UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

LUIZ FERNANDO PUTTOW SOUTHIER

CONVERSÃO ENTRE MODELOS DE SISTEMAS A EVENTOS DISCRETOS PARA SIMPLIFICAÇÃO DA SÍNTESE DE CONTROLADORES

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2017

LUIZ FERNANDO PUTTOW SOUTHIER

CONVERSÃO ENTRE MODELOS DE SISTEMAS A EVENTOS DISCRETOS PARA SIMPLIFICAÇÃO DA SÍNTESE DE CONTROLADORES

Trabalho de Conclusão de Curso 2, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso de Engenharia de Computação da Universidade Tecnológica Federal do Paraná - UTFPR, Câmpus Pato Branco, como requisito parcial para obtenção do título de Engenheiro da Computação.

Orientador: Prof. Dr. Marcelo Teixeira

PATO BRANCO

2017

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento Acadêmico de Informática
Curso de Engenharia de Computação

TERMO DE APROVAÇÃO

Às 8 horas e 20 minutos do dia 27 de junho de 2017, na sala V105, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, reuniu-se a banca examinadora composta pelos professores Marcelo Teixeira (orientador), Marco Antonio de Castro Barbosa e Cesar Rafael Claure Torrico para avaliar o trabalho de conclusão de curso com o título **Conversão entre Modelos de Sistemas a Eventos Discretos para Simplificação da Síntese de Controladores**, do aluno **Luiz Fernando Puttow Southier**, matrícula 01260804, do curso de Engenharia de Computação. Após a apresentação o candidato foi arguido pela banca examinadora. Em seguida foi realizada a deliberação pela banca examinadora que considerou o trabalho aprovado.

_____
Prof. Marcelo Teixeira
Orientador (UTFPR)

_____
Prof. Marco Antonio de Castro Barbosa
(UTFPR)

_____
Prof. Cesar Rafael Claure Torrico
(UTFPR)

_____
Profa. Beatriz Terezinha Borsoi
Coordenador de TCC

_____
Prof. Pablo Gauterio Cavalcanti
Coordenador do Curso de
Engenharia de Computação

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

I almost wish I hadn't gone down that rabbit-hole and yet — and yet — it's rather curious, you know, this sort of life!

Alice in Wonderland - Lewis Carroll

RESUMO

SOUTHIER, Luiz Fernando Puttow. Conversão entre modelos de Sistemas a Eventos Discretos para simplificação da síntese de controladores. 2017. ___f. Monografia (Trabalho de Conclusão de Curso) - Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2017.

A Teoria de Controle Supervisório (TCS) estrutura formalmente a síntese de controladores para Sistemas a Eventos Discretos (SED) com base na Teoria dos Autômatos e Linguagens. Na TCS o conjunto de eventos é dividido entre controláveis e não controláveis e sobre essa divisão é definido o cálculo do supervisor. Dentre os aspectos que limitam a aplicação prática dessa teoria estão os problemas de modelagem e síntese. Nos últimos anos, uma série de abordagens vem sendo proposta como forma de lidar com tais limitações. Dentre elas, o uso de Distinguidores e de Autômatos Finitos Estendidos (AFE). Distinguidores utilizam o conceito de refinamentos para associar contextos a determinados eventos do sistema, o que leva o problema de controle a ser resolvido utilizando um novo alfabeto construído a partir do alfabeto inicial. Um AFE, por sua vez, implementa uma ideia similar aos distinguidores, mas utiliza variáveis e fórmulas lógicas de guarda e atualização para representar contexto. Cada uma dessas abordagens traz um conjunto de melhorias para a TCS, no sentido de simplificar as etapas de modelagem e de síntese. No entanto, em geral, cada abordagem é proposta dentro de um domínio específico, de forma tal que e a integração entre elas, ou com outras abordagens de propósito similar, não é direta. Isso impossibilita que as vantagens de cada uma possam ser combinadas. Este trabalho se apoia nas abordagens por Distinguidores e por AFE para propor um método de conversão que possibilita migrar de um domínio para o outro, preservando essencialmente o mesmo comportamento. É mostrado que isso possibilita combinar vantagens ao resolver um dado problema de controle. A conversão é ilustrada por meio de exemplos. Em particular, usa-se o exemplo de um sistema de manufatura com retrabalho de materiais. O sistema é inicialmente modelado por AFE, o que agrega vantagens de modelagem. Posteriormente, o sistema é convertido em um modelo distinguido e então a síntese modular local é processada com economias computacionais.

Palavras-chave: Sistemas a Eventos Discretos, Teoria do Controle Supervisório, Distinguidores, Autômatos Finitos Estendidos.

ABSTRACT

SOUTHIER, Luiz Fernando Puttow. Conversion between Discrete Events Systems models for simplification of controller synthesis. 2017. ___f. Monografia (Trabalho de Conclusão de Curso) - Computer Engineering major, Federal University of Technology - Paraná, Campus Pato Branco. Pato Branco, 2017.

The Supervisory Control Theory (SCT) formally structures the synthesis of controllers for Discrete Event Systems (DES) based on Automata and Languages theory. In SCT, events are separated into two different sets to include controllable and uncontrollable, and the control synthesis is defined on these sets. Among the aspects that limit the practical application of this theory are modeling and synthesis problems. In recent years, some approaches have been proposed as a mean to deal with such limitations, such as Distinguishers and Extended Finite-state Automata (EFA). Distinguishers use the concept of refinements to associate contexts with certain system's events, which leads the control problem to be solved using a new alphabet constructed from the initial one. An EFA implements a similar idea, but using variables and logical formulas to represent contexts. These approaches bring some improvements to SCT, including modeling and synthesis simplification. However, in general, each approach is proposed within a specific framework, so that their integration is not straightforward. This work introduces a method for converting Distinguishers to EFA, and vice versa, while preserving the same global behavior. It is shown that the proposal allows to combine advantages from both approaches when solving a given control problem. The conversion is illustrated by examples. In particular, it is used a manufacturing system with rework of materials. The system is initially modeled by an EFA, which brings modeling advantages. Afterwards, the system model is translated to Distinguishers and a local modular control synthesis is processed with computational economies.

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

LISTA DE SÍMBOLOS

Σ	Alfabeto
w	Cadeia
$\|w\|$	Comprimento de uma cadeia w
ϵ	Cadeia Vazia
Σ^*	Conjunto de todas as cadeias sobre um alfabeto Σ
L	Linguagem
$\overline{L}$	Prefixo-fechamento da linguagem L
Q	Conjunto de estados
q°	Estado inicial
Q^ω	Conjunto de estados marcados
γ	Função de transição
$\mathcal{L}(A)$	Linguagem Gerada por um autômato A
$\mathcal{L}^\omega(A)$	Linguagem Marcada por um autômato A
$A_1 \quad A_2$	Composição Síncrona
G	Modelo da planta
E	Modelo das especificações
S	Supervisor
$\sup\mathcal{C}$	Elemento supremo para controle supervisório
Δ	Alfabeto refinado
Π	Mapa mascarador
D	Distinguidor
H_d	Modelo para distinção da planta refinada
V	Conjunto de variáveis
Q°	Conjunto de estados iniciais
P_V	Conjunto de fórmulas lógicas sobre V
$\mathfrak{D}$	Modelo refinado por Distinguidores
$\mathfrak{E}$	Modelo refinado por variáveis
$\mathfrak{D} \rightarrow \mathfrak{E}$	Método de conversão de Distinguidores para AFE
$\mathcal{C}_\mathfrak{E}$	Construtor de variável
Θ_σ	Mapeamento de refinamentos de σ
x_σ	Variável construída por $\mathcal{C}_\mathfrak{E}(\Delta^\sigma)$

$\mathfrak{E}[G_{\mathrm{d}}]$	Resultado da conversão $\mathfrak{D} - \mathfrak{E}$
$\mathcal{C}_{\mathfrak{D}}$	Construtor de refinamento
$\mathfrak{E} \rightarrow \mathfrak{D}$	Método de conversão de AFE para Distinguidores
$\mathfrak{D}_A\left[G_{\mathrm{v}}\right]$	Resultado da conversão $\mathfrak{E} - \mathfrak{D}$ sem distinções
$\mathfrak{D}_H\left[G_{\mathrm{v}}\right]$	Distinções para o resultado da conversão $\mathfrak{E} \rightarrow \mathfrak{D}$
$\mathcal{C}_Q$	Construtor de estados
Φ	Mapeamento de estados

SUMÁRIO

1 INTRODUÇÃO

Apresenta-se neste capítulo uma perspectiva geral sobre o estado da arte no qual este trabalho está contextualizado. Descreve-se, também, a problematização do assunto abordado, bem como os objetivos e sua correspondente justificativa de pesquisa destacando a importância deste tema. Expõe-se, ainda, a estrutura adotada no trabalho.

1.1 CONSIDERAÇÕES INICIAIS

Sistemas a Eventos Discretos (SED) são sistemas que se caracterizam por estarem sujeitos a eventos que ocorrem em intervalos de tempo irregulares ou desconhecidos. Exemplos de SED são a transmissão de um pacote em um sistema de comunicação, a finalização de uma tarefa ou a falha de uma máquina em um sistema de manufatura, por exemplo (CASSANDRAS; LAFORTUNE, 2009).

Uma das formas de modelar SED é por meio de Linguagens Formais e por Autômatos Finitos (AF) (HOPCROFT *et al.*, 1939). Lógicas de controle para SED podem, em geral, ser implementadas usando-se os mesmos formalismos utilizados na sua modelagem. A Teoria do Controle Supervisório (TCS), por exemplo, define um método formal para a síntese automática de supervisores ou controladores ótimos para SED (RAMADGE; WONHAM, 1987). A TCS, assim como a maioria das teorias de controle, tem por objetivo a síntese de controladores partindo de um modelo formal do sistema a ser controlado, chamado de planta, e os requisitos de comportamento do sistema controlado, chamados de especificações (THISTLE, 2004).

Embora bem fundamentada, a TCS enfrenta alguns problemas na prática que acabam limitando sua aplicabilidade na indústria. Por exemplo, para alguns problemas em particular, uma única especificação do sistema pode exigir monoliticamente combinar milhares de estados, tornando a modelagem inviável (TEIXEIRA, 2013). O conceito de distinguidor, proposto por Bouzon *et al.* (2008), tem por objetivo simplificar o processo de modelagem de especificações refinando o conjunto de eventos do modelo dos SED.

Outra abordagem de simplificação da complexidade de modelagem é o uso de Autômatos Finitos Estendidos (AFE) investigados por Chen e Lin (2000), Chen e Lin (2001), Skoldstam *et al.* (2007) , Fei *et al.* (2012) e Cheng e Krishnakumar (1993). Os AFE se caracterizam por monitorar um conjunto de elementos dos sistemas por meio de variáveis. Essas variáveis são atualizadas por fórmulas e expressões lógicas asso-

ciadas aos eventos do modelo, fazendo com que a ação de controle seja direcionada pelo valor atual das variáveis.

1.2 PROBLEMÁTICA E JUSTIFICATIVA

Informalmente, Distinguidores e AFE apresentam finalidades similares, que é simplificar a resolução de problemas para os quais a construção de um modelo é uma tarefa complexa ou mesmo inviável. O fundamento chave por trás dessa simplificação é o conceito de *refinamento*. Um refinamento é um mecanismo associado ou aos estados do modelo (no caso dos AFE) ou às suas transições (no caso dos Distinguidores), que provê ao *designer* maiores detalhes sobre como o sistema se comporta, o que pode contribuir decisivamente na hora de expressar um requisito. A ideia geral é que certas ações no sistema passem a carregar uma semântica que as diferencie em relação a outras instâncias dessas ações. A chegada de um pacote de dados, por exemplo, pode ser associada à sua ordem de chegada, facilitando assim a modelagem de uma restrição que coordene tais chegadas.

Além dos benefícios de modelagem apresentados, as abordagens por Distinguidores e AFE são ainda associados na literatura a alternativas avançadas de síntese, capazes de levar ao cálculo de controladores ótimos, de maneira mais simples do que a convencional, com menor dispêndio computacional. Tais alternativas caracterizam-se pelo uso de abstrações no caso dos AFE (TEIXEIRA *et al.*, 2015) e aproximações no caso dos Distinguidores (CURY *et al.*, 2015). Contudo, a revisão de literatura realizada indica que não existe uma relação formal e explícita entre as duas abordagens. Isso impacta negativamente no processo de modelagem e síntese, no sentido de que isso impede que as vantagens de ambas as abordagens possam ser combinadas na resolução de um mesmo problema. Ou, ainda, que possa ser feita uma comparação explícita de qual abordagem é mais adequada para cada tipo de problema de controle.

Os resultados existentes até o momento sugerem que a síntese de controle de SED utilizando Distinguidores é relativamente mais vantajosa do que a síntese com AFE. De fato, ela facilita o processo de modelagem (BOUZON *et al.*, 2008), quando associada a aproximações reduz a complexidade de síntese (CURY *et al.*, 2015), e ela ainda possui suporte ao Controle Modular Local (CML) (TEIXEIRA *et al.*, 2013), fundamental para lidar com problemas industriais de grande porte (QUEIROZ; CURY, 2000). Como principal desvantagem em relação à síntese convencional, a abordagem com Distinguidores impõe um *overhead* no sentido de que, nesse caso, se tem que cons-

truir um modelo adicional para distinguir a planta refinada do sistema, tarefa esta que não compõe o método convencional de síntese, é manual e pode ser complexa em alguns casos (FISCHER; LEAL, 2014).

Por outro lado, AFE possuem a vantagem de serem integrados com uma estrutura de variáveis, o que gera benefícios de modelagem similares aos Distinguidores, além de ser mais natural sob o ponto de vista computacional. Além disso, a literatura associa abstrações ao controle supervisório com AFE (TEIXEIRA *et al.*, 2015), o que gera benefícios computacionais similares ao uso de aproximações na síntese com Distinguidores. Como principal desvantagem, a abordagem com AFE não possui suporte ao CML. Um método de modularização de síntese com AFE foi recentemente proposto por Malik e Teixeira (2016). No entanto, a abordagem ainda é essencialmente teórica, não conta com ferramental de síntese, além de que, nessa abordagem, AFE são explorados sem a ideia de abstrações e, portanto, não geram vantagens comparáveis à aproximações no CML. Ademais, a estrutura de variáveis não é diretamente associável às ferramentas de síntese e geração de código, requerendo um passo adicional para converter AFE em autômatos ordinários que podem, então, ser tratados pelos algoritmos convencionais de síntese. Por fim, existe ainda a desvantagem de se ter que construir a estrutura lógica de atualização da planta que, embora pareça mais simples do que a modelagem de Distinguidores, não deixa de ser um esforço extra no processo de síntese.

Neste trabalho, argumenta-se que, dependendo da aplicação ou problema em que está sendo utilizado, o refinamento de eventos acaba por si só se tornando um problema de modelagem. Isso se deve ao fato de que a construção de uma estrutura de distinção precisa incorporar a ocorrência de cada instância de refinamento de acordo com o comportamento do sistema. No entanto, entende-se que, caso o mesmo problema seja modelado utilizando AFE, o esforço de modelagem poderia ser resumido à implementação de uma base de fórmulas e expressões lógicas, ao invés da construção de autômatos. Fórmulas atualizam variáveis associando valores a instâncias do comportamento de um SED. Em tese, estima-se que essa construção seja mais simples e passível de automatização, ao passo que um modelo distinguidor continua dependendo puramente da modelagem de uma máquina de estados, tarefa que recai inevitavelmente sobre o engenheiro.

Em face ao exposto, o presente trabalho visa a construção de um método de conversão que possibilita combinar vantagens de ambas abordagens, por Distinguidores e por AFE. A característica atrativa dos AFE, de facilitar a modelagem de problemas complexos, pode ser explorada durante a fase de *design* e, após obtidos os

modelos, é possível convertê-los para modelos refinados para fins de síntese. O processo inverso de conversão também é realizado a fim de que, a partir de um modelo ordinário refinado, se possa obter a sua versão equivalente em AFE.

1.3 OBJETIVO

Desenvolver um método formal de conversão entre modelos de SED construídos utilizando AFE e Distinguidores que expressem comportamento equivalente. Almeja-se tal objetivo executando-se as seguintes tarefas:

1.3.1 OBJETIVOS ESPECÍFICOS

- Modelar um exemplo de um SED por meio das abordagens clássica, Distinguidores e AFE;
- Desenvolver um método de conversão entre máquinas de estados, de Distinguidores para AFE e vice-versa;
- Aplicar método de conversão sobre o exemplo, analisando os resultados obtidos;

1.4 ESTRUTURA

O presente trabalho apresenta inicialmente as definições teóricas inerentes à modelagem de SED, tais como Teoria das Linguagens Formais e seu reconhecimento por Autômatos Finitos Determinísticos. Posteriormente, retrata os conceitos referentes ao controle de SED, por meio da Teoria do Controle Supervisório, sob as abordagens de Distinguidores e de AFE. Prossegue-se indicando o processo de desenvolvimento do método de conversão, os resultados e por fim a conclusão.

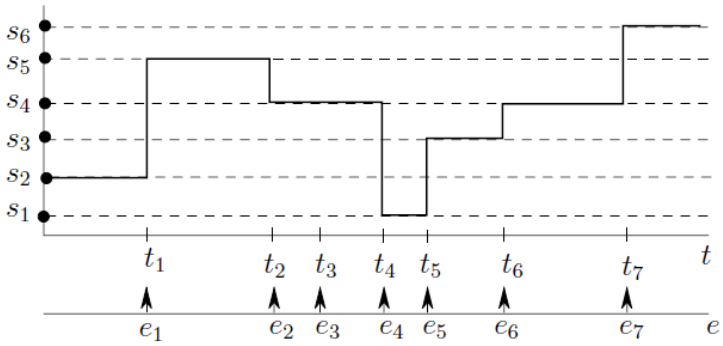
2 REFERENCIAL TEÓRICO

Apresentam-se neste capítulo os fundamentos teóricos. Descreve-se a Teoria dos Autômatos, reforçando os conceitos centrais de Alfabetos, Cadeias, Linguagens e Operações. Conceitua-se, ainda, a teoria dos Autômatos Finitos Determinísticos e sua aplicação como reconhecedores de linguagens. Por fim, é apresentada a Teoria de Controle Supervisório, usada na síntese de controladores para SED, sob a abordagem clássica, por Distinguidores e por AFE.

2.1 MODELAGEM DE SED

Ramadge e Wonham (1989) definem um SED como sendo um sistema dinâmico com um espaço de estados discreto e com trajetórias entre estados constantes. Os instantes de tempo em que as transições ocorrem, assim como a transição em si, em geral são imprevisíveis, contrastando-os com sistemas de dinâmica contínua no tempo. A Figura 1 apresenta um exemplo de transições (e_i) que ocorrem em determinados instantes de tempo (t_i) entre os estados (s_i) de um SED.

Figura 1 – Exemplo de transições de estados de um SED



Fonte: Cassandras e Lafortune (2009).

Enquanto um sistema dirigido pelo tempo pode ser descrito por equações diferenciais, para Cassandras e Lafortune (2009) os SED são mais naturalmente modelados por estruturas de diagramação, como AFD por exemplo. As próximas duas seções trazem conceitos referentes à *Teoria da Linguagens Formais* e aos *Autômatos Finitos Determinísticos* utilizados amplamente na modelagem de SED.

2.1.1 TEORIA DAS LINGUAGENS FORMAIS

As definições principais que permeiam a Teoria das Linguagens Formais (TLF) consistem em Alfabetos, Cadeias e Linguagens e são definidas a seguir de acordo

com Hopcroft *et al.* (1939).

Um *alfabeto* pode ser definido como um conjunto finito e não vazio de símbolos, denotado por Σ. Uma *cadeia* w é uma sequência finita de símbolos pertencentes a um alfabeto, cujo comprimento é denotado por $|w|$. Uma *cadeia vazia* é uma cadeia com nenhuma ocorrência de símbolos, por isso pode ser formada a partir de qualquer alfabeto, e é representada por ϵ, tal que $|\epsilon| = 0$. O conjunto de todas as cadeias sobre um alfabeto Σ é convencionalmente denotado Σ^*.

Duas cadeias $\alpha = a_1a_2...a_i$ e $\beta = b_1b_2...b_j$ podem ser concatenadas em uma única cadeia $\gamma = a_1a_2...a_ib_1b_2...b_j$. Se a cadeia $s = xyz$, com $s, x, y, z \in \Sigma^*$, então x é um *prefixo* de s, y uma *subcadeia* de s e z é um *sufixo* de s (HOPCROFT *et al.*, 1939).

A partir de Σ^*, podem ser definidos subconjuntos, denominados *linguagem* (L), que incorporam cadeias reconhecidas em determinado contexto. Ou seja, para um dado alfabeto Σ, $L \subseteq \Sigma^*$, caracteriza L como uma linguagem sobre Σ cujas cadeias apresentam um significado em determinado contexto. Enquanto que as cadeias $s \in L$ são finitas, L pode ser um conjunto infinito de cadeias finitas.

Cassandras e Lafortune (2009) definem algumas operações sobre Linguagens, complementando as operações usuais de conjuntos, tais como união, interseção e diferença. Dentre elas destaca-se o *prefixo-fechamento* ($\overline{L}$) de uma linguagem L. Este pode ser definido como

$$\overline{L} = \{s \in \Sigma^* : \exists t \in \Sigma^* \text{ tal que } st \in L\} \quad (1)$$

Em palavras, o prefixo-fechamento de L é a linguagem que consiste de todos os prefixos de todas as cadeias pertencentes à L. L é dita prefixo-fechada se $L = \overline{L}$, ou seja, uma linguagem L é prefixo-fechada se todos os prefixos das cadeias de L também são elementos de L.

