INSTITUTO MILITAR DE ENGENHARIA

1º TEN RAQUEL STELLA DA SILVA DE AGUIAR

BUSCA HEURÍSTICA NA OBTENÇÃO DE SUPERVISOR POR DISTINGUIDORES APROXIMADOS: ESTUDO DE CASO NA COORDENAÇÃO DE SISTEMAS MULTIVEICULARES

Dissertação de Mestrado apresentada ao Curso de Mestrado em Engenharia Elétrica do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Ciências em Engenharia Elétrica.

Orientador: Maj Antonio Eduardo Carrilho da Cunha - Dr.Eng.
Orientador: Prof. José Eduardo Ribeiro Cury - Dr.D'Etat

Rio de Janeiro
2013

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80-Praia Vermelha
Rio de Janeiro-RJ CEP 22290-270

621.3 Aguiar, Raquel Stella da Silva de
A282b Busca heurística na obtenção de supervisor por distinguidores aproximados: estudo de caso na coordenação de sistemas multiveiculares/Raquel Stella da Silva de Aguiar; orientada por Antonio Eduardo Carrilho da Cunha. - Rio de Janeiro: Instituto Militar de Engenharia, 2013.
número de paginas 126p.

Dissertação (mestrado) - Instituto Militar de Engenharia- Rio de Janeiro, 2013

1. Engenharia Elétrica 2. Teoria de Controle Supervisório com Distinguidores. 3. Busca Heurística. I. Cunha, Antonio Eduardo Carrilho da II. Título. III. Instituto Militar de Engenharia.

CDD 621.3

INSTITUTO MILITAR DE ENGENHARIA

1º TEN RAQUEL STELLA DA SILVA DE AGUIAR

BUSCA HEURÍSTICA NA OBTENÇÃO DE SUPERVISOR POR DISTINGUIDORES APROXIMADOS: ESTUDO DE CASO NA COORDENAÇÃO DE SISTEMAS MULTIVEICULARES

Dissertação de Mestrado apresentada ao Curso de Mestrado em Engenharia Elétrica do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Ciências em Engenharia Elétrica.

Orientador: Maj Antonio Eduardo Carrilho da Cunha - Dr.Eng.

Orientador: Prof. José Eduardo Ribeiro Cury - Dr.D'Etat

Aprovada em 25 de fevereiro de 2013 pela seguinte Banca Examinadora:

Maj Antonio Eduardo Carrilho da Cunha - Dr.Eng. do IME - Presidente

---

Prof. José Eduardo Ribeiro Cury - Dr.D'Etat da UFSC

Prof. Marcos Vicente de Brito Moreira - D.Sc. da UFRJ

Prof. Max Hering de Queiroz - Dr. Eng. da UFSC

Rio de Janeiro
2013

AGRADECIMENTOS

À minha família, minha fonte inesgotável de motivação, alento e amor.

Ao professor e orientador Antonio Eduardo Carrilho da Cunha, que com as doses certas de cobrança e confiança me muniu da segurança, do equilíbrio e da concentração necessários para produzir o trabalho e a dissertação. Agradeço pelas palavras que ora exortavam ora acalmavam, e pela empolgação com o trabalho, que me encorajou e motivou. Agradeço também pelo exemplo profissional e pessoal, tão importantes na formação do mestre.

Ao professor e orientador José Eduardo Ribeiro Cury, por me acolher em seu programa, por aceitar ser meu orientador, pela solicitude, pela confiança em mim e no trabalho e pela generosidade de indicar um caminho nos momentos de dúvida.

Ao Professor Max Queiroz, por dedicar tempo e paciência para dirimir muitas dúvidas, por compartilhar seu profundo conhecimento do tema e por contribuir com tantas idéias e orientações.

Aos meus professores do curso de pós-graduação em engenharia elétrica do Instituto Militar de Engenharia, cujas aulas, palavras e exemplos instigam a atividade científica.

À amiga Anne Elise, que sempre esteve à disposição para ajudar e assim o fez de tantas formas diversas.

Ao amigo Coelho, que sempre me socorreu no andar dos trabalhos.

Aos amigos Fernando e Nathália, cujos empenhos me possibilitaram estar dentro do prazo para realizar o curso.

Aos companheiros de pós-graduação Ângelo, Antônio, Gabriela, Guilherme, Jackler, Kirk e Nina por apoiarem e dividirem a jornada.

À CAPES, pelo apoio financeiro prestado para realização de estágio e participação em congresso.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SÍMBOLOS

ABREVIATURAS

$\mathbf{E}$	-	Linguagem da especificação
$\mathbf{G}$	-	Autômato de planta
$\mathbf{G_{aj}}$	-	Autômato de planta com aproximação qualquer
$\mathbf{G_d}$	-	Autômato de plantas distinguidas
$\mathbf{H}$	-	Autômato de distinguidor preditivo
$\mathbf{H_j}$	-	Autômato de distinguidor não preditivo
$\mathbf{K}$	-	Linguagem-alvo
$\mathbf{K_a}$	-	Linguagem-alvo de planta refinada
$\mathbf{K_{aj}}$	-	Linguagem-alvo de planta com aproximação qualquer
$\mathbf{K_d}$	-	Linguagem-alvo de planta distinguida
$\mathbf{L_D}$	-	Linguagem do distinguidor preditivo
$\mathbf{R}$	-	Autômato, $L_m(R) = K$
$\mathbf{Z}$	-	Autômato para supervisores
$\mathbf{Z_a}$	-	Autômato para supervisores obtidos de planta refinada
$\mathbf{Z_{aj}}$	-	Autômato para supervisores obtidos de planta com aproximação qualquer
$\mathbf{Z_d}$	-	Autômato para supervisores exatos

SÍMBOLOS

$\|$	-	produto síncrono
δ	-	função de transição de um autômato
$\mathbf{\Delta}$	-	conjunto de eventos refinados de um SED
$\mathbf{\Delta^\sigma}$	-	conjunto de eventos refinados de uma máscara σ
$\mathbf{\Sigma}$	-	conjunto de eventos de um SED
$\mathbf{\Sigma^*}$	-	conjunto de todas as cadeias fomadas por eventos em Σ
$\mathbf{\Sigma_u}$	-	subconjunto formado pelos eventos não controláveis de Σ
$\mathbf{\Sigma_c}$	-	subconjunto formado pelos eventos controláveis de Σ
$\mathbf{\Pi}$	-	mapa mascarador

$\mathbf{\Pi^{-1}}$	-	mapeamento inverso
$\mu(L)$	-	medida da linguagem L
$\|\mu\|(L)$	-	medida de variação total da linguagem L
$\chi(q)$	-	peso atribuído ao estado q pela função característica χ
$\widetilde{\Psi}$	-	matriz custo de evento
$\widetilde{\Psi_G}$	-	matriz custo de evento do autômato G
Ψ	-	matriz custo de transição de estado
Ψ_G	-	matriz custo de transição de estado para o autômato G

LISTA DE SIGLAS

AGV	(*Automated Guided Veichle*) Veículo Auto-Guiado
PBCT	Plano Básico de Ciência e Tecnologia
PCS	Problema do Controle Supervisório
PCS-D	Problema do Controle Supervisório Distinguido
SED	Sistemas a Eventos Discretos
SMR	Sistema Multi-Robôs
TCS	Teoria do Controle Supervisório
TCS-D	Teoria do Controle Supervisório Distinguido
TCS-A	Teoria do Controle Supervisório Aproximado
VANT	Veículo Aéreo Não Tripulado

LISTA DE ALGORITMOS

RESUMO

O emprego da TCS para síntese de supervisores para SEDs é limitado nos casos em que o SED é constituído por muitos subsistemas integrados e há explosão combinatória de estados. O uso de distinguidores foi introduzido na TCS visando a simplificar a modelagem das especificações, com a garantia de síntese do supervisor maximamente permissivo. Ao aproximar-se a linguagem do distinguidor, obtém-se um supervisor aproximado, cuja síntese é de menor complexidade, porém não há garantias de que este seja maximamente permissivo. O objetivo deste trabalho é propor um método para se obter um supervisor, no contexto das aproximações na TCS-D, o menos restritivo possível nos casos em que obter o supervisor maximamente permissivo pelo distinguidor preditivo ou inspecionando cada aproximação é proibitivo. O método consiste na busca heurística no espaço formado pelos supervisores aproximados por distinguidores pelo supervisor com a maior medida de linguagem. Algoritmos de busca baseados em Busca Tabu e Algoritmo Genético foram implementados e validados mediante testes com sistemas criados para este trabalho. Por fim um sistema de AGVs foi submetido aos algoritmos de busca para obtenção de um supervisor. Os resultados mostram que o método é eficaz na exploração do espaço de supervisores aproximados e pode ser utilizado como alternativa para obter-se o melhor supervisor possível de ser sintetizado para sistemas muito complexos.

ABSTRACT

The application of SCT to synthesize supervisors for DES is restricted in cases that the DES comprises many integrated subsystems and there is combinatorial explosion of states. The use of distinguishers was introduced for SCT in order to simplify specification modeling task while guaranteeing the synthesis of the maximally permissive supervisor. By approximatting the language of the distinguisher, a approximated supervisor is obtained, providing computational savings in the synthesis, although there is no guarantee that this is the maximally permissive supervisor for the system. The main purpose of this work is to propose a procedure to obtain the least restrictive achievable supervisor, in the context of approximations in the SCT-D , in cases which to obtain the maximally permissive supervisor by predictive distinguisher or by inspecting each possible aproximattion is prohibitive. The procedure consists of heuristic search for the supervisor with the highest language measure in the space formed by supervisors approximated by distinguishers. Search algortithms based in Tabu Search and Genetic Algorithms were implemented and validated by tests which were performed on systems that were conceived for this end. At last, the search algorithms were applied to a AGV system in order to obtain a supervisor. The results confirm the effectiveness of the procedure in explore the space formed by approximated supervisors and prove that the procedure may be an alternative to obtain the best achievable supervisor for very complex systems.

1 INTRODUÇÃO

1.1 CONTEXTO E MOTIVAÇÃO

Desde os primórdios da humanidade o homem desenvolve ferramentas visando a facilitar a execução de suas tarefas. Hoje, o nível de sofisticação atinge patamares tão altos que *ferramenta* já não parece a palavra mais adequada para denominar as complexas criações do homem. Dentre essas criações, uma das mais notáveis é a automação, solução encontrada para que os diversos sistemas desenvolvidos não dependessem da constante atuação humana para funcionarem em níveis aceitáveis.

Com o avanço tecnólogico, esses sistemas atingiram patamares elevados de sofisticação, sendo integrados por diversas e inúmeras estruturas que interagem entre si de forma tão complexa que divisar o comportamento global se torna uma tarefa árdua para um projetista. Não raro sacrifica-se a eficiência em prol da construção de um modelo compreensível e computável, fato que afeta drasticamente o desempenho dos sistemas, convergindo para prejuízos financeiros. Além disso, essa simplificação pode também impedir que sejam assegurados requisitos de confiabilidade e segurança da operação do sistema, podendo até mesmo ocasionar a perda de vidas.

Como sistema dinâmico, os sistemas fabris, de comunicação, de tráfego, entre outros, podem ser modelados como Sistemas a Eventos Discretos (SED) (CASSANDRAS e LAFORTUNE, 2008). Os SEDs possuem um comportamento dinâmico com evolução dirigida a eventos que ocorrem de forma discreta e não uniforme no tempo, em contraste com os sistemas contínuos, cujo espaço de estados varia de forma contínua e dependente do tempo.

Os Sistemas a Eventos Discretos geralmente consistem na interação entre diversos subsistemas com comportamentos autônomos que carecem de regras para um funcionamento conjunto adequado. O comportamento desejado deste sistemas deve ser então garantido pela ação de um agente externo de controle.

A Teoria de Controle Supervisório (TCS) (RAMADGE e WONHAM, 1989) provê métodos para se obter um agente de controle para SEDs, expresso na figura do supervisor. O supervisor atua observando a interação e o funcionamento dos vários subsistemas e decidindo, a cada evento ocorrido, quais os próximos passos possíveis. Ele pode determinar que alguns eventos nunca ocorram e levar o sistema a um comportamento

desejado. Os métodos propostos pela TCS garantem ainda que o supervisor sintetizado seja maximamente permissivo, isto é, permite que o SED se comporte da maneira desejada restringindo seus eventos o minímo necessário.

A Teoria de Controle Supervisório é fundamentada na Teoria de Linguagens Formais e Autômatos (CASSANDRAS e LAFORTUNE, 2008) e o primeiro passo para sua aplicação é modelar o comportamento do sistema em malha aberta, traduzido na figura da planta. A síntese dos supervisores exige ainda que sejam modeladas especificações. Especificações nada mais são do que requisitos que o sistema deve cumprir e que restringem seu funcionamento. Um simples exemplo é determinar que uma máquina só inicie operação após a chegada de suprimentos. O supervisor é obtido então a partir do comportamento da planta conjugado a essas especificações, por um algoritmo com complexidade polinomial no número de estados da planta e da especificação.

O número de estados da planta, porém, em geral cresce exponencialmente com o número de subsistemas devido à explosão combinatória de estados que ocorre ao se compor os modelos obtidos individualmente para cada subsistema (CASSANDRAS e LAFORTUNE, 2008). O mesmo é válido para a especificação. Para contornar a explosão combinatória de estados da planta e da especificação foi introduzido à TCS o controle modular (WONHAM e RAMADGE, 1988; QUEIROZ e CURY, 2000) e o controle modular local (QUEIROZ e CURY, 2002a,b).

