcal{V}_{p_t} = \{y\}$, $Dom(\mathcal{V}_{p_t}) = Dom(y)$, e as valorações $\hat{v}_{p_t}$ são na forma $(\bar{y})$. Então, $p_t(\hat{v}_{p_t})$ é *verdadeiro* para a valoração $\hat{v}_{p_t} = (3)$, e *falso*, do contrário.

Para os resultados desse trabalho, é necessário as vezes referenciar-se ao valor de uma variável específica em um tupla maior. Considere $V_1 = \{v_i, \ldots, v_j\}$ um conjunto de variáveis qualquer e $\hat{v}_1 = (\overline{v_i}, \ldots, \overline{v_j})$ uma valoração de V_1. Denota-se por $\hat{v}_1(V_2)$ os valores em $\hat{v}_1$ que correspondem às variáveis em V_2, para qualquer $V_2 \subseteq V_1$.

Ainda, assume-se que as atualizações são todas exatos, isto é, para cada valoração $\hat{v}_{p_a}$, uma atualização $p_a(\hat{v}_{p_a})$ leva a uma única valoração $\hat{v}'_{p_a}$. Isso se diferencia de abordagens na literatura que utilizam abstrações de variáveis, por exemplo, as quais em geral lidam com não determinismo dos valores das variáveis (TEIXEIRA et al., 2015; MALIK; TEIXEIRA, 2019). Ademais, considera-se apenas atualizações convergentes, isto é, duas atualizações não podem levar uma mesma variável a valores diferentes.

3.1.1 Modelos com estados parametrizados na forma explícita

Os modelos com estados parametrizados foram apresentados em sua forma implícita, isto é, com fórmulas associadas às transições. Nesse caso, o valor de cada variável que parametriza cada um dos estados é desconhecido até que as fórmulas sejam executadas. Depois disso, um AFE pode ser apresentado em sua forma *explícita*, pois o valor de cada variável em cada estado é conhecido. Neste trabalho, usa-se a forma explícita para quantificar-se os estados e transições de um AFE afim de se comparar com outras abordagens como a sem parâmetros.

Então, um *Autômato Finito Estendido na Forma Explícita* (AFEE) é um AFD que expressa o mesmo comportamento que um AFE. Segundo (CHEN; HWANG; JIANG, 2013), considerando-se um AFE qualquer $A_{\mathfrak{E}} = (\Sigma, V, Q, q^{\circ}, Q^{\omega}, P_V, \Gamma)$, no qual $V = \{v_0, \ldots, v_n\}$, sua forma explícita é o AFEE $A^{\bullet}_{\mathfrak{E}} = (\Sigma, Q_{\bullet}, q^{\circ}_{\bullet}, Q^{\omega}_{\bullet}, \Gamma_{\bullet})$ tal que:

- $Q_{\bullet} = Q \times Dom(V)$. Observa-se que cada estado em $Q_{\bullet}$ assume a forma de uma tupla $(q, \hat{v})$, tal que $q \in Q$ e $\hat{v} \in Dom(V)$;
- $q^{\circ}_{\bullet} = (q^{\circ}, (v^o_0, \ldots, v^o_n))$;
- $Q^{\omega}_{\bullet} = Q^{\omega} \times Dom(V)$;
- $\Gamma_{\bullet} \subseteq Q_{\bullet} \times \Sigma \times Q_{\bullet}$ tem transições denotadas por $(q_0, \hat{v}) \xrightarrow{\sigma} (q_1, \hat{v}')$, se existir $q_0 \xrightarrow{\sigma:p} q_1$ com fórmula $p(\hat{v}'_p) \in P_V$ que habilita a transição, tal que, $\hat{v}_p \in Dom(\mathcal{V}_p)$, $\hat{v}'_p \in Dom(\mathcal{V}'_p)$, $\hat{v}(\mathcal{V}_p) = \hat{v}_p$ e $\hat{v}'(\mathcal{V}'_p) = \hat{v}'_p$.

Essa notação pode ser estendida para cadeias $s \in \Sigma^*$ por $(q_0, \hat{v}) \xrightarrow{s} (q_1, \hat{v}')$. A notação $(q_0, \hat{v}) \xrightarrow{\sigma}$ representa $(q_0, \hat{v}) \xrightarrow{\sigma} (q_1, \hat{v}')$, isto é, a valoração $\hat{v}$ no estado q_0 habilita o evento σ levando a um estado qualquer $q_1 \in Q$ e a uma nova valoração qualquer $\hat{v}'$. A mesma notação é usada para cadeias em Σ^*, tal que $A^{\bullet}_{\mathfrak{E}} \xrightarrow{s}$ representa $(q^{\circ}, \hat{v}) \xrightarrow{s} (q, \hat{v}')$ para algum estado $q \in Q$ e alguma valoração $\hat{v}'$, enquanto que a notação $A^{\bullet}_{\mathfrak{E}} \rightarrow q$ representa $(q^{\circ}, \hat{v}) \xrightarrow{s} (q, \hat{v}')$ para uma cadeia qualquer $s \in \Sigma^*$.

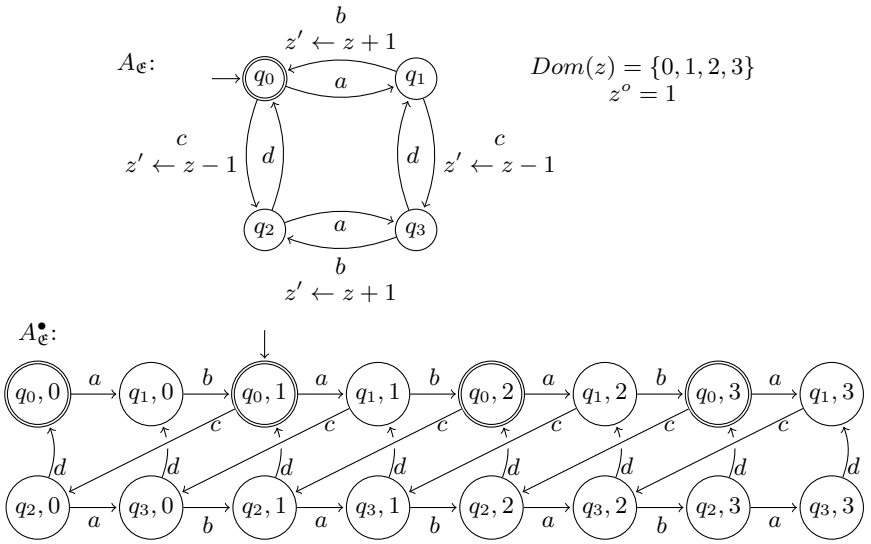
A Figura 14 apresenta um exemplo de AFE em suas formas implícita $A_{\mathfrak{E}}$ e explícita $A^{\bullet}_{\mathfrak{E}}$. Observa-se que, na forma explícita existe uma variável z com valor inicial $z^o = 1$. Então, o estado inicial na forma explícita é representado pela tupla que contém o estado inicial implícito ($q^{\circ} = q_0$) e o valor inicial da variável (1), isto é, $q^{\circ}_{\bullet} = (q_0, 1)$. Partindo-se de q° é possível executar-se duas transições: uma como evento a e outra com o evento c. Caso a transição executada seja a do evento a, $A_{\mathfrak{E}}$ parte do estado q_0 e chega ao estado q_1 sem alterar o valor de z. Isso é representado em $A^{\bullet}_{\mathfrak{E}}$ por uma transição partindo do estado inicial $(q_0, 1)$ e indo para o estado $(q_1, 1)$ sob a ocorrência do evento a. Caso a transição executada partindo de q_0 em $A_{\mathfrak{E}}$ seja a com o evento c, então o valor de z é decrementado de 1 para 0 e o estado alcançado é q_2. Nesse caso o mesmo comportamento é representado na versão explícita por uma transição partindo do estado inicial $(q_0, 1)$ e indo para o estado $(q_2, 0)$ sob a ocorrência do evento c.

Dado os AFEs construídos com estados parametrizados, a próxima seção aborda o processo de síntese do controlador tendo como entrada tais autômatos.

3.2 Controle Supervisório com estados parametrizados

Se planta e especificações de um SED são modelados por AFEs, controlabilidade e não bloqueio têm que ser estendidos de forma a considerar os valores das variáveis, além

Figura 14 – Exemplo de AFE e sua versão explícita



dos eventos (TEIXEIRA et al., 2015).

Para $G_{\mathfrak{E}}$ e $E_{\mathfrak{E}}$ respectivamente modelando planta e especificações de um SED, $E_{\mathfrak{E}}$ é dito *V-controlável* com relação a $G_{\mathfrak{E}}$ se o que segue for verdade para todas as cadeias $s \in \Sigma^*$, todos oe eventos não controláveis $\mu \in \Sigma_{\mathrm{u}}$, e todas as valorações $\hat{v}$, $\hat{v}'$: se $E_{\mathfrak{E}} \xrightarrow{s} (q_0, \hat{v})$ e $G_{\mathfrak{E}} \xrightarrow{s} (q_1, \hat{v}) \xrightarrow{\mu} (q_3, \hat{v}')$, então existe $q_2 \in Q_E$ tal que $E_{\mathfrak{E}} \xrightarrow{s} (q_0, \hat{v}) \xrightarrow{\mu} (q_2, \hat{v}')$.

V-controlabilidade difere da controlabilidade ordinária no sentido de que uma especificação deve não somente ser capaz de processar todos os eventos não controláveis possíveis na planta, mas na ocorrência de um evento não controlável deve atualizar as variáveis da mesma maneira que a planta. Diferentemente, para eventos controláveis, a especificação pode desabilitar algumas ou todas as atualizações associadas.

Síntese com AFEs é definida pela composição $K_{\mathfrak{E}} = E_{\mathfrak{E}} \parallel G_{\mathfrak{E}}$ e, similarmente à abordagem sem parâmetros, tem o supervisor denotado $S_{\mathfrak{E}}$, cuja linguagem $\mathcal{L}(S_{\mathfrak{E}}) = \sup\mathcal{C}_V(E_{\mathfrak{E}}, G_{\mathfrak{E}})$ (TEIXEIRA et al., 2015), e representa o comportamento mais permissivo que pode ser implementado por $G_{\mathfrak{E}}$ enquanto satisfaz $E_{\mathfrak{E}}$. O não bloqueio de um AFE $A_{\mathrm{v}} = (\Sigma, V, Q, q^{\circ}, Q^{\omega}, P_V, \Gamma)$ pode ser assegurado se $A_{\mathrm{v}} \xrightarrow{s} (q_0, \hat{v})$ implica em $(q_0, \hat{v}) \xrightarrow{s} (q_1, \hat{v}')$ para algum estado $q_1 \in Q^{\omega}$. Para AFEs, o não bloqueio é verificado após a síntese e, geralmente, não incluso no cálculo de $S_{\mathfrak{E}}$.

Se, além de ser controlável, $S_{\mathfrak{E}}$ é também não bloqueante, então $S_{\mathfrak{E}}$ é a solução ótima do problema de controle. Assumindo-se que eventos e variáveis são determinísticos (TEIXEIRA et al., 2015), pode ser mostrado que

$$\sup\mathcal{C}((E_{\mathfrak{E}} \parallel G_{\mathfrak{E}}), G_{\mathfrak{E}}) = \sup\mathcal{C}_V(E_{\mathfrak{E}}, G_{\mathfrak{E}}).$$

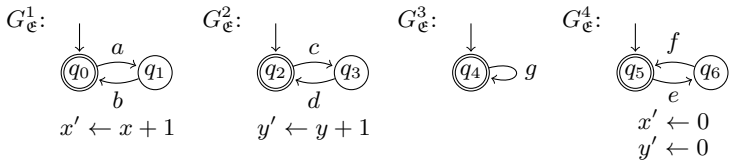
Isto é, o problema de controle com AFEs pode ser resolvido utilizando tanto a operação de síntese ordinária (sem parâmetros) ($\sup\mathcal{C}$) quanto a versão estendida ($\sup\mathcal{C}_V$).

Em ambos casos, considerando-se determinismo, o resultado esperado é o mesmo, e a escolha dentre as duas abordagens impacta somente a etapa de modelagem.

3.3 Exemplo com estados parametrizados

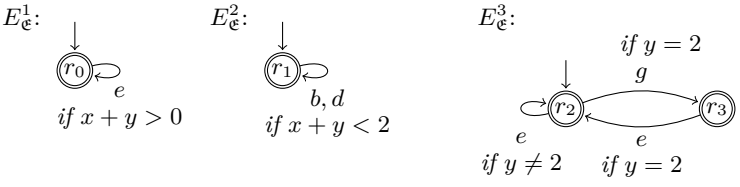
O exemplo proposto na Figura 10 pode ser modelado por meio de AFEs, tanto para a planta, quanto para as especificações. Para o exemplo proposto, criam-se duas variáveis x e y que armazenam a quantidade de peças do tipo B e D presentes no *buffer*, respectivamente. Dessa maneira, $V = \{x, y\}$, $V' = \{x', y'\}$, $Dom(x) = Dom(y) = \{0, 1, 2, 3\}$ e $x^o = y^o = 0$, pois o *buffer* está inicialmente vazio. Duas fórmulas de atualização são criadas para cada variável, $P_V = \{x' \leftarrow x + 1, x' \leftarrow 0, y' \leftarrow y + 1, y' \leftarrow 0\}$, tal que: o valor de x seja incrementado ao se inserir uma peça do tipo B no *buffer* ($x' \leftarrow x + 1$); o valor de y seja incrementado ao se inserir uma peça do tipo D no *buffer* ($y' \leftarrow y+1$); os valores de x e y são zerados quando o *buffer* é esvaziado ($x' \leftarrow 0, y' \leftarrow 0$). A Figura 15 apresenta os modelos criados para a planta.

Figura 15 – Planta - Modelo com estados parametrizados



A abordagem de estados parametrizados facilita a criação das especificações propostas no exemplo: evitar o *underflow* e o *overflow* do *buffer*, realizar a operação de emparelhamento (evento g) quando um par de peças do tipo D está no *buffer*. No modelo das especificações, é possível permitir, por meio de fórmulas de teste, que a execução de determinado evento se dê apenas para valores específicos no domínio das variáveis. Por exemplo, na Figura 16, foram criadas fórmulas de teste: para verificar se o *buffer* não está vazio (*if* $x + y > 0$); para verificar se o *buffer* não está cheio (*if* $x + y < 2$); para verificar se um par de peças do tipo D está no buffer (*if* $y = 2$) ou não (*if* $y \neq 2$). A abordagem de estados parametrizados possibilita que a memorização da quantidade de peças no *buffer* não esteja relacionada aos estados do modelo, como na abordagem não parametrizada, mas sim armazenada em variáveis.

Figura 16 – Especificações - Modelo com estados parametrizados



Considerando o exemplo, e que $K_{\mathfrak{E}} = G^1_{\mathfrak{E}} \parallel G^2_{\mathfrak{E}} \parallel G^3_{\mathfrak{E}} \parallel G^4_{\mathfrak{E}} \parallel E^1_{\mathfrak{E}} \parallel E^2_{\mathfrak{E}} \parallel E^3_{\mathfrak{E}}$ a quantidade de estado (e transições) dos modelos com estados parametrizados são mostrados na Tabela 2.

Tabela 2 – Quantidade de estados (e transições) para o exemplo.

Abordagem	Planta	Especificação	Composição	Supervisor
Sem parâmetros	8 (32)	7 (12)	56 (136)	28 (57)
Estados parametrizados	72 (380)	2 (14)	56 (136)	28 (57)

Observa-se que a quantidade de estados e transições do modelo da composição com estados parametrizados é equivalente ao modelo obtido da abordagem não parametrizada apresentado no capítulo anterior. A equivalência é mantida para os respectivos supervisores calculados. Porém, algumas características impedem que as vantagens obtidas pela utilização dos estados parametrizados na modelagem sejam transferidas diretamente para a etapa de síntese. A seção seguinte descreve essas características.

3.3.1 A limitação dos modelos com estados parametrizados

Apesar das equivalência mantida no resultado da etapa de modelagem entre a abordagem do capítulo anterior e a com estados parametrizados, sua obtenção é resultado de uma modelagem menos complexa do que a abordagem não parametrizada: as especificações $E_{\mathfrak{E}}$ são mais facilmente obtidas que E, têm o número de estados independentes do número de peças do *buffer* e, de uma maneira geral, podem modelar requisitos que não são facilmente modelados pela abordagem não parametrizada; as fórmulas em P_V proporcionam regras de atualização e teste que podem expressar comportamentos dos mais variados tipos como a saída de um algoritmo ou a execução de uma equação matemática, por exemplo.

Porém, para que um AFE tenha sua forma explícita apresentada, é necessário que o processo de explicitação seja realizado levando em consideração todos os modelos associados ao conjunto de variáveis. Isso significa que não é possível calcular o AFEE correspondente apenas ao modelo $G_{\mathfrak{E}1}$ na Figura 15, por exemplo, porque $G_{\mathfrak{E}2}$ também altera os valores da variável x. Isso significa que os modelos obtidos em $\mathfrak{E}$ não são modulares.

A literatura oferece técnicas de abstração de variáveis (TEIXEIRA et al., 2015) que funcionam modularmente (MALIK; TEIXEIRA, 2016; MALIK; TEIXEIRA, 2019) abstraindo as variáveis desnecessárias na etapa de síntese. Porém, essas técnicas não funcionam com abstrações parciais, isto é, abstrações que, além de removerem variáveis desnecessárias, também removam partes desnecessárias do domínio de variáveis necessárias. Isso se deve ao fato de um domínio de variável ser intrinsecamente indivisível. Quando o domínio de uma variável é grande, e é combinado com outros domínios grandes de variáveis, tende a gerar um espaço de estados muito maior, o que dificulta o tratamento computacional.