2.1.2 AF DETERMINÍSTICOS COMO RECONHECEDORES DE LINGUAGENS

Hopcroft *et al.* (1939) define um Autômato Finito Determinístico (AFD) como sendo a quíntupla $A = (\Sigma, Q, q^\circ, Q^\omega, \gamma)$, onde:

- Σ é o alfabeto de *símbolos de entrada*;
- Q é um *conjunto finito de estados*;
- $q^\circ \in Q$ é o *estado inicial*;
- $Q^\omega \subseteq Q$ é o *conjunto de estados finais ou de aceitação*;

- γ: $Q \times \Sigma \rightarrow Q$ é uma *função de transição*, normalmente parcial em relação ao seu domínio, que tem como argumentos um estado e um símbolo de entrada e, como retorno, um estado. Na representação de um autômato por meio de um grafo direcionado, γ é representada por nós e transições identificadas. Considerando estados q e p, e um símbolo de entrada a, e que $q \xrightarrow{a} p$, então a representação por grafo de γ possui uma transição identificada por a de q para p (CASSANDRAS; LAFORTUNE, 2009).

Para reconhecimento de linguagens, Hopcroft *et al.* (1939) sugere a utilização de AFD, isto é, ele pode ser utilizado para decidir se uma cadeia é aceita ou não (pertence ou não a uma linguagem). A linguagem de um AFD é o conjunto de todas as cadeias reconhecidas por ele (HOPCROFT *et al.*, 1939).

A *Linguagem Gerada* $\mathcal{L}(A)$ por um autômato $A = (\Sigma, Q, q^{\circ}, Q^{\omega}, \gamma)$ é definida por Cassandras e Lafortune (2009) como:

$$\mathcal{L}(A) = \{s \in \Sigma^{*} \text{ tal que } \gamma(q^{\circ}, s) \text{ é definida } \} \tag{2}$$

A *Linguagem Marcada* $\mathcal{L}^{\omega}(A)$ por um autômato $A = (\Sigma, Q, q^{\circ}, Q^{\omega}, \gamma)$ é definida por Cassandras e Lafortune (2009) como:

$$\mathcal{L}^{\omega}(A) = \{s \in \mathcal{L}(A) \text{ tal que } \gamma(q^{\circ}, s) \in Q^{\omega}\} \tag{3}$$

À classe de linguagens que podem ser reconhecidas por um Autômato Finito (LEWIS; PAPADIMITRIOU, 1997), dá-se o nome de *linguagens regulares*. Na indústria, a regularidade é importante porque define as linguagens passíveis de processamento computacional, isto é, elas podem ser representadas por autômatos que ocupam memória finita quando armazenados em um computador (CASSANDRAS; LAFORTUNE, 2009).

No caso de um SED ser modelado por um autômato G, os eventos do sistema podem ser representados por um alfabeto Σ_G, tal que as sequências de eventos possíveis no sistema podem ser identificadas por $\mathcal{L}(G)$ e suas cadeias marcadas por $\mathcal{L}^{\omega}(G)$. Sendo assim, AFD são uma ferramenta notória para expressar e modelar o comportamento de SED (RAMADGE; WONHAM, 1989).

2.1.2.1 COMPOSIÇÃO SÍNCRONA DE AFD

A *Composição Síncrona* é um método de junção dos comportamentos de um conjunto de AFD que ocorrem simultaneamente. Cassandras e Lafortune (2009) de-

finem a composição síncrona $A_1 \quad A_2$ 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:

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

$$\gamma_{1\|2}\left((q_1, q_2), s\right) = \begin{cases} (\gamma_1(q_1, s), \gamma_2(q_2, s)) \text{ se } s \in \Sigma_1 \cap \Sigma_2 \\ (\gamma_1(q_1, s), q_2) \text{ se } s \in \Sigma_1 \setminus \Sigma_2 \\ (q_1, \gamma_2(q_2, s)) \text{ se } s \in \Sigma_2 \setminus \Sigma_1 \\ \text{indefinido para outros casos} \end{cases} \tag{4}$$

Ao analisar dois autômatos que representam os comportamentos de dois processos de um SED é possível utilizar essa operação para combinar esses comportamentos em um novo autômato, de tal modo que os eventos compartilhados ocorram simultaneamente e os demais sejam intercalados em qualquer ordem (TEIXEIRA, 2013).

2.2 CONTROLE DE SED

A partir da operação de composição, se torna possível que os componentes de um SED, também chamados de subsistemas, sejam modelados individualmente e, em tese, de forma mais simples. Espera-se que o resultado da composição das partes leve a uma estrutura equivalente à obtida caso o sistema fosse modelado como um todo. O modelo que reflete a execução dos elementos do sistema em conjunto é conhecido como *planta* (CASSANDRAS; LAFORTUNE, 2009), que é dita estar em *malha aberta* quando nenhuma interferência, ação de controle ou coordenação externa atua em seu comportamento.

Na prática, uma planta em malha aberta deve ser restrita a determinado comportamento esperado para quando estiver em funcionamento. Dessa necessidade nasce o conceito de *especificação*. Uma especificação pode ser entendida como a representação de um ato proibitivo que, quando associada à planta, interfere em determinados eventos possíveis em malha aberta, de modo que o comportamento resultante atenda as condições de controle propostas. Ao comportamento de uma planta restrito por um conjunto de especificações, dá-se o nome de *malha fechada*.

Se a planta de um SED é modelada por um conjunto de autômatos $G^j,\ j = 1, \cdots, n$, então a composição síncrona $G = \|_{j=1}^n G^j$ representa o modelo do sistema em malha aberta. Da mesma forma, podem ser combinados os m autômatos E^i que representam a especificação $E = \|_{i=1}^m E^i$. Combinando planta e especificações obtêm-

se a estrutura $K = G \| E$, em que $\mathcal{L}^{\omega}(K) \subseteq \mathcal{L}^{\omega}(G)$ correspondendo ao comportamento desejado da planta sob a ação das especificações.

Pela própria definição de composição síncrona, se um evento possível na planta G é desabilitado pela especificação E, então o resultado é uma ação que desabilita esse evento em G. Logo, o próprio modelo K pode ser utilizado para realizar o controle da planta G de um SED, desde que se assuma que todos os eventos em Σ são passíveis de desabilitação externa.

Na prática, porém, existem eventos que não podem ser diretamente controlados como, por exemplo, o evento de quebra ou parada de um equipamento, o evento de recepção de um pacote em uma rede, etc. A natureza da ocorrência desse tipo de evento é não-controlável e, portanto, a estrutura que regula a habilitação ou não desses eventos na planta precisa incorporar essa característica. Uma alternativa formal para o controle adequado de SED é apresentada na sequência.

2.2.1 TEORIA DE CONTROLE SUPERVISÓRIO

A *Teoria de Controle Supervisório* (TCS) é um formalismo que define a síntese de supervisores (ou controladores) ótimos para SED. Um supervisor é ótimo se ele é controlável, minimamente restritivo e não-bloqueante. Para permitir o cálculo de tal supervisor, considerando o aspecto de controlabilidade, a TCS particiona o conjunto de eventos $\Sigma = \Sigma_{\mathrm{c}} \cup \Sigma_{\mathrm{u}}$, onde Σ_{c} é o conjunto de eventos controláveis, i.e., aqueles que podem ser desabilitados, e Σ_{u} é o conjunto de eventos não controláveis, i.e., que não podem ser desabilitados diretamente (RAMADGE; WONHAM, 1987; RAMADGE; WONHAM, 1989).

A partir desse particionamento, a TCS define a operação que calcula o *supervisor*, a entidade que efetivamente implementa a ação de controle na planta. Formalmente, um supervisor S é um mapa $S : \mathcal{L}(G) \rightarrow 2^{\Sigma_{\mathrm{c}}}$ associado a uma linguagem $M \subseteq \mathcal{L}^{\omega}(G)$ que, após qualquer cadeia s em G, desabilita eventos controláveis $S(s) \subseteq \Sigma_{\mathrm{c}}$ e marca cadeias de acordo com $\mathcal{L}^{\omega}(S/G) = \mathcal{L}(S/G) \cap M$ (BOUZON *et al.*, 2008; CASSANDRAS; LAFORTUNE, 2009; RAMADGE; WONHAM, 1989).

O cálculo de S passa pelo conceito de *controlabilidade*. Uma linguagem $K \subseteq \Sigma^*$ é controlável em relação à L se $\overline{K}\Sigma_{\mathrm{u}} \cap L \subseteq \overline{K}$, ou seja, após qualquer prefixo de K, se um evento não-controlável é possível em L, a cadeia resultante continua em $\overline{K}$. Um supervisor marcador não-bloqueante S possui $\mathcal{L}^{\omega}(S/G) = K$ e pode ser implementado por qualquer autômato V em que $K = \mathcal{L}^{\omega}(V) \cap \mathcal{L}^{\omega}(G)$ e $\overline{K} = \mathcal{L}(V) \cap \mathcal{L}(G)$. O controle ocorre por meio de S que desabilita eventos possíveis em $\mathcal{L}(G)$ que não são possíveis

em $\mathcal{L}(V)$ após uma cadeia $s \in \mathcal{L}(S/G)$ (BOUZON *et al.*, 2009).

Se K não é controlável, então se torna necessário calcular a sua máxima linguagem controlável, ou seja, o subcomportamento que mais se aproxima de K e, ao mesmo tempo, é controlável. Considerando $\mathcal{C}(K,G)$ como o conjunto das linguagens controláveis de K em relação a G em que $L \subseteq K$ tal que L é controlável em relação à $\mathcal{L}(G)$, $\mathcal{C}(K,G)$ possui um único elemento supremo, denotado por $\sup\mathcal{C}(K,G)$, que expressa o comportamento menos restritivo possível a ser implementado por controle supervisório sobre a planta, respeitando o conjunto de especificações.

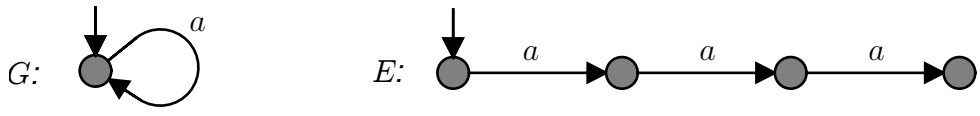
2.2.1.1 EXEMPLO: TCS

Com o objetivo de exemplificar a utilização da TCS, e suas limitações sob a abordagem clássica, é apresentado a seguir um problema simples, proposto por Teixeira (2013), que posteriormente será discutido sob as abordagens de Distinguidores e AFE.

Considera-se para este exemplo uma máquina M que está inicialmente desligada e entra em funcionamento após um evento a, que representa a chegada de uma peça ou a disponibilidade de um recurso, por exemplo. Supõe-se que essa máquina precisa ser colocada em manutenção após a ocorrência de três eventos a.

É possível modelar M utilizando os conceitos anteriormente apresentados. Dessa modelagem obtêm-se os autômatos G e E apresentados na Figura 2. G modela o comportamento da planta, sempre está habilitada a receber um evento a, enquanto que E modela o comportamento da especificação ou restrição apresentada, a máquina precisa entrar em manutenção após a ocorrência de três eventos a.

Figura 2 – Modelagem do SED sob a abordagem clássica



A TCS enfrenta problemas na prática que limitam sua aplicação diretamente na indústria. Ao considerar não mais três, mas sim 100000 ocorrências do evento a no problema acima, por exemplo, é interessante observar que a especificação E alcançaria um número de estados consideravelmente grande, forçando o desenvolvedor a combinar monoliticamente esses milhares de estados. Se ainda, em conjunto com esses eventos, o desenvolvedor tiver que considerar eventos concorrentes, a complexidade aumenta de uma maneira a tornar o problema inviável, como mostrado em Teixeira (2013), Cury *et al.* (2015), Bouzon *et al.* (2009), Bouzon *et al.* (2008).

Considerando que este exemplo retrata apenas uma fração singela do SED em problemas práticos, e que ele possui apenas uma especificação, é possível inferir que para um problema real a modelagem utilizando essa abordagem muitas vezes se torna inviável. Com isso, na literatura propõem-se abordagens para lidar com esse tipo de característica da TCS. Dentre as abordagens, as próximas duas seções tratam da abordagem por Distinguidores e da abordagem por AFE.

2.2.2 TCS COM DISTINGUIDORES

Esta seção apresenta os fundamentos do conceito de Distinguidores, proposto por Bouzon *et al.* (2008), Bouzon *et al.* (2009), que visa a simplificação do processo de modelagem de especificações identificando e distinguindo ocorrências de um evento no sistema. Uma característica interessante a se destacar é que a abordagem por Distinguidores proporciona uma solução de controle equivalente à obtida pela TCS clássica.

Seja G um autômato que modela um SED com eventos em Σ. Assume-se que cada elemento $\sigma \in \Sigma$ corresponde a uma máscara para eventos em um conjunto $\Delta^\sigma \neq \emptyset$ que refina σ (TEIXEIRA, 2013). Cada máscara está associada a um conjunto próprio de refinamentos, diferente dos outros e não vazio. Dessa forma, Σ passa a ser um conjunto de máscaras para eventos de um alfabeto refinado $\Delta = \bigcup_{\sigma\in\Sigma} \Delta^\sigma$. A estrutura de controle é definida no alfabeto refinado por $\Delta = \Delta_{\mathrm{c}} \cup \Delta_{\mathrm{u}}$, em que $\Delta_{\mathrm{u}} = \bigcup_{\sigma\in\Sigma_{\mathrm{u}}} \Delta^\sigma$ e $\Delta_{\mathrm{c}} = \bigcup_{\sigma\in\Sigma_{\mathrm{c}}} \Delta^\sigma$.

A relação entre Σ e Δ pode ser definida por um mapa mascarador $\Pi\colon \Delta^* \to \Sigma^*$ definido por Bouzon *et al.* (2008) a seguir:

$$\begin{aligned} &\Pi(\epsilon) = \epsilon \\ &\Pi(t\delta) = \Pi(t)\sigma \text{ para } t \in \Delta^*, \delta \in \Delta^\sigma \text{ e } \sigma \in \Sigma \end{aligned} \tag{5}$$

O mapa mascarador Π pode ser estendido para qualquer linguagem $L_\Delta \subseteq \Delta^*$

por $\Pi(L_\Delta) = \{s \in \Sigma^* | \exists t \in L_\Delta, \Pi(t) = s\}$. Da mesma forma, por ser definido também o mapa mascarador 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 $\Pi^{-1}(L) = \{t \in \Delta^* | \Pi(t) \in L\}$.

Por fim, dada uma linguagem $L \subseteq \Sigma^*$, um distinguidor para L pode ser definido como um mapa D: $\Sigma^* \to 2^{\Delta^*}$ tal que

$$D(L) = \Pi^{-1}(L) \cap L_{\mathrm{d}} \tag{6}$$

em que L_{d} é uma linguagem distinguidora. Dessa forma, o efeito de D sobre L é tal que ele mapeia cada cadeia de L em todas as possíveis cadeias refinadas ($\Pi^{-1}(L)$), e filtra quais delas devem de fato ocorrer (L_{d}). Pode-se notar que, no caso dos Distinguidores, resta definir um autômato complementar que reconheça L_{d} e esse autômato é chamado de H_d Teixeira (2013) define ainda um Distinguidor Preditível D como aquele que para todo $s \in \Sigma^*, |D(\sigma)| = 1$, isto é, existe um, e apenas um, evento refinado δ elegível para cada máscara $\sigma \in \Sigma$ após cada cadeia $r \in L_{\mathrm{d}}$.

Considerando um SED, cujo comportamento é modelado por autômato G, o efeito de um distinguidor D sobre G é dado por:

$$\begin{aligned} D(\mathcal{L}(G)) &= \Pi^{-1}(\mathcal{L}(G)) \cap L_{\mathrm{d}} \\ D(\mathcal{L}^\omega(G)) &= \Pi^{-1}(\mathcal{L}^\omega(G)) \cap L_{\mathrm{d}} \end{aligned} \tag{7}$$

Nesse contexto, seja G_{d} um autômato tal que $\mathcal{L}(G_{\mathrm{d}}) = D(\mathcal{L}(G))$ e $\mathcal{L}^\omega(G_{\mathrm{d}}) = D(\mathcal{L}^\omega(G))$, e seja E_{d} uma especificação definida em Δ. Os requisitos expressos por E_{d} usando as informações adicionais providas pelo distinguidor devem se referir aos mesmos requisitos expressos por E em Σ, tal que $E_{\mathrm{d}} \cap \mathcal{L}^\omega(G_{\mathrm{d}}) = K_{\mathrm{d}} = D(K)$, onde $K = E \cap \mathcal{L}^\omega(G)$. Nesse caso, a especificação $K \subseteq \mathcal{L}^\omega(G)$ seria dada por $K = \Pi(E_{\mathrm{d}} \cap \Pi^{-1}(\mathcal{L}^\omega(G)) \cap L_{\mathrm{d}})$.

Nota-se que o efeito do mapa inverso Π^{-1} pode ser representado em autômatos, simplesmente substituindo os eventos de cada transição pelos conjuntos correspondentes de eventos refinados. Equivalentemente, o efeito do mapa Π também pode ser representado, desta vez substituindo-se os eventos refinados pelas respectivas máscaras.

Então, o problema de controle com distinguidor consiste em encontrar um supervisor não-bloqueante $S_{\mathrm{d}} : \Delta^* \to 2^\Delta$, tal que $\mathcal{L}^\omega(S_{\mathrm{d}}/G_{\mathrm{d}}) \subseteq K_{\mathrm{d}}$. Cury *et al.* (2015)

demonstra que:

$$\begin{aligned} \Pi(\sup\mathcal{C}(K_{\mathrm{d}}, G_{\mathrm{d}})) &= \sup\mathcal{C}(K, G) \\ \sup\mathcal{C}(K_{\mathrm{d}}, G_{\mathrm{d}}) &= D(\sup\mathcal{C}(K, G)) \end{aligned} \tag{8}$$

Ou seja, a abordagem por Distinguidores proporciona uma solução de controle equivalente à obtida pela TCS clássica.

2.2.2.1 EXEMPLO: TCS COM DISTINGUIDORES

Baseado no exemplo proposto da seção da TCS e considerando suas limitações apresentadas, é possível adaptar a modelagem do problema utilizando Distinguidores. Nesse problema, é possível observar que o evento a se caracteriza por uma condição de modelagem que justifica sua exploração. No caso de Distinguidores, cada ocorrência do evento a, pode ser identificada por um novo evento a_i, no qual i está relacionado ao contexto em que o evento a ocorre, sendo este contexto incorporado semanticamente por um distinguidor.

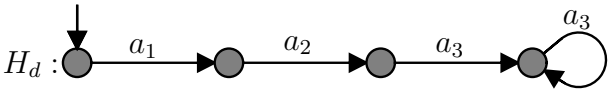
Para o exemplo proposto, ao evento a podem ser atribuídos três contextos diferentes (primeira, segunda e terceira ocorrência de a na máquina M). Nesse caso o conjunto de refinamentos do evento a é $\Delta^a = \{a_1, a_2, a_3\}$. Dessa forma, a planta G, modelada com eventos em Σ, pode ser substituída por uma versão em Δ, dada por $G_a = \Pi^{-1}(G)$. Também a especificação E, anteriormente definida, pode ser substituída pela especificação E_d, que desabilita a ocorrência de $a_i \in \Delta^a$ após a ocorrência de a_3, apresentada na Figura 3.

Figura 3 – Modelagem do SED sob a abordagem por Distinguidores

É interessante notar que o processo de modelagem é claramente facilitado pela abordagem por Distinguidores, pois a mesma especificação pode ser usada para qualquer quantidade n de eventos a a serem memorizados, enquanto um modelo equivalente dependeria de n caso fosse modelado em Σ. Como demonstrado por Teixeira (2013), os resultados dessa abordagem e pela TCS convencional são equivalentes, i.e., $K = G \parallel E = \Pi(G_a \parallel H_d \parallel E_d) = \Pi(K_d)$, o que por conseguinte leva também a um controlador equivalente.

A desvantagem dessa abordagem é que, aparentemente, ela introduz um passo a mais no processo de modelagem, pois requer a construção de um modelo para distinguir a planta refinada do sistema, o qual, por sua vez, corresponde a uma estrutura que volta a depender de n. Para esse exemplo, tal modelo é mostrado na Figura 4.

Figura 4 – Modelo para distinção da planta refinada



Ao ser associado à planta G_a, o modelo H_d distingue a ocorrência de cada instância do evento a no sistema. Isso leva a uma planta distinguida $G_d = G_a \| H_d$, que pode então ser usada na síntese de controle. Note que, embora a modelagem em Δ não mostre vantagens aparentes em relação à modelagem em Σ (já que H_d e E possuem o mesmo número de estados) H_d é, geralmente, por natureza modular (devido ao refinamento dos eventos), ao contrário de E. H_d pode, na verdade, ser modelado por um conjunto de autômatos de 2 estados (CURY *et al.*, 2015).

A próxima seção apresenta os conceitos teóricos referentes à segunda abordagem de otimização da TCS, por meio de AFE, que implementa ideia similar aos Distinguidores, mas usando outra estrutura de modelos.