Contudo, em sistemas complexos com grande número de elementos interconectados, a escrita das especificações pode se tornar uma atividade de difícil realização. Com o objetivo de facilitar essa etapa da TCS, foi introduzido o conceito de distinguidores e a TCS foi estendida para a Teoria de Controle Supervisório com Distinguidores (TCS-D) (BOUZON et al., 2008a,b, 2009).

Os distinguidores são modelos que, inseridos no sistema, permitem o refinamento de eventos, isto é, ocorre o mapeamento de cada evento original em um novo conjunto de eventos que caraterizam suas instâncias. Eles são incorporados às plantas do sistema, o que se traduz no refinamento, ou detalhamento, desta.

Os distinguidores podem ser implementados, por exemplo, pela inserção de um sensor no sistema. Em um sistema que recebe caixas de diferentes tamanhos, o evento *chegada-de-caixa* poderia ser refinado em *chegada-de-caixa-pequena* e *chegada-de-caixa-grande*. Neste caso o distinguidor seria implementado por um sensor que mede a altura das caixas na entrada do sistema.

Com o refinamento de eventos, a modelagem de especificações pode se tornar uma

tarefa mais simples. No entanto, para distinguidores que identificam uma única instância para cada ocorrência de um evento, o custo computacional permanece o mesmo que para a TCS. Esta aparente incoerência deriva do fato de que a complexidade do sistema, antes resultado das intricadas especificações, agora provém da planta refinada do sistema.

Os mesmos autores introduziram também outro conceito que busca aliar a facilidade de escrita de especificações com a redução do custo computacional: os supervisores aproximados. Nessa abordagem ainda são utilizados distinguidores para refinamento dos eventos, porém somente algumas instâncias de alguns eventos são distinguidas entre si. O custo computacional diminui, mas a permissividade do supervisor é penalizada e não se pode garantir neste tipo de síntese que o supervisor obtido será o menos restritivo possível ou que constituirá uma solução para o problema de controle.

No estado da arte, não é possível determinar previamente quais distinguidores aproximados conduzem ao supervisor maximamente permissivo e a opção existente demanda que se sintetize cada supervisor aproximado até que se encontre a solução menos restritiva dentre as possíveis. Esses problemas de busca são conhecidos como otimização combinatória e exigem que cada solução do espaço de busca seja testada, método conhecido como busca exaustiva ou busca por força bruta (PARDALOS et al., 2011), demandando tempo e recursos computacionais.

Devido a isto faz-se necessário o desenvolvimento de um método que possibilite a exploração do espaço formado pelos supervisores aproximados na busca por soluções maximamente permissivas de forma mais eficiente. Deve-se ainda garantir que, no caso de tais supervisores não serem possíveis de ser sintetizados devido à complexidade computacional envolvida, seja retornada a solução mais permissiva possível. Propõe-se então uma busca heurística que, estabelecidos os critérios apropriados, buscará dentro do espaço de supervisores aproximados possíveis aquele que melhor atenda ao critério desejado.

1.2 OBJETIVO

Buscando resolver a questão levantada, este trabalho propõe aliar a redução de complexidade computacional oferecida pelo uso de aproximação por distinguidores à exploração do espaço de soluções com busca heurística por soluções quasi-ótimas[1].

O objetivo deste trabalho é, portanto, propor um método para obtenção de supervi-

[1]O termo quasi-ótimo é empregado neste texto para qualificar as soluções retornadas como ótimos pela busca heurística, mas que não são iguais à solução maximamente permissiva.

sores quasi-ótimos para sistemas em que a explosão combinatória de estados inviabiliza a síntese de um supervisor maximamente permissivo conforme os preceitos da TCS. Desta forma garante-se que o supervisor sintetizado é a melhor solução possível dada a restrição de recursos computacionais.

1.3 METODOLOGIA

A busca heurística (BROWNLEE, 2011) é sugerida como meio alternativo de explorar o espaço de supervisores aproximados. Empregando métodos heurísticos, que incluem a determinação de parâmetros como o número máximo de soluções a examinar e o limite de complexidade da síntese, a busca garante a convergência para um solução quasi-ótima em menos tempo e examinando menos opções do que a busca exaustiva. A busca heurística, no entanto, demanda que haja uma relação de ordem entre os elementos do espaço de busca de tal forma que o espaço seja totalmente ordenado, ou seja, que as soluções sejam comparáveis entre si.

A literatura foi pesquisada (CLAVIJO, 2012; KEPHART, 2005; RAY e PHOHA, 2003; WANG e RAY, 2004) para se encontrar um meio de se ordenar ou comparar linguagens. Como resultado da pesquisa vislumbrou-se empregar a medida de linguagem proposta em WANG e RAY (2002) como meio de obter o ordenamento total do espaço de busca.

Os algoritmos de busca heurística desenvolvidos foram validados mediante testes com sistemas de díficil, porém possível, solução de controle. Os supervisores exatos para cada sistema foram sintetizados para comparação com os supervisores retornados como resultados dos testes, permitindo avaliar os algoritmos e ajustar parâmetros necessários. Finalizada a validação e estabelecidos os critérios de busca e otimidade, um sistema multiveicular de grande porte foi submetido aos algoritmos.

1.4 CONTRIBUIÇÕES

Uma vez comprovado, o novo método proposto pode ser aplicado a outros sistemas de controle em que o grande número de elementos coíbe a síntese de um supervisor ótimo. Ademais, as seguintes contribuições emergem deste trabalho como fruto de investigações realizadas para atingir o objetivo principal:

- Decomposição de um distinguidor em distinguidores modulares por instância, o que implica em menor complexidade da síntese e maior variedade de soluções

aproximadas;

- Determinação de um operador que aplicado ao espaço de supervisores aproximados produz um espaço totalmente ordenado e adaptação de medida de linguagens à Teoria de Controle Supervisório Distinguido;
- Concepção de sistemas, cuja modelagem e síntese se beneficiam do uso de distinguidores, para validação do método proposto; e
- Implementação em ferramenta computacional dos algoritmos de busca e de outros algoritmos necessários para conclusão do trabalho.

1.5 ORGANIZAÇÃO DO DOCUMENTO

No capítulo 2, são abordados os Sistemas a Eventos Discretos e as formas de representá-los. O conceito de linguagem é introduzido, bem como as operações sobre autômatos e linguagem. O capítulo também aborda a Teoria de Controle Supervisório, introduzindo o problema de controle de um SED, juntamente com a noção de controlabilidade de eventos. É explicado como uma planta, modelo do sistema, pode ser controlada pela ação de um supervisor e os passos necessários para que seja sintetizado um supervisor que restrinja a planta o mínimo possível.

Os distinguidores são o tema do capítulo 3, que busca demonstrar como a utilização de distinguidores pode facilitar o trabalho de modelagem das especificações do sistema. A definição e modelagem de distinguidores são expostas, bem como a introdução dos mesmos na síntese de um supervisor exato e aproximado.

Como solução para explorar o espaço formado pelas soluções aproximadas, o capítulo 4 apresenta a busca heurística como forma alternativa de tratar problemas com espaços de busca muito grandes ou cujas soluções exatas não podem ser calculadas. Dois métodos em particular são apresentados, o Algoritmo Genético e a Busca Tabu. Em seguida, um método para obter medidas e distâncias entre linguagens é discutido e um exemplo prático serve como ilustração. Também neste capítulo são expostos outros estudos realizados neste trabalho para a área de Controle Supervisório Distinguido, tais como o distinguidor modular por instância e a conjectura de uma nova síntese de supervisor.

A seguir, no capítulo 5, são detalhados os sistemas empregados para validar o algoritmo de busca. Para cada sistema são evidenciadas as plantas, especificações e os resultados para as sínteses distinguida e aproximada obtidos um a um mediante uso de

ferramentas consolidadas na área de controle supervisório. Em seguida são apresentados os resultados obtidos para a busca por supervisores para cada um dos sistemas e é feita uma apreciação dos resultados.

O capítulo 6 mostra a aplicação do algoritmo na resolução de um sistema composto por múltiplos veículos autônomos. Uma introdução a sistemas multiveiculares é feita e são descritos os desafios na concepção, modelagem e controle de tais sistemas, muitos dos quais encontram amplo emprego na área de defesa.

A dissertação encerra com as conclusões, revisões dos resultados, sugestões de aspectos a serem investigados futuramente e restrições do trabalho desenvolvido, todos assunto do capítulo 7. O capítulo também contém as contribuições da pesquisa às áreas do conhecimento.

2 CONCEITOS PRELIMINARES

Para melhor entendimento dos resultados apresentados neste trabalho, apresentam-se neste capítulo alguns conceitos introdutórios sobre Sistema de Eventos Discretos e Teoria de Controle Supervisório. No transcorrer do capítulo, um pequeno exemplo é reproduzido para exemplificar a aplicação dos conceitos.

2.1 SISTEMA A EVENTOS DISCRETOS

Os Sistemas a Eventos Discretos (SED) são sistemas dinâmicos cuja evolução é dirigida por eventos de ocorrência não determinista e discreta no tempo. Isto significa que não é possível prever quando um evento irá ocorrer e, a cada ocorrência, o estado do sistema como um todo é alterado instantaneamente.

Um conjunto abrangente de sistemas pode ser classificado como SED. Exemplos mais facilmente identificáveis são os sistemas de manufatura, os sistemas de controle de tráfego e os sistemas de comunicação (CASSANDRAS e LAFORTUNE, 2008). Os SEDs se opõem aos sistemas contínuos no sentido de que estes últimos possuem um espaço de estados que evolui de forma contínua e dependente do tempo. Alguns sistemas podem ter comportamento híbrido, em que eventos são disparados quando condições sobre as variáveis contínuas são atendidas.

Há várias propostas para a modelagem de SEDs, como Redes de Petri, Teoria das Filas, Álgebra max-plus, Cadeias de Markov e Linguagens Formais e Autômatos, cada qual com sua representação específica (CASSANDRAS e LAFORTUNE, 2008).

2.1.1 LINGUAGENS FORMAIS

Conforme dito anteriormente, os SEDs são sistemas dinâmicos dirigidos a eventos cuja ocorrência é discreta e não uniforme no tempo. O conjunto formado pelos eventos de um SED é chamado alfabeto e é usualmente denotado por Σ. Conforme o sistema evolui, são geradas sequências de eventos que podem ser representadas pela concatenação ou justaposição dos eventos contidos no alfabeto Σ. Essa concatenção forma sequências de eventos denominadas *cadeias*[2] e essa idéia motiva a representação dos SEDs por Linguagens Formais.

[2]Alternativamente utilizam-se também os termos *palavra* e *sequência de eventos*

A figura 2.1 mostra uma máquina M1 que manufatura uma peça. Ao se modelar essa máquina como um SED, pode-se atribuir ao início de operação da máquina M1 o evento x e ao término de operação o evento y. O alfabeto deste SED é então o conjunto $\Sigma = \{x, y\}$.

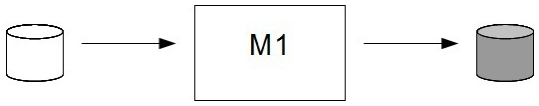


FIG. 2.1: Exemplo de SED.

As palavras podem ainda ser concatenadas gerando novas palavras. A concatenação é realizada ao se justapor uma cadeia ao fim de outra. Sejam por exemplo s, t, u e v palavras formadas por eventos em Σ e s formada pela concatenação de t, u e v, $s{=}tuv$. Nesse caso t é dita ser *prefixo* de s, v é dita ser *sufixo* de s e u é uma *subcadeia* de s. Quando uma cadeia t é prefixo de outra cadeia s, escreve-se $t \leq s$.

O comprimento de uma palavra s é representado por $|s|$ e expressa o número de eventos que formam a palavra s. A cadeia de comprimento nulo é chamada cadeia vazia, representada pelo símbolo ε, isto é, $|\varepsilon| = 0$. A cadeia vazia é o elemento neutro da concatenação, ou seja, $s = s\varepsilon = \varepsilon s$. Essas igualdades implicam que, para uma cadeia s, ε e a própria cadeia s são sufixos, prefixos e subcadeias de s.

O conjunto formado por todas as palavras de comprimento finito mais a palavra vazia é denotado por Σ^*. Uma linguagem L em Σ é qualquer subconjunto de Σ^*, isto é, $L \subseteq \Sigma^*$. Segundo esta definição, os conjuntos $\emptyset$, Σ^* e Σ também constituem linguagens em Σ.

Devido ao fato de que a linguagem é um conjunto de palavras, as operações definidas sobre conjuntos também podem ser aplicadas a linguagens, tais como a interseção $L \cup K$, a união $L \cap K$ e a diferença $L - K$, para duas linguagens $L, K \subseteq \Sigma^*$ (CASSANDRAS e LAFORTUNE, 2008). Outras operações também são definidas sobre linguagens (CASSANDRAS e LAFORTUNE, 2008):

- complemento: $L^C = \Sigma^* - L$;
- concatenação: $LK = \{uv : \ u \in L, \ v \in K\}$;
- potência: $K^0 = \varepsilon$ e $K^n = K^{n-1}K$; fechamento Kleene: $K^* = \bigcup_{n=0}^{\infty} K^n$.
- prefixo-fechamento: $\overline{K} = \{u \mid uv \in K \text{ para algum } v \in \Sigma^*\}$.

O prefixo-fechamento de uma linguagem L, $\overline{L}$, contém L, todos os sufixos de todas as palavras contidas em L e a cadeia vazia. Se $L = \overline{L}$, L é dita ser prefixo-fechada.