Dessa maneira, as vantagens da etapa de modelagem dos modelos em $\mathfrak{E}$ não são diretamente transferidas para a etapa de síntese e implementação. A próxima seção apresenta a abordagem com eventos parametrizados que proporciona a criação de modelos modularizados.

3.4 Modelagem com eventos parametrizados

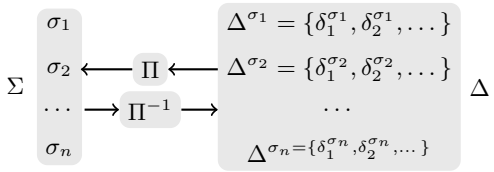
Os AFDs modelados sob a abordagem de eventos parametrizados, denotada ao longo deste trabalho por $\mathfrak{D}$, são caracterizados quando cada evento $\sigma \in \Sigma$ é associado a um conjunto de eventos parametrizados $\{\delta_1, \delta_2, \ldots, \delta_n\} = \Delta^\sigma \subseteq \Delta$, passando σ a ser uma máscara para um conjunto de eventos. A relação entre Σ e Δ pode ser definida sobre os conjuntos de cadeias por um mapa mascarador $\Pi : \Delta^* \to \Sigma^*$ (TEIXEIRA, 2013) como:

$$\begin{aligned} \Pi(\epsilon) &= \epsilon; \\ \Pi(\delta) &= \sigma \text{ para } \delta \in \Delta^\sigma; \\ \Pi(t\delta) &= \Pi(t)\sigma \text{ para } t \in \Delta^*, \delta \in \Delta^\sigma \text{ e } \sigma \in \Sigma. \end{aligned} \tag{3.1}$$

O mapa mascarador Π pode ser generalizado para qualquer linguagem $\mathcal{L}_\mathfrak{D} \subseteq \Delta^*$ por $\Pi(\mathcal{L}_\mathfrak{D}) = \{s \in \Sigma^* | \exists t \in \mathcal{L}_\mathfrak{D}, \Pi(t) = s\}$. Similarmente, por ser definido também o mapa inverso $\Pi^{-1} : \Sigma^* \to 2^{\Delta^*}$ como sendo $\Pi^{-1}(s) = \{t \in \Delta^* | \Pi(t) = s\}$ e que pode ser estendido sobre uma linguagem $\mathcal{L} \subseteq \Sigma^*$, por $\Pi^{-1}(\mathcal{L}) = \{t \in \Delta^* | \Pi(t) \in \mathcal{L}\}$.

A mesma ideia pode ser estendida para AFDs, ao invés de linguagens. Seja A um AFD definido com eventos em Σ, o mapa $\Pi^{-1}(A)$ mapeia os eventos de A em eventos parametrizados em Δ no AFD $A_\mathfrak{D}$. Esse pressuposto consiste em substituir cada evento σ, de cada transição em A, pelo respectivo conjunto de eventos parametrizados Δ^σ. A Figura 17 ilustra a relação entre o alfabeto original Σ e o parametrizado Δ.

Figura 17 – Parametrização de eventos



3.4.1 Modelagem do filtro

O processo de mapeamento de um evento em Σ em um conjunto de eventos em Δ^σ é incluído na modelagem de plantas e especificações apresentadas no capítulo anterior. Assume-se que a planta G obtida pela modelagem não parametrizada (como a da Figura 11) é agora modelada pelo AFD $G_\mathfrak{D}$ que expressa o mesmo comportamento de G, no

entanto usufruindo de eventos enriquecidos pelos contextos (CURY et al., 2015), isto é, $G_{\mathfrak{D}} = \Pi^{-1}(G)$, tal que:

$$\Pi(\Pi^{-1}(G)) = G \tag{3.2}$$

Nota-se que, com um conjunto de eventos parametrizados, uma transição pode permitir mais de uma instância para cada evento no conjunto Σ. Isso significa que o modelo $G_{\mathfrak{D}}$ expressa diferentes contextos para um evento, mas não é capaz de escolher qual contexto deve ser aplicado em cada etapa. Os diferentes contextos, representados pelos diferentes eventos parametrizados habilitados, associam uma característica de ambiguidade ao modelo no sentido de que $G_{\mathfrak{D}}$ não sabe qual instância escolher. A escolha dentre os eventos parametrizados $\{\delta_1, \delta_2, \ldots, \delta_n\}$ de um evento $\sigma \in \Sigma$ depende da construção de um modelo adicional, aqui denotado $H_{\mathfrak{D}}$, que tem a função de filtrar $G_{\mathfrak{D}}$.

O *filtro* $H_{\mathfrak{D}}$ tem o papel de selecionar o evento parametrizado que deve ocorrer a cada transição de $G_{\mathfrak{D}}$, implementando assim o modo como a *alternância de contexto* deve ocorrer. Isto é, um filtro não tem a intenção de desabilitar eventos completamente na planta, como uma especificação faz, mas sim de escolher qual evento parametrizado $\delta_i \in \Delta^{\sigma}$ deve ocorrer quando estes são ambíguos. O filtro escolhe dentre todos os contextos possíveis, qual é o que deve ocorrer a cada transição. Para isso, se assume que $H_{\mathfrak{D}}$ é sempre *preciso*.

Definição 4 *Um filtro $H_{\mathfrak{D}}$ é preciso se para cada Δ^{σ} elegível em uma transição, $H_{\mathfrak{D}}$ escolhe um, e apenas um, deles para permanecer habilitado.*

Considera-se que uma planta não parametrizada G é expressa por $G_{\mathfrak{D}} \| H_{\mathfrak{D}}$, tal que $\Pi(G_{\mathfrak{D}} \| H_{\mathfrak{D}}) = G$. Ainda, considera-se que $E_{\mathfrak{D}}$ é uma especificação com eventos parametrizados que expressa as mesmas regras de controle que a especificação não parametrizada E. Dessa forma, se espera que $\mathcal{L}(\Pi(K_{\mathfrak{D}})) = \mathcal{L}(K)$, para $K = G \| E$ e $K_{\mathfrak{D}} = G_{\mathfrak{D}} \| H_{\mathfrak{D}} \| E_{\mathfrak{D}}$. Com essas considerações, é mostrado por Cury et al. (2015) que ambos $K_{\mathfrak{D}}$ e K podem ser utilizados como entrada para o *framework* de síntese proposto por Ramadge e Wonham (1989) e a solução de controle é equivalente. A abordagem de eventos parametrizados exige a construção do modelo adicional $H_{\mathfrak{D}}$. Porém, se comparada com a abordagem não parametrizada, ela possui a vantagem de reconhecer e alternar contextos, e isso implica em criar especificações mais facilmente e possibilitar a *modularização*. A modularidade é uma característica importante que permite que $H_{\mathfrak{D}}$ possa ser modelado por uma composição $H_{\mathfrak{D}} = H_{\mathfrak{D}1} \| \ldots \| H_{\mathfrak{D}m}$ de modelos menores. Ainda, a modularidade proporciona vantagens de síntese (TEIXEIRA; CURY; QUEIROZ, 2018) e implementação (ROSA et al., 2017) que serão descritas no Capítulo 5.

3.5 Controle Supervisório com eventos parametrizados

Quando a planta e as especificações são construídas com eventos parametrizados, o algoritmo de síntese é o mesmo que o da abordagem sem parâmetros. Porém, o supervisor resultante irá controlar a planta da mesma maneira apenas se o que segue for respeitado.

Considera-se uma planta G obtida por autômatos com eventos parametrizados $G_{\mathfrak{D}} \| H_{\mathfrak{D}}$, tal que, $\Pi(G_{\mathfrak{D}} \| H_{\mathfrak{D}}) = G$. Considera-se uma especificação E, com eventos em Σ, substituída pela especificação mais simples $E_{\mathfrak{D}}$ e que usa eventos em Δ e expressa E. Nesse caso, é esperado que, para $K = G \| E$ e $K_{\mathfrak{D}} = G_{\mathfrak{D}} \| H_{\mathfrak{D}} \| E_{\mathfrak{D}}$,

$$\mathcal{L}(\Pi(K_{\mathfrak{D}})) = \mathcal{L}(K). \tag{3.3}$$

Considerando a igualdade, e que $H_{\mathfrak{D}}$ é *preciso*, a literatura (CURY et al., 2015) mostra que tanto $K_{\mathfrak{D}}$ quanto K podem ser utilizados como entrada para o algoritmo de síntese (RAMADGE; WONHAM, 1989), resultando na mesma solução de controle, isto é,

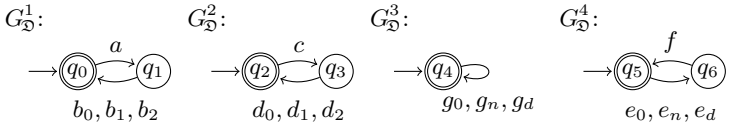
$$\sup\mathcal{C}(K, G) = \Pi(\sup\mathcal{C}(K_{\mathfrak{D}}, G_{\mathfrak{D}} \| H_{\mathfrak{D}})). \tag{3.4}$$

Isso significa que tanto a solução sem parâmetros quanto a solução com eventos parametrizados são equivalentes no processo de síntese (Eq. (3.4)), mas a abordagem em $\mathfrak{D}$ adiciona vantagens de modelagem, ao preço de garantir as igualdades (3.2) e (3.3) e construir um filtro $H_{\mathfrak{D}}$ preciso. Como essas tarefas são manuais, podem ser complexas de serem executadas.

3.6 Exemplo com eventos parametrizados

O modelo da Figura 11 é apresentado com eventos parametrizados na Figura 18. Os AFDs $G^1_{\mathfrak{D}}$, $G^2_{\mathfrak{D}}$, $G^3_{\mathfrak{D}}$ e $G^4_{\mathfrak{D}}$ são da forma $(\Delta, Q, q^\circ, Q^\omega, \Gamma)$. Os eventos b e d são parametrizados tal que $\Delta^b = \{b_0, b_1, b_2\}$ e $\Delta^d = \{d_0, d_1, d_2\}$ e carregam a informação extra da quantidade de peças dos tipos B e D presentes no *buffer*, respectivamente. Eventos e e g são parametrizados como $\Delta^e = \{e_0, e_n, e_d\}$ e $\Delta^g = \{g_0, g_n, g_d\}$ tal que: e_0 e g_0 representam que o *buffer* está vazio; e_d e g_d representam que um par de peças do tipo D está no *buffer*; e e_n e g_n representam todas as outras combinações de peças no *buffer*.

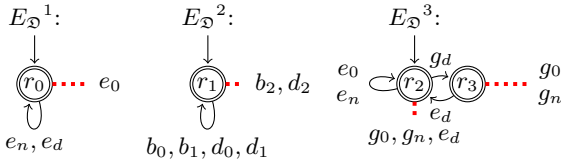
Figura 18 – Planta - Modelo com eventos parametrizados



A principal vantagem do uso de parametrizações na fase de modelagem está na criação do modelo das especificações. A parametrização de um evento proporciona ao

projetista a habilidade de desabilitar diretamente um evento com um contexto específico. No exemplo proposto, deseja-se que as especificações desabilitem b e d quando o *buffer* possuir duas peças, e quando o *buffer* estiver vazio, e que g ocorra somente para um par de peças do tipo D. No modelo não parametrizado apresentado na Figura 11, não existe distinção entre os contextos de b e c e, dessa maneira, se torna necessário memorizar a quantidade p de peças no *buffer*. A memorização implica na criação de um modelo E que possui $p+1$ estados. Utilizando eventos parametrizados é possível criar os modelos de especificações apresentados na Figura 19. Nota-se que, em relação ao modelo não parametrizado, eles possuem a vantagem de serem independentes da capacidade do *buffer*, isto é, são modelados com um e dois estado cada. Isso é possível, porque se desabilita diretamente as instâncias desejadas.

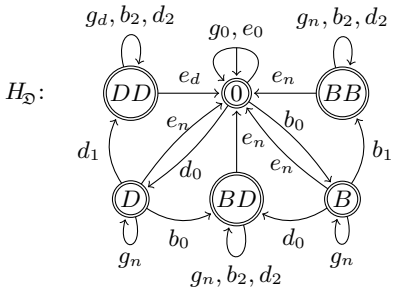
Figura 19 – Especificações - Modelo com eventos parametrizados



Em contrapartida, ao levar-se em conta apenas os modelos da Figura 18 existe ambiguidade, por exemplo, em relação a qual evento parametrizado de b (b_0, b_1, b_2) será executado primeiro. O modelo do filtro $H_{\mathfrak{D}}$ implementa a distinção dentre os eventos parametrizados, ordenando-os de modo que b_0 ocorra antes de b_1, por exemplo.

A ordem de ocorrência desses eventos era implementada no modelo não parametrizado por meio da especificação na Figura 12, mas com eventos parametrizados pode ser alcançada por $H_{\mathfrak{D}}$ mostrado na Figura 20. $H_{\mathfrak{D}}$ possibilita vantagens de síntese e implementação que serão descritas na Capítulo 5.

Figura 20 – Filtro - Modelo com eventos parametrizados



Quando composto com $G^1_{\mathfrak{D}}$, $G^2_{\mathfrak{D}}$, $G^3_{\mathfrak{D}}$ e $G^4_{\mathfrak{D}}$, $H_{\mathfrak{D}}$ leva a um caso particular da planta parametrizada que mantém uma cadeia $t \in \mathcal{L}(G^1_{\mathfrak{D}} \| G^2_{\mathfrak{D}} \| G^3_{\mathfrak{D}} \| G^4_{\mathfrak{D}} \parallel H_{\mathfrak{D}})$ para cada correspondente $s \in \Sigma^*$, enquanto simplifica a modelagem de $E_{\mathfrak{D}}$ em relação a E. A Tabela 3 compara os resultados de modelagem e síntese para o exemplo adicional sob as abordagens sem parâmetros e com estados e eventos parametrizados.

Tabela 3 – Quantidade de estados (e transições) para o exemplo.

Abordagem	Planta	Especificação	Filtro	Composição	Supervisor
Sem parâmetros	8 (32)	7 (12)	-	56 (136)	28 (57)
Estados parametrizados	72 (380)	2 (14)	-	56 (136)	28 (57)
Eventos parametrizados	8 (72)	2 (11)	6 (24)	56 (136)	28 (57)

Observa-se que, pela igualdade de estados e transições, o resultado da composição dos modelos nas três abordagens é equivalente, bem como o supervisor calculado.

3.6.1 A limitação dos modelos com eventos parametrizados

A abordagem com eventos parametrizados é mais eficiente na criação e memorização de contextos do que a abordagem sem parâmetros. Isso permite que o modelo das especificações sejam criados mais facilmente e com menos estados nessa abordagem. Todavia, modelos com eventos parametrizados requerem a criação do modelo do filtro $H_{\mathfrak{D}}$ para complementar o modelo da planta. Como pode ser visto em (CURY et al., 2015), a criação desse modelo pode ser complexa e é manual. Assim, o esforço de modelagem é transferido da especificação para o modelo do filtro que implementa a distinção de contextos.

Nesse sentido, a abordagem com estados parametrizados tem a vantagem de implementar a estrutura de filtragem de maneira mais simples, por meio de fórmulas sobre variáveis. A próxima seção recapitula as principais características das abordagens citadas bem como as vantagens e desvantagens de cada uma.

3.7 Comparação entre as abordagens

As abordagens apresentadas anteriormente são comparadas nessa seção a fim de se identificar os elementos envolvidos em cada etapa da modelagem e síntese do controlador.

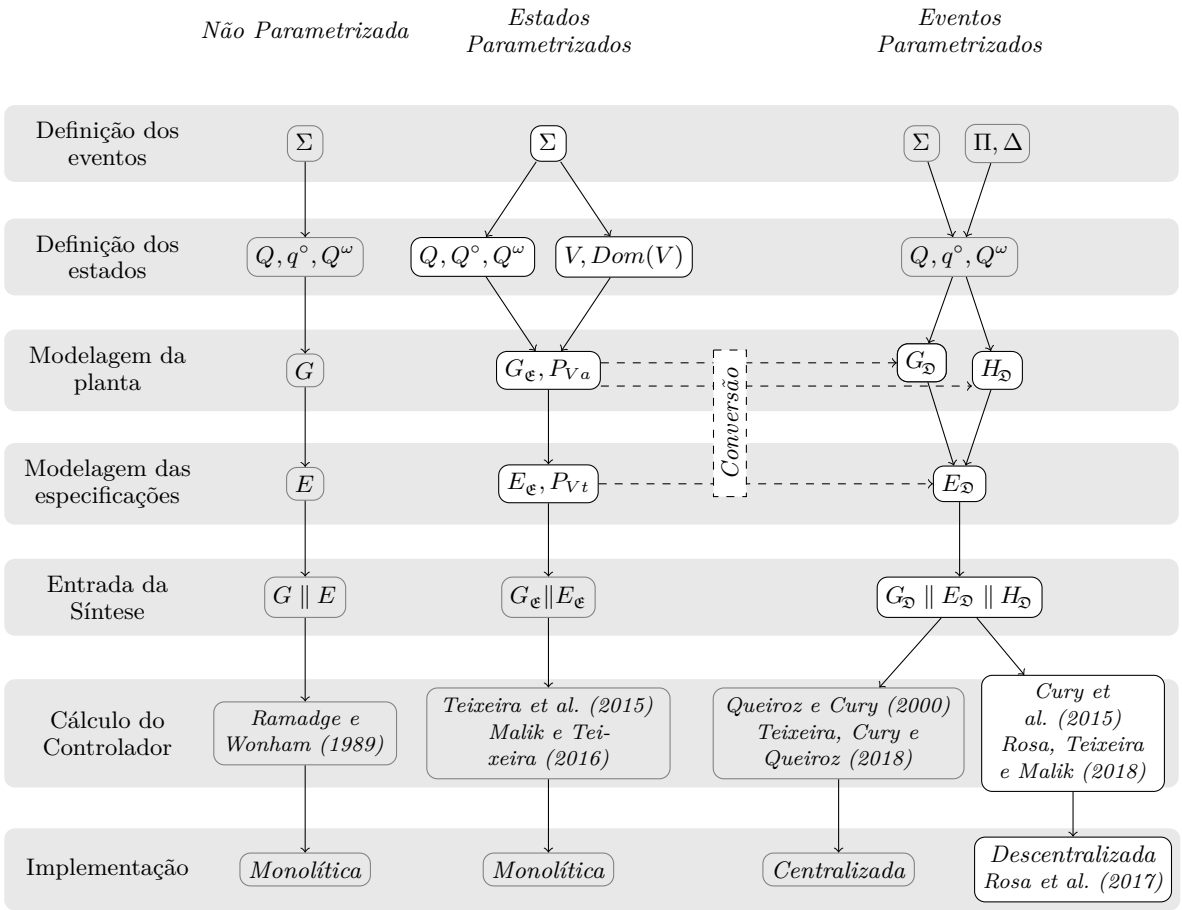
Inicialmente, a abordagem não parametrizada de construção de modelos de SEDs por meio de AFDs consiste na definição dos eventos (Σ) e estados (Q, q°, Q^{ω}) dos diferentes componentes do sistema. Então, são construídos modelos para a planta ($G = G_1 \parallel \ldots \parallel G_n$) e para as especificações ($E = E_1 \parallel \ldots \parallel E_m$) que são compostos para o cálculo do AFD ($K = G \parallel E$) de entrada para síntese do controlador, por meio da TCS. O supervisor S, tal que $\mathcal{L}(S) = \sup\mathcal{C}(K, G)$ é calculado de modo a ser não bloqueante, maximamente permissível e controlável.