2.2.3 TCS COM AUTÔMATOS FINITOS ESTENDIDOS

AFE compõem uma abordagem de modelagem que pode ser caracterizada por AF que incorporam variáveis definidas sobre um determinado domínio. As transições de um AFE possuem fórmulas de guarda, que são regras definidas sobre as variáveis, e funções de atualização dos valores dessas variáveis (CHEN; LIN, 2000; CHEN; LIN, 2001). AFE são formalmente definidos como a sétupla $A = (\Sigma, V, Q, Q^{\circ}, Q^{\omega}, \gamma)$, (TEIXEIRA, 2013) onde:

- Σ é o *alfabeto* de eventos;
- $V = \{v_1, ..., v_n\}$ é o *conjunto de variáveis*;
- Q é um *conjunto finito de estados*;
- $Q^{\circ} \in Q$ é o conjunto de *estados iniciais*;
- $Q^{\omega} \subseteq Q$ é o *conjunto de estados finais ou de aceitação*;
- $\gamma \subseteq Q \times \Sigma \times P_V \rightarrow Q$ é uma *função de transição*, normalmente parcial em relação ao seu domínio, na qual P_V é o conjunto de fórmulas sobre V. O

efeito de uma fórmula altera os valores das variáveis no estado corrente para novos valores no estado alcançado. Para representar formalmente esse efeito, pode-se definir um conjunto auxiliar, denotado por V', que contém as mesmas variáveis de V, com o mesmo domínio. Ou seja, todos os valores de variáveis possivelmente assumidos em V também podem ser assumidos em V'. Assim, P_V passa a ser um conjunto de fórmulas sobre $V \cup V'$, tal que cada $\rho \in P_V$ implementa uma condição *booleana* $\rho(v, v') = true$ ou $false$, em que $v' \in V'$ é a variável v após atualização de seu valor por meio de uma fórmula lógica. Considerando estados q e p e um evento σ, e que $q \overset{\sigma:\rho}{\rightarrow} p$, então a representação por grafo de γ possui uma transição identificada q para p com o evento $\sigma \in \Sigma$ e com a fórmula $\rho \in P_V$.

Um AFE também pode ser definido por meio da *definição explícita*, que nada mais é do que a versão materializada das *atualizações* sobre as variáveis que, então, passam a compor a semântica dos estados. Segundo Teixeira (2013) a versão explícita é dada pelo autômato $A_V = (\Sigma, Q_A, Q_A^\circ, Q_A^\omega, \gamma_A)$ onde:

- $Q_A = Q \times Dom(V)$;
- $Q_A^\circ = Q^\circ \times \{(v_1^o, ..., v_n^o)\}$;
- $Q_A^\omega = Q^\omega \times Dom(V)$;
- γ_A é tal que $(x, \overline{v}) \overset{\sigma}{\rightarrow} (y, \overline{v}')$, para $\overline{v}, \overline{v}' \in Dom(V)$, se existe $x \overset{\sigma:p}{\rightarrow} y$ com $p(\overline{v}, \overline{v}') = true$.

A mesma relação de transição pode ser aplicada a Σ^* por $(x, \overline{v}) \overset{\epsilon}{\rightarrow} (x, \overline{v})$, para todo $(x, \overline{v}) \in Q_A$ e $(x, \overline{v}) \overset{s\sigma}{\rightarrow} (x'', \overline{v}'')$, se $(x, \overline{v}) \overset{s}{\rightarrow} (x', \overline{v}') \overset{\sigma}{\rightarrow} (x'', \overline{v}'')$, para algum elemento $(x', \overline{v}') \in Q_A$. Teixeira (2013) denota por $A_V \overset{s}{\rightarrow} (x, \overline{v})$ a existência de um estado $(q^\circ, \overline{v}^o) \in Q_A^\circ$, tal que $(q^\circ, \overline{v}^o) \overset{s}{\rightarrow} (x, \overline{v})$ e define as linguagens gerada e marcada por A_V, respectivamente, como:

$$\begin{aligned} \mathcal{L}(A_V) &= \left\{ s \in \Sigma^* \text{ tal que } A_V \overset{s}{\rightarrow} (x, \overline{v}) \in Q_A \right\} \\ \mathcal{L}^\omega(A_V) &= \left\{ s \in \Sigma^* \text{ tal que } A_V \overset{s}{\rightarrow} (x, \overline{v}) \in Q_A^\omega \right\} \end{aligned} \tag{9}$$

A composição síncrona $A_1 \quad A_2$ também pode ser definida para dois AFE $A_1 = (\Sigma, V, Q_1, Q_1^\circ, Q_1^\omega, \gamma_1)$ e $A_2 = (\Sigma, V, Q_2, Q_2^\circ, Q_2^\omega, \gamma_2)$ (OUEDRAOGO *et al.*, 2011; TEIXEIRA,

2013) como:

$$A_1 \quad A_2 = \left(\Sigma_1 \cup \Sigma_2, V_1 \cup V_2, Q_1 \times Q_2, Q_1^\circ \times Q_2^\circ, Q_1^\omega \times Q_2^\omega, \gamma_{1\|2}\right), \text{ tal que}$$

$$\gamma_{1\|2} = \begin{cases} (x_1, x_2) \xrightarrow{\sigma:p_1 \wedge p_2} (y_1, y_2) \text{ se } \sigma \in \Sigma_1 \cap \Sigma_2, \; x_1 \xrightarrow{\sigma:p_1}_1 y_1 \text{ e } x_2 \xrightarrow{\sigma:p_2}_2 y_2 \\ (x_1, x_2) \xrightarrow{\sigma:p_1} (y_1, x_2) \text{ se } \sigma \in \Sigma_1 \setminus \Sigma_2 \text{ e } x_1 \xrightarrow{\sigma:p_1}_1 y_1 \\ (x_1, x_2) \xrightarrow{\sigma:p_2} (x_1, y_2) \text{ se } \sigma \in \Sigma_2 \setminus \Sigma_1 \text{ e } x_2 \xrightarrow{\sigma:p_2}_2 y_2 \\ \text{indefinido para outros casos} \end{cases} \tag{10}$$

Teixeira (2013) define as propriedades de totalidade, Q-determinismo, V-determinismo sobre os AFE: Um AFE é dito total se para todas as suas fórmulas lógicas $\rho \in P_V$, para todo $v \in Dom(V)$, existe $v' \in Dom(V')$ tal que $\rho(v, v') = true$, ou seja, se nenhuma de suas transições é desabilitada por uma fórmula lógica. Um AFE $A = (\Sigma, V, Q, Q^\circ, Q^\omega, \gamma)$ é Q-determinístico se $|Q^\circ| = 1$ e para $x \xrightarrow{\sigma:\rho_1} y_1$ e $x \xrightarrow{\sigma:\rho_2} y_2 \in \gamma$ então $y_1 = y_2$ para todo $x, y_1, y_2 \in Q$, $\sigma \in \Sigma$ e $\rho_1, \rho_2 \in P_V$. Um AFE $A = (\Sigma, Q, Q^\circ, Q^\omega, \gamma)$ é V-determinístico se $(x, \overline{v}) \xrightarrow{\sigma} (y, \overline{v}'_1)$ e $(x, \overline{v}) \xrightarrow{\sigma} (y, \overline{v}'_2) \in \gamma$ então $\overline{v}'_1 = \overline{v}'_2$ para todo $x, y \in Q$, $\sigma \in \Sigma$ e $\overline{v} \in Dom(v)$.

A controlabilidade é outra propriedade da TCS que pode ser estendida para AFE. Sejam $E_{\text{v}} = (\Sigma, V, Q_E, Q_E^\circ, Q_E^\omega, \gamma_E)$ e $G_{\text{v}} = (\Sigma, V, Q_G, Q_G^\circ, Q_G^\omega, \gamma_G)$ dois AFE, Teixeira (2013) diz que E_{v} é V-controlável em relação a G_{v}, se o seguinte for verdadeiro para todo $s \in \Sigma^*$, $\mu \in \Sigma_u$, $x_E \in Q_E$, x_G e $x'_G \in Q_G$, $\overline{v}$ e $\overline{v}' \in Dom(V)$:

$$\begin{aligned} & E_{\text{v}} \xrightarrow{s} (x_E, \overline{v}) \\ \wedge & G_{\text{v}} \xrightarrow{s} (x_G, \overline{v}) \xrightarrow{\mu} (x'_G, \overline{v}') \\ \Rightarrow \exists \, & x1_E \in Q_E \mid E_{\text{v}} \xrightarrow{s} (x_E, \overline{v}) \xrightarrow{\mu} (x'_E, \overline{v}') \end{aligned} \tag{11}$$

Sendo assim, uma especificação E_{v} é V-controlável se não proíbe eventos não-controláveis em G_{v} e se atualiza as variáveis da mesma forma como na planta. Considerando $E_{\text{v}} \parallel G_{\text{v}}$ como o modelo do comportamento esperado sob controle, define-se o conjunto:

$$C_v = \{K_v \subseteq E_{\text{v}} \parallel G_{\text{v}} | K_v \text{ é V-controlável em relação a } G_{\text{v}}\} \tag{12}$$

C_v possui um AFE supremo, denotado por $\sup\mathcal{C}_V$(E_{v},G_{v}), representando o comportamento menos restritivo a ser implementado sobre G_{v}, de modo a garantir E_{v}. Pode ser mostrado (TEIXEIRA *et al.*, 2015) que, nessas condições, $\mathcal{L}(\sup\mathcal{C}_V(E_{\text{v}}, G_{\text{v}})) = \sup\mathcal{C}(\mathcal{L}(E_{\text{v}} \| G_{\text{v}}), \mathcal{L}(G_{\text{v}}))$, ou seja, a linguagem de $\sup\mathcal{C}_V$ equivale à síntese da linguagem de um determinado AFE, ou de sua versão ordinária.

2.2.3.1 EXEMPLO: TCS COM AFE

O mesmo exemplo proposto anteriormente é reproduzido a seguir, dessa vez com a modelagem do problema utilizando AFE. Nesse caso, após cada ocorrência do evento a, o estado alcançado pode ser identificado por um valor discreto dentro de um domínio de uma variável x, onde $Dom(x) = \{0, 1, 2, 3, 4\}$ e $x^o = 0$. Sendo assim, o objetivo da planta é atribuir à variável x um valor que identifica o contexto em que o estado alcançado por a possui. Já a especificação objetiva restringir a planta de modo que a condição de controle seja satisfeita, ou seja, apenas 3 ocorrências de a devem ser possíveis. A Figura 5 mostra como a planta G_{v} e a especificação E_{v} do exemplo podem ser modeladas por AFE.

Figura 5 – Modelagem do SED sob a abordagem por AFE

A cada ocorrência do evento a, a variável x é incrementada dentro de seu domínio. O domínio 0 é usado para representar a condição do estado inicial, enquanto que o domínio 4 sustenta o fato de ser possível ocorrer algo proibido na planta, justamente o que é desabilitado via controle. Ao construir uma condição de guarda que considera o estado alcançado com base no valor atual de x, é possível identificar o contexto desse estado e impedir a ocorrência de a não desejada.

É interessante notar que os modelos, tanto da planta, quanto das especificações, possuem apenas um estado, assim como a composição entre eles. Isso indica que o esforço e complexidade de modelagem recai sobre as fórmulas lógicas implementadas para atualização e guarda das variáveis. Porém, sob o ponto de vista computacional, a versão explícita do autômato K_v possui a mesma quantidade de estados que os autômatos K e K_d, o que será exposto na sequência.

2.3 COMPARAÇÃO ENTRE OS MODELOS

A motivação principal para o uso de AFE e de Distinguidores como alternativas de extensão da TCS, é que essas abordagens facilitam expressar requisitos de controle. Uma breve comparação dos exemplos providos nas figuras 2, 3 e 5 dimensiona esse argumento. Neles, um modelo com n estados é substituído por estruturas de 1 e 2 estados fixos, para qualquer n. Esse fator, por si só, pode tornar viável o tratamento

de problemas de controle cuja solução envolvendo apenas as políticas convencionais da TCS seria intratável.

Adicionalmente, Distinguidores e AFE têm ainda a vantagem de facilitar tarefas de modelagem sem complexificar a síntese de controle como efeito. Na verdade, a literatura tem provado que o esforço de síntese, com ou sem o uso dessas abordagens é exatamente o mesmo. Isso pode ser facilmente ilustrado para o exemplo abordado anteriormente no contexto do comportamento desejado. Os modelos a seguir mostram como seria o modelo K e suas respectivas versões K_d e K_v (explícito).

$$K: \quad q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_3 \tag{13}$$

$$K_d: \quad q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} q_3 \tag{14}$$

$$K_v: \quad (q_0,0) \xrightarrow{a} (q_1,1) \xrightarrow{a} (q_2,2) \xrightarrow{a} (q_3,3) \tag{15}$$

Portanto, é esperado que $L(K) = L(K_v) = \Pi(L(K_d))$ e, assim, também é esperada uma solução de controle equivalente. Um aspecto que poderia ser visto como uma desvantagem é que a simplificação da modelagem cobra como contrapartida a construção de uma estrutura adicional de modelagem: o modelo distinguidor, no caso da abordagem com refinamentos; e a estrutura de fórmulas, no caso dos AFE. Ainda assim, essas são atividades naturalmente modulares, que podem ser conduzidas incrementalmente e, hipoteticamente, de maneira bastante simples.

Esse trabalho se desenvolve sobre problemas de controle para os quais essas estruturas de tratamento de modelos é determinante para a obtenção de uma solução. Para tais, é necessário definir, antes de mais nada, qual abordagem será usada: Distinguidores ou AFE. Sabe-se que os resultados teóricos existentes até o momento não suportam a combinação direta dessas abordagens.

Sabe-se, ainda, que para alguns casos, uma abordagem pode se mostrar mais vantajosa do que a outra. Algumas características entre as abordagens foram citadas por Teixeira (2013). O Quadro 1 estabelece um comparativo entre os principais indicadores que levariam Distinguidores e AFE a serem adotados no projeto de desenvolvimento de um controlador.

Quadro 1 – Comparação de características entre as três abordagens

Característica	TCS	Distinguidores	AFE
Otimização de modelagem	Não	Sim	Sim
Método usado	-	Refinamentos de eventos	Variáveis
Esforço extra de modelagem	-	Distinguidor H_d	Fórmulas lógicas
Modelagem de problemas complexos	Não	Sim	Sim
Disponibilidade de ferramental	Sim	Sim	Restrita
Suporte à geração de código	Sim	Sim	Restrita
Equivalência com a TCS convencional	-	Sim	Restrita
Modularização	Sim	Sim	Sim
Uso de abstrações	Não	Sim	Sim
Modularização com abstrações	Não	Sim	Não
Abstração com resultado ótimo	-	Não	Sim

Fonte: adaptado de Teixeira (2013).

Com os exemplos propostos nas seções anteriores e a comparação realizada no Quadro 1 alguns indícios de que esse método de conversão é alcançável, podem ser observados empiricamente. Se, por exemplo, identifica-se um evento b em um SED que pode ser refinado por Distinguidores, com o propósito de simplificação, em b_1, b_2, b_3 e b_4, é possível afirmar que a esse evento é atribuído quatro contextos diferentes.

Sendo assim, a mesma identificação de contextos pode ser reproduzida sobre um AFE, onde existiria uma variável y inicialmente valorada em 0, por exemplo, que tem seu valor incrementado a cada ocorrência de b e possui um domínio de 0 a 4. Essa variável então assumiria a cada incremento de seu valor o mesmo contexto e significado atribuído aos eventos refinados citados anteriormente. Partindo desses indícios e análise comparativa entre as abordagens, no próximo capítulo propõe-se métodos de conversão entre as abordagens.

3 CONVERSÃO DE MODELOS

Este capítulo apresenta uma abordagem para a conversão de plantas refinadas (e distinguidas), em plantas modeladas por AFE, e vice-versa. Para a clareza de cada contexto a ser abordado, será assumida a notação $\mathfrak{D}$ para identificar o universo dos modelos com eventos refinados e $\mathfrak{E}$ para a abordagem por AFE. Inicialmente, a sistemática de conversão de um modelo $\mathfrak{D}$ para um modelo $\mathfrak{E}$ ($\mathfrak{D} \rightarrow \mathfrak{E}$)será apresentada, e na sequência o processo inverso ($\mathfrak{E} \rightarrow \mathfrak{D}$), isto é, a conversão no sentido $\mathfrak{E}$ para $\mathfrak{D}$.

3.1 CONVERSÃO $\mathfrak{D} \rightarrow \mathfrak{E}$

O processo proposto para a conversão de modelos $\mathfrak{D} \rightarrow \mathfrak{E}$ consiste em associar as instâncias predefinidas para um certo evento, a valores pertencentes ao domínio de uma variável. Para que isso seja possível, é fundamental que se define-se uma sistemática capaz de preservar a consistência entre a ocorrência dos eventos refinados, e os valores a serem assumidos no domínio da variável correspondente. Tal sistemática é descrita na sequência.

Sejam duas plantas G e G_{d} modeladas com eventos em Σ e Δ, respectivamente, tal que $G = \Pi(G_{\mathrm{d}})$ e $G_{\mathrm{d}} = \Pi^{-1}(G) \| H_d$. Primeiro, é possível notar que, para um dado evento $\sigma \in \Sigma$, tal que Δ^σ denota o conjunto de instâncias de refinamentos de σ, é verdade que $|\Delta^\sigma| = n$, para algum natural $n \in \mathbb{N}$. Logo, pode ser assumido, sem perda de generalidade, que qualquer conjunto finito de refinamento Δ^σ pode ser associado a uma variável de Domínio finito. Sendo assim, propõe-se a seguinte definição:

Definição: 1 (Construtor de variável $\mathcal{C}_{\mathfrak{E}}$) *Dado um evento $\sigma \in \Sigma$, e seu conjunto de refinamentos associados, Δ^σ, define-se $\mathcal{C}_{\mathfrak{E}}$ como um construtor de variável que recebe um parâmetro Δ^σ e retorna uma variável associada x_σ, isto é, $\mathcal{C}_{\mathfrak{E}}(\Delta^\sigma) = x_\sigma$ tal que $x_\sigma^\circ = 1$ e $Dom(x_\sigma) = \{1, \cdots, n\}$ para $n = |\Delta^\sigma|$.*

A partir da definição anterior, pode ser construído um conjunto V de variáveis construídas a partir de $\mathcal{C}_{\mathfrak{E}}(\Delta^{\sigma i})$, para todo $\sigma_i \in \Sigma, i = 1, \cdots, m$, tal que $m = |\Sigma|$, isto é, $V = \{x_{\sigma_1}$,...,$x_{\sigma_m}\}$. Nesse contexto, note que a igualdade $|Dom(x_\sigma)| = |\Delta^\sigma|$ é sempre verdadeira. Assim sendo, cada elemento do Domínio de x_σ passa a explicitar um contexto prático do sistema modelado, ou seja, desempenha o mesmo papel semântico associado ao respectivo elemento de Δ^σ. Essa associação é constituída pela função que segue:

Definição: 2 (Mapeamento de refinamentos) *Dado um evento $\sigma \in \Sigma$, um conjunto de refinamentos Δ^σ e uma variável $x_\sigma = \mathcal{C}_\mathfrak{E}(\Delta^\sigma)$ com Domínio $Dom(x_\sigma)$, seja Θ_σ: $i \in Dom(x_\sigma) \rightarrow \delta \in \Delta^\sigma$ uma função bijetora definida tal que, para cada valor em $Dom(x_\sigma)$, existe um correspondente em Δ^σ, isto é*

$$\Theta_\sigma(i) = \delta \textit{ tal que } \delta \in \Delta^\sigma \textit{ e } i \in Dom(x_\sigma) \tag{16}$$

e, para

$$\Theta_\sigma(i_1) = \delta_1 \textit{ e } \Theta_\sigma(i_2) = \delta_2 \textit{ se } \delta_1 = \delta_2 \textit{ então } i_1 = i_2. \tag{17}$$

Da mesma forma, pode ser definida a função inversa $\Theta_\sigma^{-1} : \delta \in \Delta^\sigma \rightarrow i \in Dom(x_\sigma)$ como sendo $\Theta_\sigma^{-1}(\delta) = i$ tal que $\Theta_\sigma(i) = \delta$. Com base nas definições apresentadas, e em $G_\mathrm{d} = (\Delta, Q_\mathfrak{D}, q_\mathfrak{D}^\circ, Q_\mathfrak{D}^\omega, \gamma_\mathfrak{D})$ é então construído um AFE explícito, $\mathfrak{E}[G_\mathrm{d}] = (\Sigma, Q_\mathfrak{E}, Q_\mathfrak{E}^\circ, Q_\mathfrak{E}^\omega, \gamma_\mathfrak{E})$ por meio do *Algoritmo de Construção de $\mathfrak{E}$ [G_d]*. Cada componente de $\mathfrak{E}[G_\mathrm{d}]$ tem sua descrição nos itens a seguir:

- Σ é o *alfabeto* de eventos de $\mathfrak{E}[G_\mathrm{d}]$ e corresponde a $\Pi(\Delta)$;

- $Q_\mathfrak{E} \subseteq Q_\mathfrak{D} \times Dom(V)$ é o conjunto de estados. Em um AFE, cada estado $q_\mathfrak{E} \in Q_\mathfrak{E}$ passa a ser visto como uma tupla de tamanho $|V| + 1$, que ordena $k = 0, \cdots, n$ valores, e pode ser descrita na forma $(q_\mathfrak{D}, i_1 \in Dom(x_{\sigma_1}), ..., i_n \in Dom(x_{\sigma_n}))$. A posição 0 da tupla, denotada por $q_\mathfrak{E}[0]$, identifica o respectivo estado $q_\mathfrak{D} \in Q_\mathfrak{D}$ em G_d, enquanto as posições de $q_\mathfrak{E}[1]$ a $q_\mathfrak{E}[n]$ identificam o valor corrente de cada variável em V, notação que pode ser especificada para $q_\mathfrak{E}[x_{\sigma_k}]$;

- $Q_\mathfrak{E}^\circ = \{q_\mathfrak{D}^\circ\} \times Dom(V^\circ)$ é o conjunto de estados iniciais que sempre conterá a tupla $\left\{q_\mathfrak{D}^\circ, x_{\sigma_1}^\circ, ..., x_{\sigma_n}^\circ\right\}$. Como, pela Definição 1, $x_\sigma^\circ = 1$ para qualquer Δ^σ, então o conjunto de estados iniciais sempre conterá a tupla $(q_\mathfrak{D}^\circ, 1, ..., 1)$;

- $Q_\mathfrak{E}^\omega \subseteq Q_\mathfrak{D}^\omega \times Dom(V)$ é o subconjunto de estados marcados;

- $\gamma_\mathfrak{E} \subseteq Q_\mathfrak{E} \times \Sigma \rightarrow Q_\mathfrak{E}$ é a *função de transição* construída a partir de G_d de acordo com o Algoritmo 1.