Duas linguagens $L_1, L_2 \subseteq \Sigma^*$ são ditas não conflitantes quando todos os prefixos contidos na interseção entre os prefixo-fechamentos de cada linguagem são prefixos de pelo menos uma cadeia contida na interseção entre as linguagens. Em notação de conjunto:

$$\overline{L_1} \cap \overline{L_2} = \overline{L_1 \cap L_2}. \tag{2.1}$$

2.1.2 AUTÔMATOS

Um autômato finito determinista é representado pela quíntupla $(Q, \Sigma, \delta, q_i, Q_m)$ em que Σ é o conjunto de eventos que ocorrem no automâto, Q é conjunto de estados, Q_m é o conjunto de estados marcados, q_i é o estado inicial e $\delta : Q \times \Sigma \to Q$ é a função de transição entre os estados. A função de transição δ é parcial pois só é definida para alguns elementos do domínio $Q \times \Sigma$, uma vez que não necessariamente todos os eventos podem ocorrer em um determinado estado Q. Quando a função de transição δ é definida para um estado q e um evento σ, indica-se tal fato pela notação $\delta(q, \sigma)!$. Para um autômato determinista ainda há a restrição de que para cada elemento do domínio $Q \times \Sigma$ a função δ associa obrigatoriamente somente um elemento no contra-domínio Q.

A função δ pode ser extendida para cadeias em Σ^* de forma que $\delta : Q \times \Sigma^* \to Q$ (CASSANDRAS e LAFORTUNE, 2008). A extensão é feita de forma recursiva tal que:

1)$\delta(q, \varepsilon) = q$; e

2)$\delta(q, sa) = \delta(\delta(q, s), a)$ para $s \in \Sigma^*$ e $a \in \Sigma$.

A notação $\Sigma_G(q)$ denota os eventos que estão ativos no estado q do autômato G, isto é, $\Sigma_G(q) = \{\sigma \in \Sigma \mid \delta(q, \sigma)!\}$.

A representação gráfica de um autômato é feita por grafos orientados, em que nós representando os estados são conectados por linhas orientadas. Essas linhas são as transições entre estados e cada uma delas recebe uma etiqueta com o nome do evento que produz a transição de estado. O estado inicial q_i é indicado por uma pequena seta sem origem apontando para o estado.

Para a máquina da figura 2.1, o autômato seria igual ao ilustrado na figura 2.2. O estado inicial é o nó 1 e a transição do estado 1 para o estado 2 acontece instantaneamente na ocorrência do evento x. O nó 1 possui uma duplo círculo, que indica que este estado

é marcado, ou seja, pertence a Q_m. O alfabeto é o conjunto $\Sigma = \{x, y\}$. A função de transição δ é tal que:

- $\delta(1, x) = 2$;
- $\delta(2, y) = 1$; e
- $\delta(1, y)$ e $\delta(2, x)$ são não definidas.

O conjunto de eventos ativos em cada estado é, portanto:

- $\Sigma_G(1) = \{x\}$; e
- $\Sigma_G(2) = \{y\}$.

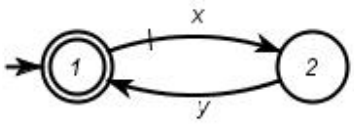


FIG. 2.2: Exemplo de autômato para um SED.

A determinação de que estados $q \in Q$ devem ser marcados, de forma que $q \in Q_m$, é tarefa de modelagem do SED. Os estados marcados denotam que, após a cadeia de eventos correspondente, o SED encontra-se em uma condição relevante para o seu comportamento. Isto pode significar que uma tarefa foi concluída, que o sistema está inoperante ou qualquer característica importante.

Um estado é acessível quando é possível chegar até ele a partir do estado inicial com sucessivas transições entre os estados, ou seja, para $q_k \in Q$, existe $s \in \Sigma^*$ tal que $\delta(q_i, s) = q_k$. Um automâto é acessível se todos os seus estados são acessíveis. É possível obter a componente acessível de um automâto eliminando os estados que não são acessíveis. A componente acessível de um automâto $G = (Q, \Sigma, \delta, q_i, Q_m)$ é representada por $Ac(G) = (Q_{ac}, \Sigma, \delta_{ac}, q_i, Q_{mac})$ em que Q_{ac} é o conjunto de estados acessíveis, $Q_{mac} = Q_m \cap Q_{ac}$ é o conjunto de estados marcados acessíveis e $\delta_{ac} : Q_{ac} \times \Sigma \to Q_{ac}$ é a restrição de δ aos estados de Q_{ac}.

Um estado é co-acessível se a partir dele é possível chegar em algum estado marcado, ou seja, para $q_k \in Q$, existe $s \in \Sigma^*$ tal que $\delta(q_k, s) = q_m \in Q_m$. Os estados não co-acessíveis são estados que correspondem a situações de bloqueio no SED, pois assim como no SED não será possível completar uma tarefa, no autômato não se poderá alcançar um estado marcado. Um automâto é co-acessível se todos os seus estados são co-acessíveis.

A componente co-acessível de um automâto qualquer é obtida eliminando-se os estados que não são co-acessíveis.

Os automâtos simultaneamente acessíveis e co-acessíveis são ditos Trim e modelam SEDs cujos comportamentos são não bloqueantes.

2.1.3 RELAÇÃO ENTRE AUTÔMATOS E LINGUAGENS

O comportamento de um SED pode ser modelado alternativamente por um autômato ou por um par de linguagens que modelam seu comportamento gerado, formada por todas as cadeias possíveis de eventos, e seu comportamento marcado, formada pelas cadeias que representam tarefas concluídas CURY (2001).

A relação entre as duas representações é tal que, para um autômato G, a linguagem gerada $L(G)$ que modela o comportamento gerado e a linguagem marcada $L_m(G)$ que modela o comportamento marcado, são definidas como:

$$L(G) \triangleq \{s \in \Sigma \mid \delta(q_i, s)!\} \text{ e}$$

$$L_m(G) \triangleq \{s \in L(G) \mid \delta(q_i, s) \in Q_m\}.$$

A linguagem gerada por um autômato $L(G)$ é, por definição, prefixo-fechada, ou seja, $L(G) = \overline{L(G)}$, já que para toda palavra $s \in L(G)$, os seus prefixos também estão contidos em $L(G)$. Isso se dá pela definição de que se $s \in L(G)$ então $\delta(q_i, s)$ é definida, o que implica que para $s = uv,\ \delta(q_i, s) = \delta(q_i, uv) = \delta(\delta(q_i, u), v)$, logo $\delta(q_i, u)$ é definida para todo prefixo u de s.

Em contrapartida, para a linguagem marcada $L_m(G)$ tem-se que $L_m(G) \subseteq \overline{L_m(G)} \subseteq L(G) = \overline{L(G)}$. Apenas para as linguagens geradas por um autômato trim se verifica a igualdade $\overline{L_m(G)} = L(G)$, isto é, todas as palavras da linguagem gerada são um sufixo para alguma palavra da linguagem marcada. Para uma linguagem $M = L_m(G)$ é dito que G marca a linguagem M.

2.1.4 COMPOSIÇÃO DE AUTÔMATOS E LINGUAGENS

Os SEDs por vezes incluem vários subsistemas que podem ser modelados individualmente. O modelo completo do sistema então pode ser obtido por intermédio da composição dos diferentes modelos desses subsistemas. Ao entrarem em operação simultaneamente, os subsistemas podem dividir recursos ou mesmo terem seu funcionamento condicionado ao de outros subsistemas. Neste contexto, cada um dos susbsistemas possui

um alfabeto que pode ser único ou não e que é um subconjunto do alfabeto global. Os eventos compartilhados por mais de um subsistema devem ocorrer de forma síncrona, pois eles ocorrem no sistema como um todo. A composição entre os modelos dos diversos subsistemas, linguagem ou autômato, deve levar este fato em consideração. A operação que realiza esta composição é chamada *produto síncrono*.

Para definir o produto síncrono de linguagens, antes deve-se definir a operação *projeção natural*. A projeção natural apaga de uma palavra formada no alfabeto Σ os eventos que não pertencem a um subalfabeto $\Sigma_i \subseteq \Sigma$. Seja $s \in \Sigma^*$ e $a \in \Sigma$. A projeção natural $P_i : \Sigma^* \to \Sigma_i^*$ é definida recursivamente como:

$$\begin{aligned} &1) P_i(\varepsilon) = \varepsilon; \\ &2) P_i(a) = \begin{cases} \varepsilon & \text{se} \quad a \notin \Sigma_i; \\ a & \text{se} \quad a \in \Sigma_i; e \end{cases} \\ &3) P_i(sa) = P_i(s)P_i(a), s \in \Sigma^*, a \in \Sigma. \end{aligned} \tag{2.2}$$

A projeção natural aplicada a uma linguagem $L \subseteq \Sigma^*$ é tal que:

$$P_i(L) = \{u_i \in \Sigma_i^* \mid u_i = P_i(u), u \in L\}. \tag{2.3}$$

Define-se também a projeção inversa $P_i^{-1} : \Sigma_i^* \to 2^{\Sigma^*}$ como:

$$P_i^{-1}(s_i) = \{s \in \Sigma^* \mid P_i(s) = s_i\}. \tag{2.4}$$

A projeção inversa de uma cadeia $s_i \in \Sigma_i$ é o conjunto formado por todas as palavras em Σ cuja projeção é igual s_i. Aplicada a uma linguagem $L_i \subseteq \Sigma_i^*$, a projeção inversa é:

$$P_i^{-1}(L_i) = \{t \in \Sigma \mid P_i(t) = t_i, t_i \in L_i\}. \tag{2.5}$$

Sejam agora $L_1 \subseteq \Sigma_1^*$ e $L_2 \subseteq \Sigma_2^*$ tais que $\Sigma = \Sigma_1 \cup \Sigma_2$.
O produto síncrono $L_1 \parallel L_2 \subseteq \Sigma^*$ é definido como:

$$L_1 \parallel L_2 \triangleq P_1^{-1}(L_1) \cap P_2^{-1}(L_2).$$

Em notação de conjunto:

$$L_1 \parallel L_2 = \{u \mid P_1(u) \in L_1 \ e \ P_2(u) \in L_2\}. \tag{2.6}$$

Dois casos particulares do produto síncrono podem ocorrer. O primeiro decorre de

$\Sigma_1 \cap \Sigma_2 = \emptyset$, quando a operação é chamada produto assíncrono. O nome deriva do fato de que como não há eventos em comum, todos eles podem ocorrer de forma assíncrona.

O segundo caso decorre se $\Sigma_1 = \Sigma_2$, neste caso, o resultado do produto síncrono é igual a interseção entre as linguagens.

A *composição síncrona* entre autômatos estabelece que um evento em comum só ocorre se ocorre para os dois modelos simultaneamente. Isto é alcançado ao se, verificar se para os estados em que cada autômato se encontra, a transição para este evento é definida ou não. A correspondência entre composição síncrona entre os autômatos e as suas linguagens é tal que $L(G_1 \| G_2) = L(G_1) \| L(G_2)$ e $L_m(G_1 \| G_2) = L_m(G_1) \| L_m(G_2)$.

Sejam os autômatos $G_1 = (\Sigma_1, Q_1, \delta_1, q_{01}, Q_{m1})$ e $G_2 = (\Sigma_2, Q_2, \delta_2, q_{02}, Q_{m2})$. O autômato $G = G_1 \| G_2 = Ac(\Sigma, Q, \delta, q_0, Q_m)$ é tal que:

- $\Sigma = \Sigma_1 \cup \Sigma_2$;
- $q_0 = (q_{01}, q_{02})$;
- $Q = Q_1 \times Q_2$;
- $Q_m = Q_{m1} \times Q_{m2}$; e
- $\delta((x,y),\sigma) = \begin{cases} (\delta_1(x,\sigma), \delta_2(y,\sigma)), & \text{se} \quad \sigma \in \Sigma_1 \cap \Sigma_2, \delta_1(x,\sigma)! \ e \ \delta_2(y,\sigma)! \\ (\delta_1(x,\sigma), y), & \text{se} \quad \sigma \in \Sigma_1, \ \sigma \notin \Sigma_2 \ e \ \delta_1(x,\sigma)! \\ (x, \delta_2(y,\sigma)), & \text{se} \quad \sigma \notin \Sigma_1, \ \sigma \in \Sigma_2 \ e \ \delta_2(y,\sigma)! \\ \text{não definida}, & \text{caso contrário.} \end{cases}$

Outra operação definida para autômatos é o *produto*. O produto entre autômatos estabelece que um evento só ocorre se ocorre para os dois modelos simultaneamente e eventos que não pertençam à interseção entre os alfabetos dos dois autômatos não ocorrem. Sejam os autômatos G_1 e G_2 definidos acima. O autômato $G = G_1 \times G_2 = Ac(\Sigma, Q, \delta, q_0, Q_m)$, que representa o produto entre G_1 e G_2, é tal que:

- $\Sigma = \Sigma_1 \cup \Sigma_2$;
- $q_0 = (q_{01}, q_{02})$;
- $Q = Q_1 \times Q_2$;
- $Q_m = Q_{m1} \times Q_{m2}$; e
- $\delta((x,y),\sigma) = \begin{cases} (\delta_1(x,\sigma), \delta_2(y,\sigma)), & \text{se} \quad \sigma \in \Sigma_1 \cap \Sigma_2, \delta_1(x,\sigma)! \ e \ \delta_2(y,\sigma)! \\ \text{não definida}, & \text{caso contrário.} \end{cases}$

A correspondência entre o produto dos autômatos e as suas linguagens é tal que $L(G_1 \times G_2) = L(G_1) \cap L(G2)$ e $L_m(G_1 \times G_2) = L_m(G_1) \cap L_m(G_2)$.