Depois, relembra-se que os modelos com estados parametrizados ($\mathfrak{E}$) utilizam eventos (Σ) e estados (Q, Q°, Q^{ω}) que são indiretamente parametrizados por variáveis ($V, Dom(V)$). O modelo da planta ($G_{\mathfrak{E}} = G_{\mathfrak{E}1} \parallel \ldots \parallel G_{\mathfrak{E}n}$) é criado com fórmulas de atualização (P_{Va}) e o modelo das especificações ($E_{\mathfrak{E}} = E_{\mathfrak{E}1} \parallel \ldots \parallel E_{\mathfrak{E}m}$) com fórmulas de teste (P_{Vt}) avaliadas

sobre os valores das variáveis. As fórmulas e variáveis são mais adequadas à percepção do engenheiro e podem capturar de maneira mais favorável características de SEDs. Assim, as especificações são criadas de maneira mais simples que na abordagem sem parâmetros. O supervisor não bloqueante e controlável $S_{\mathfrak{E}}$ é calculado. O não bloqueio é verificado em uma etapa pós-síntese.

Por fim, recapitula-se que na construção de modelos com eventos parametrizados ($\mathfrak{D}$) se parte dos eventos σ (Σ) que são parametrizados por um mapa mascarador (Π) resultando-se em um novo conjunto de eventos (Δ). Com esse conjunto de eventos parametrizados, são definidos o modelo da planta ($G_{\mathfrak{D}} = G_{\mathfrak{D}1} \parallel \ldots \parallel G_{\mathfrak{D}n}$) e das especificações ($E_{\mathfrak{D}} = E_{\mathfrak{D}1} \parallel \ldots \parallel E_{\mathfrak{D}m}$) que possuem a vantagem de serem menos complexas sob essa abordagem. Ainda, uma etapa adicional e manual é a obtenção do filtro ($H_{\mathfrak{D}} = H_{\mathfrak{D}1} \parallel \ldots \parallel H_{\mathfrak{D}i}$). O cálculo do AFD K se dá pela utilização do mapa mascarador e dos modelos compostos $K = \Pi(G_{\mathfrak{D}} \parallel E_{\mathfrak{D}} \parallel H_{\mathfrak{D}})$. O supervisor $S_{\mathfrak{D}}$, tal que $\mathcal{L}(S_{\mathfrak{D}}) = \sup\mathcal{C}(K, \Pi(G_{\mathfrak{D}}))$ é calculado de modo a ser não bloqueante, maximamente permissível e controlável.

Figura 21 – Comparação estrutural entre as abordagens



A Figura 21 sintetiza o processo de obtenção do controlador nas três abordagens. Nela, é possível observar que as abordagens com parametrização, se comparadas com a clássica, implicam na criação de modelos adicionais, eventos parametrizados e estrutura de

filtragem, no caso de $\mathfrak{D}$, e variáveis e fórmulas, no caso de $\mathfrak{E}$. Embora os modelos adicionais signifiquem uma etapa extra na modelagem, vantagens decisivas podem ser observadas na modelagem das *especificações* e isso permite tratar de uma gama maior de problemas envolvendo SEDs caracterizados por retroalimentação, paralelismo, memorização, etc.

Além disso, a abordagem por estados parametrizados tem a vantagem de modelagem de incluir fórmulas que podem ser utilizadas para facilitar o processo de obtenção dos modelos. Porém, resulta em uma estrutura monolítica que não possibilita a modularidade diretamente, uma vez que, o domínio de uma variável é atômico e, quando combinado com outros domínios, de outras variáveis, pode levar à explosão do espaço de estados. Ainda, na fase de implementação, a abordagem por estados parametrizados utiliza uma estrutura monolítica como a da abordagem sem parametrizações. Contudo, a abordagem por eventos parametrizados permite a criação de modelos modulares que possuem a característica de facilitar o processo de síntese e de implementação, que pode ser descentralizada. No entanto, ela implica na criação direta e manual do modelo do filtro e dos contextos a serem reconhecidos.

Com o objetivo de combinar vantagens de ambas as abordagens apresentadas, o próximo capítulo propõe os algoritmos de conversão dos modelos em $\mathfrak{E}$ para os modelos em $\mathfrak{D}$. Nesse sentido, tais métodos possibilitam a criação dos modelos utilizando-se fórmulas e variáveis (vantagem na etapa de modelagem), e conversão destes modelos para eventos parametrizados (o filtro é gerado automaticamente) para que as vantagens de síntese e implementação possam ser exploradas, conforme descritas no Capítulo 5.

O método de conversão proposto está estruturalmente ilustrado na Figura 21 que destaca em branco o fluxograma de modelagem, síntese e implementação propostos por essa dissertação.

4 Conversão de modelos: estados parametrizados para eventos parametrizados

Inicialmente, relembra-se que em AFEs, uma transição com um evento $\sigma \in \Sigma$ implementa fórmulas $p \in P_V$ que usam ou testam valores de variáveis $(\overline{v_i}, \ldots, \overline{v_j}) = \hat{v}_p \in Dom(\mathcal{V}_p)$, levando a uma valoração determinística $(\overline{v_k}', \ldots, \overline{v_l}') = \hat{v}'_p \in Dom(\mathcal{V}'_p)$ em cada estado.

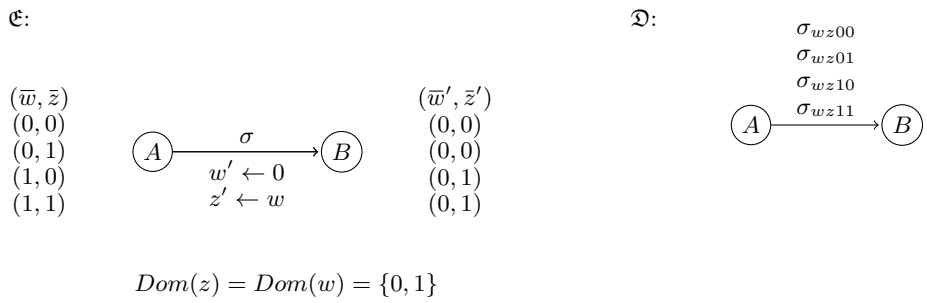
Em um AFE, os conjuntos de variáveis $\mathcal{V}_p$ e $\mathcal{V}'_p$, e suas valorações $\hat{v}_p$ e $\hat{v}'_p$ são parâmetros que representam determinado contexto no estado atual e próximo estado, respectivamente. Para representar os mesmos parâmetros usando eventos, uma opção é concatenar cada evento original $\sigma \in \Sigma$ a um parâmetro que represente a troca de contexto. Essa ideia leva à criação de um conjunto de eventos parametrizados.

Sistematicamente, se não houve troca de contexto ao se executar uma transição, isto é, a transição não implementou nenhuma fórmula e $\mathcal{V}_p \cup \mathcal{V}'_p = \emptyset$, então não existe necessidade de se criar um evento parametrizado. Porém, se $\mathcal{V}_p \cup \mathcal{V}'_p \neq \emptyset$, então ao menos um valor de variável foi modificado ou testado na transição, e um parâmetro representando essa modificação deve ser adicionado ao autômato em construção. Essa construção é formalizada inicialmente definindo-se um conjunto de variáveis $\mathbb{V}^\sigma = \bigcup_j \mathcal{V}_{p_j} \cup \mathcal{V}'_{p_j}$, para todas as transições $\xrightarrow{\sigma:p_j}$. Então, para uma valoração $\hat{\mathbb{V}}^\sigma_i \in \hat{\mathbb{V}}^\sigma$, tal que $\hat{\mathbb{V}}^\sigma = Dom(\mathbb{V}^\sigma)$, a troca de contexto pode ser representada pelo evento parametrizado $\sigma_{\mathbb{V}^\sigma \hat{\mathbb{V}}^\sigma_i}$, significando que as variáveis em $\mathbb{V}^\sigma$ tem a valoração $\hat{\mathbb{V}}^\sigma_i$ antes da transição com o evento σ ocorrer.

A Figura 22 apresenta duas transições de exemplo com o intuito de ilustrar como os contextos são representados e convertidos de uma abordagem para outra. Considera-se as transições $\xrightarrow{\sigma:w'\leftarrow 0}$ e $\xrightarrow{\sigma:z'\leftarrow w}$ de estados parametrizados, tal que $Dom(z) = Dom(w) = \{0, 1\}$. A cada diferente combinação de valores das variáveis w e z o estado A é parametrizado. Primeiro, $\mathbb{V}^\sigma = \{w, z\}$ é criado, porque $\mathcal{V}_{w'\leftarrow 0} = \{w\}$, $\mathcal{V}'_{w'\leftarrow 0} = \emptyset$, $\mathcal{V}_{z'\leftarrow w} = \{z\}$, e $\mathcal{V}'_{z'\leftarrow w} = \{w\}$. Observa-se que todas as variáveis envolvidas em atualizações de σ são elementos de $\mathbb{V}^\sigma$. Então, $\hat{\mathbb{V}}^\sigma = Dom(\mathbb{V}^\sigma) = \{(0,0), (0,1), (1,0), (1,1)\}$ inclui todas as combinações válidas para variáveis em $\mathbb{V}^\sigma$. Assim, concatenando o evento σ, as variáveis $\mathbb{V}^\sigma$, e cada possível combinação $\hat{\mathbb{V}}^\sigma_i \in \hat{\mathbb{V}}^\sigma$, é possível criar os eventos parametrizados como $\sigma_{\mathbb{V}^\sigma \hat{\mathbb{V}}^\sigma_i}$, isto é, σ_{wz00}, σ_{wz01}, σ_{wz10}, e σ_{wz11}.

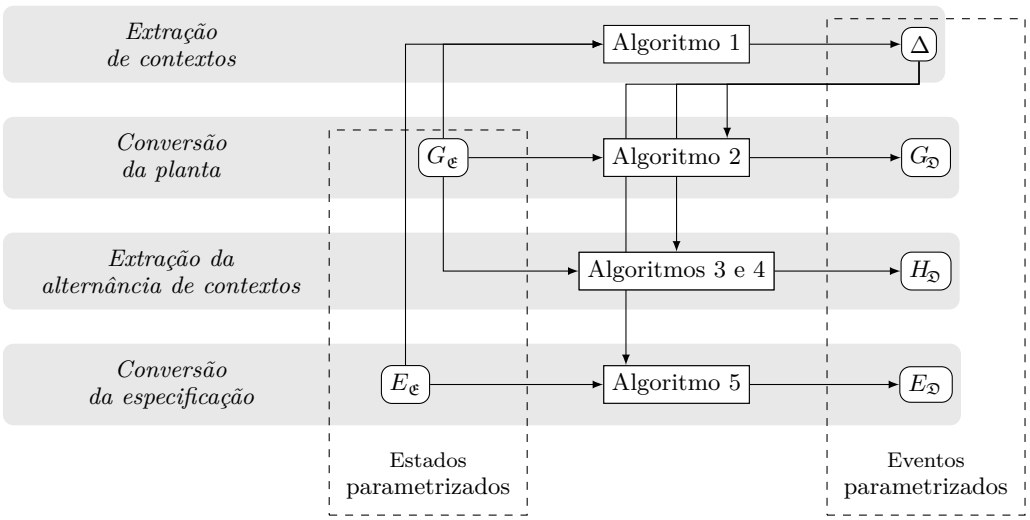
Se o processo for repetido para todos os eventos, e todas as alterações de valores de variáveis, um autômato da abordagem $\mathfrak{D}$ preserva a mesma semântica de atualizações que um AFE, mas usando um mecanismo diferente, com eventos parametrizados em vez de estados.

Figura 22 – Representação de contexto de $\mathfrak{E}$ para $\mathfrak{D}$



Com base nisso, se propõe a estrutura de conversão de $\mathfrak{E}$ para $\mathfrak{D}$ que pode ser conduzida pelo processo de quatro etapas mostrado na Figura 23.

Figura 23 – Estrutura de conversão de $\mathfrak{E}$ para $\mathfrak{D}$



A primeira etapa extrai todos os contextos possíveis assumidos pelo planta $G_{\mathfrak{E}}$ e especificação $E_{\mathfrak{E}}$ (Algoritmo 1). Essa etapa gera um conjunto de eventos parametrizados Δ, cada um representando um contexto. Então, a planta $G_{\mathfrak{E}}$ na abordagem $\mathfrak{E}$ é convertida para a planta $G_{\mathfrak{D}}$ na abordagem $\mathfrak{D}$ (Algoritmo 2). Como inicialmente $G_{\mathfrak{E}}$ não é precisa, o modelo do filtro $H_{\mathfrak{D}}$ é construído na terceira etapa extraindo as trocas de contextos das atualizações de variáveis (Algoritmo 3). O Algoritmo 4 modulariza o modelo do filtro $H_{\mathfrak{D}}$. Finalmente, a quarta etapa mapeia as restrições modeladas pelas especificações, de $\mathfrak{E}$ para $\mathfrak{D}$ (Algoritmo 5). Cada algoritmo é introduzido, discutido e exemplificado em detalhes a seguir.

4.1 Extração dos contextos

Inicialmente, o Algoritmo 1 inicializa o novo alfabeto parametrizado Δ como vazio (linha 2). Então, para cada evento $\sigma \in \Sigma$, ele constrói um conjunto de eventos

parametrizados Δ^σ representando todos os possíveis contextos que σ pode assumir. $\mathbb{V}^\sigma$ é o conjunto de variáveis envolvidas na representação de contextos de σ, e $\hat{\mathbb{V}}^\sigma$ é o conjunto de possíveis valorações para $\mathbb{V}^\sigma$, isto é, $\hat{\mathbb{V}}_i^\sigma \in \hat{\mathbb{V}}^\sigma$ tal que $\hat{\mathbb{V}}_i^\sigma \in Dom(\mathbb{V}^\sigma)$.

Inicialmente, Δ^σ, $\mathbb{V}^\sigma$ e $\hat{\mathbb{V}}^\sigma$ são inicializados como vazios (linha 4). Então, todas as transições com $\sigma : p$ são lidas dos AFEs de entrada $G_{\mathfrak{E}}$ e $E_{\mathfrak{E}}$ (linha 5) para identificar as variáveis que representam contextos para σ. Variáveis em $\mathcal{V}_p \cup \mathcal{V}'_p$ são adicionadas no conjunto $\mathbb{V}^\sigma$. Depois, cada valoração $\hat{\mathbb{V}}_i^\sigma \in Dom(\mathbb{V}^\sigma)$ é inserida no conjunto $\hat{\mathbb{V}}^\sigma$, contanto que$\hat{\mathbb{V}}_i^\sigma$ seja válida com respeito a todas as fórmulas em transições com σ (linha 8).

Finalmente, cada evento parametrizado é criado associando-se a σ as variáveis em $\mathbb{V}^\sigma$ e cada uma das valorações válidas $\hat{\mathbb{V}}_i^\sigma$ em $\hat{\mathbb{V}}^\sigma$. Se nenhum evento parametrizado precisa ser criado (isto é $\mathbb{V}^\sigma = \emptyset$) então σ é adicionado a Δ^σ. Como resultado, o Algoritmo 1 identifica todos os possíveis contextos em $G_{\mathfrak{E}}$ e $E_{\mathfrak{E}}$, e os reproduz em Δ usando um mecanismo livre de fórmulas e variáveis.