Algoritmo 1: CONSTRUÇÃO DE $\mathfrak{E}$ [G_{d}]

Entrada: $(\Delta, Q_{\mathfrak{D}}, q^{\circ}_{\mathfrak{D}}, Q^{\omega}_{\mathfrak{D}}, \gamma_{\mathfrak{D}}), \Pi$

Saída: $\mathfrak{E}[G_{\mathrm{d}}] = (\Sigma, Q_{\mathfrak{E}}, Q^{\circ}_{\mathfrak{E}}, Q^{\omega}_{\mathfrak{E}}, \gamma_{\mathfrak{E}})$

1 **início**
2 $\quad \Sigma \leftarrow \Pi(\Delta)$
3 $\quad$ **para** *cada* $\sigma \in \Sigma$ **faça**
4 $\quad\quad x_\sigma \leftarrow \mathcal{C}_{\mathfrak{E}}(\Delta^\sigma)$ tal que $x^{\circ}_{\sigma} = 1, \Theta_\sigma(i) = \delta$ e $\Theta^{-1}_{\sigma}(\delta) = i$ para $i \in Dom(x_\sigma)$ e $\delta \in \Delta^\sigma$
5 $\quad$ **fim**
6 $\quad q_{aux} \leftarrow (q^{\circ}_{\mathfrak{D}}, x^{\circ}_{\sigma_1}, ..., x^{\circ}_{\sigma_n})$
7 $\quad Q_{\mathfrak{E}} \leftarrow \{q_{aux}\}$
8 $\quad Q^{\circ}_{\mathfrak{E}} \leftarrow \{q_{aux}\}$
9 $\quad Q^{\omega}_{\mathfrak{E}} \leftarrow \emptyset$
10 $\quad$ **se** $q^{\circ} \in Q^{\omega}_{\mathfrak{D}}$ **então**
11 $\quad\quad Q^{\omega}_{\mathfrak{E}} \leftarrow \{q_{aux}\}$
12 $\quad$ **fim**
13 $\quad \gamma_{\mathfrak{E}} \leftarrow \emptyset$
14 $\quad$ CONSTROITRANSICAO(q_{aux})
15 $\quad$ **retorna** $(\Sigma, Q_{\mathfrak{E}}, Q^{\circ}_{\mathfrak{E}}, Q^{\omega}_{\mathfrak{E}}, \gamma_{\mathfrak{E}})$
16 **fim**
17 **Função** CONSTROITRANSICAO$(p_{\mathfrak{E}})$
18 $\quad p_{\mathfrak{D}} \leftarrow p_{\mathfrak{E}}[0]$
19 $\quad$ **para** *cada transição* $p_{\mathfrak{D}} \xrightarrow{\delta} q_{\mathfrak{D}} \in \gamma_{\mathfrak{D}}$ *tal que* $p_{\mathfrak{D}}$ *e* $q_{\mathfrak{D}} \in Q_{\mathfrak{D}}$ *e* $\delta \in \Delta$ **faça**
20 $\quad\quad \sigma \leftarrow \Pi(\delta)$
21 $\quad\quad i \leftarrow \Theta^{-1}_{\sigma}(\delta)$
22 $\quad\quad q_{\mathfrak{E}} \leftarrow p_{\mathfrak{E}}$
23 $\quad\quad q_{\mathfrak{E}}[0] \leftarrow q_{\mathfrak{D}}$
24 $\quad\quad q_{\mathfrak{E}}[x_\sigma] \leftarrow i$
25 $\quad\quad \gamma_{\mathfrak{E}} \leftarrow \gamma_{\mathfrak{E}} \cup \left\{ p_{\mathfrak{E}} \xrightarrow{\sigma} q_{\mathfrak{E}} \right\}$
26 $\quad\quad$ **se** $q_{\mathfrak{E}} \notin Q_{\mathfrak{E}}$ **então**
27 $\quad\quad\quad Q_{\mathfrak{E}} \leftarrow Q_{\mathfrak{E}} \cup \{q_{\mathfrak{E}}\}$
28 $\quad\quad\quad$ **se** $q_{\mathfrak{D}} \in Q^{\omega}_{\mathfrak{D}}$ **então**
29 $\quad\quad\quad\quad Q^{\omega}_{\mathfrak{E}} \leftarrow Q^{\omega}_{\mathfrak{E}} \cup \{q_{\mathfrak{E}}\}$
30 $\quad\quad\quad$ **fim**
31 $\quad\quad\quad$ CONSTROITRANSICAO$(q_{\mathfrak{E}})$
32 $\quad\quad$ **fim**
33 $\quad$ **fim**

Resumidamente, este algoritmo, a cada iteração recursiva, lê um estado de G_{d} e suas transições e constrói um estado em $\mathfrak{E}$ [G_{d}] com suas transições. A linha 2 apresenta a construção de Σ por meio de Δ e do mapa Π. A linha 4 faz a construção da variável x_σ por meio de $\mathcal{C}_{\mathfrak{E}}$, para cada $\sigma \in \Sigma$. A linha 6 mostra a criação do primeiro estado de $\mathfrak{E}[G_{\mathrm{d}}]$, que é criado por meio de $q^{\circ}_{\mathfrak{D}} \in G_{\mathrm{d}}$ e das variáveis em seu estado inicial

e consiste em $q_{aux} = \left\{q^{\circ}_{\mathfrak{D}}, (x^{\circ}_{\sigma_1}, ..., x^{\circ}_{\sigma_n})\right\} = (q^{\circ}_{\mathfrak{D}}, 1, ..., 1)$. Esse estado é inserido (linhas 7 e 8) nos conjuntos de estados $Q_{\mathfrak{E}}$ e de estados iniciais $Q^{\circ}_{\mathfrak{E}}$. O conjunto de estados marcados é inicializado na linha 9 e caso $q^{\circ}_{\mathfrak{D}}$ seja um estado marcado, q_{aux} é adicionado a este conjunto pela linha 11. Essa etapa inicial se encerra com a inicialização de $\gamma_{\mathfrak{E}}$ e pela chamada da função recursiva CONSTROITRANSICAO que tem como parâmetro o estado q_{aux} (linhas 13 e 14).

A função CONSTROITRANSICAO tem por objetivo construir efetivamente os estados e transições de $\mathfrak{E}$ [G_{d}], tendo como parâmetro um estado já criado $p_{\mathfrak{E}} \in Q_{\mathfrak{E}}$. Na linha 18 a função extrai da tupla/estado $p_{\mathfrak{E}} \in Q_{\mathfrak{E}}$ a informação de qual estado em G_{d} foi utilizado para construí-la. Essa informação é armazenada em $p_{\mathfrak{D}}$. Lendo G_{d} e utilizando $p_{\mathfrak{D}}$ é possível encontrar todas as transições que partem de $p_{\mathfrak{D}}$ em G_{d}. Tais transições possuem a ocorrência de um refinamento δ e chegam a um estado $q_{\mathfrak{D}} \in Q_{\mathfrak{D}}$. Para cada uma dessas transições $p_{\mathfrak{D}} \xrightarrow{\delta} q_{\mathfrak{D}}$, em $\gamma_{\mathfrak{D}}$, constrói-se uma transição em $\gamma_{\mathfrak{E}}$ da seguinte forma:

- Com base no refinamento δ de uma dada transição, extraem-se as informações de qual evento σ está relacionado a tal refinamento (pelo mapa Π) e a qual valor i de domínio da variável x_{σ} ele está associado. (linhas 20 e 21);

- $p_{\mathfrak{E}}$ é a tupla/estado que será utilizada como início da transição (estado de partida). É feito uma cópia $q_{\mathfrak{E}}$ desta tupla, que será utilizada como fim de transição (estado de chegada). (linha 22);

- A tupla de chegada $q_{\mathfrak{E}}$ tem as entradas que precisam ser atualizadas alteradas. $q_{\mathfrak{E}}[0]$ recebe o estado de chegada $q_{\mathfrak{D}}$ da transição lida em $\gamma_{\mathfrak{D}}$ e $q_{\mathfrak{E}}[\sigma]$ recebe a informação do refinamento representada por i. (linhas 23 e 24);

- A transição é construída de $p_{\mathfrak{E}}$ para $q_{\mathfrak{E}}$ na ocorrência de σ e inserida em $\gamma_{\mathfrak{E}}$. (linha 25);

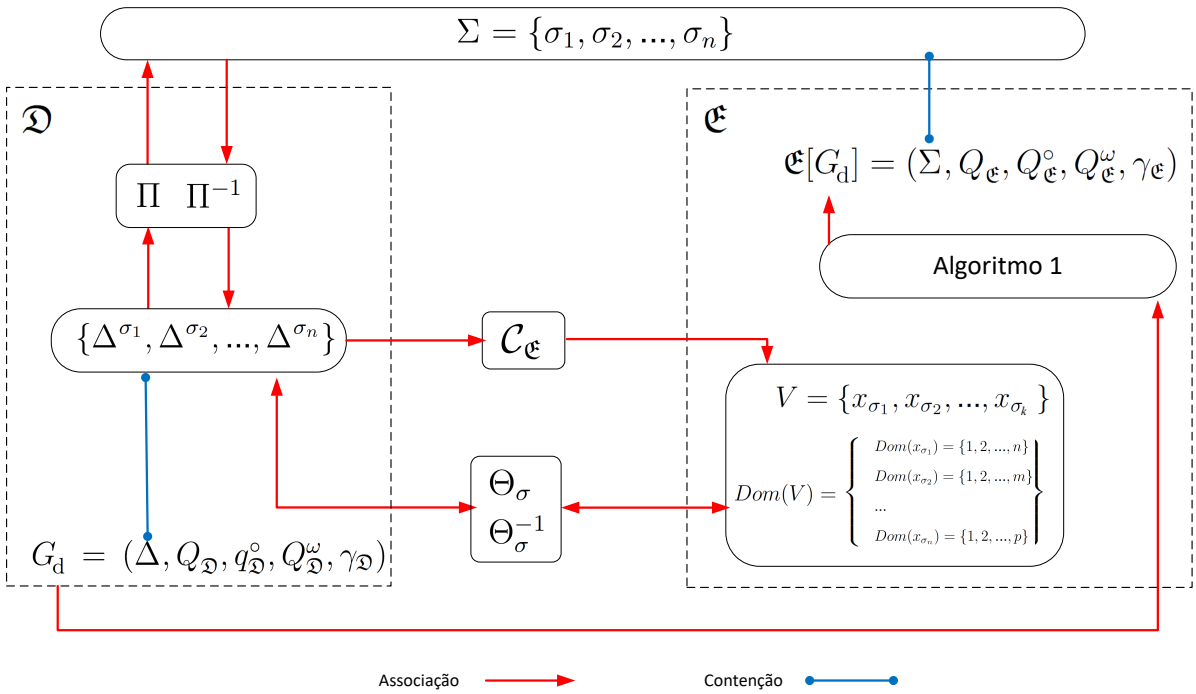
- Caso o estado criado $q_{\mathfrak{E}}$ já exista em $Q_{\mathfrak{E}}$ a função é encerrada;

- Caso contrário, $q_{\mathfrak{E}}$ é inserido no conjunto de estados $Q_{\mathfrak{E}}$ (linha 27). Se $q_{\mathfrak{D}}$ for um estado marcado em G_{d}, $q_{\mathfrak{E}}$ é inserido no conjunto de estados marcados $Q^{\omega}_{\mathfrak{E}}$ (linha 29). A função recursiva é chamada tendo como parâmetro o estado $q_{\mathfrak{E}}$ (linha 31).

A Figura 6 apresenta um resumo dos conjuntos utilizados na sistemática apresentada, bem como as associações constituídas entre eles por meio das funções,

mapas e construtores.

Figura 6 – Esquema de associação entre os conjuntos, funções e mapas utilizados na Conversão $\mathfrak{D} \rightarrow \mathfrak{E}$

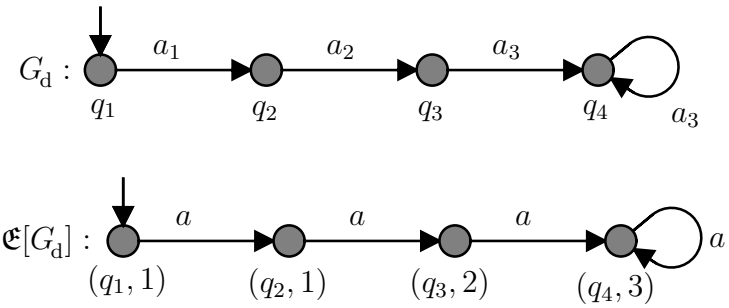


3.1.1 EXEMPLO: $\mathfrak{D} \rightarrow \mathfrak{E}$

Para melhor ilustrar o procedimento de conversão $\mathfrak{D} \rightarrow \mathfrak{E}$, utiliza-se o exemplo proposto nos capítulos anteriores. Considera-se uma máquina M que está inicialmente desligada e entra em funcionamento após um evento a e supõe-se que a mesma precisa ser colocada em manutenção após a ocorrência de três eventos a. Tal exemplo já foi resolvido sob a abordagem de distinguidores e as Figuras 2 e 4 apresentam os autômatos obtidos sob esta abordagem. É possível calcular $G_{\mathrm{d}} = G_a \| H_d$, mostrado na Figura 7, que será então utilizado na conversão $\mathfrak{D}$-$\mathfrak{E}$. O resultado da conversão $\mathfrak{E}[G_{\mathrm{d}}]$ também é apresentado.

Destaca-se que no exemplo citado $\Sigma = \{a\}$, $\Delta^a = \{a_1, a_2, a_3\}$ e $\Pi(a_1) = \Pi(a_2) = \Pi(a_3) = a$. Com base nesses conjuntos, utiliza-se $\mathcal{C}_{\mathfrak{E}}$ para gerar a variável x_a cujo domínio está associado aos refinamentos do evento a: $\mathcal{C}_{\mathfrak{E}}(\Delta^a) = x_a$ e $Dom(x_a) = \{1, 2, 3\}$. Constrói-se então a função bijetiva $\Theta_a : Dom(x_a) \rightarrow \Delta^a$ que neste caso é definida como $\Theta_a(1) = a_1$, $\Theta_a(2) = a_2$ e $\Theta_a(3) = a_3$. Os itens seguintes descrevem os passos do Algoritmo 1 executados para o exemplo proposto:

Figura 7 – Entrada e saída do exemplo de conversão $\mathfrak{D} \rightarrow \mathfrak{E}$



- $\Sigma \leftarrow \Pi(\Delta) = \{a\}$;
- $x_a \leftarrow \mathcal{C}_{\mathfrak{E}}(\Delta^{\sigma}) = \mathcal{C}_{\mathfrak{E}}(\{a_1, a_2, a_3\})$;
- $q_{aux} \leftarrow (q^{\circ}_{\mathfrak{D}}, x^{\circ}_{\sigma_1}, ..., x^{\circ}_{\sigma_n}) = (q_1, 1)$;
- $Q_{\mathfrak{E}} \leftarrow \{q_{aux}\} = \{(q_1, 1)\}$;
- $Q^{\circ}_{\mathfrak{E}} \leftarrow \{q_{aux}\} = \{(q_1, 1)\}$;
- $Q^{\omega}_{\mathfrak{E}} \leftarrow \emptyset$;
- $Q^{\omega}_{\mathfrak{E}} \leftarrow \{q_{aux}\} = \{(q_1, 1)\}$;
- $\gamma_{\mathfrak{E}} \leftarrow \emptyset$;
- CONSTROITRANSICAO($(q_1, 1)$);
  - $p_{\mathfrak{D}} \leftarrow p_{\mathfrak{E}}[0] = (q_1, 1)[0] = q_1$;
  - Para a transição $p_{\mathfrak{D}} \xrightarrow{\delta} q_{\mathfrak{D}} = q_1 \xrightarrow{a_1} q_2$;
  - $\sigma \leftarrow \Pi(\delta) = \Pi(a_1) = a$;
  - $i \leftarrow \Theta^{-1}_{\sigma}(\delta) = \Theta^{-1}_{a}(a_1) = 1$;
  - $q_{\mathfrak{E}} \leftarrow p_{\mathfrak{E}} = (q_1, 1)$;
  - $q_{\mathfrak{E}}[0] \leftarrow q_{\mathfrak{D}} = q_2$;
  - $q_{\mathfrak{E}}[x_{\sigma}] \leftarrow i = 1$. Nesse passo $q_{\mathfrak{E}} = (q_2, 1)$;
  - $\gamma_{\mathfrak{E}} \leftarrow \gamma_{\mathfrak{E}} \cup \left\{p_{\mathfrak{E}} \xrightarrow{\sigma} q_{\mathfrak{E}}\right\} = \left\{(q_1, 1) \xrightarrow{a} (q_2, 1)\right\}$;
  - $Q_{\mathfrak{E}} \leftarrow Q_{\mathfrak{E}} \cup \{q_{\mathfrak{E}}\} = \{(q_2, 1)\}$;
  - $Q^{\omega}_{\mathfrak{E}} \leftarrow Q^{\omega}_{\mathfrak{E}} \cup \{q_{\mathfrak{E}}\} = \{(q_2, 1)\}$;
- CONSTROITRANSICAO($(q_2, 1)$);
  - $p_{\mathfrak{D}} \leftarrow p_{\mathfrak{E}}[0] = (q_2, 1)[0] = q_2$;
  - Para a transição $p_{\mathfrak{D}} \xrightarrow{\delta} q_{\mathfrak{D}} = q_2 \xrightarrow{a_2} q_3$;

- $\sigma \leftarrow \Pi(\delta) = \Pi(a_2) = a$;
- $i \leftarrow \Theta_\sigma^{-1}(\delta) = \Theta_a^{-1}(a_2) = 2$;
- $q_{\mathfrak{E}} \leftarrow p_{\mathfrak{E}} = (q_2, 1)$;
- $q_{\mathfrak{E}}[0] \leftarrow q_{\mathfrak{D}} = q_3$;
- $q_{\mathfrak{E}}[x_\sigma] \leftarrow i = 2$. Nesse passo $q_{\mathfrak{E}} = (q_3, 2)$;
- $\gamma_{\mathfrak{E}} \leftarrow \gamma_{\mathfrak{E}} \cup \left\{p_{\mathfrak{E}} \xrightarrow{\sigma} q_{\mathfrak{E}}\right\} = \left\{(q_2, 1) \xrightarrow{a} (q_3, 2)\right\}$;
- $Q_{\mathfrak{E}} \leftarrow Q_{\mathfrak{E}} \cup \{q_{\mathfrak{E}}\} = \{(q_3, 2)\}$;
- $Q_{\mathfrak{E}}^{\omega} \leftarrow Q_{\mathfrak{E}}^{\omega} \cup \{q_{\mathfrak{E}}\} = \{(q_3, 2)\}$;

- CONSTROITRANSICAO$((q_3, 2))$;
  - $p_{\mathfrak{D}} \leftarrow p_{\mathfrak{E}}[0] = (q_3, 2)[0] = q_3$;
  - Para a transição $p_{\mathfrak{D}} \xrightarrow{\delta} q_{\mathfrak{D}} = q_3 \xrightarrow{a_3} q_4$;
  - $\sigma \leftarrow \Pi(\delta) = \Pi(a_3) = a$;
  - $i \leftarrow \Theta_\sigma^{-1}(\delta) = \Theta_a^{-1}(a_3) = 3$;
  - $q_{\mathfrak{E}} \leftarrow p_{\mathfrak{E}} = (q_3, 2)$;
  - $q_{\mathfrak{E}}[0] \leftarrow q_{\mathfrak{D}} = q_4$;
  - $q_{\mathfrak{E}}[x_\sigma] \leftarrow i = 3$. Nesse passo $q_{\mathfrak{E}} = (q_4, 3)$;
  - $\gamma_{\mathfrak{E}} \leftarrow \gamma_{\mathfrak{E}} \cup \left\{p_{\mathfrak{E}} \xrightarrow{\sigma} q_{\mathfrak{E}}\right\} = \left\{(q_3, 2) \xrightarrow{a} (q_4, 3)\right\}$;
  - $Q_{\mathfrak{E}} \leftarrow Q_{\mathfrak{E}} \cup \{q_{\mathfrak{E}}\} = \{(q_4, 3)\}$;
  - $Q_{\mathfrak{E}}^{\omega} \leftarrow Q_{\mathfrak{E}}^{\omega} \cup \{q_{\mathfrak{E}}\} = \{(q_4, 3)\}$;

- CONSTROITRANSICAO$((q_4, 3))$;
  - $p_{\mathfrak{D}} \leftarrow p_{\mathfrak{E}}[0] = (q_4, 3)[0] = q_4$;
  - Para a transição $p_{\mathfrak{D}} \xrightarrow{\delta} q_{\mathfrak{D}} = q_4 \xrightarrow{a_3} q_4$;
  - $\sigma \leftarrow \Pi(\delta) = \Pi(a_3) = a$;
  - $i \leftarrow \Theta_\sigma^{-1}(\delta) = \Theta_a^{-1}(a_3) = 3$;
  - $q_{\mathfrak{E}} \leftarrow p_{\mathfrak{E}} = (q_4, 3)$;
  - $q_{\mathfrak{E}}[0] \leftarrow q_{\mathfrak{D}} = q_4$;
  - $q_{\mathfrak{E}}[x_\sigma] \leftarrow i = 3$. Nesse passo $q_{\mathfrak{E}} = (q_4, 3)$;
  - $\gamma_{\mathfrak{E}} \leftarrow \gamma_{\mathfrak{E}} \cup \left\{p_{\mathfrak{E}} \xrightarrow{\sigma} q_{\mathfrak{E}}\right\} = \left\{(q_4, 3) \xrightarrow{a} (q_4, 3)\right\}$.