Embora cada subsistema individualmente possa ser modelado por um autômato trim, a operação concorrente destes subsistemas pode levar a um comportamento que inclua estados bloqueantes. Isso acontece porque ao dividir recursos, o funcionamento não coordenado do sistema pode levar a situações em que o mesmo fique impedido de evoluir.

Isso pode ser observado quando a composição síncrona dos autômatos resulta em um automato que não é co-acessível ou quando as linguagens dos modelos são conflitantes entre si.

Ao se conceber um sistema, além de se evitar que ele atinja estados a partir dos quais ele não possa mais evoluir, muitas vezes é necessário ou desejável que ele obedeça a certas regras de operação. A seção seguinte apresenta uma importante teoria que permite que estes objetivos sejam cumpridos, a Teoria de Controle Supervisório.

2.2 TEORIA DE CONTROLE SUPERVISÓRIO

A Teoria de Controle Supervisório (TCS) foi introduzida em (RAMADGE e WONHAM, 1989) e estabele métodos formais fundamentados na Teoria de Linguagens e Autômatos para obter um controle qualitativo de um SED. Esse controle é exercido por um agente externo denonimado supervisor. Segundo CURY (2001), o supervisor interage com o sistema numa estrutura de malha fechada de modo que, a cada evento ocorrido na planta, o supervisor define quais eventos, dentre os possíveis, são permitidos ocorrerem em sucessão imediata. O conjunto de eventos habilitados a cada sequência gerada pela planta constitui uma entrada de controle $\gamma \subseteq \Sigma$.

O supervisor também assume a função de determinar quais tarefas definidas em malha aberta permanecerão definidas em malha fechada, isto é, o supervisor pode alterar a marcação dos estados da planta, determinando que cadeia de eventos leva a conclusão de uma tarefa ou situação de destaque na planta[3].

O modelo do comportamento em malha aberta do sistema denomina-se *planta* e é representado pelo autômato $G = (\Sigma, Q, \delta, q_0, Q_m)$ ou por suas linguagens gerada e marcada, $L(G)$ e $L_m(G)$. Os modelos dos subsistemas também são chamados planta, porém adiciona-se o nome do subsistema correspondente de forma a diferenciá-la da

[3]Esse supervisor é denominado *supervisor marcador* ou *Marking Nonblocking Supervisory Controller* em (RAMADGE e WONHAM, 1989) em contraste com o *supervisor ordinário*, que apenas desabilita eventos.

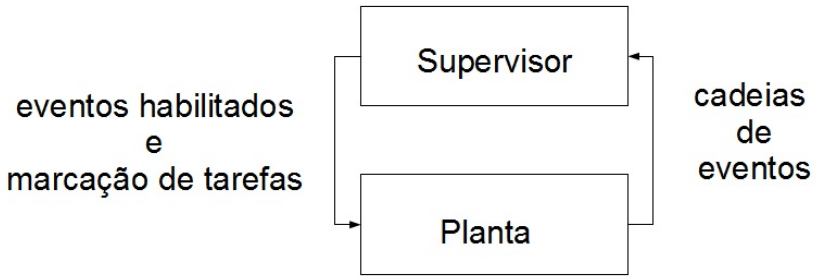


FIG. 2.3: Esquema do sistema em malha fechada com planta sob supervisão.

planta do sistema como um todo. A planta do sistema é obtida por intermédio da composição síncrona de todas as plantas dos subsistemas que o integram.

A planta gera os eventos espontaneamente e o supervisor apenas desabilita a ocorrência dos eventos necessários, sem obrigar a ocorrência de nenhum evento. Neste sentido, diz-se que o supervisor impõe um controle permissivo à planta (CURY, 2001).

Os eventos gerados pela planta representam ações ou alterações das condições do sistema. Naturalmente há ações e alterações que podem ser impedidas de acontecer durante a execução do sistema e há outras que depois de iniciado um certo processo, independentemente de interferência externa não podem ser evitadas. Ao se modelar a planta de um SED, os eventos recebem o atributo da controlabilidade, em que os eventos cujas ocorrências podem ser impedidas pela ação de um supervisor são denominados controláveis em oposição aos eventos cujas ocorrências não podem ser impedidas, denominados não controláveis. Este atributo resulta num particionamento do alfabeto tal que $\Sigma = \Sigma_u \dot{\cup} \Sigma_c$, em que Σ_c contém os eventos controláveis e Σ_u, os não controláveis. Como um evento não pode ser simultaneamente controlável e não controlável, $\Sigma_u \cap \Sigma_c = \emptyset$. A representação de um evento controlável num autômato é feita por intermédio de um traço no arco orientado correspondente. Salvo indicação contrária no autômato, o evento deve ser considerado não controlável.

Um supervisor S representado pelo autômato $Z = (P, \Sigma, \zeta, p_0, P_m)$ e um mapa de desabilitação de eventos[4] $\Phi : Z \rightarrow 2^{\Sigma}$. Se S é um supervisor para a planta G, então as transições do autômato Z são dirigidas pelos eventos gerados pela planta G. A cada estado de Z são mapeados os eventos controláveis cuja ocorrência estão impedidas de

[4]O mapa Φ é obtido pelo comando *condat* das ferramentas, para controle supervisório de SEDs, TCT e Grail.

ocorrer, ou seja, são mapeados os eventos desabiblitados. Para cada estado $p \in P$ de Z:

$$\begin{aligned}\Phi(p) = \{\sigma \in \Sigma \mid \exists s \in \Sigma^* \ e \ \exists q \in Q \text{ tal que,} \\ \zeta(p_0, s) = p, \delta(q_0, s) = q \ e \ \sigma \in \Sigma_G(q) - \Sigma_Z(p)\}\end{aligned} \tag{2.7}$$

O comportamento da planta em malha fechada, ou seja, sob supervisão de S, é denotado por S/G e é representado por $Z\|G$ tal que:

$$L_m(S/G) = L_m(Z\|G) = L_m(Z)\|L_m(G) \ e \tag{2.8}$$

$$L(S/G) = L(Z\|G) = L(Z)\|L(G). \tag{2.9}$$

O supervisor é não bloqueante se $\overline{L_m(S/G)} = L(S/G)$, ou seja, o comportamento da planta G em malha fechada induzido pelo supervisor S não contém estados bloqueantes.

2.2.1 SÍNTESE DE SUPERVISOR

O objetivo de um supervisor é fazer com que a planta opere livre de bloqueio em malha fechada e respeitando condições pré-estabelecidas de fucionamento. As condições de funcionamento atuam como restrição ao comportamento livre da planta. A linguagem que modela essa restrição denomina-se *especificação* indicada por $E \subseteq \Sigma^*$. As especificações podem englobar de restrições para garantia da operacionalidade do sistema a determinação do número de vezes que o sistema pode funcionar. Cada restrição pode ser modelada individualmente e a especificação global será então obtida da composição síncrona entre elas, $E = E_1\|E_2\| \ldots \|E_n$. O autômato que marca a linguagem E é denotado por $G(E)$ tal que $L_m(G(E)) = E$, mas ao longo do trabalho o símbolo E será utilizado para denotar tanto a linguagem E quanto o seu autômato $G(E)$.

A *Linguagem-alvo* $K = E \cap L_m(G)$ é o comportamento fisicamente possível do sistema que atende à especificação E no sentido que $K \subseteq E$ e $K \subseteq L_m(G)$. A linguagem-alvo será representada por um autômato R obtido pela componente trim do produto entre os autômatos da especificação e da planta, $R = Trim(G(E) \times G)$. Observe que $L_m(R) = K$ e $L(R) = \overline{K}$.

No entanto, o comportamento desejado do sistema pode exigir que eventos não controláveis sejam impedidos de ocorrer, o que de fato não pode ser feito. A solução é então extrair de K uma linguagem que não impeça nenhum evento não controlável de ocorrer e que ao mesmo tempo restrinja o comportamento da planta o mínimo possível. Esse é o Problema do Controle Supervisório (PCS).

Problema do Controle Supervisório (RAMADGE e WONHAM, 1989): *Dada uma planta G com alfabeto Σ e uma especificação genérica $E \subseteq \Sigma^*$ e o comportamento desejado $K = E \cap L_m(G)$. Encontrar um supervisor não bloqueante S tal que $\emptyset \neq L_m(S/G) \subseteq K$.*

O conceito envolvido na não desabilitação por uma linguagem $K' \subseteq K$ de um evento não controlável em $L(G)$ é a controlabilidade de linguagens. Uma linguagem K' é dita ser controlável em relação a (e.r.a) linguagem prefixo-fechada L e a um conjunto de eventos não controláveis Σ_u se $\overline{K'}\Sigma_u \cap L \subseteq \overline{K'}$.

Dada a planta G e a linguagem-alvo K, seja $C(K, G)$, um conjunto de sublinguagens de K, como definido a seguir:

$$C(K, G) = \left\{ J \subseteq K \mid \overline{J}\Sigma_u \cap L(G) \subseteq \overline{J} \right\} \tag{2.10}$$

Pode ser demonstrado que o conjunto $C(K, G)$ é não vazio, é fechado para a união e, portanto, possui um único elemento supremo (RAMADGE e WONHAM, 1989). A linguagem procurada, denominada máxima linguagem controlável e.r.a G contida em K e denotada por $SupC(K, G)$, é o supremo deste conjunto, que restringe a planta o mínimo possível enquanto a mantém operando dentro dos requisitos especificados. Em notaçao de conjunto:

$$SupC(K, G) = \bigcup_{J \in C(K)} J. \tag{2.11}$$

A solução para o PCS é então a linguagem $L_m(S/G) = SupC(K, G) = L_m(Z)$, em que Z é o autômato que marca a linguagem $SupC(K, G)$. Em (RAMADGE e WONHAM, 1989) é apresentado o método e o algoritmo para síntese de um supervisor a partir da planta G e o autômato que marca a linguagem K.

A complexidade do algoritmo de síntese de um supervisor é polinomial em m e n, em que m é o número de estados da planta G e n é o número de estados da especificação E. No entanto, m cresce exponencialmente com o número de subsistemas e seus respectivos números de estados em função da composição síncrona RAMADGE e WONHAM (1989). O número de estados n da especificação também cresce exponencialmente em razão da composição síncrona dos autômatos das diversas restrições.

Como exemplo da aplicação da TCS, seja o sistema da pequena fábrica da figura 2.4

(WONHAM, 2011), em que duas máquinas manufaturam uma peça em sequência, conectadas por um *buffer* B1 de capacidade unitária. As plantas de cada um dos subsistemas M1 e M2 estão expostas na figura 2.5.

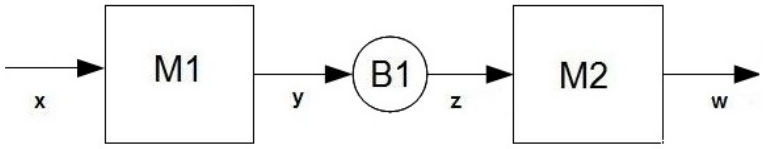


FIG. 2.4: Exemplo da pequena fábrica.

O funcionamento de cada máquina consiste somente em iniciar operação e terminar operação, representados pelos eventos x e y para a máquina M1 e z e w para a máquina M2. O início da operação das máquinas poder ser impedido, porém uma vez iniciada, a máquina obrigatoriamente deverá terminar. Isto determina o atributo de controlabilidade dos eventos, em que x e z são controláveis e y e w são não controláveis. No instante inicial, ambas as máquinas estão ociosas, representado pelo estado inicial 1. O estado marcado também é o estado 1, pois o estado desejado para as máquinas é que não estejam processando peças. O estado 2 naturalmente representa a máquina em funcionamento.

(a) M1 (b) M2

FIG. 2.5: Plantas de dois subsistemas do exemplo.

A planta do sistema é obtida pela composição síncrona destes autômatos, gerando o automato exposto na figura 2.6. Esta planta expressa o comportamento livre do sistema e pode ser notado que é possível que a máquina M2 entre em funcionamento antes de M1, o que não é desejado, pois o funcionamento é sequencial.

Cria-se então uma especificação E, cujo autômato pode ser visto na figura 2.7. Esta especificação condiciona o ínicio de operação da máquina M2 ao término de operação da máquina M1. A especificação determina que uma peça só pode ser retirada por M2 de B1 se antes M1 depositá-la, não *underflow* do *buffer*, e que M1 só pode depositar uma nova peça após a primeira ser retirada de B1 por M2, não *overflow* do *buffer*. O estado inicial é aquele em que não há nenhuma peça em B1 e esse também é o estado desejado, portanto marcado.

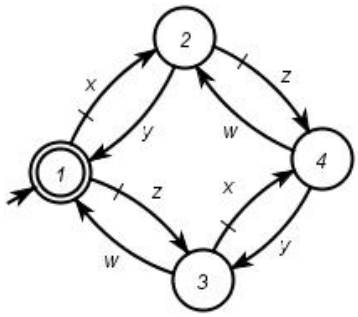


FIG. 2.6: Planta global do sistema exemplo.

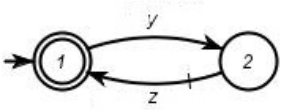


FIG. 2.7: Especificação para o sistema exemplo.

O comportamento desejado do sistema é expresso pelo autômato R da figura 2.8, obtido do produto síncrono entre os autômatos das figuras 2.6 e 2.7. Na figura são indicados, por setas pontilhadas, que eventos são desabilitados na planta para cada estado do autômato R. Nota-se que nos estados 4 e 8, o evento não controlável y está impedido de ocorrer, ou seja, esse autômato não marca uma linguagem controlável em relação à linguagem da planta.