Algoritmo 1: EXTRAÇÃO DE CONTEXTO DE $\mathfrak{E}$ PARA $\mathfrak{D}$

entradas: plantas $G_{\mathfrak{E}} = (\Sigma_G, V_G, Q_G, q_G^\circ, Q_G^\omega, P_{VG}, \Gamma_G)$
especificações $E_{\mathfrak{E}} = (\Sigma_E, V_E, Q_E, q_E^\circ, Q_E^\omega, P_{VE}, \Gamma_E)$
saída : conjunto de eventos parametrizados Δ

1 **início**
2 $\Delta \leftarrow \emptyset$
3 **para cada** $\sigma \in \Sigma_G$ **faça**
4 $\Delta^\sigma \leftarrow \emptyset$, $\mathbb{V}^\sigma \leftarrow \emptyset$, $\hat{\mathbb{V}}^\sigma \leftarrow \emptyset$
5 **para cada** *transição* $q_1 \xrightarrow{\sigma:p} q_2 \in \Gamma_G \cup \Gamma_E$, *tal que* $q_1, q_2 \in Q_G \cup Q_E$, $p \in P_{VG} \cup P_{VE}$ *e* $\mathcal{V}_p \cup \mathcal{V}'_p \neq \emptyset$ **faça**
6 $\mathbb{V}^\sigma \leftarrow \mathbb{V}^\sigma \cup \mathcal{V}_p \cup \mathcal{V}'_p$
7 **fim**
8 **para cada** $\hat{\mathbb{V}}_i^\sigma \in Dom(\mathbb{V}^\sigma)$, *tal que* $\mathbb{V}^\sigma = \{v_j, \ldots, v_k\}$ *e* $Dom(\mathbb{V}^\sigma) = Dom(v_j) \times \cdots \times Dom(v_k)$ **faça**
9 $\hat{\mathbb{V}}^\sigma \leftarrow \hat{\mathbb{V}}^\sigma \cup \hat{\mathbb{V}}_i^\sigma$, tal que $\hat{\mathbb{V}}_i^\sigma(\mathcal{V}_p \cup \mathcal{V}'_p)$ é válida em relação a p $\forall$ transições $q_1 \xrightarrow{\sigma:p} q_2 \in \Gamma \cup \Gamma_E$
10 **fim**
11 $\Delta^\sigma \leftarrow \Delta^\sigma \cup \{\sigma_{\mathbb{V}^\sigma \hat{\mathbb{V}}_i^\sigma}\}$, para cada $\hat{\mathbb{V}}_i^\sigma \in \hat{\mathbb{V}}^\sigma$
12 **se** $\mathbb{V}^\sigma = \emptyset$ **então**
13 $\Delta^\sigma \leftarrow \sigma$
14 **fim**
15 $\Delta \leftarrow \Delta \cup \Delta^\sigma$
16 **fim**
17 **retorna** Δ
18 **fim**

4.1.1 Exemplo - Algoritmo 1

Considerando os modelos de entrada nas Figuras 15 e 16, é possível obter o alfabeto parametrizado Δ usando o Algoritmo 1. Os eventos a, c e f não possuem transições com fórmulas nas Figuras 15 e 16 ($\mathcal{V}_p = \mathcal{V}'_p = \emptyset$ para as transições com esses eventos), então $\Delta^a = \{a\}$, $\Delta^c = \{c\}$, e $\Delta^f = \{f\}$. As fórmulas $x' \leftarrow x + 1$ e

$if\ x + y < 2$ estão associadas com o evento b, então o conjunto de variáveis utilizadas para representar contextos de b é $\mathbb{V}^b = \{x, y\}$. O conjunto de valorações de $\mathbb{V}^b$, na forma (x, y), que são válidas com respeito a todas as fórmulas associadas com b é $\hat{\mathbb{V}}^b = \{(0,0), (0,1), (0,2), (0,3), (1,0), (1,1), (1,2), (1,3), (2,0), (2,1), (2,2), (2,3)\}$. O valor $\bar{x} = 3$ atribuiria um valor a x fora de seu domínio, então as valorações $(3,0), (3,1), (3,2)$ e $(3,3)$ são inválidas. Assim, o conjunto de eventos parametrizados de b é $\Delta^b = \{b_{xy00}, b_{xy01}, b_{xy02}, b_{xy03}, b_{xy10}, b_{xy11}, b_{xy12}, b_{xy13}, b_{xy20}, b_{xy21}, b_{xy22}, b_{xy23}\}$. Analogamente é possível calcular os eventos parametrizados de todos os eventos em Σ. O Quadro 1 mostra os eventos do exemplo e os eventos parametrizados calculados pelo Algoritmo 1.

Quadro 1 – Eventos, variáveis e respectivos eventos parametrizados para o exemplo

Evento σ	**Variáveis utilizadas $\mathbb{V}^\sigma$**	**Valorações válidas $\hat{\mathbb{V}}^\sigma$**	**Eventos parametrizados Δ^σ**
a	$\mathbb{V}^a = \emptyset$	-	$\{a\}$
b	$\mathbb{V}^b = \{x, y\}$	$\begin{Bmatrix}(0,0),(0,1),(0,2),(0,3),\\(1,0),(1,1),(1,2),(1,3),\\(2,0),(2,1),(2,2),(2,3)\end{Bmatrix}$	$\begin{Bmatrix}b_{xy00}, b_{xy01}, b_{xy02}, b_{xy03},\\b_{xy10}, b_{xy11}, b_{xy12}, b_{xy13},\\b_{xy20}, b_{xy21}, b_{xy22}, b_{xy23}\end{Bmatrix}$
c	$\mathbb{V}^c = \emptyset$	-	$\{c\}$
d	$\mathbb{V}^d = \{x, y\}$	$\begin{Bmatrix}(0,0),(1,0),(2,0),(3,0),\\(0,1),(1,1),(2,1),(3,1),\\(0,2),(1,2),(2,2),(3,2)\end{Bmatrix}$	$\begin{Bmatrix}d_{xy00}, d_{xy10}, d_{xy20}, d_{xy30},\\d_{xy01}, d_{xy11}, d_{xy21}, d_{xy31},\\d_{xy02}, d_{xy12}, d_{xy22}, d_{xy32}\end{Bmatrix}$
e	$\mathbb{V}^e = \{x, y\}$	$\begin{Bmatrix}(0,0),(0,1),(0,2),(0,3),\\(1,0),(1,1),(1,2),(1,3),\\(2,0),(2,1),(2,2),(2,3),\\(3,0),(3,1),(3,2),(3,3)\end{Bmatrix}$	$\begin{Bmatrix}e_{xy00}, e_{xy01}, e_{xy02}, e_{xy03},\\e_{xy10}, e_{xy11}, e_{xy12}, e_{xy13},\\e_{xy20}, e_{xy21}, e_{xy22}, e_{xy23},\\e_{xy30}, e_{xy31}, e_{xy32}, e_{xy33}\end{Bmatrix}$
f	$\mathbb{V}^f = \emptyset$	-	$\{f\}$
g	$\mathbb{V}^g = \{y\}$	$\{(0), (1), (2), (3)\}$	$\{g_{y0}, g_{y1}, g_{y2}, g_{y3}\}$

4.2 Conversão da planta

Para converter os modelos das plantas $G_{\mathfrak{E}}$ para $G_{\mathfrak{D}}$, o Algoritmo 2 substitui cada evento σ de uma transição em $G^i_{\mathfrak{E}}$ pelos eventos no conjunto de eventos parametrizados correspondentes $\Delta^\sigma \in \Delta$. Para isso, inicialmente a estrutura de estados é construída exatamente como a do AFE de entrada. O alfabeto Δ e a relação de transição são inicializadas no início do algoritmo.

Para cada transição com um evento σ no AFE de entrada $G^i_{\mathfrak{E}}$ (linha 4), os eventos correspondentes $\Delta^\sigma \in \Delta$ são incluídos no alfabeto do autômato de saída $\Delta_{\mathfrak{D}}$ (linha 5). Então, novas transições são criadas substituindo o evento σ por cada um dos seus correspondentes eventos parametrizados em Δ^σ (linha 6). Finalmente, essas transições são incluídas no conjunto $\Gamma_{\mathfrak{D}}$.

Algoritmo 2: CONVERSÃO DA PLANTA DE $\mathfrak{E}$ PARA $\mathfrak{D}$

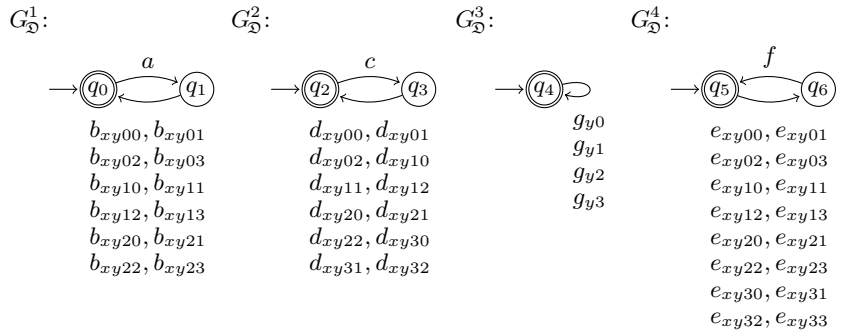
entradas: plantas com estados parametrizados $G^i_{\mathfrak{E}} = (\Sigma, V, Q, q^\circ, Q^\omega, P_V, \Gamma)$
conjunto de eventos parametrizados Δ
saída : plantas com eventos parametrizados $G^i_{\mathfrak{D}} = (\Delta_{\mathfrak{D}}, Q_{\mathfrak{D}}, q^\circ_{\mathfrak{D}}, Q^\omega_{\mathfrak{D}}, \Gamma_{\mathfrak{D}})$

início
  $Q_{\mathfrak{D}} \leftarrow Q$, $Q^\omega_{\mathfrak{D}} \leftarrow Q^\omega$, $\Delta_{\mathfrak{D}} \leftarrow \emptyset$, $\Gamma_{\mathfrak{D}} \leftarrow \emptyset$
  $q^\circ_{\mathfrak{D}} \leftarrow q^\circ$
  para cada *transição* $q_1 \xrightarrow{\sigma:p} q_2 \in \Gamma$, *tal que* $q_1, q_2 \in Q$, $\sigma \in \Sigma$ *e* $p \in P_V$ **faça**
    $\Delta_{\mathfrak{D}} \leftarrow \Delta_{\mathfrak{D}} \cup \Delta^\sigma$
    $\Gamma_{\mathfrak{D}} \leftarrow \Gamma_{\mathfrak{D}} \cup \left\{ q_1 \xrightarrow{\delta} q_2 \right\}$, para todos os eventos parametrizados $\delta \in \Delta^\sigma$
  fim
  retorna $G^i_{\mathfrak{D}}$
fim

4.2.1 Exemplo - Algoritmo 2

Utilizando o alfabeto Δ, gerado pelo Algoritmo 1 e mostrado no Quadro 1, é possível criar os autômatos com eventos parametrizados mostrados na Figura 24, fornecendo como entrada os AFE da Figura 15. Observa-se que cada evento $\sigma \in \Sigma$ é substituído pelo correspondente $\Delta^\sigma \in \Delta$.

Figura 24 – Planta - Modelo convertido de $\mathfrak{E}$ para $\mathfrak{D}$



4.3 Extração da alternância dos contextos

Nota-se que os modelos resultantes do Algoritmo 2 precisam de um modelo complementar para determinar a alternância de contextos (modelo do filtro). Sem o filtro, eventos parametrizados se tornam ambíguos com relação ao evento original representado na planta do SED. Por exemplo, na Figura 24 a ocorrência dos eventos parametrizados de b, d, g e e em Δ, isto é, eventos em Δ^b, Δ^d, Δ^e e Δ^g são ambíguos.

Para evitar esse problema, os Algoritmos 3 e 4 usam as fórmulas de atualização no modelo da planta $G_{\mathfrak{E}}$ para criar um conjunto, denotado H, de modelos adicionais chamados filtros. Os modelos em H implementam a alternância de contexto, com base em como as variáveis em V alteram seus valores quando tais atualizações ocorrem.

Neste trabalho, por questões de clareza, a construção de H é ilustrada em duas etapas: Inicialmente, utiliza-se o Algoritmo 3 para calcular um conjunto intermediário (H_V) de filtros H_{v_i}, cada um implementando a alternância de contexto de uma variável particular $v_i \in V$; Então, utiliza-se o conjunto H_V e o Algoritmo 4 para se calcular o conjunto final H de filtros modulares de dois estados, tal que $H_{\mathfrak{D}} = H_V^{\|} = H^{\|}$.

Algoritmo 3: EXTRAÇÃO DA ALTERNÂNCIA DE CONTEXTOS DE $\mathfrak{E}$ PARA $\mathfrak{D}$

entradas : modelo da planta $G_{\mathfrak{E}} = (\Sigma, V, Q, q^{\circ}, Q^{\omega}, P_V, \Gamma)$
conjunto de eventos parametrizados Δ
saída : conjunto H_V de autômatos com eventos parametrizados não modulares implementando a alternância de contextos

1 **início**
2 $H_V \leftarrow \emptyset$
3 **para cada** *variável* $v_i \in V$, *tal que* $Dom(v_i) = \{\overline{v_i}_1, \cdots, \overline{v_i}_n\}$ *e* $v_i^o \in Dom(v_i)$ **faça**
4 $Q_{v_i} \leftarrow \{q_{\overline{v_i}_1}, \cdots, q_{\overline{v_i}_n}\}$
5 $q_{v_i}^{\circ} \leftarrow q_{v_i^o}$, $Q_{v_i}^{\omega} \leftarrow Q_{v_i}$
6 $\Delta^{v_i} \leftarrow \emptyset$, $\Gamma_{v_i} \leftarrow \emptyset$
7 **para cada** *transição* $q \xrightarrow{\sigma:p} \in \Gamma$ **faça**
8 **para cada** *evento parametrizado* $\sigma_{\mathbb{V}^\sigma \hat{\mathbb{V}}^\sigma} \in \Delta^\sigma$, *tal que* $v_i \in \mathbb{V}^\sigma$ **faça**
9 $atual \leftarrow \hat{\mathbb{V}}^\sigma(v_i)$
10 **se** $v_i \in \mathcal{V}_p'$ **então**
11 $prox \leftarrow p(\hat{\mathbb{V}}^\sigma(\mathcal{V}_p))$
12 **senão**
13 $prox \leftarrow atual$
14 **fim**
15 $\Delta^{v_i} \leftarrow \Delta^{v_i} \cup \left\{\sigma_{\mathbb{V}^\sigma \hat{\mathbb{V}}^\sigma}\right\}$
16 $\Gamma_{v_i} \leftarrow \Gamma_{v_i} \cup \left\{q_{atual} \xrightarrow{\sigma_{\mathbb{V}^\sigma \hat{\mathbb{V}}^\sigma}} q_{prox}\right\}$
17 **fim**
18 **fim**
19 $H_{v_i} \leftarrow (\Delta^{v_i}, Q_{v_i}, q_{v_i}^{\circ}, Q_{v_i}^{\omega}, \Gamma_{v_i})$
20 $H_V \leftarrow H_V \cup \{H_{v_i}\}$
21 **fim**
22 **retorna** H_V
23 **fim**

No Algoritmo 3, inicialmente o conjunto H_V é inicializado como vazio. Então, para cada variável $v_i \in V$ um filtro com eventos parametrizados H_{v_i} é criado e adicionado ao conjunto H_V. O processo de criar cada autômato H_{v_i} é: primeiro, um conjunto de estados Q_{v_i}, representando cada possível valor no domínio da variável v_i, é criado (linha 4); segundo, o estado inicial $q_{v_i}^{\circ}$ é definido como o estado correspondente ao valor inicial $v_i^o \in Dom(v_i)$ da variável v_i; terceiro, todos os estados são marcados ($Q_{v_i}^{\omega} = Q_{v_i}$) e os conjuntos Δ^{v_i} e Γ_{v_i} são inicializados (linha 6); então , todas as transições e eventos parametrizados de H_{v_i} são criados com base em como cada variável v_i altera seu valor nas transições.

Nesse sentido, cada evento σ em uma transição $q \xrightarrow{\sigma:p} \in \Gamma$ tem um conjunto de eventos parametrizados correspondente Δ^σ. Ainda, cada evento parametrizado $\sigma_{\mathbb{V}^\sigma \hat{\mathbb{V}}^\sigma} \in \Delta^\sigma$ identifica: (i) quais variáveis influenciam na representação dos contextos de σ, em $\mathbb{V}^\sigma$; e (ii) quais os valores dessas variáveis antes de σ ocorrer, em $\hat{\mathbb{V}}^\sigma$, isto é, a valoração de estado

atual. O valor assumido depois de σ pode ser calculado aplicando a fórmula p em $\hat{\mathbb{V}}^{\sigma}$.

Portanto, como cada estado em Q_{v_i} representa um valor no domínio da variável v_i, é possível criar transições em Γ_{v_i} identificadas $\sigma_{\mathbb{V}^{\sigma}\hat{\mathbb{V}}^{\sigma}}$, tal que: o estado atual (q_{atual}) identifica o valor atual de v_i (isto é, $\hat{\mathbb{V}}^{\sigma}(v_i)$ (linha 9)); e o próximo estado (q_{prox}) identifica o valor assumido por v_i, depois da transição (isto é, $p(\hat{\mathbb{V}}^{\sigma}(\mathcal{V}_p))$ (linha 11)). Quando $q \xrightarrow{\sigma:p}$ não altera o valor de v_i, isto é, $v_i \notin \mathcal{V}'_p$, os estados atual e próximo são tidos como o mesmo (linha 13).

O evento $\sigma_{\mathbb{V}^{\sigma}\hat{\mathbb{V}}^{\sigma}} \in \Delta^{\sigma}$ é adicionado ao conjunto de eventos Δ^{v_i} (linha 15) e a transição criada é adicionada a Γ_{v_i}. O modelo do filtro não modular criado H_{v_i} é adicionado ao conjunto H_V, que é retornado no final do algoritmo.

4.3.1 Exemplo - Algoritmo 3

Dado o alfabeto Δ, resultado do Algoritmo 1, é possível automaticamente criar os filtros na Figura 25, dando como entrada as plantas na Figura 15 ao Algoritmo 3. Como $V = \{x, y\}$, o Algoritmo 3 cria os filtros H_x e H_y, para as variáveis x e y, respectivamente.