O AFE $\mathfrak{E}[G_{\mathrm{d}}] = (\Sigma, Q_{\mathfrak{E}}, Q^{\circ}_{\mathfrak{E}}, Q^{\omega}_{\mathfrak{E}}, \gamma_{\mathfrak{E}})$ construído de acordo com a sistemática apresentada é:

- $\Sigma = \{a\}$;
- $Q_{\mathfrak{E}} = \{(q_1, 1), (q_2, 1), (q_3, 2), (q_4, 3)\}$;
- $Q^{\circ}_{\mathfrak{E}} = \{(q_1, 1)\}$;
- $Q^{\omega}_{\mathfrak{E}} = \{(q_1, 1), (q_2, 1), (q_3, 2), (q_4, 3)\}$;
- $\gamma_{\mathfrak{E}} = \Big\{(q_1, 1) \xrightarrow{a} (q_2, 1), (q_2, 1) \xrightarrow{a} (q_3, 2), (q_3, 2) \xrightarrow{a} (q_4, 3), (q_4, 3) \xrightarrow{a} (q_4, 3)\Big\}$.

3.2 CONVERSÃO $\mathfrak{E} \rightarrow \mathfrak{D}$

O processo de conversão de modelos $\mathfrak{E} \rightarrow \mathfrak{D}$ é análogo ao apresentado anteriormente. O principal intuito é associar os domínios de variáveis existentes a refinamentos de eventos que serão criados. A principal diferença surge em como os refinamentos são caracterizados.

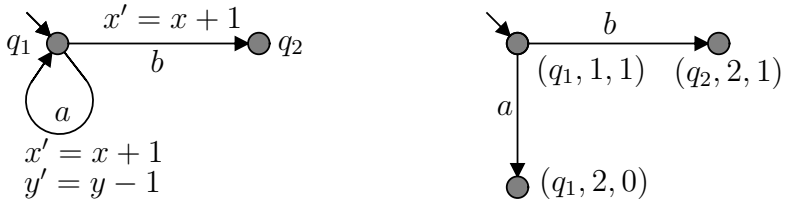
Ao se analisar um dado refinamento $\delta \in \Delta$, na seção anterior, era sabido que esse refinamento estava associado (por meio do mapa Π), a um único evento $\sigma \in \Sigma$, a uma única variável x_σ (criada por Ger) e a um único valor no domínio dessa variável $Dom(x_\sigma)$ (por meio da função Θ). A sustentação de tais pressupostos se dá sobre a *bijetividade* de Θ (Definição 2) e sobre a *preditividade* de D, o que levou à construção de uma planta refinada G_{d}, resultado da aplicação de D sobre a planta original G.

Nesse sentido, é intuitivo pensar que o resultado da conversão de G_{d} em $\mathfrak{E}$ [G_{d}] leva naturalmente a um modelo Q e V determinístico, já que a entrada G_{d} é um modelo determinístico e preditível. No entanto, essa hipótese pode não ser verdadeira no contexto dos AFE. Note que, em um AFE, uma transição com um dado evento $\sigma \in \Sigma$ pode implementar fórmulas lógicas. De maneira irrestrita, essas fórmulas são capazes de alterar os valores das variáveis nas mais diferentes formas, levando a combinações de valores (nos estados) que não necessariamente preservam o mesmo contexto do sistema o qual era descrito pelo modelo de entrada (G_{d}).

Por exemplo, uma fórmula pode implementar uma *guarda*, ou seja, uma restrição sobre a transição; duas fórmulas podem implementar *atualizações* divergentes sobre uma mesma variável; dois eventos podem atualizar uma mesma variável; um mesmo evento pode alterar mais de uma variável; etc. Para exemplificar algumas dessas características, a Figura 8 apresenta um AFE em suas formas implícita e explícita.

Considere que as variáveis x e y são inicialmente valoradas em 1 e que as tuplas da forma explícita representam os valores assumidos por x na segunda entrada da tupla e de y na terceira, isto é, as tuplas são da forma (estado, valor de x, valor de y).

Figura 8 – Exemplo de associação entre mais de um evento e mais de uma variável



É possível observar que a ocorrência do evento a faz com que os valores de ambas as variáveis, x e y, sejam atualizados para 2 e 0 respectivamente. Observa-se ainda, que a atualização dos valores de x ocorre também na ocorrência do evento b. Sendo assim, o contexto da ocorrência de um evento está associado não somente ao valor que uma variável recebe nessa ocorrência, mas a qual é esta variável, ou seja, a caracterização do contexto de um evento depende da variável atualizada e do valor assumido por ela. Esse tipo de característica complexifica o processo de conversão de AFE para Distinguidores e requer a complementação teórica. Se por um lado é necessário preservar a equivalência comportamental em relação ao modelo de entrada, por outro lado é necessário lidar com a dinamicidade nativa de um AFE. No sentido, apresenta-se a seguir um conjunto de definições que delimitam o cenário possível de atuação sobre AFE, de modo que estes possam ser utilizados de maneira consistente para a geração de modelos em $\mathfrak{D}$.

Na conversão $\mathfrak{E} \rightarrow \mathfrak{D}$, considerada neste trabalho, adota-se a forma explícita AFE $G_{\text{v}} = (\Sigma, Q_{\mathfrak{E}}, Q^{\circ}_{\mathfrak{E}}, Q^{\omega}_{\mathfrak{E}}, \gamma_{\mathfrak{E}})$ como formato de representação, uma vez que esse formato tende a facilitar a visualização dos modelos. Sem perda de generalidade, a forma implícita pode ser obtida, conversão esta que é sugerida como trabalhos futuros. Na sua forma explícita, uma variável é representada por uma entrada na tupla que define o estado do autômato. Então, a caracterização do contexto de um evento, depende do valor assumido por uma determinada entrada da tupla, na ocorrência desse evento. Considerando essas características, propõe-se a seguinte definição:

Definição: 3 (Construtor de refinamento $\mathcal{C}_{\mathfrak{D}}$) *Dado um AFE $G_{\text{v}} = (\Sigma, Q_{\mathfrak{E}}, Q^{\circ}_{\mathfrak{E}}, Q^{\omega}_{\mathfrak{E}}, \gamma_{\mathfrak{E}})$, seja $q_{\mathfrak{E}} \in Q_{\mathfrak{E}}$ um estado expresso pela tupla $(q, i_1 \in Dom(v_1), ..., i_n \in Dom(v_n))$ que tem $n+1$ posições. Para o intervalo $k = 1, \cdots, n$, denote por $q_{\mathfrak{E}}[k]$ o valor corrente assumido pela variável da posição k da tupla $q_{\mathfrak{E}}$. Para um evento $\sigma \in \Sigma$ e uma tupla $q_{\mathfrak{E}} \in Q_{\mathfrak{E}}$ define-se o construtor $\mathcal{C}_{\mathfrak{D}}(\sigma, k, q_{\mathfrak{E}}[k]) = \delta = \sigma_{(k, q_{\mathfrak{E}}[k])}$, tal que $\Pi(\mathcal{C}_{\mathfrak{D}}(\sigma, k, q_{\mathfrak{E}}[k])) = \sigma$.*

A sistemática de conversão $\mathfrak{E} \rightarrow \mathfrak{D}$ propõe a construção dos autômatos $\mathfrak{D}_A\left[G_{\mathrm{v}}\right]$ e $\mathfrak{D}_H\left[G_{\mathrm{v}}\right]$. $\mathfrak{D}_A\left[G_{\mathrm{v}}\right]$ corresponde ao comportamento de G_{v} agora sob a abordagem por Distinguidores, mas sem as distinções, isto é, $\Pi^{-1}(G)$ onde G é a planta em TCS clássica sobre a qual G_{v} foi construído. $\mathfrak{D}_H\left[G_{\mathrm{v}}\right]$ por sua vez, corresponde ao modelo que realiza a distinção, chamado anteriormente de H_d, onde $\mathcal{L}(\mathfrak{D}_H\left[G_{\mathrm{v}}\right]) = L_{\mathrm{d}}$. O autômato sob a abordagem de Distinguidores que se comporta em correspondência com G_{v} é denotado por $\mathfrak{D}$ [G_{v}] e pode ser obtido por $\mathfrak{D}[G_{\mathrm{v}}] = \mathfrak{D}_A\left[G_{\mathrm{v}}\right] \parallel \mathfrak{D}_H\left[G_{\mathrm{v}}\right]$. A geração de $\mathfrak{D}_A\left[G_{\mathrm{v}}\right]$ e $\mathfrak{D}_H\left[G_{\mathrm{v}}\right]$ é descrita nas seções que seguem.

3.2.1 GERAÇÃO DE $\mathfrak{D}_A\left[G_{\mathrm{v}}\right]$

Utilizando um AFE G_{v} e as definições apresentadas é possível construir por meio dos algoritmos 2 e 3 o AF $\mathfrak{D}_A\left[G_{\mathrm{v}}\right] = \left(\Delta_{\mathfrak{D}_A}, Q_{\mathfrak{D}_A}, q^{\circ}_{\mathfrak{D}_A}, Q^{\omega}_{\mathfrak{D}_A}, \gamma_{\mathfrak{D}_A}\right)$ tal que:

- $\Delta_{\mathfrak{D}_A}$ é o conjunto $\bigcup_{\sigma\in\Sigma}\Delta^{\sigma}$ de eventos refinados;
- $Q_{\mathfrak{D}_A}$ é o conjunto de estados tal que $Q_{\mathfrak{D}_A} = \bigcup_{q_{\mathfrak{E}}\in Q_{\mathfrak{E}}}\{q_{\mathfrak{E}}[0]\}$;
- $q^{\circ}_{\mathfrak{D}_A}$ é o estado inicial tal que $q^{\circ}_{\mathfrak{D}_A} = q_{\mathfrak{E}}[0]$ e $q_{\mathfrak{E}} \in Q^{\circ}_{\mathfrak{E}}$;
- $Q^{\omega}_{\mathfrak{D}_A}$ é o subconjunto de estados marcados tal que $Q^{\omega}_{\mathfrak{D}_A} = \bigcup_{q_{\mathfrak{E}}\in Q^{\omega}_{\mathfrak{E}}}\{q_{\mathfrak{E}}[0]\}$;
- $\gamma_{\mathfrak{D}_A} \subseteq Q_{\mathfrak{D}_A} \times \Delta_{\mathfrak{D}_A} \rightarrow Q_{\mathfrak{D}_A}$ é a *função de transição* construída a partir de G_{v} de acordo com os algoritmos 2 e 3.

Algoritmo 2: CONSTRUÇÃO DE $\mathfrak{D}_A\left[G_{\text{v}}\right]$

Entrada: $G_{\text{v}} = (\Sigma, Q_{\mathfrak{E}}, Q^{\circ}_{\mathfrak{E}}, Q^{\omega}_{\mathfrak{E}}, \gamma_{\mathfrak{E}})$

Saída: $\mathfrak{D}_A\left[G_{\text{v}}\right] = \left(\Delta_{\mathfrak{D}_A}, Q_{\mathfrak{D}_A}, q^{\circ}_{\mathfrak{D}_A}, Q^{\omega}_{\mathfrak{D}_A}, \gamma_{\mathfrak{D}_A}\right)$

início
- $N = |q_{\mathfrak{E}}| - 1$ tal que $q_{\mathfrak{E}}$ é qualquer $q_{\mathfrak{E}} \in Q_{\mathfrak{E}}$
- **para** *cada* $\sigma \in \Sigma$ **faça**
  - $\Delta^{\sigma} \leftarrow \emptyset$
- **fim**
- $Q_{\mathfrak{D}_A} \leftarrow \{q_{\mathfrak{E}}[0]\}$ tal que $q_{\mathfrak{E}}$ é qualquer $q_{\mathfrak{E}} \in Q^{\circ}_{\mathfrak{E}}$
- $q^{\circ}_{\mathfrak{D}_A} \leftarrow q_{\mathfrak{E}}[0]$ tal que $q_{\mathfrak{E}}$ é qualquer $q_{\mathfrak{E}} \in Q^{\circ}_{\mathfrak{E}}$
- $Q^{\omega}_{\mathfrak{D}_A} \leftarrow \emptyset$
- **se** $Q^{\circ}_{\mathfrak{E}} \subseteq Q^{\omega}_{\mathfrak{E}}$ **então**
  - $Q^{\omega}_{\mathfrak{D}_A} \leftarrow q_{\mathfrak{E}}[0]$ tal que $q_{\mathfrak{E}}$ é qualquer $q_{\mathfrak{E}} \in Q^{\circ}_{\mathfrak{E}}$
- **fim**
- $\gamma_{\mathfrak{D}_A} \leftarrow \emptyset$
- $Q_{aux} \leftarrow Q^{\circ}_{\mathfrak{E}}$
- **para** *cada* $q_{\mathfrak{E}} \in Q^{\circ}_{\mathfrak{E}}$ **faça**
  - CONSTROITRANSICAO2$(q_{\mathfrak{E}})$
- **fim**
- $\Delta_{\mathfrak{D}_A} \leftarrow \emptyset$
- **para** *cada* $\sigma \in \Sigma$ **faça**
  - $\Delta_{\mathfrak{D}_A} \leftarrow \Delta_{\mathfrak{D}_A} \cup \{\Delta^{\sigma}\}$
- **fim**

fim

retorna $\left(\Delta_{\mathfrak{D}_A}, Q_{\mathfrak{D}_A}, q^{\circ}_{\mathfrak{D}_A}, Q^{\omega}_{\mathfrak{D}_A}, \gamma_{\mathfrak{D}_A}\right)$

Algoritmo 3: CONSTRÓI TRANSIÇÃO 2

```
1  Função ConstroiTransicao2(p_E)
2      para cada transição p_E -σ-> q_E ∈ γ_E tal que p_E e q_E ∈ Q_E e σ ∈ Σ faça
3          para i de 1 a N faça
4              se p_E[i] ≠ q_E[i] então
5                  δ ← C_D(σ, i, q_E[i])
6                  Δ^σ ← Δ^σ ∪ {δ}
7                  γ_{D_A} ← γ_{D_A} ∪ {p_E[0] -δ-> q_E[0]}
8                  Q_{D_A} ← Q_{D_A} ∪ {q_E[0]}
9                  se q_E ∈ Q^ω então
10                     Q^ω_{D_A} ← Q^ω_{D_A} ∪ {q_E[0]}
11                 fim
12                 se q_E ∉ Q_aux então
13                     Q_aux ← Q_aux ∪ {q_E}
14                     ConstroiTransicao2(q_E)
15                 fim
16             fim
17         fim
18         se p_E[i] = q_E[i] ∀i de 1 a N então
19             Δ^σ ← Δ^σ ∪ {σ}
20             γ_{D_A} ← γ_{D_A} ∪ {p_E[0] -σ-> q_E[0]}
21             Q_{D_A} ← Q_D ∪ {q_E[0]}
22             se q_E ∈ Q^ω então
23                 Q^ω_{D_A} ← Q^ω_{D_A} ∪ {q_E[0]}
24             fim
25             se q_E ∉ Q_aux então
26                 Q_aux ← Q_aux ∪ {q}
27                 ConstroiTransicao2(q_E)
28             fim
29         fim
30     fim
```

O Algoritmo 2 inicializa os conjuntos componentes de $\mathfrak{D}_A\left[G_{\mathrm{v}}\right]$. O Algoritmo 3, a cada iteração recursiva, lê um estado de G_{v} e suas transições e constrói um estado em $\mathfrak{D}_A\left[G_{\mathrm{v}}\right]$ com suas transições. Na linha 2, N obtém a informação de quantas variáveis foram utilizadas na construção de $\mathfrak{D}_A\left[G_{\mathrm{v}}\right]$, isto é, para um estado $q_{\mathfrak{E}} = (p_{\mathfrak{D}}, v_1, ..., v_N)$, obtém-se N. São criados e inicializados os conjuntos Δ^σ que receberão os refinamentos dos eventos $\sigma \in \Sigma$ na linha 4. É possível acessar com base em um $q_{\mathfrak{E}} \in Q^\circ_{\mathfrak{E}}$ a informação de estado que está na posição 0 da tupla. Essa informação $q_{\mathfrak{E}}[0]$ é um estado inicial para $\mathfrak{D}_A\left[G_{\mathrm{v}}\right]$, é inserido no conjunto de estados de $\mathfrak{D}_A\left[G_{\mathrm{v}}\right]$, $Q_{\mathfrak{D}_A}$, e é tido como estado inicial $q^\circ_{\mathfrak{D}_A}$ de $\mathfrak{D}_A\left[G_{\mathrm{v}}\right]$ (linha 7). Nas linhas 8, e 10 ocorre a inicialização

do conjunto de estados marcados $Q^{\omega}_{\mathfrak{D}_A}$ e inserção do estado inicial $q_{\mathfrak{E}}[0]$ a ele, caso $q_{\mathfrak{E}}$ seja um estado marcado em G_{v}.

A função de transição $\gamma_{\mathfrak{D}_A}$ de $\mathfrak{D}_A\,[G_{\text{v}}]$ é inicializada na linha 12. O conjunto auxiliar de estados Q_{aux} tem o papel de registrar os estados já visitados de $Q_{\mathfrak{E}}$ e é inicializado com $Q^{\circ}_{\mathfrak{E}}$ na linha 13. Para cada estado em $Q_{\mathfrak{E}}$ é chamada na linha 15 a função CONSTROITRANSICAO2 que tem seu comportamento descrito no próximo parágrafo. Por fim, o algoritmo se encerra (linha 19) com a construção do conjunto $\Delta_{\mathfrak{D}_A}$ pelos diferentes Δ^{σ} que tiveram seu valor alterado em CONSTROITRANSICAO2.

A função CONSTROITRANSICAO2 tem por objetivo construir efetivamente os estados e transições de $\mathfrak{D}_A\,[G_{\text{v}}]$, tendo como parâmetro um estado já criado $p_{\mathfrak{E}} \in Q_{\mathfrak{E}}$. Lendo G_{v} e utilizando $p_{\mathfrak{E}}$ é possível encontrar todas as transições que partem de $p_{\mathfrak{E}}$ em G_{v}. Tais transições possuem a ocorrência de um evento σ e chegam a um estado $q_{\mathfrak{E}} \in Q_{\mathfrak{E}}$. Para cada uma dessas transições $p_{\mathfrak{E}} \xrightarrow{\sigma} q_{\mathfrak{E}}$, em $\gamma_{\mathfrak{E}}$, compara-se as entradas das tuplas/estado $p_{\mathfrak{E}}$ e $q_{\mathfrak{E}}$. Caso alguma das entradas i correspondente às variáveis ($i \neq 0$) das tuplas $p_{\mathfrak{E}}$ e $q_{\mathfrak{E}}$ sejam diferentes, ($p_{\mathfrak{E}}[i] \neq q_{\mathfrak{E}}[i]$), cria-se a transição em $\gamma_{\mathfrak{D}_A}$:

- Com base no evento σ, na posição i da tupla que teve seu valor alterado com a ocorrência de σ e no valor que foi assumido por essa posição, $q_{\mathfrak{E}}[i]$, cria-se, por meio de $\mathcal{C}_{\mathfrak{D}}$, o refinamento δ. (linha 5);

- O refinamento δ criado é inserido no conjunto Δ^{σ}. (linha 6);

- A transição é criada partindo de $p_{\mathfrak{E}}[0]$, informação do estado que está em $p_{\mathfrak{E}}$, e chegando em $q_{\mathfrak{E}}[0]$, informação do estado que está em $q_{\mathfrak{E}}$, sob a execução do evento refinado δ. Essa transição é atribuída a $\gamma_{\mathfrak{D}_A}$.(linha 7);

- Esse estado $q_{\mathfrak{E}}[0]$ é atribuído ao conjunto de estados $Q_{\mathfrak{D}_A}$ e, em caso de $q_{\mathfrak{E}}$ ser um estado marcado em G_{v}, é atribuído ao conjunto de estados marcados $Q^{\omega}_{\mathfrak{D}_A}$ (linhas 8 e 10);

- Se o estado $q_{\mathfrak{E}}$ já havia sido visitado, a função se encerra. Do contrário, o estado $q_{\mathfrak{E}} \in Q_{\mathfrak{E}}$ é adicionado ao conjunto de estados que já foram visitados Q_{aux} e a função é chamada recursivamente tendo ele como parâmetro. (linhas 13 e 14).