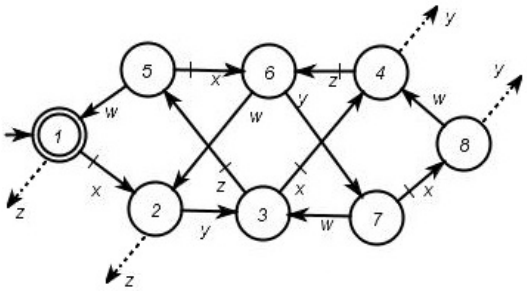


FIG. 2.8: Autômato que marca a linguagem-alvo do sistema exemplo.

A máxima linguagem controlável é marcada pelo autômato da figura 2.9, que não contém estados bloqueantes nem estados nos quais eventos não controláveis são impedidos de ocorrer.

2.2.2 CONTROLE MODULAR

Baseado no fato de uma especificação para um determinado SED pode ser obtida pela composição de várias especificações elementares, que modelam uma restrição somente

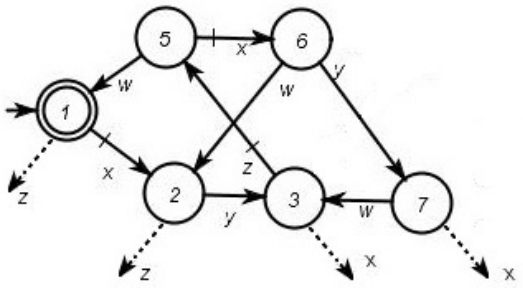


FIG. 2.9: Supervisor para o sistema exemplo.

cada, foi introduzido o conceito de Controle Modular como extensão da TCS (WONHAM e RAMADGE, 1988).

Nesta abordagem, um *supervisor modular* para cada especificação elementar é sintetizado individualmente. Com isso, evita-se a potencial explosão combinatória do número de estados da especificação global para o sistema, implicando uma redução da complexidade da síntese do supervisor modular. O comportamento total da planta em malha fechada é obtido ao se compor todos os supervisores modulares (WONHAM e RAMADGE, 1988), ou seja, para dois supervisores S e T para uma planta G tem-se que:

$$\begin{aligned} &1) L(S \times T/G) = L(S/G) \cap L(T/G);\ e \\ &2) L_m(S \times T/G) = L_m(S/G) \cap L_m(T/G). \end{aligned} \tag{2.12}$$

No entanto, é necessário verificar se os supervisores assim obtidos, quando aplicados concorrentemente ao sistema, não resultaram em um comportamento bloqueante da planta em malha fechada. Como dito na subseção 2.1.4, o comportamento conjunto de duas linguagens conflitantes implica bloqueio no SED, logo, para se garantir que não haverá bloqueio causado pela síntese modular dos supervisores, deve ser verificado se os mesmos são não conflitantes entre si.

Sejam a planta G e as especificações elementares $E_1, E_2 \subseteq \Sigma^*$ tal que $K_1 = L_m(G) \cap E_1$ e $K_2 = L_m(G) \cap E_2$ são as linguagens alvo para a planta associada a cada especificação individualmente e os supervisores modulares não bloqueantes $L_m(S_1/G) = supC(K_1, G)$ e $L_m(S_2/G) = supC(K_2, G)$.

Em WONHAM e RAMADGE (1988) é demonstrado que o supervisor obtido pela composição dos supervisores S_1 *e* S_2 é a solução ótima para o PCS se:

$1) L_m(S_1/G) \cap L_m(S_2/G) \neq \emptyset;\ e$
$2) \overline{L_m(S_1/G)} \cap \overline{L_m(S_2/G)} = \overline{L_m(S_1/G) \cap L_m(S_2/G)}$, isto é, $L_m(S_1/G)$ *e* $L_m(S_2/G)$ são não conflitante

(2.13)

Posteriormente, o Controle Modular foi também extendido por outras abordagens, como por exemplo o Controle MOdular Local (QUEIROZ e CURY, 2002a).

2.3 CONCLUSÃO

Este capítulo apresentou os Sistemas a Eventos Discretos e a Teoria de Controle Supervisório como método para sintetizar supervisores para tais sistemas. Foi exposto que a planta de um sistema é obtida pela composição das plantas dos vários subsistemas e que as restrições ao comportamento da planta são expressas por especificações genéricas.

No entanto, para sistemas compostos de grande de porte há dois grandes obstáculos a serem superados. O primeiro diz respeito à escrita das especifições. O exemplo que ilustra a aplicação da TCS para SED é simplório e não exprime a dificuldade encontrada para modelar certas especificações. Por mais direta que uma restrição seja, sua modelagem pode exigir sofisticadas especificações, que de tão complexas, podem tornar a tarefa inexequível.

O segundo obstáculo é a complexidade computacional envolvida na síntese de um supervisor. Conforme exposto na seção 2.2, o número de estados da planta de um sistema composto de grande porte cresce exponencialmente com o número de subsistemas e pode atingir proporções em que a síntese de um supervisor demanda muitos recursos computacionais e não pode ser realizada pelos meios disponíveis.

O capítulo seguinte aborda uma extensão do TCS, a Teoria de Controle Supervisório Distinguido que propõe uma solução para estes dois problemas.

3 TEORIA DE CONTROLE SUPERVISÓRIO COM DISTINGUIDORES

A Teoria de Controle Supervisório, exposta no capítulo anterior, provê métodos para síntese de supervisores minimamente restritivos para Sistemas a Eventos Discretos. A aplicação destes métodos encontra algumas restrições porém. A especificação para um sistema, embora conceitualmente simples, pode ser difícil de ser modelada na prática, particularmente nos casos em que a restrição é baseada na ocorrência de muitos eventos anteriores.

A síntese de supervisores também pode se tornar inviável para sistemas de grande porte, constituídos por muitos subsistemas e com várias restrições de funcionamento. Devido ao problema do crescimento exponencial dos números de estados da planta e da especificação, conforme exposto na seção 2.2, a síntese de um supervisor para o sistema esgota os recursos computacionais disponíveis.

Com o objetivo de contornar essas limitações, criou-se o conceito de distinguidores e dando origem à Teoria de Controle Supervisório com Distinguidores(TCS-D), que incorpora os distinguidores na obtenção de supervisores (BOUZON et al., 2008a). Os distinguidores são modelos que diferenciam instâncias de um mesmo evento, e envolve o refinamento do alfabeto original em um novo alfabeto cujos eventos descrevem um comportamento mais detalhado do sistema.

A inserção de distinguidores no sistema pode ser realizada pela da adição de sensores no sistema físico, cuja função é distinguir ocorrências de mesmo evento, por exemplo diferenciando a chegada de uma caixa amarela da chegada de uma caixa vermelha. Não obstante, o distinguidor não precisa resultar das observações de um sensor físico, podendo ser simplesmente um modelo que representa a evolução do sistema, explicitando características antes ignoradas.

Por vezes, o simples refinamento de eventos do sistema é suficiente para simplificar a tarefa de compor especificações para o mesmo e nenhum distinguidor precisa ser utilizado. Em outros casos, a distinção de apenas alguns eventos é necessária para se obter um supervisor ótimo. Esse é o foco do estudo de aproximações por distinguidores, cujo principal benefício é a redução da complexidade da síntese de supervisores.

As seções a seguir fornecem as informações necessárias para se compreender a Teoria de Controle Supervisório Distinguido e Aproximado e suas vantagens. Um exemplo

ilustrativo mostra a aplicação dos conceitos para síntese de supervisores. Ao final, é feita uma pequena apreciação e são apontados os pontos que esse trabalho explora.

3.1 DISTINGUIDORES

O refinamento do conjunto de eventos de um SED objetiva criar um novo conjunto cujos eventos representam instâncias dos eventos do alfabeto original. O evento do alfabeto original é denominado *máscara* para os eventos, no novo alfabeto, que representam suas instâncias.

Seja Σ o alfabeto original e Δ o novo alfabeto refinado. Para cada máscara $\sigma \in \Sigma$ existe um conjunto $\Delta^\sigma \neq \emptyset$ formado pelos refinamentos de σ. O novo alfabeto Δ contém então os refinamentos de todas máscaras, ou seja, $\Delta = \bigcup_{\sigma\in\Sigma} \Delta^\sigma$. Como Δ^σ representa os refinamentos de uma mesma máscara, para $\sigma_1, \sigma_2 \in \Sigma$ tal que $\sigma_1 \neq \sigma_2$ tem-se que $\Delta^{\sigma_1} \cap \Delta^{\sigma_2} = \emptyset$.

O *Mapa Mascarador* $\Pi : \Delta^* \to \Sigma^*$ estabele a correspondência entre as cadeias em Δ^* e Σ^*. Esse mapeamento é definido recursivamente em TEIXEIRA et al. (2011) como:

$$\begin{aligned} &1)\Pi(\epsilon) = \epsilon; \\ &2)\Pi(t\rho) = \Pi(t)\sigma \text{ para t } \in \Delta^*, \rho \in \Delta^\sigma \text{ e } \sigma \in \Sigma. \end{aligned} \tag{3.1}$$

Naturalmente o mapa Π pode ser aplicado a linguagens $L_d \subseteq \Delta^*$, de modo que:

$$\Pi(L_d) = \{s \in \Sigma \mid \exists t \in L_d, \Pi(t) = s\}. \tag{3.2}$$

O mapa Π é ainda prefixo-preservante, pois $s \leq t$ implica $\Pi(s) \leq \Pi(t)$, e preserva o prefixo-fechamento de uma linguagem tal que $\overline{\Pi(L_d)} = \Pi(\overline{L_d})$ (BOUZON et al., 2008b).

O Mapeamento inverso $\Pi^{-1} : \Sigma^* \to 2^{\Delta^*}$ associa às cadeias em Σ^* todas as possíveis combinações dos refinamentos de seus eventos. O mapeamento inverso Π^{-1} é definido como (CURY et al., 2013):

$$\Pi^{-1}(s) = \{t \in \Delta^* \mid \Pi(t) = s\}. \tag{3.3}$$

E para uma linguagem $L \subseteq \Sigma^*$:

$$\Pi^{-1}(L) = \{t \in \Delta^* \mid \Pi(t) \in L\}. \tag{3.4}$$

O *Distinguidor* (CURY et al., 2013) é então definido no sentido de assegurar que para

cada palavra em Σ^* haja pelo menos uma palavra correspondente formada por eventos refinados em Δ^*. Sejam os alfabetos Σ e $\Delta = \bigcup_{\sigma\in\sigma} \Delta^\sigma$. O mapeamento $D : \Sigma^* \to 2^{\Delta^*}$ é um distinguidor se $\forall s \in \Sigma^*, \sigma \in \Sigma, r \in \Delta^*$ e $\rho \in \Delta$:

$$\begin{aligned} &1) D(\epsilon) = \{\epsilon\}\,; \\ &2) r\rho \in D(s\sigma) \Rightarrow r \in D(s) \text{ e } \rho \in \Delta^\sigma; \\ &3) r \in D(s) \Rightarrow \exists \rho' \in \Delta^\sigma \text{ tal que } r\rho' \in D(s\sigma). \end{aligned} \tag{3.5}$$

Pela definição apresentada acima, o mapa Π^{-1} é um caso particular de distinguidor. Ao aplicar-se o distinguidor a uma linguagem $L \subseteq \Sigma^*$, a linguagem resultante é:

$$D(L) = \{r \in \Delta^* \mid \exists s \in L, r \in D(s)\}\,. \tag{3.6}$$

Em CURY et al. (2013) demonstra-se que $\Pi(D(L)) = L$ e $\overline{D(L)} = D(\overline{L})$.

Alternativamente pode-se caracterizar o distinguidor por sua linguagem. A *Linguagem do Distinguidor* é dada por $L_D = D(\Sigma^*) \subseteq \Delta^*$. Além disso, conforme demonstrado em (CURY et al., 2013):

$$D(L) = \Pi^{-1}(L) \cap L_D. \tag{3.7}$$

Um distinguidor é dito ser *preditivo* ou *exato* quando ele associa uma única cadeia $t \in \Delta^*$ para cada cadeia $s \in \Sigma^*$, isto é, se $\forall s \in \Sigma^*\ |D(s)| = 1$. Caso contrário, o distinguidor é dito não preditivo (CURY et al., 2013). Além disso, é demonstrado em CURY et al. (2013) que, se $D : \Sigma^* \to 2^{\Delta^*}$ é um distinguidor preditivo, $\forall L_d \subseteq L_D \subseteq \Delta^*$:

$$D(\Pi(L_d)) = L_d. \tag{3.8}$$

Seja, por exemplo, um alfabeto $\Sigma = \{a, b\}$ e a linguagem $L \subseteq \Sigma^* = \{aab, b\}$. Seja ainda para este exemplo, o alfabeto refinado $\Delta = \{a1, a2, b\}$ em que os eventos $a, b \in \Sigma$ são máscaras para os eventos em Δ de forma que $\Delta^b = \{b\}$ e $\Delta^a = \{a1, a2\}$.

O mapa Π^{-1} aplicado à linguagem L determina a linguagem $L_a = \Pi^{-1}(L) = \{a1a1b, a1a2b, a2a1b, a2a2b, b\}$. Ao se aplicar à linguagem L um distinguidor preditivo D_p que determine que a primeira ocorrência do evento $a \in \Sigma$ corresponde à instância $a1 \in \Delta$ e todas as ocorrências seguintes correspondem à instância $a2 \in \Delta$, obtém-se a linguagem $D_p(L) = L_d \subseteq \Delta^*$ tal que $L_d = \{a1a2b, b\}$.