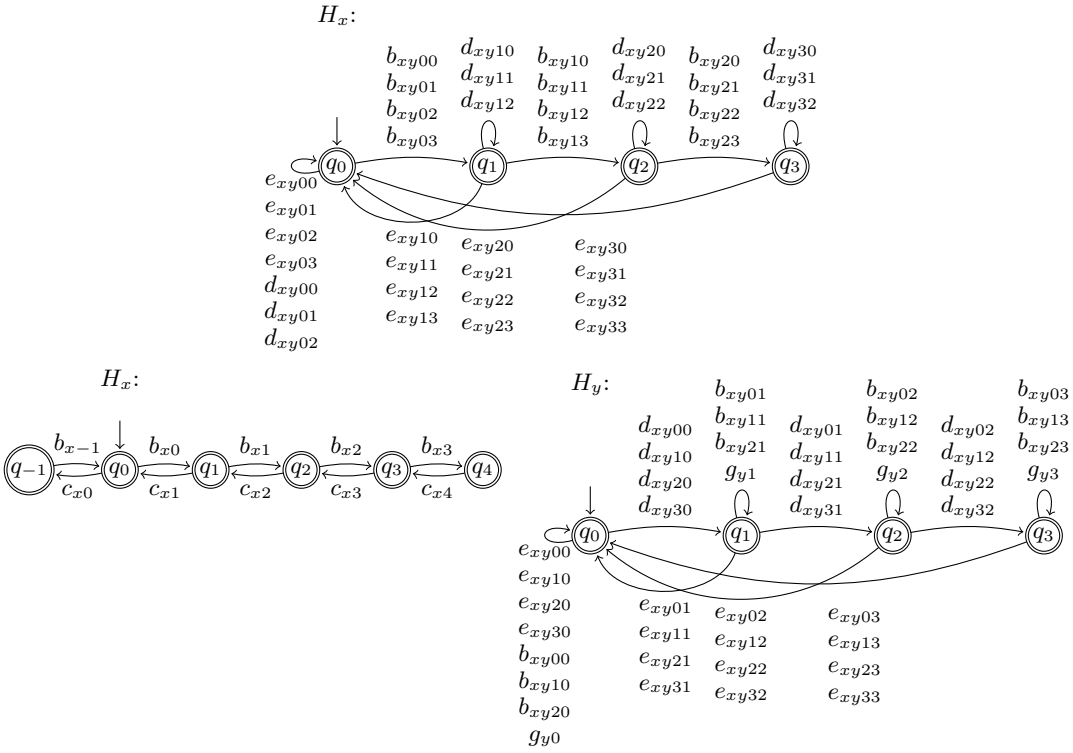
Considera-se a variável x como exemplo. O conjunto de estados $Q_x = \{q_0, q_1, q_2, q_3\}$ é criado mapeando-se os valores $Dom(x) = \{0, 1, 2, 3\}$ (linha 4). O estado inicial é definido como q_0, porque $x^o = 0$, e todos os estados são marcados. Então, para cada transição $q \xrightarrow{\sigma:p} \in \Gamma$ no modelo da planta, e com base no evento corresponde $\sigma_{\mathbb{V}^{\sigma}\hat{\mathbb{V}}^{\sigma}} \in \Delta^{\sigma}$, para $x \in \mathbb{V}^{\sigma}$, as transições de H_x são criados como segue:

- para $q \xrightarrow{b:p}$, com $p = x' \leftarrow x + 1$:
  - transições $q_0 \xrightarrow{b_{xy00}} q_1$, $q_0 \xrightarrow{b_{xy01}} q_1$, $q_0 \xrightarrow{b_{xy02}} q_1$, e $q_0 \xrightarrow{b_{xy03}} q_1$ são criadas, porque o valor de estado atual de x é 0 e o valor no próximo estado de x é $p(0) = 1$;
  - transições $q_1 \xrightarrow{b_{xy10}} q_2$, $q_1 \xrightarrow{b_{xy11}} q_2$, $q_1 \xrightarrow{b_{xy12}} q_2$, e $q_1 \xrightarrow{b_{xy13}} q_2$ são criadas, porque o valor de estado atual de x é 1 e o valor no próximo estado de x é $p(1) = 2$;
  - transições $q_2 \xrightarrow{b_{xy20}} q_3$, $q_2 \xrightarrow{b_{xy21}} q_3$, $q_2 \xrightarrow{b_{xy22}} q_3$, e $q_2 \xrightarrow{b_{xy23}} q_3$ são criadas, porque o valor de estado atual de x é 1 e o valor no próximo estado de x é $p(2) = 3$;
- para $q \xrightarrow{e:p}$, com $p = x' \leftarrow 0$:
  - transições $q_0 \xrightarrow{e_{xy00}} q_0$, $q_0 \xrightarrow{e_{xy01}} q_0$, $q_0 \xrightarrow{e_{xy02}} q_0$, $q_0 \xrightarrow{e_{xy03}} q_0$, $q_1 \xrightarrow{e_{xy10}} q_0$, $q_1 \xrightarrow{e_{xy11}} q_0$, $q_1 \xrightarrow{e_{xy12}} q_0$, $q_1 \xrightarrow{e_{xy13}} q_0$, $q_2 \xrightarrow{e_{xy20}} q_0$, $q_2 \xrightarrow{e_{xy21}} q_0$, $q_2 \xrightarrow{e_{xy22}} q_0$, $q_3 \xrightarrow{e_{xy23}} q_0$, $q_3 \xrightarrow{e_{xy30}} q_0$, $q_3 \xrightarrow{e_{xy31}} q_0$, $q_3 \xrightarrow{e_{xy32}} q_0$ e $q_3 \xrightarrow{e_{xy33}} q_0$ são criadas, porque independente do valor de estado atual de x, o valor de próximo estado de x é sempre 0;
- para $\xrightarrow{d:p}$, tal que p não atualiza x, segue que:

– transições $q_0 \xrightarrow{d_{xy00}} q_0$, $q_0 \xrightarrow{d_{xy01}} q_0$, $q_0 \xrightarrow{d_{xy02}} q_0$, $q_1 \xrightarrow{d_{xy10}} q_1$, $q_1 \xrightarrow{d_{xy11}} q_1$, $q_1 \xrightarrow{d_{xy12}} q_1$, $q_2 \xrightarrow{d_{xy20}} q_2$, $q_2 \xrightarrow{d_{xy21}} q_2$, $q_2 \xrightarrow{d_{xy22}} q_2$, $q_3 \xrightarrow{d_{xy30}} q_3$, $q_3 \xrightarrow{d_{xy31}} q_3$ e $q_3 \xrightarrow{d_{xy32}} q_3$ são criadas como auto-laços, porque independente do valor atual de x, o valor de próximo estado se mantém inalterado;

O mesmo pode ser aplicado à variável y, resultando no filtro H_y mostrado na Figura 25. Assim, $H_{\mathfrak{D}} = H_V^{\parallel} = \{H_x, H_y\}$ pode ser expressado como um conjunto de filtros não modulares.

Figura 25 – Filtros não modulares



A seguir mostra-se como obter a versão modular de H_V.

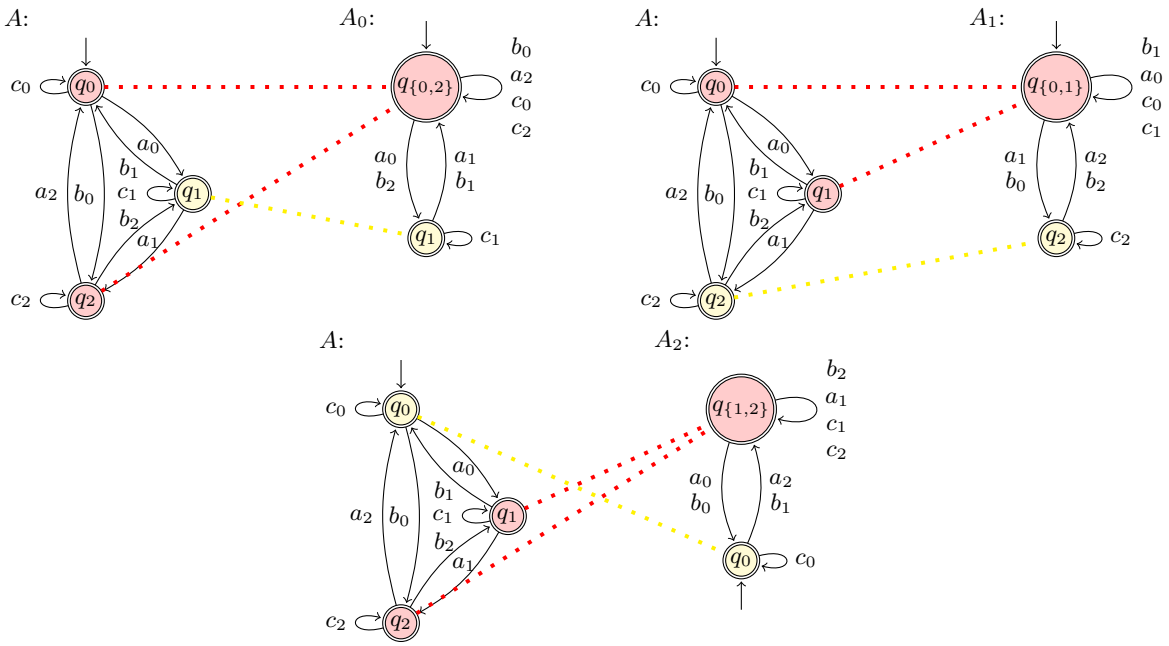
4.3.2 Conversão de filtro não modular para modular

O conceito de *superestados* tem importância central na construção dos modelos modulares a seguir. Para um AFD A, é possível criar um AFD A_1, tal que um grupo de estados em A é representado por apenas um estado em A_1. Esse estado é chamado de *superestado* (BASSINO; BÉAL; PERRIN, 1998). Cada maneira diferente de agrupar estados em A leva a um AFD A_i menos restritivo em termos de comportamento, tal que $A \sqsubseteq A_i$.

A Figura 26 ilustra a ideia de superestados com um exemplo: o AFD A possui 3 estados (q_0, q_1 e q_2); e os AFDs A_0 A_1 e A_2 possuem 2 estados cada. Em A_0, os estados q_1

e q_2 de A são combinados no superestado $q_{\{1,2\}}$, enquanto que o q_0 continua inalterado. As transições entre os estados q_1 e q_2 em A se tornam auto-laços no superestado $q_{\{1,2\}}$. O mesmo ocorre para A_1 e A_2, que possuem um superestado cada, $q_{\{0,2\}}$ e $q_{\{0,1\}}$. Note que $A_0 \| A_1 \| A_2 = A$.

Figura 26 – Exemplo de superestados



A ideia de superestados é utilizada para modularizar os modelos $H_{v_i} \in H_V$. Isso leva a modelos menores de 2 estados, que como mostrado no Capítulo 5, tem potencial de benefício na etapa de síntese. O Algoritmo 4 recebe como entrada o conjunto H_V e constrói o conjunto H que inclui somente modelos de filtro de 2 estados.

A ideia principal do algoritmo é estabelecido na linha 4, que cria todas as possíveis combinações j de dois conjuntos de estados u_{ij} e c_{ij}. Assume-se que u_{ij} é um conjunto unitário incluindo um estado de Q_{v_i}, e que c_{ij} é o seu conjunto complemento que inclui todos os outros estados de Q_{v_i}, exceto u_{ij}. Em conjunção, cada combinação j de u_{ij} e c_{ij} modela um estado $q_{u_{ij}}$ e um superestado $q_{c_{ij}}$ de um modelo modular de filtro H_{ij} com dois estados.

O Algoritmo 4 primeiramente define o alfabeto Δ_{ij} como o mesmo que Δ_{v_i}. Então, o conjunto de estados é criado com base nos dois conjuntos, u_{ij} e c_{ij}, tal que $Q_{ij} = \{q_{u_{ij}}, q_{c_{ij}}\}$ (linhas 6 e 7). O estado inicial q°_{ij} é definido como $q_{u_{ij}}$, se $q^{\circ}_{vi} \in u_{ij}$, ou como $q_{c_{ij}}$, caso contrário. O conjunto de estados marcados e a relação de transição são inicializados.

Para cada transição $q_1 \xrightarrow{\delta} q_2 \in \Gamma_{vi}$ em H_{v_i}, uma nova transição é criada em H_{ij} verificando se os estados q_1 e q_2 ou são o estado $q_{u_{ij}}$ ou estão contidos no superestado $q_{c_{ij}}$ (linhas 17 até 23). Na linha 26 o filtro modular H_{ij} é construído e adicionado ao conjunto H. Repetindo esse processo para todos os elementos em H_V, o algoritmo leva a

Algoritmo 4: MODULARIZAÇÃO DOS MODELOS DO FILTRO $H_{\mathfrak{D}}$

entradas: conjunto H_V de filtros não modulares
saída : conjunto H de filtros modulares de 2 estados

início
  $H \leftarrow \emptyset$
  para cada *modelo* $H_{vi} \in H_V$, *tal que* $H_{vi} = (\Delta_{vi}, Q_{vi}, q^{\circ}_{vi}, Q^{\omega}_{vi}, \Gamma_{vi})$ **faça**
    $C \leftarrow \bigcup\{u_{ij}, c_{ij}\}$, tal que $u_{ij} \neq \emptyset$, $c_{ij} \neq \emptyset$, $u_{ij} \cup c_{ij} = Q_{vi}$, $u_{ij} \cap c_{ij} = \emptyset$ e $|u_{ij}| = 1$
    para cada $\{u_{ij}, c_{ij}\} \in C$ **faça**
      $\Delta_{ij} \leftarrow \Delta_{vi}$
      $Q_{ij} \leftarrow \{q_{u_{ij}}, q_{c_{ij}}\}$
      se $q^{\circ}_{vi} \in u_{ij}$ **então**
        $q^{\circ}_{ij} \leftarrow q_{u_{ij}}$
      senão
        $q^{\circ}_{ij} \leftarrow q_{c_{ij}}$
      fim
      $Q^{\omega}_{ij} \leftarrow Q_{ij}$
      $\Gamma_{ij} \leftarrow \emptyset$
      para cada *transição* $q_1 \xrightarrow{\delta} q_2 \in \Gamma_{vi}$ **faça**
        se $q_1 \in u_{ij}$ *e* $q_2 \in u_{ij}$ **então**
          $\Gamma_{ij} \leftarrow \Gamma_{ij} \cup \left\{q_{u_{ij}} \xrightarrow{\delta} q_{u_{ij}}\right\}$
        senão se $q_1 \in c_{ij}$ *e* $q_2 \in c_{ij}$ **então**
          $\Gamma_{ij} \leftarrow \Gamma_{ij} \cup \left\{q_{c_{ij}} \xrightarrow{\delta} q_{c_{ij}}\right\}$
        senão se $q_1 \in u_{ij}$ *e* $q_2 \in c_{ij}$ **então**
          $\Gamma_{ij} \leftarrow \Gamma_{ij} \cup \left\{q_{u_{ij}} \xrightarrow{\delta} q_{c_{ij}}\right\}$
        senão
          $\Gamma_{ij} \leftarrow \Gamma_{ij} \cup \left\{q_{c_{ij}} \xrightarrow{\delta} q_{u_{ij}}\right\}$
        fim
      fim
      $H_{ij} \leftarrow (\Delta_{ij}, Q_{ij}, q^{\circ}_{ij}, Q^{\omega}_{ij}, \Gamma_{ij})$
      $H \leftarrow H \cup \{H_{ij}\}$
    fim
  fim
  return H
fim

um conjunto $H = \{H_1, \ldots, H_{ij}\}$ que pode ser composto para formar $H_{\mathfrak{D}} = H^{\|}$. Quando composto com o modelo da planta $G_{\mathfrak{D}}$, $H_{\mathfrak{D}}$ representa equivalentemente os comportamento dos contextos implementados pelas variáveis e fórmulas em $G_{\mathfrak{C}}$, usando um mecanismo diferente.

4.3.3 Exemplo - Algoritmo 4

Partindo dos modelos não modulares na Figura 25, o Algoritmo 4 calcula o conjunto de filtro modulares de dois estados. Por exemplo, considere o modelo $H_x = (\Delta_x, Q_x, q^{\circ}_x, Q^{\omega}_x, \Gamma_x)$ e os modelos de dois estados correspondentes na Figura 27. O conjunto de possíveis combinações de Q_x é $C = \{\{\{q_0\}, \{q_1, q_2, q_3\}\}, \{\{q_1\}, \{q_0, q_2, q_3\}\}, \{\{q_2\}, \{q_0, q_1, q_3\}\}, \{\{q_3\}, \{q_0, q_1, q_2\}\}\}$, isto é, $u_1 = \{q_0\}$ implica em $c_1 = \{q_1, q_2, q_3\}$; $u_2 = \{q_1\}$ implica em $c_2 = \{q_0, q_2, q_3\}$; $u_3 = \{q_2\}$ implica em $c_3 = \{q_0, q_1, q_3\}$; e $u_4 = \{q_3\}$ implica

$c_4 = \{q_0, q_1, q_2\}$. Assim, para cada u_j existe um modelo de filtro de dois estados H_{x_j} com construção exemplificada a seguir.