Dada a transição $p_{\mathfrak{E}} \xrightarrow{\sigma} q_{\mathfrak{E}}$, em $\gamma_{\mathfrak{E}}$, se ao analisar $p_{\mathfrak{E}}[i]$ e $q_{\mathfrak{E}}[i]$ para todo i que representa variáveis (i de 1 a N), e observar-se que não houve alterações, significa que a ocorrência do evento σ não está atribuída a nenhum contexto por meio das variáveis, isto é, os valores das variáveis não se alteram quando o evento ocorre.

Assim sendo, tem-se que σ é o seu próprio refinamento e por si só já contempla seu contexto $(\Delta^{\sigma} = \{\sigma\})$. (linha 19):

- A transição é criada partindo de $p_{\mathfrak{E}}[0]$, informação do estado que está em $p_{\mathfrak{E}}$, e chegando em $q_{\mathfrak{E}}[0]$, informação do estado que está em $q_{\mathfrak{E}}$, sob a execução do evento refinado σ. Essa transição é atribuída a $\gamma_{\mathfrak{D}_A}$.(linha 20);

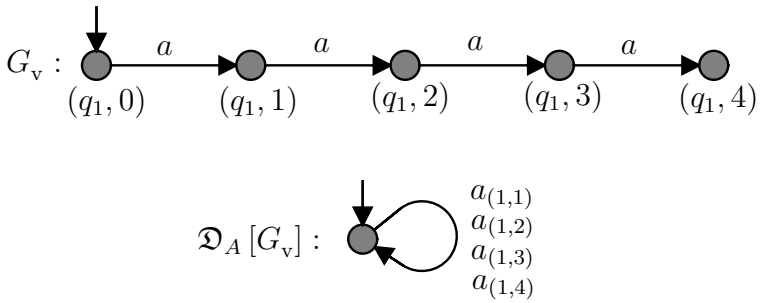
- Esse estado $q_{\mathfrak{E}}[0]$ é atribuído ao conjunto de estados $Q_{\mathfrak{D}_A}$ e, em caso de $q_{\mathfrak{E}}$ ser um estado marcado em G_{v}, é atribuído ao conjunto de estados marcados $Q^{\omega}_{\mathfrak{D}_A}$ (linhas 21 e 23);

- Se o estado $q_{\mathfrak{E}}$ já havia sido visitado, a função se encerra. Do contrário, o estado $q_{\mathfrak{E}} \in Q_{\mathfrak{E}}$ é adicionado ao conjunto de estados que já foram visitados Q_{aux} e a função é chamada recursivamente tendo ele como parâmetro. (linhas 26 e 27).

3.2.1.1 EXEMPLO: $\mathfrak{E} \rightarrow \mathfrak{D}$ GERANDO $\mathfrak{D}_A\,[G_{\mathrm{v}}]$

O procedimento de conversão $\mathfrak{E} \rightarrow \mathfrak{D}$ e geração de $\mathfrak{D}_A\,[G_{\mathrm{v}}]$ é ilustrado por meio de sua aplicação sobre o exemplo proposto anteriormente: Uma máquina M entra em funcionamento com o acontecimento de um evento a que em sua terceira ocorrência coloca a máquina em estado de manutenção. A modelagem sob a abordagem de AFE desse comportamento foi apresentada na Figura 5, onde $Dom(x) = \{0, 1, 2, 3, 4\}$ e $x^o = 0$. A Figura 9 mostra esse modelo em sua forma explícita, que será considerada como entrada para exemplificação da sistemática de conversão $\mathfrak{E}$-$\mathfrak{D}$ e a saída da geração de $\mathfrak{D}_A\,[G_{\mathrm{v}}]$.

Figura 9 – Entrada e saída do exemplo de conversão $\mathfrak{E} \rightarrow \mathfrak{D}$ com geração de $\mathfrak{D}_A\,[G_{\mathrm{v}}]$



Destaca-se que no exemplo citado $\Sigma = \{a\}$. Os itens seguintes descrevem os passos do Algoritmo 2 e 3 executados para o exemplo proposto:

- $q_{\mathfrak{E}} = (q_1, 0)$;

- $N = |q_{\mathfrak{E}}| - 1 = |(q_1, 0)| - 1 = 1$;
- $\Delta^a \leftarrow \emptyset$;
- $Q_{\mathfrak{D}_A} \leftarrow \{q_{\mathfrak{E}}[0]\} = \{q_1\}$;
- $q^{\circ}_{\mathfrak{D}_A} \leftarrow q_{\mathfrak{E}}[0] = q_1$;
- $Q^{\omega}_{\mathfrak{D}_A} \leftarrow \emptyset$;
- $Q^{\omega}_{\mathfrak{D}_A} \leftarrow q_{\mathfrak{E}}[0] = \{q_1\}$;
- $\gamma_{\mathfrak{D}_A} \leftarrow \emptyset$;
- $Q_{aux} \leftarrow Q^{\circ}_{\mathfrak{E}} = \{(q_1, 0)\}$;
- CONSTROITRANSICAO2$((q_1, 0))$;
  - para a transição $p_{\mathfrak{E}} \xrightarrow{\sigma} q_{\mathfrak{E}} = (q_1, 0) \xrightarrow{a} (q_1, 1)$;
  - $\delta \leftarrow \mathcal{C}_{\mathfrak{D}}(\sigma, i, q_{\mathfrak{E}}[i]) = \mathcal{C}_{\mathfrak{D}}(a, 1, 1) = a_{(1,1)}$;
  - $\Delta^a \leftarrow \Delta^a \cup \{\delta\} = \left\{a_{(1,1)}\right\}$;
  - $\gamma_{\mathfrak{D}_A} \leftarrow \gamma_{\mathfrak{D}_A} \cup \left\{p_{\mathfrak{E}}[0] \xrightarrow{\delta} q_{\mathfrak{E}}[0]\right\} = \left\{q_1 \xrightarrow{a_{(1,1)}} q_1\right\}$;
  - $Q_{\mathfrak{D}_A} \leftarrow Q_{\mathfrak{D}_A} \cup \{q_{\mathfrak{E}}[0]\} = \{q_1\}$;
  - $Q^{\omega}_{\mathfrak{D}_A} \leftarrow Q^{\omega}_{\mathfrak{D}_A} \cup \{q_{\mathfrak{E}}[0]\} = \{q_1\}$;
  - $Q_{aux} \leftarrow Q_{aux} \cup \{q_{\mathfrak{E}}\} = \{(q_1, 1)\}$;
- CONSTROITRANSICAO2$((q_1, 1))$;
  - para a transição $p_{\mathfrak{E}} \xrightarrow{\sigma} q_{\mathfrak{E}} = (q_1, 1) \xrightarrow{a} (q_1, 2)$;
  - $\delta \leftarrow \mathcal{C}_{\mathfrak{D}}(\sigma, i, q_{\mathfrak{E}}[i]) = \mathcal{C}_{\mathfrak{D}}(a, 1, 2) = a_{(1,2)}$;
  - $\Delta^a \leftarrow \Delta^a \cup \{\delta\} = \left\{a_{(1,2)}\right\}$;
  - $\gamma_{\mathfrak{D}_A} \leftarrow \gamma_{\mathfrak{D}_A} \cup \left\{p_{\mathfrak{E}}[0] \xrightarrow{\delta} q_{\mathfrak{E}}[0]\right\} = \left\{q_1 \xrightarrow{a_{(1,2)}} q_1\right\}$;
  - $Q_{\mathfrak{D}_A} \leftarrow Q_{\mathfrak{D}} \cup \{q_{\mathfrak{E}}[0]\} = \{q_1\}$;
  - $Q^{\omega}_{\mathfrak{D}_A} \leftarrow Q^{\omega}_{\mathfrak{D}_A} \cup \{q_{\mathfrak{E}}[0]\} = \{q_1\}$;
  - $Q_{aux} \leftarrow Q_{aux} \cup \{q_{\mathfrak{E}}\} = \{(q_1, 2)\}$;
- CONSTROITRANSICAO2$((q_1, 2))$;
  - para a transição $p_{\mathfrak{E}} \xrightarrow{\sigma} q_{\mathfrak{E}} = (q_1, 2) \xrightarrow{a} (q_1, 3)$;
  - $\delta \leftarrow \mathcal{C}_{\mathfrak{D}}(\sigma, i, q_{\mathfrak{E}}[i]) = \mathcal{C}_{\mathfrak{D}}(a, 1, 3) = a_{(1,3)}$;

  - $\Delta^a \leftarrow \Delta^a \cup \{\delta\} = \left\{a_{(1,3)}\right\}$;
  - $\gamma_{\mathfrak{D}_A} \leftarrow \gamma_{\mathfrak{D}_A} \cup \left\{p_{\mathfrak{E}}[0] \xrightarrow{\delta} q_{\mathfrak{E}}[0]\right\} = \left\{q_1 \xrightarrow{a_{(1,3)}} q_1\right\}$;
  - $Q_{\mathfrak{D}_A} \leftarrow Q_{\mathfrak{D}_A} \cup \{q_{\mathfrak{E}}[0]\} = \{q_1\}$;
  - $Q^{\omega}_{\mathfrak{D}_A} \leftarrow Q^{\omega}_{\mathfrak{D}_A} \cup \{q_{\mathfrak{E}}[0]\} = \{q_1\}$;
  - $Q_{aux} \leftarrow Q_{aux} \cup \{q_{\mathfrak{E}}\} = \{(q_1, 3)\}$;

- CONSTROITRANSICAO2$((q_1, 3))$;
  - para a transição $p_{\mathfrak{E}} \xrightarrow{\sigma} q_{\mathfrak{E}} = (q_1, 3) \xrightarrow{a} (q_1, 4)$;
  - $\delta \leftarrow \mathcal{C}_{\mathfrak{D}}(\sigma, i, q_{\mathfrak{E}}[i]) = \mathcal{C}_{\mathfrak{D}}(a, 1, 4) = a_{(1,4)}$;
  - $\Delta^a \leftarrow \Delta^a \cup \{\delta\} = \left\{a_{(1,4)}\right\}$;
  - $\gamma_{\mathfrak{D}_A} \leftarrow \gamma_{\mathfrak{D}_A} \cup \left\{p_{\mathfrak{E}}[0] \xrightarrow{\delta} q_{\mathfrak{E}}[0]\right\} = \left\{q_1 \xrightarrow{a_{(1,4)}} q_1\right\}$;
  - $Q_{\mathfrak{D}_A} \leftarrow Q_{\mathfrak{D}_A} \cup \{q_{\mathfrak{E}}[0]\} = \{q_1\}$;
  - $Q^{\omega}_{\mathfrak{D}_A} \leftarrow Q^{\omega}_{\mathfrak{D}_A} \cup \{q_{\mathfrak{E}}[0]\} = \{q_1\}$;
  - $Q_{aux} \leftarrow Q_{aux} \cup \{q_{\mathfrak{E}}\} = \{(q_1, 4)\}$;

- $\Delta_{\mathfrak{D}_A} \leftarrow \emptyset$;
- $\Delta_{\mathfrak{D}_A} \leftarrow \Delta_{\mathfrak{D}_A} \cup \{\Delta^a\}$;

O Autômato $\mathfrak{D}_A[G_{\mathrm{v}}] = \left(\Delta_{\mathfrak{D}_A}, Q_{\mathfrak{D}_A}, q^{\circ}_{\mathfrak{D}_A}, Q^{\omega}_{\mathfrak{D}_A}, \gamma_{\mathfrak{D}_A}\right)$ construído de acordo com a sistemática apresentada é composto por:

- $\Delta^a = \{a_{(1,1)}, a_{(1,2)}, a_{(1,3)}, a_{(1,4)}\}$
- $Q_{\mathfrak{D}_A} = \{q_1\}$;
- $Q^{\circ}_{\mathfrak{D}_A} = \{q_1\}$;
- $Q^{\omega}_{\mathfrak{D}_A} = \{q_1\}$;
- $\gamma_{\mathfrak{D}_A} = \left\{q_1 \xrightarrow{a_{(1,1)}} q_1, q_1 \xrightarrow{a_{(1,2)}} q_1, q_1 \xrightarrow{a_{(1,3)}} q_1, q_1 \xrightarrow{a_{(1,4)}} q_1\right\}$

3.2.2 GERAÇÃO DE $\mathfrak{D}_H[G_{\mathrm{v}}]$

Esta seção propõe a construção do modelo $\mathfrak{D}_H[G_{\mathrm{v}}]$ que será o responsável por realizar a distinção do modelo gerado anteriormente $\mathfrak{D}_A[G_{\mathrm{v}}]$. Para essa construção propõe-se as definições:

Definição: 4 (Construtor de estado $\mathcal{C}_Q$) *Dado um AFE $G_{\text{v}} = (\Sigma, Q_{\mathfrak{E}}, Q^{\circ}_{\mathfrak{E}}, Q^{\omega}_{\mathfrak{E}}, \gamma_{\mathfrak{E}})$, tal que o seu conjunto de estados é $Q_{\mathfrak{E}}$ e n é a quantidade de estados $n = |Q_{\mathfrak{E}}|$, define-se $\mathcal{C}_Q$ como um construtor de estados que recebe um parâmetro n e retorna um conjunto de estados associados $Q_h = \{q_1, \cdots, q_n\}$ tal que $n = |Q_h|$, isto é, $\mathcal{C}_Q(n) = Q_h$ e $n = |Q_h| = |Q_{\mathfrak{E}}|$.*

Definição: 5 (Mapeamento de estados) *Dado um conjunto de estados $Q_{\mathfrak{E}} \in G_{\text{v}}$ e um conjunto de estados gerados $Q_h = \mathcal{C}_Q(n)$ tal que $n = |Q_h| = |Q_{\mathfrak{E}}|$, seja $\Phi\colon q \in Q_h \rightarrow q_{\mathfrak{E}} \in Q_{\mathfrak{E}}$ uma função bijetora definida tal que, para cada estado em $Q_{\mathfrak{E}}$, existe um correspondente em Q_h, isto é*

$$\Phi(q) = q_{\mathfrak{E}} \text{ tal que } q \in Q_h \text{ e } q_{\mathfrak{E}} \in Q_{\mathfrak{E}} \tag{18}$$

e, para

$$\Phi(q_1) = q_{\mathfrak{E}1} \text{ e } \Phi(q_2) = q_{\mathfrak{E}2} \text{ se } q_1 = q_2 \text{ então } q_{\mathfrak{E}1} = q_{\mathfrak{E}2}. \tag{19}$$

Da mesma forma, pode ser definida a função inversa $\Phi^{-1} : q_{\mathfrak{E}} \in Q_{\mathfrak{E}} \rightarrow q \in Q_h$ como sendo $\Phi^{-1}(q_{\mathfrak{E}}) = q$ tal que $\Phi(q) = q_{\mathfrak{E}}$. Utilizando G_{v} e $\mathfrak{D}_A[G_{\text{v}}]$, constrói-se por meio do Algoritmo 4 o AF $\mathfrak{D}_H[G_{\text{v}}] = \left(\Delta_{\mathfrak{D}_H}, Q_{\mathfrak{D}_H}, q^{\circ}_{\mathfrak{D}_H}, Q^{\omega}_{\mathfrak{D}_H}, \gamma_{\mathfrak{D}_H}\right)$ responsável pela distinção, tal que:

- $\Delta_{\mathfrak{D}_H}$ é o conjunto de eventos refinados calculado pelo Algoritmo 4;
- $Q_{\mathfrak{D}_H}$ é o conjunto de estados tal que $Q_{\mathfrak{D}_H} = \mathcal{C}_Q(Q_{\mathfrak{E}})$;
- $q^{\circ}_{\mathfrak{D}_H}$ é o estado inicial tal que $q^{\circ}_{\mathfrak{D}_H} = \Phi^{-1}(q_{\mathfrak{E}})$ e $q_{\mathfrak{E}} \in Q^{\circ}_{\mathfrak{E}}$;
- $Q^{\omega}_{\mathfrak{D}_H}$ é o subconjunto de estados marcados tal que $Q^{\omega}_{\mathfrak{D}_H} = \bigcup_{q_{\mathfrak{E}} \in Q^{\omega}_{\mathfrak{E}}} \{\Phi^{-1}(q_{\mathfrak{E}})\}$;
- $\gamma_{\mathfrak{D}_H} \subseteq Q_{\mathfrak{D}_H} \times \Delta_{\mathfrak{D}_H} \rightarrow Q_{\mathfrak{D}_H}$ é a *função de transição* construída a partir de G_{v} de acordo com o Algoritmo 4:

Algoritmo 4: CONSTRUÇÃO DE $\mathfrak{D}_H\left[G_{\mathrm{v}}\right]$

Entrada: $G_{\mathrm{v}} = (\Sigma, Q_{\mathfrak{E}}, Q^{\circ}_{\mathfrak{E}}, Q^{\omega}_{\mathfrak{E}}, \gamma_{\mathfrak{E}})$

Saída: $\mathfrak{D}_H\left[G_{\mathrm{v}}\right] = \left(\Delta_{\mathfrak{D}_H}, Q_{\mathfrak{D}_H}, q^{\circ}_{\mathfrak{D}_H}, Q^{\omega}_{\mathfrak{D}_H}, \gamma_{\mathfrak{D}_H}\right)$

1 **início**
2 $N = |q_{\mathfrak{E}}| - 1$ tal que $q_{\mathfrak{E}}$ é qualquer $q_{\mathfrak{E}} \in Q_{\mathfrak{E}}$
3 **para** *cada* $\sigma \in \Sigma$ **faça**
4 $\Delta^{\sigma} \leftarrow \emptyset$
5 **fim**
6 $Q_{\mathfrak{D}_H} \leftarrow \mathcal{C}_Q(Q_{\mathfrak{E}})$ tal que $\Phi(q_{\mathfrak{D}_H}) = q_{\mathfrak{E}}$ e $\Phi^{-1}(q_{\mathfrak{E}}) = q_{\mathfrak{D}_H}$ para todo $q_{\mathfrak{D}_H} \in Q_{\mathfrak{D}_H}$ e $q_{\mathfrak{E}} \in Q_{\mathfrak{E}}$
7 $q^{\circ}_{\mathfrak{D}_H} \leftarrow \Phi^{-1}(q_{\mathfrak{E}})$ tal que $q_{\mathfrak{E}} \in Q^{\circ}_{\mathfrak{E}}$
8 $Q^{\omega}_{\mathfrak{D}_H} \leftarrow \emptyset$
9 **para** *cada estado* $q_{\mathfrak{E}} \in Q^{\omega}_{\mathfrak{E}}$ **faça**
10 $Q^{\omega}_{\mathfrak{D}_H} \leftarrow Q^{\omega}_{\mathfrak{D}_H} \cup \left\{\Phi^{-1}(q_{\mathfrak{E}})\right\}$
11 **fim**
12 $\gamma_{\mathfrak{D}_H} \leftarrow \emptyset$
13 **para** *cada transição* $p_{\mathfrak{E}} \xrightarrow{\sigma} q_{\mathfrak{E}} \in \gamma_{\mathfrak{E}}$ *tal que* $p_{\mathfrak{E}}$ *e* $q_{\mathfrak{E}} \in Q_{\mathfrak{E}}$ *e* $\sigma \in \Sigma$ **faça**
14 **para** i *de* 1 *a* N **faça**
15 **se** $p_{\mathfrak{E}}[i] \neq q_{\mathfrak{E}}[i]$ **então**
16 $\delta \leftarrow \mathcal{C}_{\mathfrak{D}}(\sigma, i, q_{\mathfrak{E}}[i])$
17 $\Delta^{\sigma} \leftarrow \Delta^{\sigma} \cup \{\delta\}$
18 $\gamma_{\mathfrak{D}_H} \leftarrow \gamma_{\mathfrak{D}_H} \cup \left\{\Phi^{-1}(p_{\mathfrak{E}}) \xrightarrow{\delta} \Phi^{-1}(q_{\mathfrak{E}})\right\}$
19 **fim**
20 **fim**
21 **fim**
22 $\Delta_{\mathfrak{D}_H} \leftarrow \emptyset$
23 **para** *cada* $\sigma \in \Sigma$ *tal que* $|\Delta^{\sigma}| > 1$ **faça**
24 $\Delta_{\mathfrak{D}_H} \leftarrow \Delta_{\mathfrak{D}_H} \cup \{\Delta^{\sigma}\}$
25 **fim**
26 **fim**
27 **retorna** $\left(\Delta_{\mathfrak{D}_H}, Q_{\mathfrak{D}_H}, q^{\circ}_{\mathfrak{D}_H}, Q^{\omega}_{\mathfrak{D}_H}, \gamma_{\mathfrak{D}_H}\right)$

Este algoritmo se baseia na estrutura de G_{v} para construir $\mathfrak{D}_H\left[G_{\mathrm{v}}\right]$. Na linha 2 é calculada a quantidade de entradas N nos estados de G_{v} correspondentes a variáveis. Os conjuntos de refinamentos são inicializados na linha 4. Em seguida, para cada estado $q_{\mathfrak{E}} \in Q_{\mathfrak{E}}$, constrói-se um estado correspondente $q_{\mathfrak{D}_H} \in Q_{\mathfrak{D}_H}$ cuja relação com $q_{\mathfrak{E}}$ é dada por Φ e Φ^{-1} (linha 6). O estado inicial $q^{\circ}_{\mathfrak{D}_H}$ é criado na linha 7. Nas linhas 8 e 10 constrói-se o conjunto de estados marcados $Q^{\omega}_{\mathfrak{D}_H}$ tal que os correspondentes sejam marcados em G_{v}.