3.2 SUPERVISORES COM USO DE DISTINGUIDORES

Seja G_d com $L(G_d) = D(L(G)) = \Pi^{-1}(L(G)) \cap L_D$ e $L_m(G_d) = D(L_m(G)) = \Pi^{-1}L_m(G) \cap L_D$ a representação desta planta modelada com eventos em Δ^* e $E_d \subseteq \Delta^*$ uma especificação genérica escrita diretamente no alfabeto Δ.

A escrita da especificação E_d é beneficiada pela informação agregada pelo refinamento dos eventos, sendo tarefa mais simples que a de obter uma especificação $E \subseteq \Sigma^*$. Mesmo em alfabetos diferentes, ambas as especificações devem exprimir exatamente as mesmas restrições do sistema, ainda que expressas por modelos também diferentes. ´

Por exemplo, seja uma máquina cujo início de operação é modelado pelo evento *ini_ maq* e seu alarme cujo acionamento é modelado pelo evento *soar_ alarme*. Uma possível especificação seria determinar que após a máquina ser iniciada 5 vezes, o alarme deve soar. O modelo da especificação no alfabeto original deve contabilizar o número de vezes que o evento *ini_ maq* ocorreu antes de permitir que ocorra o evento *soar_ alarme*. No alfabeto refinado, em que o evento *ini_ maq* foi refinado em instâncias que indicam o número do acionamento em questão, o modelo para tal especificação seria somente permitir o evento *soar_ alarme* após o evento refinado *ini_ maq_ 5*.

seja uma especificação de um sistema que determina que ao ser acionada pela quinta vez, uma máquina deve soar um alarme. Para a especificação escrita no alfabeto refinado, bastaria determinar que o evento *soar_ alarme* só ocorra após a quinta instância de *acionamento*. A especificação no alfabeto original para expressar a mesma restrição deve *contar* quantas vezes o evento *acionamento* ocorreu e então determinar que, após o número correto de acionamentos, o evento *soar_ alarme* pode ocorrer.

A especificação global distinguida é obtida por $K_d = E_d \cap L_m(G_d) = D(K) = \Pi^{-1}(K) \cap L_D$, em que, $K = E \cap L_m(G) = \Pi(E_d \cap \Pi^{-1}(L_m(G)) \cap L_D)$.

Cabe ressaltar que as instâncias do eventos devem preservar o atributo de controlabilidade, ou seja, instâncias de eventos não controláveis também são não controláveis, valendo o mesmo para eventos controláveis. Desse modo $\Delta_u = \bigcup_{\sigma \in \Sigma_u} \Delta^\sigma$ e $\Delta_c = \bigcup_{\sigma \in \Sigma_c} \Delta^\sigma$, com $\Delta = \Delta_c \dot{\cup} \Delta_u$.

A ação dos supervisores sobre as plantas distinguida e original será a mesma se em ambas as plantas os supervisores desabilitarem a mesma máscara. Isso significa que para um $t \in L(G_d)$ se o supervisor distinguido desabilitar um $\rho \in \Delta^\sigma$, então ele deve desabilitar todos os refinamentos restantes da máscara Δ^σ.

Um *supervisor não ambíguo* ou *supervisor exato* atua segundo o preceito anterior,

em que, para $S_d : L(G_d) \rightarrow 2^{\Delta_c}$[5], para todo $t \in L(G_d)$ e $\sigma \in \Pi(S_d(t))$, se $\rho \in \Delta^\sigma$ então $\rho \in S_d(t)$ (BOUZON et al., 2009). A partir disso é formulado o Problema do Controle Supervisório Distinguido (PCS-D).

Problema do Controle Supervisório Distinguido (CURY et al., 2013): *Dado G e $K \subseteq L_m(G)$ como no PCS e um distinguidor não ambíguo D, seja G_d tal que $L(G_d) = D(L(G))$ e $L_m(G_d) = D(L_m(G))$. Para uma especificação genérica $E_d \subseteq \Delta^*$ definindo um comportamento desejado $K_d = E_d \cap L_m(G_d)$ em que $K_d = D(K)$, encontre um supervisor não ambíguo não bloqueante $S_d : \Delta^* \rightarrow 2^{\Delta_c}$ tal que $\emptyset \neq L_m(S_d/G_d) \subseteq K_d$.*

O teorema 3.1 indica que a solução para o PCS-D equivale à solução do PCS, sendo maximamente permissivo e imprimindo o mesmo comportamento em malha fechada à planta em questão.

Teorema 3.1. *(Equivalência das soluções para o PCS e o PCS-D (BOUZON et al., 2008a; CURY et al., 2013)) Dados G e K como no PCS e o distinguidor preditivo D, seja G_d um autômato para G distinguidor por D e $K_d = D(K)$, então:*

$$\Pi(supC(K_d, G_d)) = supC(K, G) \; e \tag{3.9}$$

$$supC(K_d, G_d) = D(supC(K, G)). \tag{3.10}$$

A seguir, o uso de distinguidores para a síntese de supervisores é ilustrado no exemplo de um sistema de manufatura com retrabalho de peças, inspirado no sistema apresentado em TEIXEIRA et al. (2011). O sistema do exemplo, figura 3.1, consiste em duas máquinas, M1 e M2, uma unidade de testes, TU, um robô, RB e 4 *buffers* de capacidade unitária que conectam as máquinas.

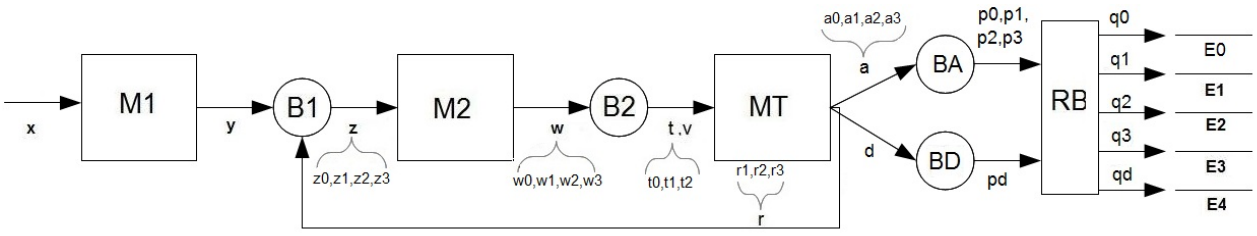
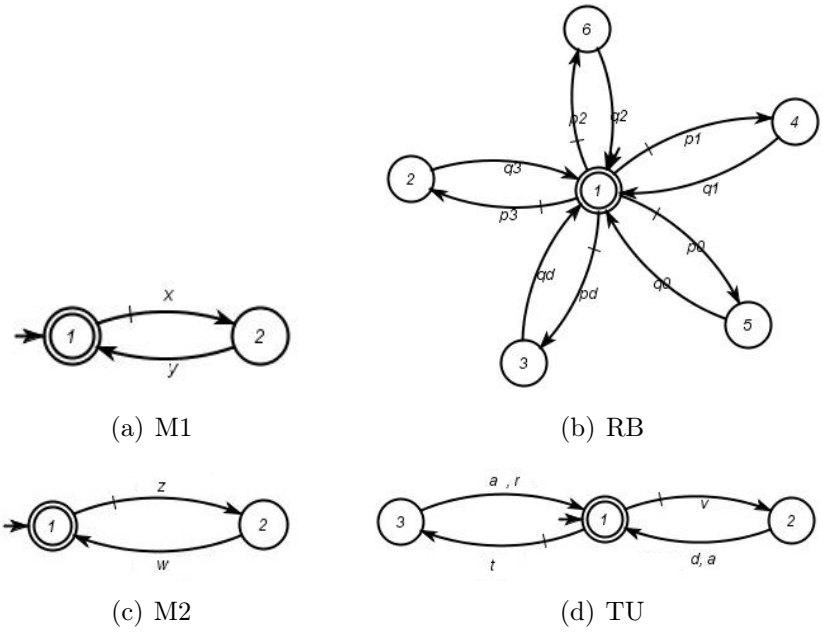


FIG. 3.1: Exemplo ilustrativo de um sistema de manufatura com retrabalho.

[5]O mapa S_d é o mapa de eventos desabilitados em $L(G_d)$ conforme seção 2.2.

As máquinas M1 e M2 processam uma peça que em seguida é verificada em TU. Dois tipos de verificação são possíveis em TU: o teste, cujo resultado é a aprovação da peça ou encaminhamento para retrabalho na máquina M2 e a validação que determina a aprovação ou descarte da peça. As peças aprovadas seguem para o *buffer* BA, as descartadas para o *buffer* BD e as encaminhadas para retrabalho para o *buffer* B2. O robô RB executa cinco tipos de movimento, cada um representando o ato de pegar uma peça no *buffer* BD ou BA e depositar em uma das cinco esteiras de saída. As plantas dos elementos do sistema são apresentadas na figura 3.2.



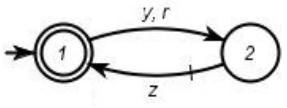
(a) M1 (b) RB

(c) M2 (d) TU

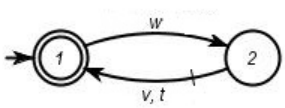
FIG. 3.2: Plantas do sistema exemplo.

Na modelagem das plantas, considerou-se que o início de operação de M1, M2 e RB são eventos controláveis, assim como os eventos que representam o teste e a validação de peças em TU, respectivamente t e v. O fim de operação é considerado não controlável, assim como o resultado das verificações, a para aprovado, d para descartado e r para deve ser retrabalhado.

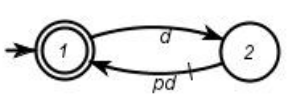
As especificações para o sistema incluem as necessárias para se evitar que os *buffers* tenham sua capacidade execedida e para o correto direcionamento das peças nas esteiras de saída. O robô RB deve depositar as peças nas esteiras de acordo com o número de retrabalho a que foram submetidas antes da aprovação, ou no caso de descarte, na esteira de peças descartadas. Adicionamente, deseja-se que peça seja submetida à validação em TU somente após 3 ciclos de retrabalho. Os autômatos que representam os modelos para as especificações relativas à capacidade dos *buffers*, encontram-se na figura 3.3.



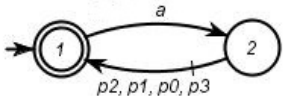
(a) *buffer* B1 - E_1



(b) *buffer* B2 - E_2



(c) *buffer* Bd - E_3



(d) *buffer* Ba - E_4

FIG. 3.3: Especificações do sistema exemplo.

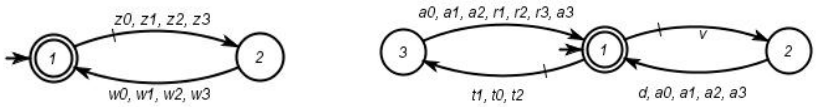
Os autômatos das outras duas especificações, porém, são complexos demais para serem criados manualmente. Cada uma delas deve rastear as peças no sistema para que se determine em qual ciclo de retrabalho cada uma está a fim de se decidir a qual tipo de verificação esta será submetida, uma vez que o sistema comporta mais de uma peça na linha de produção. O refinamento dos eventos do sistema então implica uma grande simplificação destas especificações.

Primeiramente determina-se que eventos devem ser refinados em mais de uma instância, trazendo vantagens para a modelagem. Para o presente caso, a restrição do número de ciclos de retrabalho sugere refinar o evento w, término de operação de M2, pois após a peça ser processada pela quarta vez, totalizando três retrabalhos, deve ocorrer a validação. Ou seja, o evento w deve ser refinado em quatro instâncias, cada uma se referindo ao número de retrabalhos a que a peça já foi submetida. Para isso, o evento r da unidade de testes TU também deve ser refinado para que se saiba para qual ciclo de retrabalho a peça esta seguindo. No total, cinco eventos devem ser refinados em mais de uma instância. Abaixo segue o mapeamento do alfabeto Σ para o alfabeto Δ com as respectivas máscaras:

- $\Delta^x = \{x\}$;
- $\Delta^y = \{y\}$;
- $\Delta^z = \{z1,z2,z3,z4\}$;
- $\Delta^w = \{w0,w1,w2,w3\}$;
- $\Delta^t = \{t0,t1,t2\}$;
- $\Delta^v = \{v\}$;
- $\Delta^r = \{r1,r2,r3\}$;

- Δ^a = {a0,a1,a2,a3};
- Δ^{px} ={px}, x ∈ {0,1,2,3,d};
- Δ^{qx} = {qx}, x ∈ {0,1,2,3,d}.

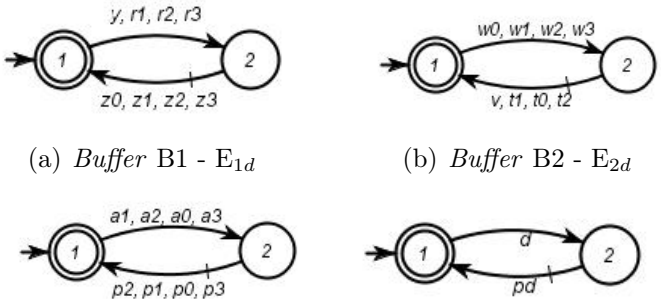
A única alteração na planta dos elementos que tiveram eventos refinados é a substituição da máscara em Σ pelos refinamentos da máscara correspondente. Isso porque o comportamento modelado ainda é o mesmo. Para a planta da máquina M2, por exemplo, independente da instância de início de operação que ocorre, ó autômato muda do estado *1*, máquina ociosa, para o estado *2*, máquina operando e independente da instância de término de operação, o autômato volta ao estado *1*. A figura 3.4 mostra como ficam as plantas dos elementos que refinados.