Dado $u_1 = \{q_0\}$ e $c_1 = \{q_1, q_2, q_3\}$, o estado inicial é q_{u_1}, porque $q_x^\circ \in u_1$ (linha 9) e o conjunto de estados é $Q_{x_1} = Q_{x_1}^\omega = \{q_{u_1}, q_{c_1}\} = \{q_{\{0\}}, q_{\{1,2,3\}}\}$. Ainda, para cada transição em Γ_x, uma transição é criada em Γ_{x_1}, tal que:

- transições com e_{xy00}, e_{xy01}, e_{xy02}, e_{xy03}, d_{xy00}, d_{xy01}, e d_{xy02}, são auto-laços no estado $q_0 \in Q_x$. Portanto, as transições criadas correspondentes em Γ_{x_1} também são auto-laços no estado q_{u_1}, porque $q_0 \in u_1$;
- transições com b_{xy00}, b_{xy01}, b_{xy02}, e b_{xy03}, partem de q_0 para $q_1 \in Q_x$. Portanto, as transições criadas correspondentes em Γ_{x_1} são de q_{u_1} para q_{c_1}, porque $q_0 \in u_1$ e $q_1 \in c_1$;
- transições com e_{xy10}, e_{xy11}, e_{xy12}, e_{xy13}, e_{xy20}, e_{xy21}, e_{xy22}, e_{xy23}, e_{xy30}, e_{xy31}, e_{xy32}, e e_{xy33}, partem de q_1, q_2 ou q_3, para $q_0 \in Q_x$. Portanto, as transições criadas correspondentes em Γ_{x_1} são de q_{c_1} para q_{u_1}, porque $q_0 \in u_1$ e $q_1, q_2, q_3 \in c_1$;
- todas as outras transições de Γ_x não partem ou alcançam o estado q_0. Portanto, as transições criadas correspondentes em Γ_{x_1} são auto-laços em q_{c_1}, porque $q_0 \notin c_1$;

É possível repetir esse processo para: $u_2 = \{q_1\}$ e $c_2 = \{q_0, q_2, q_3\}$, que resulta no filtro H_{x_2}; $u_3 = \{q_2\}$ e $c_3 = \{q_0, q_1, q_3\}$, que resulta no filtro H_{x_3}; e $u_4 = \{q_3\}$ e $c_4 = \{q_0, q_1, q_2\}$, que resulta no filtro H_{x_4}. Similarmente, os filtros $H_{y_1}, H_{y_2}, H_{y_3}, H_{y_4}$ podem ser obtidos de H_y, e são mostrados na Figura 28.

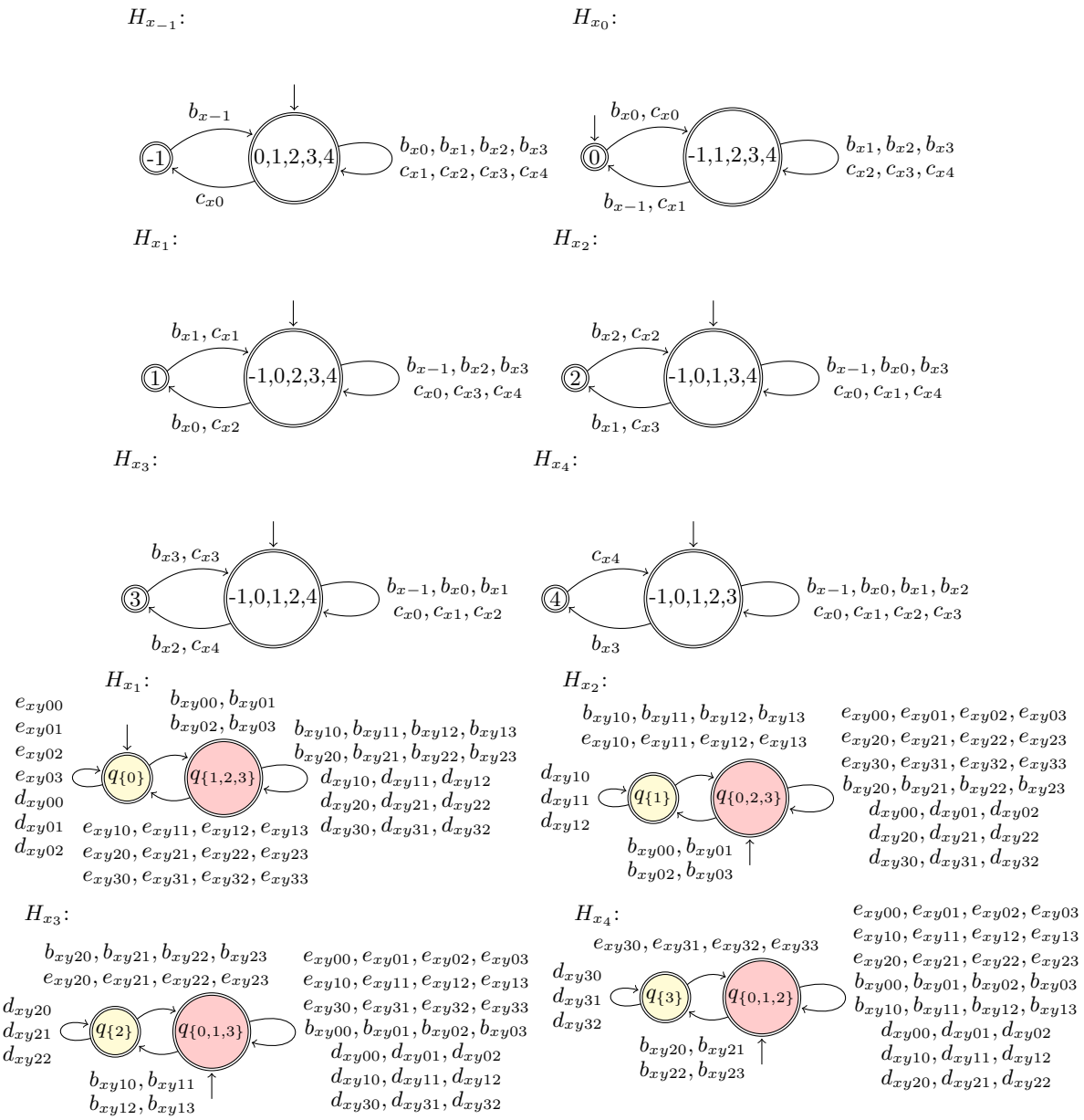
4.4 Extração das regras de controle

Além dos modelos da planta e do filtro, as regras de controle impostas pelas especificações $E_{\mathfrak{E}}$ podem ser convertidas para o modelo das especificações $E_{\mathfrak{D}}$ pelo Algoritmo 5. Para cada modelo $E_{\mathfrak{E}}^i$, cria-se um modelo com eventos parametrizados ${E_{\mathfrak{D}}}^i$ que expressa as mesmas restrições sobre a planta. Porém, em vez de desabilitar transições usando fórmulas de teste, ${E_{\mathfrak{D}}}^i$ explora o alfabeto de eventos parametrizados para reproduzir o mesmo efeito.

Para construir ${E_{\mathfrak{D}}}^i$, o Algoritmo 5 lê $E_{\mathfrak{E}}^i$ e identifica os valores de variáveis que resultam em falso nas fórmulas de teste. Então, desabilita em ${E_{\mathfrak{D}}}^i$ transições incluindo eventos que correspondem exatamente aqueles valores de variáveis.

Inicialmente, o Algoritmo 5 cria a estrutura de estados idêntica ao autômato de entrada ${E_{\mathfrak{D}}}^i$ e inicializa como vazio o alfabeto e a relação de transição. Depois, ele lê todas as transições $\xrightarrow{\sigma:p}$ de $E_{\mathfrak{E}}^i$ para criar o alfabeto e as transições em ${E_{\mathfrak{D}}}^i$. Com base em casa evento σ lido, os eventos parametrizados em Δ^σ são adicionados no alfabeto

Figura 27 – Filtros modulares para a variável x

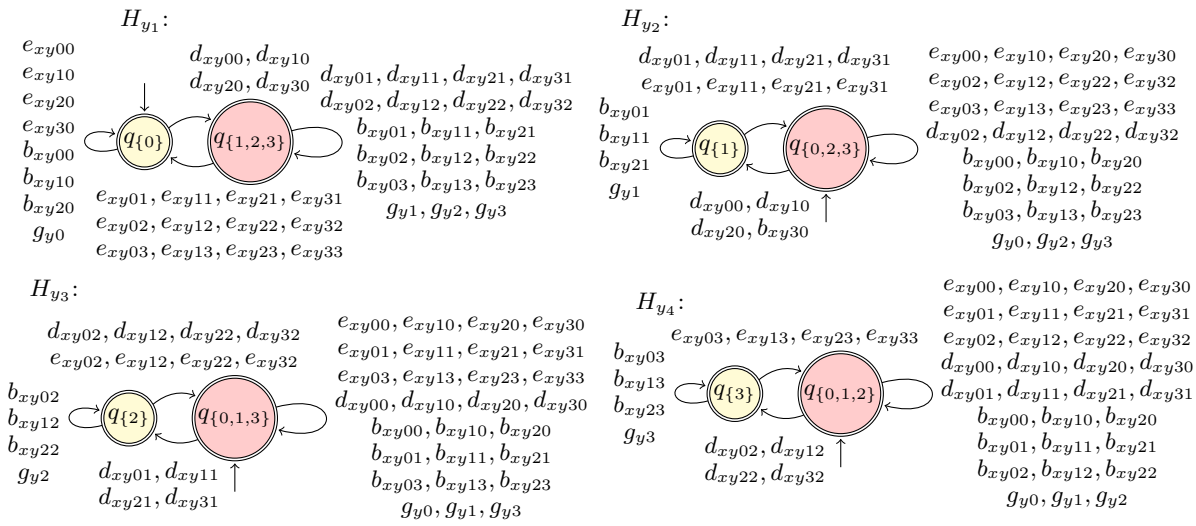


de saída (linha 5). Depois, transições no autômato de saída são criadas para os eventos parametrizados $\sigma_{\mathbb{V}\hat{\mathbb{V}}}$ em Δ^{σ}, tal que $p(\hat{\mathbb{V}}(\mathcal{V}_p)) = true$, ou seja, transições para qual a valoração $\hat{\mathbb{V}}$ é avaliada como verdadeira pela fórmula p (linha 7). Transições em $E^i_{\mathfrak{E}}$ que não tem fórmulas associadas, são diretamente adicionadas ao conjunto de transições $\Gamma_{\mathfrak{D}}$ (linha 9). Todas os outros eventos são desabilitadas, porque $p(\hat{\mathbb{V}}(\mathcal{V}_p)) = false$.

4.4.1 Exemplo - Algoritmo 5

O resultado da conversão dos AFEs na Figura 16 para modelos com eventos parametrizados é mostrado na Figura 29. Por exemplo, $E^1_{\mathfrak{E}}$ habilita um evento e somente quando $x + y > 0$ é verdadeiro. Equivalentemente, $E_{\mathfrak{D}}{}^1$ desabilita o evento e_{xy00} que

Figura 28 – Filtros modulares para a variável y



Algoritmo 5: Conversão de especificações de $\mathfrak{E}$ para $\mathfrak{D}$

entrada : especificação com estados parametrizado $E^i_{\mathfrak{E}} = (\Sigma, V, Q, q^\circ, Q^\omega, P_V, \Gamma)$
conjunto de eventos parametrizados Δ
saída : especificação com eventos parametrizados ${E_{\mathfrak{D}}}^i = (\Delta_{\mathfrak{D}}, Q_{\mathfrak{D}}, q^\circ_{\mathfrak{D}}, Q^\omega_{\mathfrak{D}}, \Gamma_{\mathfrak{D}})$

início
  $Q_{\mathfrak{D}} \leftarrow Q$, $Q^\omega_{\mathfrak{D}} \leftarrow Q^\omega$, $\Delta_{\mathfrak{D}} \leftarrow \emptyset$, $\Gamma_{\mathfrak{D}} \leftarrow \emptyset$
  $q^\circ_{\mathfrak{D}} \leftarrow q^\circ$
  para cada *transição* $q_1 \xrightarrow{\sigma:p} q_2 \in \Gamma$, *tal que* $q_1, q_2 \in Q$, $\sigma \in \Sigma$, $p \in P_V$ *e* $\mathcal{V}_p$ *é o conjunto de variáveis de* p **faça**
    $\Delta_{\mathfrak{D}} \leftarrow \Delta_{\mathfrak{D}} \cup \Delta^\sigma$
    se $\mathcal{V}_p \neq \emptyset$ **então**
      $\Gamma_{\mathfrak{D}} \leftarrow \Gamma_{\mathfrak{D}} \cup \left\{ q_1 \xrightarrow{\sigma_{\mathbb{V}\hat{\mathbb{V}}}} q_2 \right\}$, para todos os eventos $\sigma_{\mathbb{V}\hat{\mathbb{V}}} \in \Delta^\sigma$ tal que $p(\hat{\mathbb{V}}(\mathcal{V}_p)) = true$
    senão
      $\Gamma_{\mathfrak{D}} \leftarrow \Gamma_{\mathfrak{D}} \cup \left\{ q_1 \xrightarrow{\sigma} q_2 \right\}$, para todos os eventos $\sigma \in \Delta^\sigma$
    fim
  fim
  retorna ${E_{\mathfrak{D}}}^i$
fim

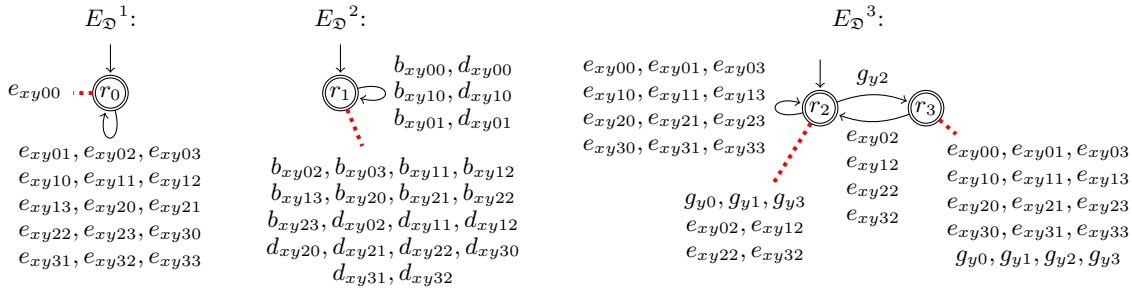
representa $\bar{x} = \bar{y} = 0$, pois $p(0,0) = falso$. Da mesma maneira $E^2_{\mathfrak{E}}$ habilita os eventos b e d somente quando $x + y < 2$ é verdadeiro. Então, ${E_{\mathfrak{D}}}^2$ desabilita os eventos parametrizados que modelam $p(\bar{x}, \bar{y}) = falso$.

4.5 Equivalência entre modelos da planta

A seguinte proposição confirma que os métodos de conversão propostos preservam o mesmo comportamento depois da conversão, isto é, que $G_{\mathfrak{D}} \| H_{\mathfrak{D}}$ é equivalente a versão explícita de $G_{\mathfrak{E}}$.

Proposição 1 *Considera-se* $G_{\mathfrak{E}}$ *um AFE que modela uma planta de um SED, e* $G_{\mathfrak{D}}$ *e* $H_{\mathfrak{D}}$

Figura 29 – Especificações - Modelo convertido de $\mathfrak{E}$ para $\mathfrak{D}$



os AFDs com eventos parametrizados calculados pelos Algoritmos 2, 3, e 4. Propõe-se que

$$L(G_{\mathfrak{E}}) = L(\Pi(G_{\mathfrak{D}} \| H_{\mathfrak{D}})).$$

Prova 1 *Dado um AFE que modela uma planta* $G_{\mathfrak{E}} = (\Sigma_{\mathfrak{E}}, V_{\mathfrak{E}}, Q_{\mathfrak{E}}, q^{\circ}_{\mathfrak{E}}, Q^{\omega}_{\mathfrak{E}}, P_{V\mathfrak{E}}, \Gamma_{\mathfrak{E}})$*, um AFD convertido* $G_{\mathfrak{D}} = (\Delta_{\mathfrak{D}}, Q_{\mathfrak{D}}, q^{\circ}_{\mathfrak{D}}, Q^{\omega}_{\mathfrak{D}}, \Gamma_{\mathfrak{D}})$ *e um modelo de filtro* $H_{\mathfrak{D}} = (\Delta_H, Q_H, q^{\circ}_H, Q^{\omega}_H, \Gamma_H)$*, considera-se* $s \in \Sigma^*$ *e* $\sigma \in \Sigma_{\mathfrak{E}}$*, tal que* $s\sigma \in \mathcal{L}(G_{\mathfrak{E}})$*, a seguir mostra-se que* $s\sigma \in \mathcal{L}(\Pi(G_{\mathfrak{D}} \parallel H_{\mathfrak{D}}))$*:*

Primeiro, $s\sigma \in \mathcal{L}(G_{\mathfrak{E}})$ *implica por construção que existe uma valoração* $\hat{v}_p \in Dom(\mathcal{V}_p)$ *válida para algum* $p \in P_V$*, e que existe também uma valoração de próximo estado* $\hat{v}'_p \in Dom(\mathcal{V}'_p)$ *tal que, para* $q_0, q_1 \in Q_{\mathfrak{E}}$*, a transição* $G_{\mathfrak{E}} \xrightarrow{s} (q_0, \hat{v}_p) \xrightarrow{\sigma} (q_1, \hat{v}'_p)$ *existe. Ainda, assume-se nesse trabalho que atualizações de variáveis são sempre exatas, assim a valoração* $\hat{v}_p$ *leva a uma única valoração de próximo estado* $\hat{v}'_p$*.*