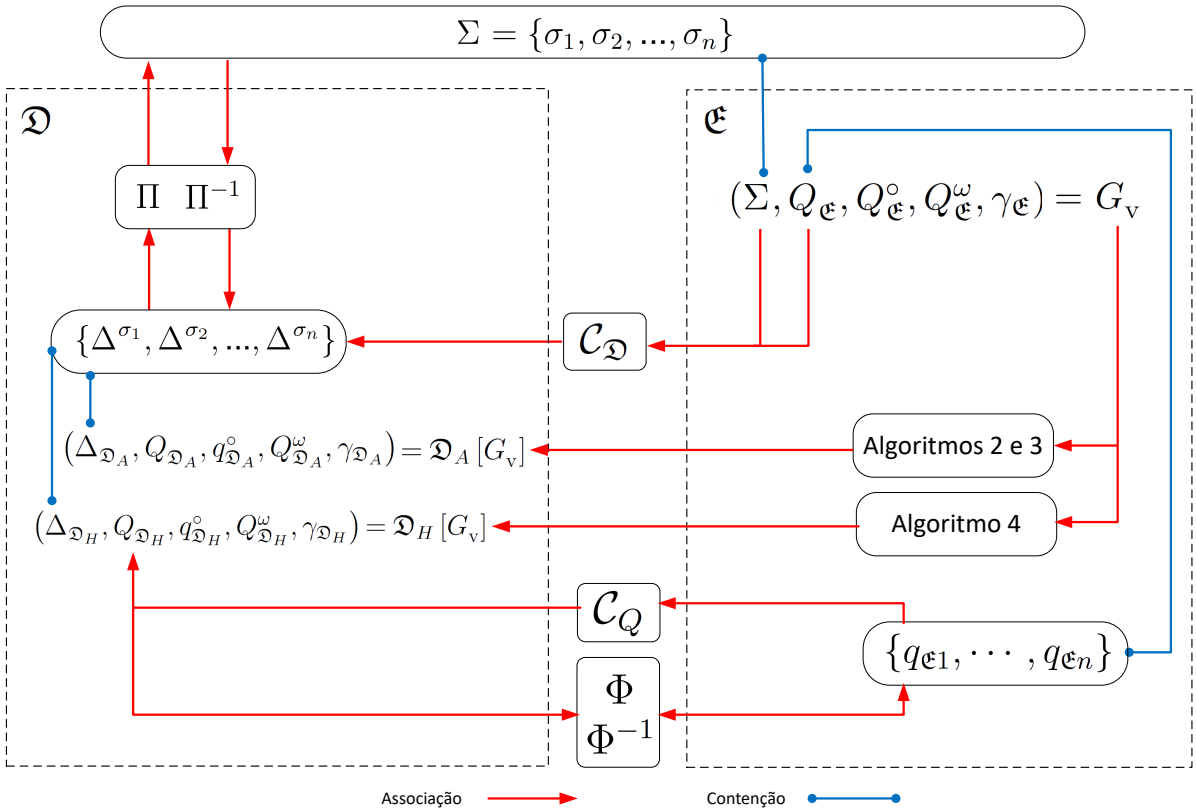
Em seguida as transições de G_{v} são analisadas. Tais transições partem de um estado $p_{\mathfrak{E}}$, na ocorrência de um evento σ e chegam a um estado $q_{\mathfrak{E}} \in Q_{\mathfrak{E}}$. Para cada uma dessas transições $p_{\mathfrak{E}} \xrightarrow{\sigma} q_{\mathfrak{E}}$, em $\gamma_{\mathfrak{E}}$, compara-se as entradas das tuplas/estado $p_{\mathfrak{E}}$

e $q_{\mathfrak{E}}$. Caso alguma das entradas i, correspondente às variáveis ($i \neq 0$) das tuplas $p_{\mathfrak{E}}$ e $q_{\mathfrak{E}}$, sejam diferentes (i.e. $p_{\mathfrak{E}}[i] \neq q_{\mathfrak{E}}[i]$), então cria-se a transição em $\gamma_{\mathfrak{D}_H}$ com base nos seguintes aspectos:

- Com base no evento σ, na posição i da tupla que teve seu valor alterado com a ocorrência de σ e no valor que foi assumido por essa posição ($q_{\mathfrak{E}}[i]$) cria-se, por meio de $\mathcal{C}_{\mathfrak{D}}$, o refinamento δ. (linha 16);
- O refinamento δ criado é inserido no conjunto Δ^{σ}. (linha 17);
- A transição é criada partindo do correspondente a $p_{\mathfrak{E}}$, calculado por Φ^{-1}, e chegando em $q_{\mathfrak{E}}$, sob a execução do evento refinado δ. Essa transição é atribuída a $\gamma_{\mathfrak{D}_H}$.(linha 18).

O algoritmo se encerra com a construção de $\Delta_{\mathfrak{D}_H}$ cujos elementos são os Δ^{σ} que possuem refinamentos (linha 24). A Figura 10 adapta a Figura 6 para mostrar os elementos do processo inverso de conversão.

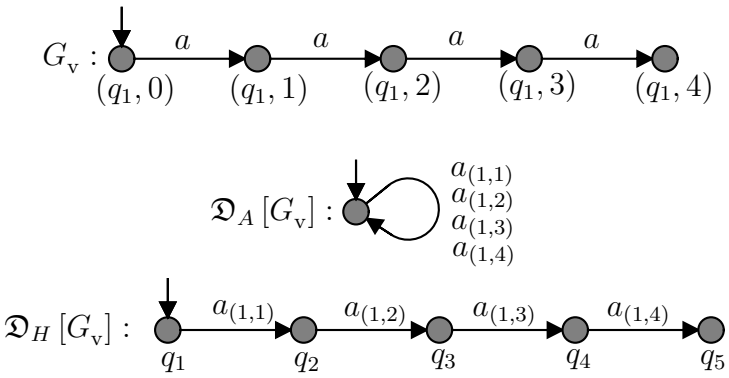
Figura 10 – Esquema de associação entre os conjuntos, funções e mapas utilizados na Conversão $\mathfrak{E} \rightarrow \mathfrak{D}$



3.2.2.1 EXEMPLO: $\mathfrak{E} \rightarrow \mathfrak{D}$ GERANDO $\mathfrak{D}_H\left[G_{\mathrm{v}}\right]$

O procedimento de conversão de $\mathfrak{E} \rightarrow \mathfrak{D}$ e geração de $\mathfrak{D}_H\left[G_{\mathrm{v}}\right]$ é ilustrado a seguir. A modelagem sob a abordagem de AFE desse comportamento foi apresentada na Figura 5, onde $Dom(x) = \{0, 1, 2, 3, 4\}$, $\Sigma = \{a\}$ e $x^o = 0$. A Figura 11 mostra G_{v} e as saídas do processo de conversão $\mathfrak{D}_A\left[G_{\mathrm{v}}\right]$ e $\mathfrak{D}_H\left[G_{\mathrm{v}}\right]$.

Figura 11 – Entrada e saída do exemplo de conversão $\mathfrak{E} \rightarrow \mathfrak{D}$ com geração de $\mathfrak{D}_A\left[G_{\mathrm{v}}\right]$ e $\mathfrak{D}_H\left[G_{\mathrm{v}}\right]$



Os itens seguintes descrevem os passos do Algoritmo 4 executados para o exemplo gerando $\mathfrak{D}_H\left[G_{\mathrm{v}}\right]$:

- $q_{\mathfrak{E}} = (q_1, 0)$;
- $N = |q_{\mathfrak{E}}| - 1 = |(q_1, 0)| - 1 = 1$;
- $\Delta^a \leftarrow \emptyset$;
- $Q_{\mathfrak{D}_H} \leftarrow \mathcal{C}_Q(Q_{\mathfrak{E}}) = \{q_1, q_2, q_3, q_4, q_5\} = \mathcal{C}_Q(\{(q_1, 0), (q_1, 1), (q_1, 2), (q_1, 3), (q_1, 4)\})$ tal que $\Phi(q_1) = (q_1, 0)$, $\Phi(q_2) = (q_1, 1)$, $\Phi(q_3) = (q_1, 2)$, $\Phi(q_4) = (q_1, 3)$, $\Phi(q_5) = (q_1, 4)$, $\Phi^{-1}((q_1, 0)) = q_1$, $\Phi^{-1}((q_1, 1)) = q_2$, $\Phi^{-1}((q_1, 2)) = q_3$, $\Phi^{-1}((q_1, 3)) = q_4$e $\Phi^{-1}((q_1, 4)) = q_5$;
- $q^{\circ}_{\mathfrak{D}_H} \leftarrow \{\Phi^{-1}((q_1, 0)) = q_1\}$;
- $Q^{\omega}_{\mathfrak{D}_H} \leftarrow \{q_1, q_2, q_3, q_4, q_5\}$;
- $\gamma_{\mathfrak{D}_H} \leftarrow \emptyset$;
- para a transição $p_{\mathfrak{E}} \xrightarrow{\sigma} q_{\mathfrak{E}} = (q_1, 0) \xrightarrow{a} (q_1, 1)$;
  - $\delta \leftarrow \mathcal{C}_{\mathfrak{D}}(\sigma, i, q_{\mathfrak{E}}[i]) = \mathcal{C}_{\mathfrak{D}}(a, 1, 1) = a_{(1,1)}$;
  - $\Delta^a \leftarrow \Delta^a \cup \{\delta\} = \{a_{(1,1)}\}$;

  - $\gamma_{\mathfrak{D}_H} \leftarrow \gamma_{\mathfrak{D}_H} \cup \left\{\Phi^{-1}(p_{\mathfrak{E}}) \xrightarrow{\delta} \Phi^{-1}(q_{\mathfrak{E}})\right\} = \left\{\Phi^{-1}(q_1, 0) \xrightarrow{a_{(1,1)}} \Phi^{-1}(q_1, 1)\right\} = \left\{q_1 \xrightarrow{a_{(1,1)}} q_2\right\}$;

- para a transição $p_{\mathfrak{E}} \xrightarrow{\sigma} q_{\mathfrak{E}} = (q_1, 1) \xrightarrow{a} (q_1, 2)$;

  - $\delta \leftarrow \mathcal{C}_{\mathfrak{D}}(\sigma, i, q_{\mathfrak{E}}[i]) = \mathcal{C}_{\mathfrak{D}}(a, 1, 2) = a_{(1,2)}$;
  - $\Delta^a \leftarrow \Delta^a \cup \{\delta\} = \left\{a_{(1,2)}\right\}$;
  - $\gamma_{\mathfrak{D}_H} \leftarrow \gamma_{\mathfrak{D}_H} \cup \left\{\Phi^{-1}(p_{\mathfrak{E}}) \xrightarrow{\delta} \Phi^{-1}(q_{\mathfrak{E}})\right\} = \left\{\Phi^{-1}(q_1, 1) \xrightarrow{a_{(1,2)}} \Phi^{-1}(q_1, 2)\right\} = \left\{q_2 \xrightarrow{a_{(1,2)}} q_3\right\}$;

- para a transição $p_{\mathfrak{E}} \xrightarrow{\sigma} q_{\mathfrak{E}} = (q_1, 2) \xrightarrow{a} (q_1, 3)$;

  - $\delta \leftarrow \mathcal{C}_{\mathfrak{D}}(\sigma, i, q_{\mathfrak{E}}[i]) = \mathcal{C}_{\mathfrak{D}}(a, 1, 3) = a_{(1,3)}$;
  - $\Delta^a \leftarrow \Delta^a \cup \{\delta\} = \left\{a_{(1,3)}\right\}$;
  - $\gamma_{\mathfrak{D}_H} \leftarrow \gamma_{\mathfrak{D}_H} \cup \left\{\Phi^{-1}(p_{\mathfrak{E}}) \xrightarrow{\delta} \Phi^{-1}(q_{\mathfrak{E}})\right\} = \left\{\Phi^{-1}(q_1, 2) \xrightarrow{a_{(1,3)}} \Phi^{-1}(q_1, 3)\right\} = \left\{q_3 \xrightarrow{a_{(1,3)}} q_4\right\}$;

- para a transição $p_{\mathfrak{E}} \xrightarrow{\sigma} q_{\mathfrak{E}} = (q_1, 3) \xrightarrow{a} (q_1, 4)$;

  - $\delta \leftarrow \mathcal{C}_{\mathfrak{D}}(\sigma, i, q_{\mathfrak{E}}[i]) = \mathcal{C}_{\mathfrak{D}}(a, 1, 4) = a_{(1,4)}$;
  - $\Delta^a \leftarrow \Delta^a \cup \{\delta\} = \left\{a_{(1,4)}\right\}$;
  - $\gamma_{\mathfrak{D}_H} \leftarrow \gamma_{\mathfrak{D}_H} \cup \left\{\Phi^{-1}(p_{\mathfrak{E}}) \xrightarrow{\delta} \Phi^{-1}(q_{\mathfrak{E}})\right\} = \left\{\Phi^{-1}(q_1, 3) \xrightarrow{a_{(1,4)}} \Phi^{-1}(q_1, 4)\right\} = \left\{q_4 \xrightarrow{a_{(1,4)}} q_5\right\}$;

- $\Delta_{\mathfrak{D}_A} \leftarrow \emptyset$;
- $\Delta_{\mathfrak{D}_A} \leftarrow \Delta_{\mathfrak{D}_A} \cup \{\Delta^a\}$;

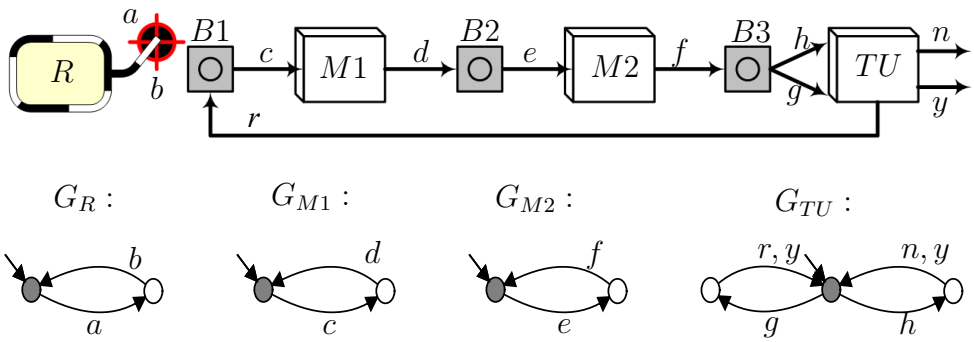
O Autômato $\mathfrak{D}_H\left[G_{\text{v}}\right] = \left(\Delta_{\mathfrak{D}_A}, Q_{\mathfrak{D}_A}, q^{\circ}_{\mathfrak{D}_A}, Q^{\omega}_{\mathfrak{D}_A}, \gamma_{\mathfrak{D}_A}\right)$ construído de acordo com a sistemática apresentada é composto por:

- $\Delta^a = \{a_{(1,1)}, a_{(1,2)}, a_{(1,3)}, a_{(1,4)}\}$
- $Q_{\mathfrak{D}_H} = \{q_1, q_2, q_3, q_4, q_5\}$;
- $Q^{\circ}_{\mathfrak{D}_H} = \{q_1\}$;
- $Q^{\omega}_{\mathfrak{D}_H} = \{q_1, q_2, q_3, q_4, q_5\}$;
- $\gamma_{\mathfrak{D}_H} = \left\{q_1 \xrightarrow{a_{(1,1)}} q_2, q_2 \xrightarrow{a_{(1,2)}} q_3, q_3 \xrightarrow{a_{(1,3)}} q_4, q_4 \xrightarrow{a_{(1,4)}} q_5\right\}$

4 EXEMPLO DE APLICAÇÃO

Para ilustrar a aplicabilidade dos métodos propostos, apresenta-se na Figura 12 o exemplo de um processo de manufatura com retrabalho de materiais, adaptado de Teixeira (2013). Neste exemplo considera-se a existência de um robô R que alimenta com materiais um *buffer* unitário $B1$, que por sua vez alimenta uma máquina $M1$. A saída de $M1$ vai para um *buffer* unitário $B2$ que alimenta a máquina $M2$. Tal máquina produz peças que são testadas pela estação de trabalho TU e em seguida podem ser aprovadas (y), descartadas (n) ou enviadas para retrabalho (r). $M3$ e TU são interconectados por um *buffer* unitário $B3$. Os modelos dos componentes do sistema $M1$, $M2$, R e TU são respectivamente G_{M1}, G_{M2}, G_R, e G_{TU}.

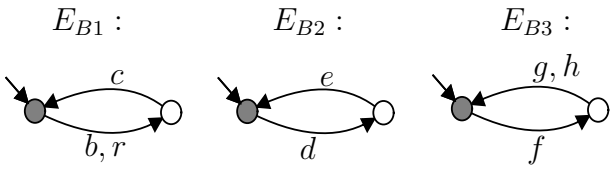
Figura 12 – Manufatura com retrabalho e modelagem das plantas



Fonte: Adaptado de Teixeira (2013)

Os modelos do robô e das máquinas possuem eventos de início (a, c, e) e fim (b, d, f) de funcionamento. O modelo da estação de trabalho, G_{TU} executa duas operações distintas, um teste geral g, que pode levar a aprovação y ou retrabalho r, evitando o descarte, e um teste final h, que pode levar a aprovação y ou descarte n, evitando o retrabalho. Supondo que o objetivo de controle seja evitar o *underflow* e *overflow* dos *buffers* $B1$, $B2$ e $B3$, as especificações são apresentadas na Figura 13.

Figura 13 – Modelagem de especificações para o problema de retrabalho



Os modelos de especificação são semanticamente equivalentes, ou seja, evitam a retirada de material do *buffer* caso ele esteja vazio, e a inserção caso ele já contenha alguma peça. Considera-se agora uma nova especificação E_N que deve evitar a rejeição de uma peça sem que não tenha sido retrabalhada ao menos uma vez e evitar que uma mesma peça seja retrabalhada mais de uma vez, isto é, toda

peça, caso defeituosa, deve percorrer um ciclo de retrabalho. Conforme a abordagem proposta por Teixeira (2013), pode ser mostrado que a modelagem de E_N, para um ciclo de retrabalho, envolve a construção de um autômato com 32 estado.s Para dois ciclos, a quantidade de estados aumenta para 560, enquanto o controle de três ciclos requer estruturar 15803 estados.

Ressalta-se, ainda, que essas considerações se baseiam em capacidade unitária de *buffers*, que se alteradas para uma valor maior que um, aumentariam substancialmente o número de estados destes modelos, bem como a complexidade para obtê-los. Isso motiva a utilização das abordagens, Distinguidores ou AFE, para modelagem deste problema, visto que sob a TCS clássica, tal modelagem se torna inviável

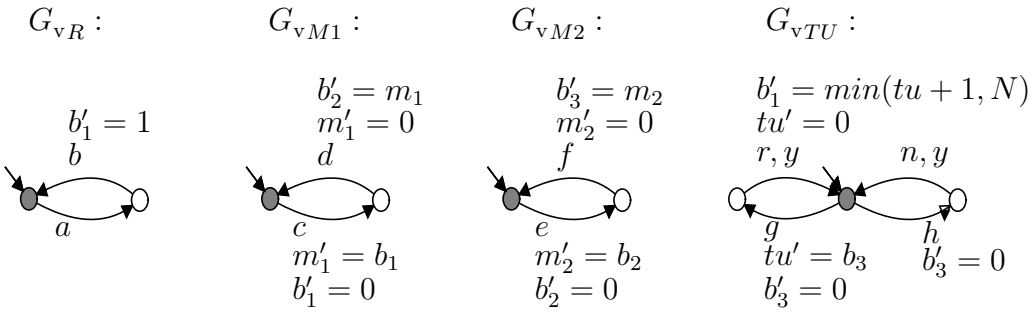
Considerando que, na abordagem de Distinguidores, para cada ciclo de retrabalho se faz necessária a construção manual dos refinamentos para os eventos c, d, e e f, e que essa tarefa varia conforme a quantidade de ciclos de retrabalhos a ser considerado no problema, então o processo de construção dos refinamentos e respectivos distinguidores se torna trabalhoso na prática. Nesse sentido, é intuitivo pensar que, sob o ponto de vista de construção, é praticamente mais vantajoso trabalhar com a abordagem com AFE.

Suponha, por exemplo, que o modelo exige que uma peça seja retrabalhada 100.000 vezes antes de ser descartada. Isso implicaria na criação de um refinamento c_0, tal que $\Pi(c_0) = c$, para identificar a peça sem nenhum retrabalho, e $c_1, \cdots, c_{100000}$, tal que $\Pi(c_1) = \cdots = \Pi(c_{100000}) = c$, para identificar a peça em cada um dos ciclos de retrabalho. O equivalente acontece com os eventos d, e e f que necessitariam de refinamentos. Após esse procedimento, toda a estrutura de distinção ainda dependeria do *designer* para ser construída o que, embora modular, seria uma tarefa extensa.

Abreviar essa etapa de construção é possível por meio dos AFE. Nesse caso a modelagem, considerando uma quantidade de ciclos N, pode ser realizada pela criação variáveis $b_1, b_2, b_3, m_1, m_2, tu$, com domínio $Dom(b_1) = Dom(b_2) = Dom(b_3) = Dom(m_1) = Dom(m_2) = Dom(tu) = \{0, 1, \cdots N\}$ e valor inicial $b_1^\circ = b_2^\circ = b_3^\circ = m_1^\circ = m_2^\circ = tu^\circ = 0$, cujos valores guardam em qual ciclo está a peça em $B1$, $B2$, $B3$, $M1$, $M2$ e TU, respectivamente. O valor 0 representa a ausência de peças em todos os modelos. A semântica de distinção de valores de variáveis, nesse caso, seria dadas por fórmulas lógicas que independem do domínio das variáveis. A Figura 14 apresenta como seria a modelagem por AFE do problema considerado nesse trabalho.