(a) M2 (b) TU

FIG. 3.4: Plantas com eventos refinados para o sistema exemplo.

O mesmo procedimento é aplicado aos autômatos das especificações que tratam a manutenção da capacidade dos *buffers*. Os autômatos no alfabeto Δ estão expostos na figura 3.5.



(a) *Buffer* B1 - E_{1d} (b) *Buffer* B2 - E_{2d}

(c) *Buffer* Ba - E_{3d} (d) *Buffer* Bd - E_{4d}

FIG. 3.5: Especificações com eventos refinados para o sistema exemplo.

A especificação relativa ao roteamento de peças, figura 3.6, pode agora ser facilmente modelada. Com o refinamento do evento que representa a aprovação da peça na verificação em TU, por meio de teste ou validação, associa-se cada instância de *a*, *a0,a1,a2* e *a3*, ao movimento de RB que conduz a esteira de destino. Acrescenta-se também o

destino de peças descartadas, associando-se o evento d ao movimento de RB iniciado por *pd*, que conduz à esteira de peças descartadas.

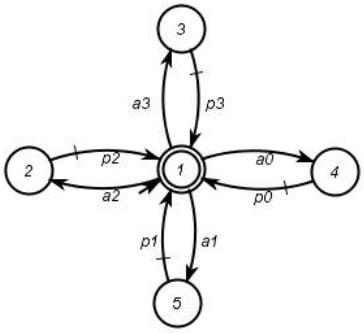


FIG. 3.6: Especificação de roteamento de peças para o sistema exemplo - E5d.

Por fim, modela-se a especificação que determina que após três ciclos de retrabalho a peça deve obrigatoriamente ser validada em TU e não mais testada, implicando que a mesma seja ou aprovada ou descartada. Com o uso de refinamento, a modelagem simplesmente permite que todas as instâncias do evento t, teste, ocorram livremente até a ocorrência do evento *w3*, que indica que a peça terminou de ser retrabalhada pela terceira vez em M2. O único evento liberado então é v, ou seja, a peça deve ser validada. O autômato pode ser conferido na figura 3.7.

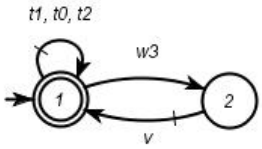


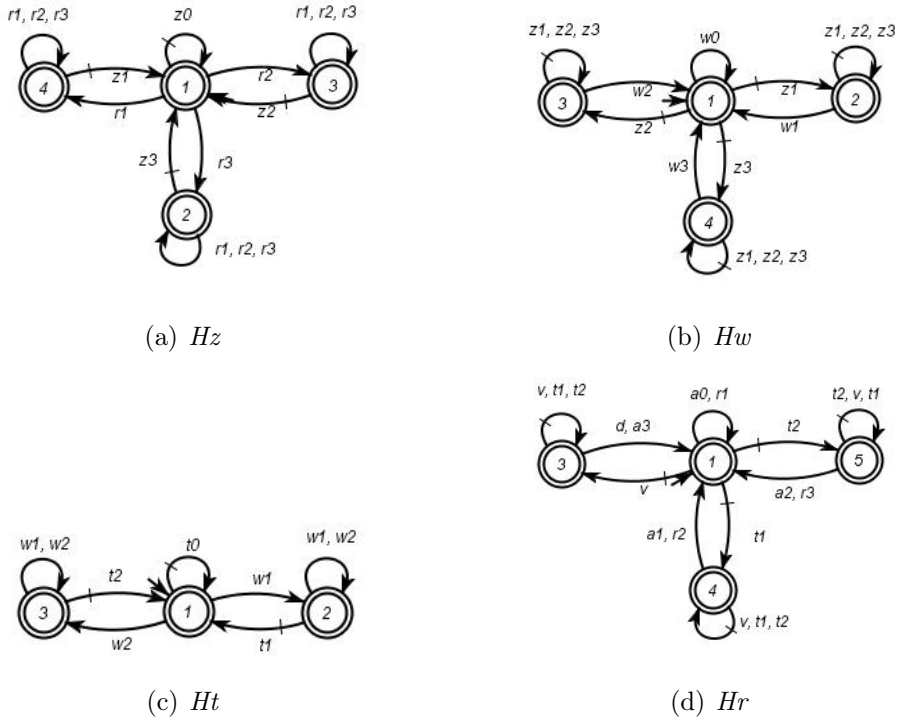
FIG. 3.7: Especificação para número máximo de ciclos de retrabalho - E6d.

Após o refinamento dos eventos em Σ para o alfabeto Δ, é preciso gerar a linguagem do distinguidor L_D para que o sistema tenha conhecimento de qual instância ocorre em dado momento. Para isso, cria-se o autômato H tal que $L_m(H) = L_D$. O autômato H pode ser criado modularmente, ao se compor um módulo distinguidor para cada conjunto de instâncias de uma máscara. Ao final, a composição de todos os módulos, mais Δ^*, resulta no autômato H.

A figura 3.8 mostra os autômatos dos módulos de H. Deve-se ter em mente que a função do distinguidor é apenas proceder à distinção entre a ocorrência das diversas instâncias de uma mesma máscara, razão pela qual nenhuma máscara deve ser impedida de ocorrer na construção de seu autômato. Na construção de cada módulo, para os eventos da mesma máscara, deve ser assegurado que sempre haja uma instância que ocorra por

padrão, de modo que a ocorrência da máscara esteja habilitada e o distinguidor não atue como restrição o sistema.

O módulo *Hz* distingue os eventos da máscara *z* e a instância *z0* é escolhida como padrão. Se o evento que precede o depósito da peça a ser processada em M2 for *r1*, ou seja, a peça vai para seu primeiro retrabalho, então a instância correspondente é *z1*. O mesmo se aplica às instâncias restantes *z2* e *z3*, figura 3.8.



(a) *Hz* (b) *Hw*

(c) *Ht* (d) *Hr*

FIG. 3.8: Módulos do distinguidor *H* para o sistema exemplo.

O módulo *Hw* define a instância *w0* como instância padrão e as outras ocorrem após a instância da máscara *z* de indíce correspondente. O módulo *Ht*, distingue as instância da máscara *t*, definindo *t0* como padrão e que a instância *t1* ocorrer após o primeiro retrabalho da peça e a instância *t2* após o segundo retrabalho.

O módulo *Hr* procede à distinção dos refinamentos das máscaras *a* e *r* simultaneamente. As instâncias padrão são respectivamente *a0* e *r1*, que denotam *aprovado após nenhum ciclo de retrabalho* e *encaminhado para o primeiro retrabalho*. Após a ocorrência da instância *t1*, que denota que a peça está sendo testada pela segunda vez, é definido a distinção das instâncias *a1* e *r2*, respectivamente *aprovado após um ciclo de retrabalho* e *encaminhado para o segundo retrabalho*. Analogamente, após *t2* tem-se *a2* e *r3* e após *v*, a instância *a3*. Nota-se que após a ocorrência de *v*, não há instância definida para a máscara *r* e além disso, adiciona-se o evento *d*, habilitado somente neste estado.

Embora conceitualmente incorreto, esta modelagem não implica restrição ao sistema, pois é direcionado pelo comportamento da planta de TU, que não prevê a ocorrência da máscara r imediatamente após a ocorrência de v enquanto que a máscara d só ocorre após a mesma.

A seguir os resultados para a síntese de um supervisor a partir do uso de distinguidor preditivo é apresentado na tabela 3.1. O autômato $G_d = G_a \| H$ e $Ed = E_{1d} \| E_{2d} \| E_{3d}] \| E_{4d} \| E_{5d} \| E_{6d}$. O autômato $R_d = Ed \| G_d$ é tal que $L_m(R_d) = K_d$ e $Z_d = SupC(K_d, G_d)$. O autômato H tem um total de 156 estados e G_a, 72.

TAB. 3.1: Resultados da síntese de supervisores para sistema exemplo utilizando TCS e TCS-D

G	E	R	Z
72	2920	51960	492
G_d	Ed	R_d	Z_d
2736	60	51960	492

Para efeitos de comparação, também na tabela 3.1 são apresentados os resultados da síntese de um supervisor na solução do PCS. Observa-se que os autômatos para a especificação global, K, e o supervisor, Z, têm o mesmo número de estados para ambas as sínteses. Cabe ressaltar que os autômatos para as especificações E_5 e E_6 para resolução do PCS foram obtidos a partir de $\Pi(E_{5d} \| H)$ e $\Pi(E_{6d} \| H)$. Isso se faz necessário devido à grande complexidade envolvida na composição de tais autômatos, que possuem 211 e 780 estados cada.

3.3 APROXIMAÇÕES NA SÍNTESE DE SUPERVISORES

A síntese para supervisores distinguidos embora solucione os aspectos referentes a simplificação da especificação, não altera a complexidade computacional para obtenção do supervisor. Nota-se contudo, que a complexidade outrora derivada da especificação genérica, agora é advém da planta distinguida, mais particularmente pela composição da linguagem da planta refinada com a linguagem do distinguidor.

Visando a reduzir esta complexidade, utilizam-se aproximações da planta distinguida. Essas aproximações no entanto influenciam a obtenção de supervisores, não havendo garantia de que para qualquer aproximação é possível obter um supervisor maximamente permissivo ou mesmo um supervisor que solucione o problema de controle (BOUZON et al., 2008b).

Uma aproximação de uma planta distinguida é obtida ao se utilizar um distinguidor não preditivo para distinguir os eventos de $\Pi^{-1}(L(G))$, que é a planta modelada com eventos refinados. A relação entre as linguagens do distinguidor não preditivo D_{aj} e o preditivo D é $L_D \subseteq L_{Daj} \subseteq \Delta^*$. Enquanto o distinguidor D estabelece uma uma distinção única para cada evento ocorrido na planta, D_{aj} distingue parcialmente as instâncias entre si, podendo diferenciar totalmente as instâncias de uma máscara ao passo que promove diferenciação apenas entre algumas instâncias de outra máscara (BOUZON et al., 2008b).

Seja então uma *planta aproximada* G_{aj} derivada de uma aproximação para G_d. As linguagens gerada e marcada de G_{aj} são tais que:

$$L(G_{aj}) = D_{aj}(L(G)) = \Pi^{-1}(L(G)) \cap L_{Daj} \; e \tag{3.11}$$

$$L_m(G_{aj}) = D_{aj}(L_m(G)) = \Pi^{-1}(L_m(G)) \cap L_{Daj}. \tag{3.12}$$

Dadas as linguagens $L_1 \subseteq L_2 \subseteq \Delta^*$ e um conjunto de máscaras $\Lambda \subseteq \Sigma$, define-se que L_1 é Λ-preservante e.r.a L_2 se $\forall \sigma \in \Lambda$ e $t \in \overline{L_1}$ tem se que $t\Delta^\sigma \cap \overline{L_2} \neq \emptyset \Rightarrow t\Delta^\sigma \cap \overline{L_1} \neq \emptyset$ (CURY et al., 2013).

Define-se também o conjunto $P_\Lambda(L_1, L_2)$ de linguagens contidas em L_1 que são Λ-preservante e.r.a L_2 (CURY et al., 2013):

$$P_\Lambda(L_1, L_2) = \{K \subseteq L_1 \mid K \text{ é } \Lambda\text{-preservante e.r.a } L_2\} \tag{3.13}$$

O elemento supremo deste conjunto é denotado por $SupP_\Lambda(L_1, L_2)$ (CURY et al., 2013).

Em CURY et al. (2013) é demonstrado que $L(G_d)$ é Σ-preservante e.r.a $L(G_{aj})$.

A síntese de supervisores a partir de plantas aproximadas segue o mesmo método para soluções distinguidas e clássica, em que se modela a planta e especificações e obtém-se a máxima linguagem controlável da especificação global. O supervisor $L_m(Z_a) = supC(K_a, G_a)$ obtido desta forma porém não é solução para o sistema físico, que é modelado pela planta distinguida.

Teorema 3.2. *(Supervisores aproximados como solução do PCS-D (CURY et al., 2013)) Dado G_d e E_d como no PCS-D e G_{aj} uma aproximação para G_d, para $K_{aj} = E_d \cap L_m(G_{aj})$, se $supC(K_{aj}, G_{aj})$ e L_D são não conflitantes, um supervisor S_{aj} que implementa $supC(K_{aj}, G_{aj}) \cap L_D$ é uma solução para o PCS-D. Além disso:*

$$supC(K_{aj}, G_{aj}) \cap L_D \subseteq supC(K_d, G_d) \subseteq sup\Sigma_u(K_{aj}, G_{aj}) \cap L_D. \tag{3.14}$$

Em que $sup\Sigma_u(K_{aj}, G_{aj}) = SupP_\Lambda(K_{aj}, G_{aj})$, para $\Lambda = \Sigma_u$.

A ação do supervisor cuja linguagem é S_{aj} sobre a planta G_d é realizada ao se implementar concorrentemente o autômato para $supC(K_{aj}, G_{aj})$ e o autômato H na planta G_{aj}. Para cada evento gerado pela planta G_{aj}, o autômato H procede à distinção de qual instância de fato ocorre e o supervisor Z_{aj} então exerce a ação de controle correspondente, ao gerar a entrada de controle com os eventos permitidos a seguir.

A condição de não conflito entre L_D e $L_m(Z_{aj})$ é fundamental pois caso contrário a ação paralela dos dois autômatos sobre a planta pode levar o sistema a uma situação de bloqueio.