Do Algoritmo 1, as valorações válidas $\hat{v}_p$ *associadas em uma transição a um evento* σ *levam a um conjunto* $\{\delta_0, \ldots, \delta_n\} = \Delta^{\sigma}$ *de eventos (Linha 11). Então, pelo Algoritmo 2, existem transições* $q_3 \xrightarrow{\delta_i} q_4$*, para todos os* $\delta_i \in \Delta^{\sigma}$ *com* $q_3, q_4 \in Q_{\mathfrak{D}}$*. Ainda, pelo Algoritmo 4, cada valoração possível é representada por um estado, isto é,* $\hat{v}_p$ *e* $\hat{v}'_p$ *podem ser representadas por estados* q_5 *e* $q_6 \in Q_H$*. Como as atualizações são exatas, para cada diferente valoração* $\hat{v}_p$ *existe apenas uma transição correspondente* $q_5 \xrightarrow{\delta_i} q_6 \in \Gamma_H$*, tal que* $\delta_i \in \Delta^{\sigma}$*. Portanto, depois de* $\gamma \in \mathcal{L}(G_{\mathfrak{D}} \parallel H_{\mathfrak{D}})$*, para* $\Pi(\gamma) = s$*, também* $\gamma\delta_i \in \mathcal{L}(G_{\mathfrak{D}} \parallel H_{\mathfrak{D}})$*, e como* $\Pi(\delta_i) = \sigma$ *por construção, então* $s\sigma \in \mathcal{L}(\Pi(G_{\mathfrak{D}} \parallel H_{\mathfrak{D}}))$*, que generalizado mostra que* $\mathcal{L}(G_{\mathfrak{E}}) \subseteq \mathcal{L}(\Pi(G_{\mathfrak{D}} \parallel H_{\mathfrak{D}}))$*.*

Para inclusão inversa, considera-se $\delta \in \Delta$*, e* $\gamma \in \Delta^*$*, tal que* $\gamma\delta \in \mathcal{L}(G_{\mathfrak{D}} \parallel H_{\mathfrak{D}})$ *e mostra-se que* $\Pi(\gamma\delta) \in \mathcal{L}(G_{\mathfrak{E}})$*:*

Constrói-se $G_{\mathfrak{D}} \parallel H_{\mathfrak{D}} \xrightarrow{\gamma} q_0 \xrightarrow{\delta} q_1$*, para* $q_0, q_1 \in Q_H$*. Do Algoritmo 4, a transição* $q_0 \xrightarrow{\delta} q_1$ *é construída de um conjunto de transições (Algoritmo 2) com eventos em* Δ^{σ}*, tal que* $\Pi(\delta_i) = \sigma$*, para cada* $\delta_i \in \Delta^{\sigma}$ *(Algoritmo 1). Como* $\Delta^{\sigma} \neq \emptyset$*, existe no mínimo uma valoração válida tal que* $q_2 \xrightarrow{\sigma:p} q_3$ *em* $G_{\mathfrak{E}}$*, para* $\sigma \in \Sigma_{\mathfrak{E}}$ *e* $q_2, q_3 \in Q_{\mathfrak{E}}$*. Em notação explícita, existe* $G_{\mathfrak{E}} \xrightarrow{\Pi(\gamma)} (q_2, \hat{v}_p) \xrightarrow{\sigma} (q_3, \hat{v}'_p)$ *e* $\Pi(\gamma)\sigma \in \mathcal{L}(G_{\mathfrak{E}})$*.*

4.6 Resultados experimentais

Os algoritmos propostos foram testados em alguns exemplos da literatura. Para isso implementou-se uma ferramenta disponível em https://luizsouthier.github.io/conversor. Quatro casos foram selecionados, representando processos comumente encontrados em ambientes fabris, além do exemplo proposto da Figura 10, e estão descritos como se segue:

- Exemplo 1 (WONHAM, 2010): duas máquinas interconectadas por um *buffer* com capacidade n (fixou-se $n = 2$ para testes). Esse exemplo explora a memória necessária para rastrear todas as possibilidades de inserção e retirada do *buffer*;
- Exemplo 2 (AGUIAR et al., 2013): uma linha de manufatura na qual duas máquinas concorrentes produzem diferentes tipos de peças, que são entregues a um *buffer* de 2 posições compartilhado. Dependendo da ordem em que as diferentes peças são produzidas, são tratadas diferentemente ao longo do processo. Isso significa que o modelo para este exemplo precisa memorizar cada diferente combinação produzida de peças durante todo a linha de manufatura (ROSA; TEIXEIRA; MALIK, 2018).
- Exemplo 3 (Zhong; Wonham, 1990): uma linha de produção industrial que permite retrabalho. Esse exemplo de Zhong e Wonham (1990) tem sido explorado inúmeras vezes pela literatura. Ele apresenta problemas de bloqueio (WONHAM, 2010), além de ser um bom critério de medição de complexidade de modelagem e de síntese (CURY et al., 2015), e alternativas de modularização (TEIXEIRA; CURY; QUEIROZ, 2018);
- Exemplo 4 (SILVA; RIBEIRO; TEIXEIRA, 2017): um sistema de manufatura flexível que recebe dois tipos de peças de um fornecedor externo. Máquinas e operação ao longo do sistema são coordenadas de acordo com o tipo de peça que entrou na linha de produção. Como o sistema pode produzir múltiplos tipos de peças simultaneamente, o modelo deve rastrear cada combinação possível de produtos ao longo da linha de produção, o que potencialmente pode requerer uma grande quantidade de memória e esforço de modelagem (MALIK; TEIXEIRA, 2019);
- Exemplo 5: é o exemplo mostrado na Figura 10 e utilizado ao longo desse trabalho para exemplificar os conceitos de parametrizações e ilustrar os métodos de conversão propostos.

Os exemplos foram inicialmente modelados utilizando AFEs na abordagem $\mathfrak{E}$, e então convertidos em AFDs com eventos parametrizados na abordagem $\mathfrak{D}$ pelos algoritmos propostos. Por questões de clareza os modelos não são mostrados explicitamente, mas são sumarizados em termos de quantidade de estados e transições: da planta ($G_{\mathfrak{E}}$) e especificação ($E_{\mathfrak{E}}$) com estados parametrizados bem como para a composição entrada da

etapa de síntese ($K_{\mathfrak{E}} = G_{\mathfrak{E}} \| E_{\mathfrak{E}}$) e para o supervisor ($S_{\mathfrak{E}}$); da planta ($G_{\mathfrak{D}}$), especificação ($E_{\mathfrak{D}}$) e filtro ($H_{\mathfrak{D}}$) obtidos pelos métodos de conversão, bem como o modelo de entrada da síntese $K_{\mathfrak{D}} = G_{\mathfrak{D}} \| E_{\mathfrak{D}} \| H_{\mathfrak{D}}$ e o supervisor ($S_{\mathfrak{D}}$). A Tabela 4 mostra os resultados encontrados:

Tabela 4 – Número de estados (e transições) de modelos dos problemas da literatura e modelos convertidos correspondentes

Modelo\Exemplo	**1**	**2**	**3**	**4**	**5**
$G_{\mathfrak{E}}$	14 (22)	3640 (15604)	288 (888)	921600 (7937280)	72 (380)
$E_{\mathfrak{E}}$	1 (2)	21 (193)	4 (26)	128 (2438)	2 (11)
$\boldsymbol{K_{\mathfrak{E}}}$	**10 (14)**	**3500 (9236)**	**624 (1180)**	**14580 (56376)**	**56 (136)**
$\boldsymbol{S_{\mathfrak{E}}}$	**8 (10)**	**742 (1816)**	**58 (102)**	**1551 (4596)**	**28 (57)**
$G_{\mathfrak{D}}$	4 (16)	40 (7104)	12 (200)	2048 (108032)	8 (72)
$E_{\mathfrak{D}}$	1 (6)	21 (2443)	4 (70)	128 (5632)	2 (14)
$H_{\mathfrak{D}}$	4 (6)	169 (546)	65 (306)	1200 (13200)	6 (24)
$\boldsymbol{K_{\mathfrak{D}}}$	**10 (14)**	**3500 (9236)**	**624 (1180)**	**14580 (56376)**	**56 (136)**
$\boldsymbol{S_{\mathfrak{D}}}$	**8 (10)**	**742 (1816)**	**58 (102)**	**1551 (4596)**	**28 (57)**

Como esperado, $K_{\mathfrak{E}}$ e $K_{\mathfrak{D}}$ tem o mesmo número de estados e transições para todos os exemplos, assim como os supervisores $S_{\mathfrak{E}}$ e $S_{\mathfrak{D}}$ correspondentes, o que pode ser verificado, por exemplo, por intrusão de linguagem (CASSANDRAS; LAFORTUNE, 2009) ou bissimulação (BARRETT; LAFORTUNE, 1998).

No próximo Capítulo, exploram-se as etapas de projeto de controladores com síntese eficiente, isto é, que tiram vantagens dos modelos de filtros modularizados de 2 estados criados pelos algoritmos de conversão, além de implementação descentralizada.

5 Projeto de controladores com síntese e implementação eficientes

Esse capítulo evidencia como os métodos de conversão propostos e os modelos obtidos, especialmente os filtros modularizados, podem ser utilizados em conjunção com abordagens de síntese e implementação presentes na literatura. Inicialmente relembra-se que os métodos de conversão propostos permitem que o projetista construa modelos de planta $G_{\mathfrak{E}}$ e especificação $E_{\mathfrak{E}}$ utilizando fórmulas que são avaliadas sobre variáveis. Dessa maneira a etapa de modelagem atinge um conjunto maior de problemas que a modelagem sem parâmetros, visto que as variáveis podem ser utilizadas para memorização de contextos e as fórmulas implementam facilmente a alternância entre contextos.

Utilizando os métodos de conversão propostos, os modelos $G_{\mathfrak{E}}$ e $E_{\mathfrak{E}}$ são automaticamente convertidos para os modelos com eventos parametrizados $G_{\mathfrak{D}}$, $E_{\mathfrak{D}}$ e $H_{\mathfrak{D}}$. Evidencia-se que o modelo que implementa a alternância de contextos, $H_{\mathfrak{D}}$, é implementado por um conjuntos e autômatos modulares de 2 estados. Evidencia-se também que o projetista não tem o esforço de criar o modelo do filtro e nem de realizar a modularização, o que normalmente é uma tarefa manual na literatura, mas com os algoritmos propostos por esse trabalho se dá automaticamente.

De posse dos modelos $G_{\mathfrak{D}}$, $E_{\mathfrak{D}}$ e $H_{\mathfrak{D}}$, tal que $H_{\mathfrak{D}}$ é representado por um conjunto de autômatos modulares, é possível utilizar o método de síntese eficiente proposta por (CURY et al., 2015) com a técnica de Rosa, Teixeira e Malik (2018) e o método de implementação eficiente herdada de Rosa et al. (2017). As seções a seguir detalham e exemplificam tais métodos.

5.1 Síntese eficiente com AFDs com eventos parametrizados

O filtro $H_{\mathfrak{D}}$ calculado é preciso. Porém, a literatura propõe (CURY et al., 2015) que, sob certas condições, a síntese com modelos com eventos parametrizados seja simplificada incluindo apenas *parte* (alguns módulos) do modelo do filtro $H_{\mathfrak{D}}$. Todavia, o supervisor $S_{\mathfrak{D}}$ calculado mantem-se o mesmo. O modelo da planta que *abstrai* uma parte de um filtro $H_{\mathfrak{D}}$ preciso é chamado de *aproximação*.

Definição 1 *Para uma planta G, considera-se que $G_{\mathfrak{D}} = \Pi^{-1}(G)$ e que $H_{\mathfrak{D}}$ é um filtro* preciso *para $G_{\mathfrak{D}}$. $G_{\mathfrak{A}}$ é dita uma aproximação para $G_{\mathfrak{D}} \| H_{\mathfrak{D}}$ se existir ao menos um evento $\sigma \in \Sigma$ em G tal que $|\Gamma^{\Delta^\sigma}(q)| > 1$, for $q \in Q_{G_{\mathfrak{A}}}$.*

Definição 1 implica que existe um par $\delta_i, \delta_j \in \Delta^\sigma$, tal que sua ocorrência não é precisa em ao menos um estado em $G_{\mathfrak{A}}$. Na prática isso significa que um contexto a ser assumido pela planta será ambíguo, eventualmente.

Em termos de controle, aproximações podem levar a soluções de controle mais restritivas. Já que a síntese pode não conseguir distinguir os contextos, então deve desabilitar qualquer contexto associado com um evento indesejável, levando a controladores que, em geral, são mais restritivos. Para uma dada aproximação $G_{\mathfrak{A}}$, um filtro preciso $H_{\mathfrak{D}}$, e uma especificação $E_{\mathfrak{D}}$, tal que $K_{\mathfrak{A}} = G_{\mathfrak{A}} \| E_{\mathfrak{D}}$, é mostrado por Cury et al. (2015) que $\text{sup}\mathcal{C}(K_{\mathfrak{A}}, G_{\mathfrak{A}}) \cap \mathcal{L}(H_{\mathfrak{D}}) \subseteq \text{sup}\mathcal{C}(K_{\mathfrak{D}}, G_{\mathfrak{D}})$, tal que $\text{sup}\mathcal{C}(K_{\mathfrak{A}}, G_{\mathfrak{A}}) = \mathcal{L}(S_{\mathfrak{A}})$

Porém, em geral, o comportamento de controle mais restritivo não é desejado. A igualdade pode ser obtida sistematicamente selecionando *partes* de $H_{\mathfrak{D}}$ para construir $G_{\mathfrak{A}}$ (ROSA; TEIXEIRA; MALIK, 2018). Para isso é essencial que $H_{\mathfrak{D}}$ seja construído modularmente, o que é um dos resultados desse trabalho. Quando $H_{\mathfrak{D}}$ é dado por um conjunto de autômatos modulares, como no resultado do Algoritmo 4, a abordagem de Rosa, Teixeira e Malik (2018) seleciona quais partes de $H_{\mathfrak{D}}$ são necessárias para a síntese. Isso leva a composição $H_{\mathfrak{A}}$ de módulos, para $\mathcal{L}(H_{\mathfrak{D}}) \subseteq \mathcal{L}(H_{\mathfrak{A}})$.

Então, a aproximação eficiente $G_{\mathfrak{A}}\text{`} = G_{\mathfrak{D}} \| H_{\mathfrak{A}}$ pode ser obtida tal que, para $K_{\mathfrak{A}}\text{`} = G_{\mathfrak{A}}\text{`} \| E_{\mathfrak{D}}$ implicando em $\mathcal{L}(S_{\mathfrak{A}}\text{`} \| H_{\mathfrak{D}}) = \mathcal{L}(S_{\mathfrak{D}})$. Isso significa que o supervisor abstraído $S_{\mathfrak{A}}\text{`}$ pode ser obtido pelo cálculo de síntese sobre o modelo $K_{\mathfrak{A}}\text{`}$, que inclui partes de $H_{\mathfrak{D}}$, e depois composto com $H_{\mathfrak{D}}$ para obter a solução equivalente de controle, isto é, composição obtida $S_{\mathfrak{A}}\text{`} \| H_{\mathfrak{D}}$ é equivalente em comportamento à solução implementada por $S_{\mathfrak{D}}$.

Os ganhos dessa abordagem são quantificados na Tabela 5 utilizando-se os exemplos da literatura apresentados no capítulo anterior.

É possível observar que os supervisores resultantes da síntese eficiente ($S_{\mathfrak{A}}\text{`} \| H_{\mathfrak{D}}$) têm a quantidade de estados e transições equivalente aos supervisores calculados tanto pela abordagem com estados parametrizados ($S_{\mathfrak{E}}$), quanto pela abordagem com eventos parametrizados ($S_{\mathfrak{D}}$). Em contrapartida, o cálculo da síntese eficiente se dá sobre modelos abstraídos ($K_{\mathfrak{A}}\text{`}$) com quantidade de estados e transições menor que os modelos $K_{\mathfrak{E}}$ e $K_{\mathfrak{D}}$. Isso significa que o esforço computacional para se calcular supervisores equivalentes é menor ou igual na síntese eficiente. Destaca-se que, como a síntese eficiente depende de modelos de filtro modularizados, os métodos de conversão propostos no capítulo anterior se integram diretamente a essa abordagem. A próxima seção explora características de implementação descentralizada de modelos com eventos parametrizados.