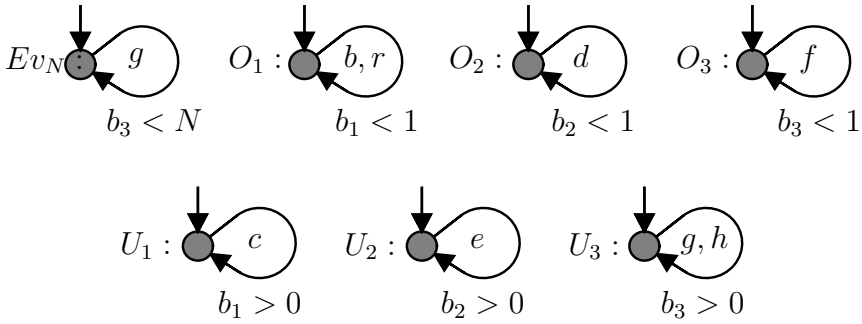
A partir das variáveis atualizadas na planta, constroem-se as especificações de

Figura 14 – Plantas do problema de retrabalho modelado por AFE



underflow, U_1, U_2 e U_3, e *overflow*, O_1, O_2 e O_3, para os *buffers* e uma especificação Ev_N, para controlar a quantidade de ciclos N, conforme mostra a Figura 15.

Figura 15 – Especificações do problema de retrabalho modelado por AFE



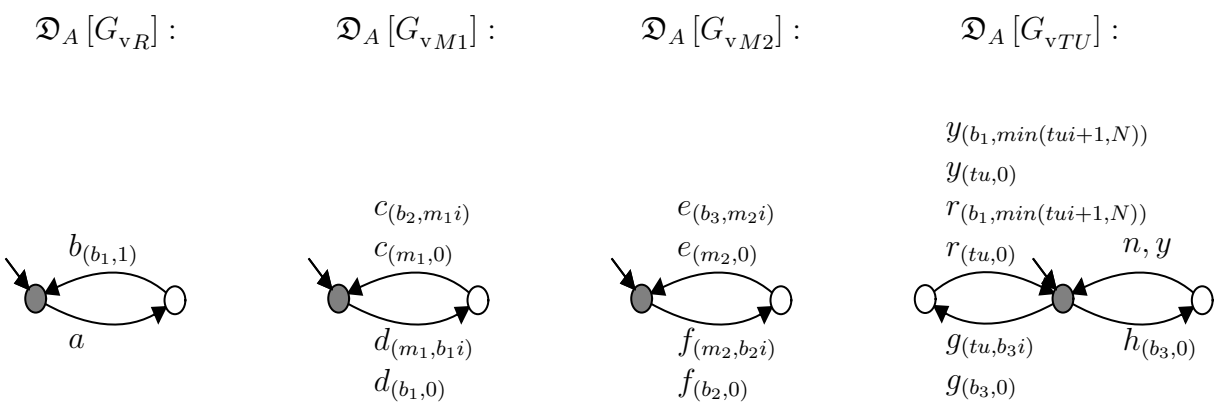
Em termos de esforço de modelagem, planta e especificações são representadas por AFE bastante simples. Efeito similar seria reproduzido caso a opção fosse por Distinguidores, conforme já evidencia a literatura (CURY *et al.*, 2015). Computacionalmente, uma constatação análoga pode ser construída a partir de evidências na literatura. Os trabalhos de Teixeira *et al.* (2015) e Cury *et al.* (2015), por exemplo, mostram que ambas as abordagens suportam a síntese do controlador com vantagens computacionais provenientes do uso de técnicas de abstrações. Ainda os trabalhos de Teixeira *et al.* (2013) e Malik e Teixeira (2016) mostram, respectivamente, que a síntese com distinguidores e AFE pode ser modularizada e que isso reduz substancialmente o esforço computacional inerente ao cálculo do controlador.

Entretanto, o atual estado da arte dá conta de que a modularização na abordagem com AFE pode ser mais limitada em comparação à síntese modular com Distinguidores. Embora AFE suportem um formato de modularização, proposto em Malik e Teixeira (2016), este formato não é diretamente compatível com o uso de abstrações. Assim, embora a síntese modular com AFE seja mais vantajosa em relação à monolítica, ela ainda é processada sobre todo o conjunto de variáveis. Diferentemente, a abordagem com distinguidores fornece resultados teóricos que permitem que uma abstração do distinguidor seja usada tanto na síntese monolítica quanto na modular.

Nesse sentido, é apresentada na sequência a conversão dos modelos correspondentes ao exemplo proposto, de AFE para Distinguidores, para serem então sintetizados na forma modular com abstrações. Posteriormente, serão apresentados os resultados de síntese também para a versão em AFE, para que os possíveis ganhos sejam quantificados.

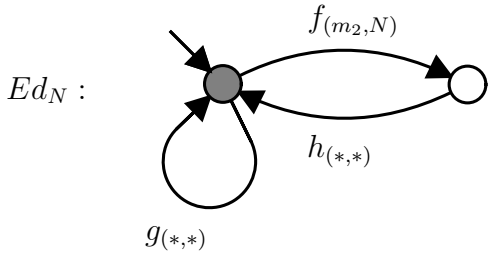
Por meio da conversão $\mathfrak{E} \rightarrow \mathfrak{D}$, são obtidos os modelos $\mathfrak{D}_A\left[G_{M1}\right]$, $\mathfrak{D}_A\left[G_{M2}\right]$, $\mathfrak{D}_A\left[G_R\right]$, e $\mathfrak{D}_A\left[G_{TU}\right]$ como descritos na Figura 16. Considera-se, nessa figura, que um refinamento qualquer $\delta \in \Delta^{\sigma}$ ao ser, por exemplo, representado como $b_{(b_1,m_1 i)}$ que existe para o evento b um refinamento para cada valor $i \in Dom(m_1)$.

Figura 16 – Resultado $\mathfrak{D}_A$ da conversão $\mathfrak{E} \rightarrow \mathfrak{D}$ do problema de retrabalho



Analogamente os modelos $\mathfrak{D}_H$podem ser obtidos para distinção da planta. Utilizando os modelos $\mathfrak{D}_A$e $\mathfrak{D}_H$, é possível construir as especificações sob a abordagem por Distinguidores. Dentre tais especificações, destaca-se a construção de Ed_N, que é a especificação responsável por controlar a quantidade de ciclos de retrabalho, apresentada na Figura 17. A notação $\sigma_{(*,*)}$ representa qualquer refinamento criado sobre o evento σ, isto é, $\sigma_{(*,*)} = \delta | \delta \in \Delta^{\sigma}$.

Figura 17 – Especificação para retrabalho sob abordagem por Distinguidores



As características do exemplo abordado permitem que ele seja ilustrativo para possíveis aplicações do método de conversão proposto. Os principais aspectos que sustentam esse argumento são sumarizados a seguir:

- Analisando o exemplo, identificou-se que uma dada especificação (E_N) é dificilmente modelada pela abordagem clássica. Considerou-se as abordagens que possuem refinamentos, Distinguidores e AFE, para a solução do problema;

- Observou-se que, por Distinguidores, o trabalho de modelagem da planta estaria relacionado à quantidade de ciclos N, e dependendo desse valor, se tornaria impraticável monoliticamente. Optou-se pela utilização de AFE;

- As plantas foram modeladas sob a abordagem de AFE (Figura 14);

- As plantas sob a abordagem por Distinguidores foram obtidas utilizando a conversão $\mathfrak{E} \rightarrow \mathfrak{D}$ (Figura 16), fruto da preparação dos modelos para o processo de síntese;

- A especificação de controle dos ciclos de retrabalho Ed_N foi concebida sob a abordagem por Distinguidores;

- A síntese foi processada.

Ou seja, inicialmente parte-se de uma dificuldade de modelagem de especificação e escolhe-se a alternativa que melhor acomode a tarefa de modelagem. No entanto, pode ocorrer que a melhor abordagem de modelagem não seja exatamente a mais indicada para executar um procedimento de síntese que reflita as extensões da TCS em sua amplitude. Nesse sentido, o método de conversão se torna essencial, por habilitar a melhor escolha para a síntese. No exemplo, com as especificações e plantas modeladas no ambiente refinado, o *designer* pode usufruir dos benefícios de ferramental, modularização e geração de código, o que é mais natural para a abordagem por Distinguidores do que por AFE. Nota-se que a abordagem por distinguidores remete simplesmente a uma instância da TCS, resolvida em um alfabeto particular e, portanto, as abordagens de síntese são nativamente válidas. Diferentemente, a abordagem por AFE não contempla uma associação direta e intuitiva com a TCS convencional, por envolver uma estrutura separada de variáveis e por tratar de modo bastante particular das atualizações dessas variáveis.

No sentido de exemplificar os benefícios da conversão proposta, apresenta-se a seguir um quadro comparativo que explora as fronteiras das abordagens de síntese da TCS. Muitas das abordagens não são apresentadas em detalhes neste trabalho, por questões de escopo, mas foram usadas como parâmetro para sustentar a obtenção do controlador para fins comparativos. Em particular, a síntese monolítica convencional é processada conforme o *framework* inicial de (RAMADGE; WONHAM, 1989); a síntese monolítica com distinguidores e com aproximações é processada conforme (CURY *et al.*,

2015); a síntese monolítica com variáveis e com abstrações é processada conforme (TEIXEIRA *et al.*, 2015); a síntese modular local é processada conforme (QUEIROZ; CURY, 2000); a síntese modular local com distinguidores é processada conforme (TEIXEIRA *et al.*, 2013); a síntese modular local com distinguidores e aproximações é processada conforme (TEIXEIRA *et al.*, 2016); e, finalmente, a síntese modular com variáveis não pode ser processada conforme (MALIK; TEIXEIRA, 2016), dado que a abordagem não dispõe até o momento de uma implementação automatizada, o que inviabiliza a síntese manual para problemas maiores. Não é de conhecimento dos autores uma abordagem que permita usufruir de abstrações no controle modular com variáveis. Aliás, a própria abordagem em (MALIK; TEIXEIRA, 2016) não acomoda naturalmente propriedades importantes do controlador, como o não-bloqueio, por exemplo.

No Quadro 2, são evidenciadas as abordagens de modelagem, monolíticas e modulares, utilizando a quantidade de estados em cada uma delas para o exemplo do retrabalho com um ciclo de retrabalho.

Quadro 2 – Comparação entre abordagens sobre o exemplo do retrabalho

	SCT	**SCT-D**	**SCT-V**
Abordagens Monolíticas			
Entrada	$K = G \| E$	$K_{\mathrm{d}} = E_{\mathrm{d}} \| (G_{\mathrm{d}} = G_a \| H_{\mathrm{D}})$	$K_v = E_{\mathrm{v}} \| (G_{\mathrm{v}} = Ga_v \| H_v)$
estados	7220	7220	7220
Supervisor	$\sup\mathcal{C}(K, G)$	$\sup\mathcal{C}(K_{\mathrm{d}}, G_{\mathrm{d}})$	$\sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})$
estados	212	212	212
Abstrações	-	$\sup\mathcal{C}(K_a, G_a) \| H_{\mathrm{D}}$	$\sup\mathcal{C}_V(E_{\mathrm{v}}, Ga_v) \| H_v$
estados	-	212	212
Abordagens modulares			
Supervisor	$_{i\in I} S_{loc}[i]$	$_{i\in I} S_{dloc}[i]$	-
estados	276	276	-
Combinações	-	$(\ _{i\in I'} S_{loc}[i]) \| \Pi(\ _{i\in I''} S_{dloc}[i])$	-
estados	-	276	-
Abstrações	-	$(\ _{i\in I'} S_{loc}[i]) \| \Pi(\ _{i\in I''} S_{dloc}[i]) \| \Pi(\ _{i\in I'''} S_{aloc}[i] \| H_{\mathrm{D}})$	-
estados	-	276	-

Nota-se que na síntese modular, o número de estados do controlador difere da solução monolítica. Isso ocorre porque esse processo é bloqueante localmente, em função do refluxo de trabalho. Alternativas para resolver o problema do bloqueio são propostas na literatura, como o uso de *coordenadores* (WONG; WONHAM, 1998). Como esse aspecto foge do escopo deste trabalho, optou-se por manter a solução bloqueante, mas preservando o mesmo número de estados em todas as abordagens

modulares testadas.

5 CONCLUSÃO

Neste trabalho foi proposto um método de conversão entre as abordagens de Distinguidores e de AFE, composto pelas sistemáticas $\mathfrak{D} \rightarrow \mathfrak{E}$ e $\mathfrak{E} \rightarrow \mathfrak{D}$. Tal conversão visa a utilização, por parte do *designer*, das vantagens de ambas as abordagens. Inicialmente esse *designer* pode modelar as plantas de um dado problema de SED sob a abordagem por AFE, e por meio do método de conversão proposto, obter o seu equivalente em comportamento sob a abordagem por Distinguidores, e vice-versa. Esse procedimento é motivado pela característica modular de Distinguidores e pela facilidade de modelagem por AFE, tanto de plantas quanto de especificações.

A conversão se deu pela ideia de associação de refinamentos. Os eventos refinados de um modelo sob a abordagem de Distinguidores foram associados aos valores do domínio de uma variável correspondente, na abordagem por AFE. Na conversão $\mathfrak{D} \rightarrow \mathfrak{E}$ utilizou-se os eventos refinados existentes para a criação, por meio de $\mathcal{C}_{\mathfrak{E}}$, de variáveis associadas a esses refinamentos, por meio de Θ. Tal conversão resultou em um autômato $\mathfrak{E}[G_{\mathrm{d}}]$. Na conversão $\mathfrak{E} \rightarrow \mathfrak{D}$, analisou-se a ocorrência da mudança de valor em uma dada variável para a criação de um evento refinado, por meio de $\mathcal{C}_{\mathfrak{D}}$. Essa conversão gera o autômato $\mathfrak{D}_A\left[G_{\mathrm{v}}\right]$ que representa o comportamento sem distinções e $\mathfrak{D}_H\left[G_{\mathrm{v}}\right]$ que representa o comportamento de distinção. Para este segundo, a geração (por meio de $\mathcal{C}_Q$) e associação de estados (por meio de Φ) com G_{v} se fizeram necessárias.

Almeja-se por trabalhos futuros a demonstração de possíveis equivalências entre as linguagens reconhecidas pelos autômatos. Para a conversão $\mathfrak{D} \rightarrow \mathfrak{E}$ que tem como entrada o modelo G_{d}, sugere-se a possível equivalência $\mathcal{L}(\Pi(G_{\mathrm{d}})) = \mathcal{L}(\mathfrak{E}[G_{\mathrm{d}}])$ e para a conversão $\mathfrak{E} \rightarrow \mathfrak{D}$ que tem como entrada o modelo G_{v}, pretende-se comprovar que $\mathcal{L}(\Pi(\mathfrak{D}_A\left[G_{\mathrm{v}}\right] || \mathfrak{D}_H\left[G_{\mathrm{v}}\right])) = \mathcal{L}(G_{\mathrm{v}})$. Considera-se ainda a implementação do método de conversão aqui proposto, para utilização em companhia de ferramentas de modelagem que já existem. Isso incentivaria o uso das sistemáticas e auxiliariam no teste das conversões.

Outra característica interessante que sugere-se para implantação em trabalhos futuros é a representação implícita de um AFE concebido explicitamente. Na conversão $\mathfrak{D} \rightarrow \mathfrak{E}$ o autômato resultante $\mathfrak{E}[G_{\mathrm{d}}]$ se apresenta em sua forma explícita. Seria interessante a concepção de um processo que transforma esse AFE explícito em um AFE implícito de equivalente linguagem. Analisando os exemplos comuns na literatura, tal processo pode ser generalizado, tendo em conta decrementos, incrementos e *resets*. Ao observar um AFE explícito que em uma dada posição da tupla tem seus va-

lores incrementando em transições conseguintes, sugere-se a incorporação de uma fórmula do tipo $x = x + 1$, tal que x representa a dada posição da tupla. Analogamente, ao se observar um decremento de valores em transições que se seguem, incorporaria-se uma fórmula $x = x - 1$. O *reset* acontece quando uma transição chega a um estado e uma determinada posição na tupla recebeu um valor que não corresponde a um incremento ou decremento, e pode ser de duas formas: um valor em uma entrada qualquer na tupla de partida da transição (nesse caso a fórmula seria $x = y$, tal que xe y são as variáveis correspondente às posições nas tuplas de chegada e partida, respectivamente) ou um valor qualquer m (a fórmula seria $x = m$).

Destaca-se ainda que, inicialmente, ao se constatar que a utilização de uma das abordagens de refinamento se fazia necessária, a opção por tal abordagem fadava o *designer* a utilizá-la até o momento de cálculo do controlador. Com o uso do método de conversão proposto, ambas as abordagens podem ser utilizadas em partes distintas da modelagem, e por consequência podem facilitar o seu processo. Espera-se que esta conversão contribua para a ampliação da aplicação da TCS na indústria.

REFERÊNCIAS

BOUZON, Gustavo; QUEIROZ, Max H de; CURY, José ER. Supervisory control of des with distinguishing sensors. In: IEEE. **Discrete Event Systems, 2008. WODES 2008. 9th International Workshop on**. [S.l.], 2008. p. 22–27.

BOUZON, Gustavo; QUEIROZ, Max H de; CURY, José ER. Exploiting distinguishing sensors in supervisory control of des. In: **IEEE International Conference on Control and Automation, ICCA**. [S.l.: s.n.], 2009. v. 9, p. 442–447.

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

CHEN, Yi-Liang; LIN, Feng. Modeling of discrete event systems using finite state machines with parameters. In: IEEE. **Control Applications, 2000. Proceedings of the 2000 IEEE International Conference on**. [S.l.], 2000. p. 941–946.

CHEN, Yi-Liang; LIN, Feng. 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.

CHENG, Kwang Ting; KRISHNAKUMAR, Avinash S. Automatic functional test generation using the extended finite state machine model. In: ACM. **Proceedings of the 30th international Design Automation Conference**. [S.l.], 1993. p. 86–91.

CURY, José ER; QUEIROZ, Max Hering de; BOUZON, Gustavo; TEIXEIRA, Marcelo. Supervisory control of discrete event systems with distinguishers. **Automatica**, Elsevier, v. 56, p. 93–104, 2015.

FEI, Zhennan; MIREMADI, Sajed; ÅKESSON, Knut; LENNARTSON, Bengt. A symbolic approach to large-scale discrete event systems modeled as finite automata with variables. In: IEEE. **2012 IEEE International Conference on Automation Science and Engineering (CASE)**. [S.l.], 2012. p. 502–507.

FISCHER, G.; LEAL, A. Investigação do uso de distinguidores na síntese de supervisores em função de mudanças na planta. In: **Congresso Brasileiro de Automática**. [S.l.: s.n.], 2014.

HOPCROFT, John E.; MOTWANI, Rajeev; ULLMAN, Jeffrey D. **Introduction to Automata Theory, Languages, and Computation**. 2. ed. United States of America: Pearson Education, 1939.

LEWIS, Harry R; PAPADIMITRIOU, Christos H. **Elements of the Theory of Computation**. [S.l.]: Prentice Hall PTR, 1997.

MALIK, Robi; TEIXEIRA, Marcelo. 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.

OUEDRAOGO, Lucien; KUMAR, Ratnesh; MALIK, Robi; AKESSON, Knut. Nonblocking and safe control of discrete-event systems modeled as extended finite automata. **IEEE Transactions on Automation Science and Engineering**, IEEE, v. 8, n. 3, p. 560–569, 2011.

QUEIROZ, Max H De; CURY, José ER. Modular control of composed systems. In: IEEE. **American Control Conference, 2000. Proceedings of the 2000**. [S.l.], 2000. v. 6, p. 4051–4055.

RAMADGE, Peter J; WONHAM, W Murray. Supervisory control of a class of discrete event processes. **SIAM journal on control and optimization**, SIAM, v. 25, n. 1, p. 206–230, 1987.

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

SKOLDSTAM, Markus; AKESSON, Knut; FABIAN, Martin. Modeling of discrete event systems using finite automata with variables. In: IEEE. **Decision and Control, 2007 46th IEEE Conference on**. [S.l.], 2007. p. 3387–3392.

TEIXEIRA, Marcelo. 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.

TEIXEIRA, Marcelo; CURY, José ER; QUEIROZ, Max H de. Local modular control with distinguishers applied to a manufacturing system. **IFAC Proceedings Volumes**, Elsevier, v. 46, n. 9, p. 263–268, 2013.

TEIXEIRA, Marcelo; CURY, José ER; QUEIROZ, Max H de. Exploiting distinguishers in local modular control of discrete event systems. **IEEE Transactions on Automation Science and Engineering**, IEEE, p. 825, 2016.

TEIXEIRA, Marcelo; MALIK, Robi; CURY, José ER; QUEIROZ, Max H de. 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.

THISTLE, John G. Synthesis of supervisory controls for discrete event systems. In: IEEE. **Application of Concurrency to System Design, 2004. ACSD 2004. Proceedings. Fourth International Conference on**. [S.l.], 2004. p. 151.

WONG, K C; WONHAM, W M. Modular control and coordination of discrete event systems. **Discrete Event Dynamic Systems**, v. 8, n. 3, p. 241–273, 1998.