Tabela 5 – Número de estados (e transições) de modelos dos problemas da literatura, modelos convertidos e modelos da síntese eficiente

Modelo\Exemplo	**1**	**2**	**3**	**4**	**5**
$G_{\mathfrak{E}}$	14 (22)	3640 (15604)	288 (888)	921600 (7937280)	72 (380)
$E_{\mathfrak{E}}$	1 (2)	21 (193)	4 (26)	128 (2438)	2 (11)
$\boldsymbol{K_{\mathfrak{E}}}$	**10 (14)**	**3500 (9236)**	**624 (1180)**	**14580 (56376)**	**56 (136)**
$\boldsymbol{S_{\mathfrak{E}}}$	**8 (10)**	**742 (1816)**	**58 (102)**	**1551 (4596)**	**28 (57)**
$G_{\mathfrak{D}}$	4 (16)	40 (7104)	12 (200)	2048 (108032)	8 (72)
$E_{\mathfrak{D}}$	1 (6)	21 (2443)	4 (70)	128 (5632)	2 (14)
$H_{\mathfrak{D}}$	4 (6)	169 (546)	65 (306)	1200 (13200)	6 (24)
$\boldsymbol{K_{\mathfrak{D}}}$	**10 (14)**	**3500 (9236)**	**624 (1180)**	**14580 (56376)**	**56 (136)**
$\boldsymbol{S_{\mathfrak{D}}}$	**8 (10)**	**742 (1816)**	**58 (102)**	**1551 (4596)**	**28 (57)**
$H_{\mathfrak{A}}$	2 (6)	81 (702)	2 (18)	24 (876)	16 (56)
$G_{\mathfrak{A}}{}' = G_{\mathfrak{D}} \| H_{\mathfrak{A}}$	8 (20)	1800 (11040)	24 (304)	49152 (1173504)	128 (480)
$\boldsymbol{K_{\mathfrak{A}}{}' = G_{\mathfrak{A}}{}' \| E_{\mathfrak{D}}}$	**8 (16)**	**2400 (8976)**	**68 (502)**	**13680 (153616)**	**56 (136)**
$S_{\mathfrak{A}}{}'$	6 (11)	605 (2134)	20 (112)	2256 (20572)	28 (57)
$\boldsymbol{S_{\mathfrak{A}}{}' \| H_{\mathfrak{D}}}$	**8 (10)**	**742 (1816)**	**58 (102)**	**1551 (4596)**	**28 (57)**

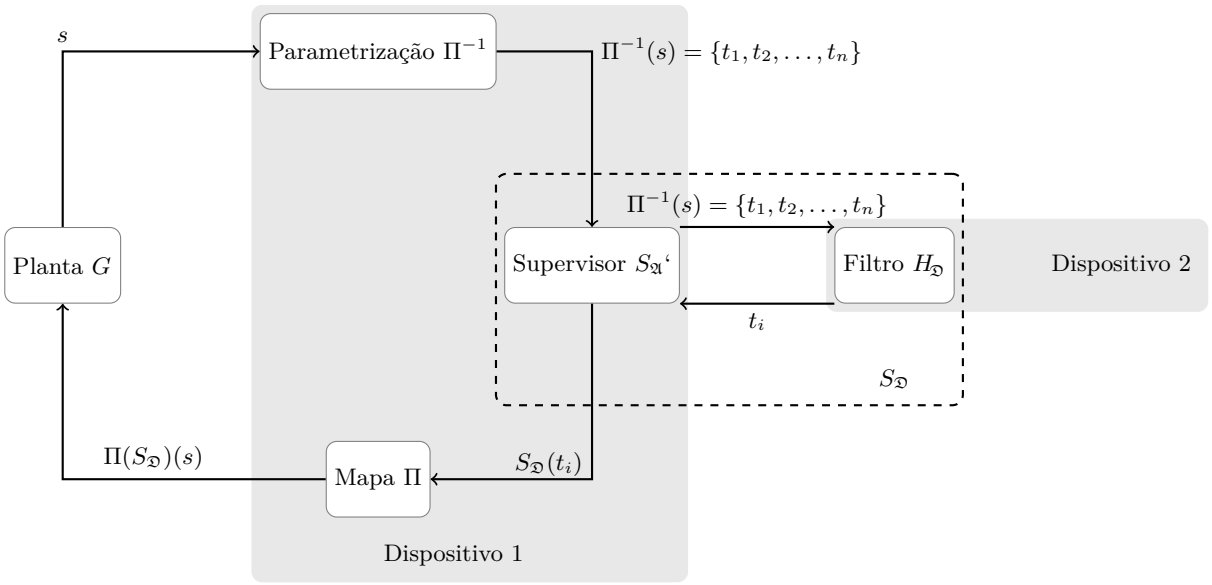
5.2 Implementação descentralizada com AFDs com eventos parametrizados

A fim de se explorar todas as vantagens do uso de parâmetros no projeto de SEDs, a síntese eficiente proposta na seção anterior pode ser implementada de maneira descentralizada como mostrado por Rosa, Teixeira e Malik (2018). Considerando um supervisor abstraído $S_{\mathfrak{A}}{}'$ que foi obtido pelo cálculo de síntese sobre o modelo $K_{\mathfrak{A}}{}' = G_{\mathfrak{A}}{}' \| E_{\mathfrak{D}}$, que inclui partes de um filtro $H_{\mathfrak{D}}$, e que $S_{\mathfrak{A}}{}' \| H_{\mathfrak{D}}$ implementa a solução ótima de controle tal como $S_{\mathfrak{D}}$, é possível implementar tal solução em dispositivos separados, tal como mostra a Figura 30.

Observa-se que a implementação não é realizada por meio de um único autômato $S_{\mathfrak{D}}$, mas por dois conjuntos de autômatos em dispositivos diferentes que se comunicam a fim de garantir a sincronia $S_{\mathfrak{A}}{}' \| H_{\mathfrak{D}}$ concorrentemente. Depois de uma cadeia de eventos $s \in \mathcal{L}(G)$, a planta G espera que o supervisor $S_{\mathfrak{D}}$ responda habilitando um conjunto de eventos tal que a planta esteja sob controle. Como o controle é realizado em Δ e a planta reconhece eventos em Σ, então a planta espera uma resposta $\Pi(S_{\mathfrak{D}})$ do supervisor. Porém, o supervisor $S_{\mathfrak{D}}$ não está diretamente implementado: o seu comportamento é realizado pela composição $S_{\mathfrak{A}}{}' \| H_{\mathfrak{D}}$.

Após uma cadeia $s \in \Sigma^*$ o Dispositivo 1 mapeia essa cadeia tal que um conjunto $\Pi^{-1}(s) = \{t_1, t_2, \ldots, t_n\}$ de possíveis cadeias correspondentes podem ser habilitadas em

Figura 30 – Implementação descentralizada de controladores



Δ^*. Como $\Pi^{-1}(s)$ não é necessariamente preciso ($|\Pi^{-1}(s)| \geq 1$), então o supervisor $S_{\mathfrak{A}}$‘ não é capaz de por si só implementar a ação de controle. Assim, o conjunto $\Pi^{-1}(s) = \{t_1, t_2, \ldots, t_n\}$ é enviado para o Dispositivo 2 que implementa o filtro preciso $H_{\mathfrak{D}}$. O filtro escolhe qual contexto t_i em $\{t_1, t_2, \ldots, t_n\}$ deve ser utilizado em cada momento e devolve tal informação para o Dispositivo 1.

De posse da cadeia t_i, $S_{\mathfrak{A}}$‘ aplica a ação de controle $S_{\mathfrak{A}}$‘(t_i) e devolve a ação à planta. Para isso, é necessário que a ação de controle seja representada em termos do alfabeto original Σ. Assim, o mapa Π realiza a conversão de $S_{\mathfrak{A}}$‘(t_i) para $\Pi(S_{\mathfrak{D}})(s)$. Observa-se em tracejado que a ação de controle equivalente a $S_{\mathfrak{D}}$ é implementada pelas estruturas $S_{\mathfrak{A}}$‘ e $H_{\mathfrak{D}}$. Pode ser mostrado (ROSA; TEIXEIRA; MALIK, 2018), que o *overhead* imposto na malha de controle pela comunicação entre os dispositivos é de ordem insignificante diante do tempo que, em geral, dois atuadores físicos operam sob controle.

6 Conclusão

Este trabalho explora a utilização de parâmetros para representação de contextos, a fim de simplificar o processo de modelagem, síntese e implementação de SEDs. Apesar da relevância teórica, a TCS clássica enfrenta problemas sobretudo na construção de especificações complexas. Duas abordagens parametrizadas são apresentadas pela literatura com o intuito de resolver as limitações da TCS clássica Entende-se que a abordagem com estados parametrizados seja mais adequada à etapa de modelagem, pois utiliza fórmulas e variáveis para representar e alternar contextos. Ainda, a abordagem com eventos parametrizados permite que a síntese seja calcula com abstrações e possibilita a implementação descentralizada do controlador.

Algoritmos de complexidade polinomial foram apresentados para permitir a conversão de modelos construídos com estados parametrizados para eventos parametrizados, combinando assim as vantagens do uso de parâmetros em todas as etapas da construção de controladores de SEDs. Exemplos são utilizados para ilustrar e testar o método de conversão: alguns desenvolvidos diretamente para esse trabalho e outros obtidos da literatura. Os exemplos compartilham características de memorização de sequências para tomada de decisões de controle. Utilizando parâmetros, é possível controlar e armazenar facilmente essas informações e obter controladores equivalentes aos da TCS clássica, mas por com menos esforço de modelagem e de síntese.

Os resultados experimentais obtidos dos métodos de conversão foram utilizados na síntese com abstrações de Cury et al. (2015) combinados com o critério de escolha de Rosa, Teixeira e Malik (2018). Apesar dos supervisores encontrados terem sido equivalente aos da síntese convencional, foram calculados sobre modelos com menor quantidade de estados e transições. Todavia, a redução do tamanho dos modelos não foi tão vantajosa quanto esperado e como é reportado pela literatura de abstrações. Dessa maneira, pretende-se identificar em trabalhos futuros quais características impediram as abstrações de atingirem o potencial total.

Além disso, métodos de síntese modular não foram testados no escopo deste trabalho sobre os modelos convertidos. Pretende-se explorar os efeitos da síntese modular sobre os modelos convertidos futuramente, assim como realizar a prova formal da equivalência das especificações.

Referências

AGUIAR, R. S. S. et al. Heuristic search of supervisors by approximated distinguishers. In: *Dependable Control of Discrete Systems (DCDS'13)*. York, UK: [s.n.], 2013. p. 121–126. Citado na página 53.

BARRETT, G.; LAFORTUNE, S. Bisimulation, the supervisory control problem and strong model matching for finite state machines. *Discrete Event Dynamic Systems*, Springer, v. 8, n. 4, p. 377–429, 1998. Citado na página 54.

BASSINO, F.; BÉAL, M.-P.; PERRIN, D. Super-state automata and rational trees. In: SPRINGER. *Latin American Symposium on Theoretical Informatics*. [S.l.], 1998. p. 42–52. Citado na página 46.

CAMPOS, G. R. de; ROSSA, F. D.; COLOMBO, A. Safety verification methods for human-driven vehicles at traffic intersections: Optimal driver-adaptive supervisory control. *IEEE Transactions on Human-Machine Systems*, v. 48, n. 1, p. 72–84, fev. 2018. ISSN 2168-2291. Citado na página 13.

CASSANDRAS, C. G.; LAFORTUNE, S. *Introduction to discrete event systems*. [S.l.]: Springer Science & Business Media, 2009. Citado 7 vezes nas páginas 11, 14, 15, 17, 18, 26 e 54.

CHEN, Y.-J.; HWANG, K.-S.; JIANG, W.-C. Policy sharing between multiple mobile robots using decision trees. *Information Sciences*, v. 234, p. 112–120, 2013. Citado 2 vezes nas páginas 11 e 28.

CHEN, Y.-L.; LIN, F. Safety control of discrete event systems using finite state machines with parameters. In: IEEE. *American Control Conference, 2001. Proceedings of the 2001*. [S.l.], 2001. v. 2, p. 975–980. Citado na página 25.

CHIEN, T. L. et al. Dynamic programming algorithm based path planning of the multiple robot system. In: *Int. Conf. on Digital Manufacturing and Automation*. [S.l.: s.n.], 2011. p. 469–474. Citado na página 11.

CLAESSEN, K. et al. Using valued booleans to find simpler counterexamples in random testing of cyber-physical systems. *IFAC-PapersOnLine*, Elsevier, v. 51, n. 7, p. 408–415, 2018. Citado 2 vezes nas páginas 12 e 13.

CURY, J. E. et al. Supervisory control of discrete event systems with distinguishers. *Automatica*, Elsevier, v. 56, p. 93–104, 2015. Citado 13 vezes nas páginas 12, 13, 23, 24, 25, 33, 34, 36, 37, 53, 55, 56 e 59.

DAVYDOV, A.; LARIONOV, A.; NAGUL, N. The formal logic approach for checking the observability of a specification language on des functioning. In: *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. [S.l.: s.n.], 2018. p. 0938–0943. Citado na página 13.

DORF, R. C.; BISHOP, R. H. Sistemas de controle modernos, 8ª edição. *Editora LTC, Rio de Janeiro*, 2001. Citado 2 vezes nas páginas 14 e 15.

ESMAEILIAN, B.; BEHDAD, S.; WANG, B. The evolution and future of manufacturing: A review. *Journal of Manufacturing Systems*, v. 39, p. 79 – 100, 2016. Citado na página 11.

GOHARI, P.; WONHAM, W. M. On the complexity of supervisory control design in the rw framework. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, v. 30, n. 5, p. 643–652, 2000. Citado na página 23.

HOPCROFT, J. E.; MOTWANI, R.; ULLMAN, J. D. *Introduction to Automata Theory, Languages, and Computation*. 2. ed. United States of America: Pearson Education, 1939. Citado na página 17.

JHA, S. S.; NAIR, S. B. A logic programming interface for multiple robots. In: *3rd Nat. Conf. on Emerging Trends and Applications in Computer Science*. [S.l.: s.n.], 2012. p. 152–156. Citado na página 11.

KORSSEN, T. et al. Systematic model-based design and implementation of supervisors for advanced driver assistance systems. *IEEE Transactions on Intelligent Transportation Systems*, v. 19, n. 2, p. 533–544, fev. 2018. ISSN 1524-9050. Citado na página 13.

MALIK, R. et al. Supremica–an efficient tool for large-scale discrete event systems. *IFAC-PapersOnLine*, Elsevier, v. 50, n. 1, p. 5794–5799, 2017. Citado na página 12.

MALIK, R.; TEIXEIRA, M. Modular supervisor synthesis for extended finite-state machines subject to controllability. In: IEEE. *Discrete Event Systems (WODES), 2016 13th International Workshop on*. [S.l.], 2016. p. 91–96. Citado 2 vezes nas páginas 31 e 37.

MALIK, R.; TEIXEIRA, M. *Synthesis of Least Restrictive Controllable Supervisors for Extended Finite-State Machines with Variable Abstraction*. 2019. Citado 3 vezes nas páginas 27, 31 e 53.

MOHAJERANI, S.; MALIK, R.; FABIAN, M. Compositional synthesis of supervisors in the form of state machines and state maps. *Automatica*, Elsevier, v. 76, p. 277–281, 2017. Citado 2 vezes nas páginas 11 e 12.

MOHAJERANI, S.; MALIK, R.; FABIAN, M. Compositional synthesis of supervisors in the form of state machines and state maps. *Automatica*, Elsevier, v. 76, p. 277–281, 2017. Citado na página 23.

NISE, N. S. *Control Systems Engineering*. 6. ed. California State Polytechnic University, Pomona: John Wiley and Sons, Inc, 2011. Citado na página 14.

OGATA, K. *Discrete-time Control Systems*. 2. ed. University of Minnesota: Prentice-Hall International, Inc., 1996. Citado na página 14.

QAMSANE, Y.; ABDELOUAHED, T.; PHILIPPOT, A. A synthesis approach to distributed supervisory control design for manufacturing systems with grafcet implementation. *International Journal of Production Research*, p. 1–21, 9 2016. Citado na página 11.

QUEIROZ, M. H. D.; CURY, J. E. Modular control of composed systems. In: IEEE. *American Control Conference, 2000. Proceedings of the 2000*. [S.l.], 2000. v. 6, p. 4051–4055. Citado na página 37.

RAMADGE, P. J.; WONHAM, W. M. Supervisory control of a class of discrete event processes. *SIAM journal on control and optimization*, SIAM, v. 25, n. 1, p. 206–230, 1987. Citado 3 vezes nas páginas 16, 20 e 21.

RAMADGE, P. J.; WONHAM, W. M. The control of discrete event systems. *Proceedings of the IEEE*, IEEE, v. 77, n. 1, p. 81–98, 1989. Citado 6 vezes nas páginas 11, 20, 21, 33, 34 e 37.

ROSA, M. et al. Efficient implementation of distinguished controllers for discrete-event systems. *IFAC-PapersOnLine*, Elsevier, v. 50, n. 1, p. 1187–1192, 2017. Citado 4 vezes nas páginas 13, 33, 37 e 55.

ROSA, M.; TEIXEIRA, M.; MALIK, R. Exploiting approximations in supervisory control with distinguishers. In: *International Workshop on Discrete Event Systems*. Sorrento, Italy: Elsevier, 2018. v. 51, n. 7, p. 13–18. Citado 9 vezes nas páginas 13, 23, 37, 53, 55, 56, 57, 58 e 59.

SILVA, A. L.; RIBEIRO, R.; TEIXEIRA, M. Modeling and control of flexible context-dependent manufacturing systems. *Information Sciences*, v. 421, p. 1–14, 2017. Citado 3 vezes nas páginas 12, 25 e 53.

SOUTHIER, L.; TEIXEIRA, M. Combining advantages from parameters in modeling and control of discrete event systems. In: *Submetido: Emerging Technologies and Factory Automation*. [S.l.: s.n.], 2019. p. 1–8. Citado na página 12.

TEIXEIRA, M. Explorando o uso de distinguidores e de autômatos finitos estendidos na teoria do controle supervisório de sistemas a eventos discretos. Tese de Doutorado - Doutorado em Engenharia de Automação e Sistemas - Programa de Pós-Graduação em Engenharia de Automação e Sistemas, Universidade Federal de Santa Catarina - UFSC, Florianópolis, 2013. Citado na página 32.

TEIXEIRA, M.; CURY, J. E. R.; QUEIROZ, M. H. de. Exploiting distinguishers in local modular control of discrete-event systems. *IEEE Transactions on Automation Science and Engineering*, IEEE, v. 15, n. 3, p. 1431–1437, jul. 2018. Citado 4 vezes nas páginas 23, 33, 37 e 53.

TEIXEIRA, M. et al. Supervisory control of des with extended finite-state machines and variable abstraction. *IEEE Transactions on Automatic Control*, IEEE, v. 60, n. 1, p. 118–129, 2015. Citado 8 vezes nas páginas 12, 23, 24, 25, 27, 29, 31 e 37.

TSAI, C. K. Multiple robot coordination and programming. In: *IEEE Int. Conf. on Robotics and Automation*. [S.l.: s.n.], 1991. v. 2, p. 978–985. Citado na página 11.

WARE, S.; SU, R. Time optimal synthesis based upon sequential abstraction and its application to cluster tools. *IEEE Transactions on Automation Science and Engineering*, IEEE, v. 14, n. 2, p. 772–784, 2017. Citado na página 12.

WONHAM, W. *Supervisory Control of Discrete Event Systems*. [S.l.]: University of Toronto, 2010. Citado na página 53.

Zhong, H.; Wonham, W. M. On the consistency of hierarchical supervision in discrete-event systems. *IEEE Transactions on Automatic Control*, v. 35, n. 10, p. 1125–1134, 1990. Citado na página 53.