O corolário 1 a seguir estabelece uma condição suficiente, mas não necessária, para assegurar que S_a seja tão permissivo quanto o supervisor ótimo.

Corolário 1 (CURY et al., 2013): Para $supC(K_{aj}, G_{aj})$ não conflitante com a linguagem L_D do distinguidor preditivo, tem-se que:

$$supC(K_{aj}, G_{aj}) = sup\Sigma_u(K_{aj}, G_{aj}) \Rightarrow supC(K_d, G_d) = supC(K_{aj}, G_{aj}) \cap L_D. \quad (3.15)$$

Uma vez mais utiliza-se o exemplo do sistema de manufatura com retrabalho, desta vez para ilustrar o ganho obtido com as aproximações. Para este exemplo, utiliza-se a notação $\Pi^{-1}(G)$ para denotar que os eventos da planta G foram refinados, de modo que $L(\Pi^{-1}(G)) = \Pi^{-1}(L(G))$ e $L_m(\Pi^{-1}(G)) = \Pi^{-1}(L_m(G))$.

A tabela 3.2 contém os resultados para as sínteses de supervisores obtidos a partir de diversas aproximações da planta. Para $\varsigma \in \{0, t, r, w\}$ o índice do módulo distinguidor usado na aproximação, foram obtidos $G_\varsigma = \Pi^{-1}(G) \| H_\varsigma \| H_{\Delta^*}$, $R_\varsigma = E_d \| G_\varsigma$, $L_m(R_\varsigma) = K_\varsigma$ e $L_m(Z_\varsigma) = SupC(K_\varsigma, G_\varsigma)$. Os resultados para síntese com a planta distinguida $G_d = \Pi^{-1}(G) \| H$ são reapresentados na última linha para fins de comparação.

TAB. 3.2: Resultados da síntese de supervisores para sistema exemplo utilizando aproximações

módulo	G_ς	R_ς	Z_ς	$Z_\varsigma \| H$
0	72	4320	540	468
Ht	216	7200	672	468
Hr	120	7200	516	468
Hz	288	9360	576	468
Hw	180	10800	870	492
H	2736	51960	492	492

A primeira linha mostra os resultados para a aproximação $G_0 = \Pi^{-1}(G)$, em que

nenhum módulo do distinguidor H é utilizado e a linguagem $L(Z_0)$ do supervisor obtido é não conflitante com L_D. Apesar do supervisor não ambíguo $S_0 = Z_0 \| H$ ser mais restritivo que Z_d, o número de estados para a planta aproximada é 38 vezes menor e do autômato da especificação global é 12 vezes menor, o que implica uma grande diminuição da complexidade computacional. As aproximações $G_t = \Pi^{-1}(G) \| Ht \| H_{\Delta^*}$, $G_r = \Pi^{-1}(G) \| Hr \| H_{\Delta^*}$, $G_z = \Pi^{-1}(G) \| Hz \| H_{\Delta^*}$, levam ao mesmo supervisor $S_0 = Z_0 \| H$, embora com reduções inferiores na complexidade computacional.

Para a aproximação $G_w = \Pi^{-1}(G) \| Hw \| H_{\Delta^*}$, verifica-se que com uma planta cujo número de estados é 15 vezes menor e uma especificação global quase 5 vezes menor, sintetiza-se um supervisor que, implementado concorrentemente ao autômato do distinguidor, é tão permissivo quanto o suprevisor obtido da planta distinguida. De fato, as linguagens de $S_w = L_m(Z_w \| H)$ e $L_m(Z_d)$ são isomorfas, ou seja, iguais. Além disso, $SupC(K_w, G_w) = Sup\Sigma_u(K_w, G_w)$, o que confirma que $S_w = L_m(Z_w \| H)$ é solução do PCS-D.

A explicação para tal comportamento destes supervisores aproximados deriva da informação agregada pela distinção do refinamento *w3*, fim de operação em M2 do terceiro ciclo de retrabalho para uma mesma peça. Com a informação indisponível, o supervisor proibe que novas peças entrem na linha de manufatura para impedir que haja um eventual bloqueio do sistema ocasionado, por exemplo, quando o *buffer* B1 recebe uma peça de M1 e a unidade de testes TU precisa encaminhar uma peça para o mesmo buffer. Com esta informação, o supervisor pode permitir a entrada de uma nova peça no sistema, além da que estava sendo trabalhada, pois a peça em questão não pode mais ser retrabalhada e consequentemente encaminhada para o *buffer* B1.

3.4 CONCLUSÃO

A síntese de supervisores para SED com uso de distinguidores se mostrou vantajosa ao simplificar a tarefa de modelar as restrições para o sistema e ao reduzir a complexidade desta síntese por meio de aproximações.

Os supervisores obtidos a partir da síntese com plantas aproximadas podem ser mais restritivos que aqueles obtidos da planta distinguida. Esse comportamento mais restritivo, embora indesejável do ponto de vista da flexibilidade do sistema, pode ser relevado.

Para sistemas em que a composição das plantas dos subsistema leva a uma explosão combinatória de estados, a síntese aproximada pode ser a única viável computacional-

mente. As aproximações possíveis para um mesmo sistema são inúmeras e cada uma pode levar a um supervisor diferente, com diferentes graus de permissividade. O exemplo que ilustra a aplicação da síntese aproximada mostra este resultado.

A exploração de aproximações distintas para sistemas de grande porte em busca de uma solução tão permissiva quanto possível é então a resposta para encontrar a melhor solução de controle disponível. O espaço de busca no entanto é deveras extenso e uma inspeção manual é impraticável. O próximo capítulo propõe métodos para uma exploração baseada em métodos heurísticos, que otimiza a busca por soluções neste espaço.

4 METODOLOGIA PARA A OBTENÇÃO DE SUPERVISORES NA TCS-D COM APROXIMAÇÕES

Este capítulo apresenta os trabalhos desenvolvidos para implementação de um algoritmo que realize a busca no espaço de supervisores aproximados.

O capítulo anterior abordou a Teoria de Controle Supervisório Distinguido, apresentando suas vantagens em relação à TCS original proposta por RAMADGE e WONHAM (1987), como a simplificação da modelagem das especificações e redução da complexidade computacional da síntese de supervisores valendo-se da aproximação da linguagem do distinguidor. No entanto, o espaço formado por estas aproximações é vasto e a tarefa de encontrar, caso exista, uma aproximação que leve a um supervisor tão permissivo quanto o supervisor ótimo pode se tornar árdua e extenuante. Para solucionar este problema, o presente trabalho sugere a utilização de busca heurística para explorar o espaço formado por estas aproximações.

A seção 4.1 a seguir mostra os benefícios da busca heurística, aprofundando-se em dois tipos de busca: a Busca Tabu e a busca baseada em Algoritmos Genéticos. Também apresenta os algoritmos criados especificamente para a busca por supervisores aproximados.

A busca heurística, no entanto, para promover a melhora contínua das soluções e explorar o espaço em direção a ótimos locais e globais, necessita que sejam atribuídos índices às soluções. A seção 4.2 aborda então uma medida de linguagem que é empregada para este fim. Nesta seção são apresentados a definição da medida de linguagem, as formas de obtenção e o algoritmo para cômputo da mesma. Como contribuição a esta área, são feitas adaptações para aplicação ao TCS-D.

O presente capítulo inclui ainda mais duas seções. A seção 4.3 expõe a modularização do distinguidor por instâncias e as vantagens agregadas e, por fim, a seção 4.4 mostra os estudos realizados na tentativa de resolver o potencial conflito entre as linguagens dos supervisores aproximados e do distinguidor.

Os algoritmos desenvolvidos foram implementados em Matlab versão 7.9 para Windows 7 64 bits. A ferramenta Grail[6] foi empregada para operações auxiliares, como a composição síncrona, a minimização e o produto de autômatos. O algoritmo do Grail para

[6]A ferramenta Grail pode ser obtida em http://www.das.ufsc.br/ cury/grail.html .

a síntese de supervisores foi reimplementada em C++ para o compilador gcc versão 4.6.2, assim como o algoritmo apresentado em (CURY et al., 2013) para obtenção do supremo $sup\Sigma_u(L_1, L2)$, citado na seção 3.3. O software IDES versão 3b1[7] foi utilizado para criação gráfica dos autômatos, que foram exportados para o formato JPEG, para utilização como figuras neste trabalho, e para o formato aceito pelo Grail.

4.1 BUSCA HEURÍSTICA

O objetivo deste trabalho é propor um método para se obter um supervisor não bloqueante tão permissivo quanto possível quando há severas restrições para a síntese de um supervisor exato para um sistema. Essa restrição é imposta pelo grande número de estados que o modelo para um sistema de grande porte e composto produz. Uma solução para supervisão desses sistemas nesse casos é sintetizar um supervisor a partir de aproximações da planta, aqui obtidas com auxílio dos distinguidores não preditivos, capítulo 3.

Esses supervisores assim obtidos podem ser mais restritivos que o supervisor sintetizado com o modelo completo da planta, isto é, com um distinguidor preditivo. Dentro do espaço formado pelos supervisores aproximados, alguns podem ser mais ou menos permissivos do que outros. Para uma melhor solução de controle, o ideal seria encontrar dentre os supervisores aproximados possíveis de serem sintetizados, aquele que é o mais permissivo. Essa tarefa contudo exige que se construam todas as soluções possíveis e se analise cada uma delas, ou seja, deve-se realizar uma busca exaustiva.

Para um sistema em que haja m módulos possíveis para a construção de um distinguidor, haverá 2^m combinações para a formação de um distingindor, o que significa 2^m supervisores aproximados. Para um sistema com dez distinguidores isso implicaria um espaço de 1024 soluções. Tais problemas são chamados problemas combinatórios e a otimização de soluções deste tipo de problema é chamada otimização combinatória (PARDALOS et al., 2011). A introdução de heurísticas é então aconselhável.

No presente contexto, heurísticas são estratégias para guiar o processo de busca e que visam explorar eficientemente o espaço de busca para encontrar soluções quasi-ótimas, geralmente empregando métodos não deterministas. Seu objetivo é melhorar iterativamente uma candidata a solução até encontrar soluções que satisfaçam critérios mínimos de aceitabilidade, possivelmente sacrificando exatidão em favor de um menor

[7] O software IDES pode ser obtido em https://qshare.queensu.ca/Users01/rudie/www/software.html .

esforço computacional. As heurísticas englobam desde algoritmos para obtenção de um ótimo local até sofisticados mecanismos de aprendizagem por inteligência artificial (BROWNLEE, 2011). Para tanto, necessitam que sejam atribuídos índices às soluções, cuja função é informar a heurística como prosseguir na exploração do espaço, indicando se estão convergindo ou não para um ótimo.

Ao confrontar a natureza do problema aqui tratado com as várias heurísticas existentes, se sobressaem os Algoritmos Genéticos e a Busca Tabu, cada uma exposta nas subseções seguintes. Essas heurísticas são embutidas no algoritmo principal de busca, explorando cada uma a seu modo, o espaço de soluções do problema.

O algoritmo principal, algoritmo 4.1, começa com a função *Inicializa* que inicializa o *sup_ otimo* com a solução vazia e determina o número de iterações da busca, em que a variável *sup_ otimo* representa a melhor solução encontrada pelo algoritmo. A seguir, a cada iteração um supervisor é retornado como ótimo local pela função *Busca heurística*, cujos algoritmos são detalhados nas subseções seguintes. Caso o supervisor retornado seja conflitante com a linguagem L_D do distinguidor, o algoritmo passa direto para a próxima iteração, pois esse supervisor não constitui solução para o PCS-D, permanecendo a solução corrente.

O supervisor é então testado pela função *Compara* ao se comparar sua linguagem a de $sup_{\Sigma_u}(K_{ap}, G_{ap})$, seção 3.3, para a mesma planta aproximada usada na síntese do supervisor. Conforme estabelecido na seção 3.3, se as linguagens forem iguais, o supervisor aproximado implementado concorrentemente ao distinguidor preditivo é a solução maximamente permissiva. A busca é interrompida em seguida e este supervisor é retornado como solução ótima.

Por fim, se o índice do supervisor retornado pela heurística embutida for melhor que o do atual, *o sup_ otimo* é atualizado e o *loop* prossegue para a próxima iteração. Caso o índice seja igual ou pior do que o atual, o *loop* apenas prossegue. A função *Indice* é implementada pelos algoritmos 4.4 e 4.5, consonante com a teoria apresentada na seção 4.2.

Após o número determinado de iterações (*n_ iter*), a busca retorna *sup_ otimo* como o supervisor com melhor índice encontrado pelo algoritmo. Os parâmetros da busca devem ser escolhidos de forma que *sup_ otimo* corresponda ao ótimo global.

Algoritmo 4.1 - Busca por supervisores aproximados

```
Inicializa (sup_otimo, n_iter);
para i de 1 até n_iter faça
    Busca_heuristica(sup);
    se Conflito(sup, L_D) então
        próxima iteração;
    fim
    se Compara(sup, supΣ_u) então
        sup_otimo := sup;
        parar loop i;
    fim
    se Indice(sup) > Indice (sup_otimo) então
        sup_otimo := sup;
    fim
fim
retorna sup_otimo;
```

4.1.1 BUSCA BASEADA EM ALGORITMO GENÉTICO

O Algoritmo Genético (BROWNLEE, 2011; JANG et al., 1997) é inspirado na evolução das populações em termos genéticos, como a recombinação e mutação de cromossomos, genes e alelos. Ela se baseia na interpretação de que os indíviduos mais adaptados ao meio em que vivem geram mais descendentes, ou seja, disseminam mais o seu material genético nas gerações seguintes. Neste entendimento, a carga genética de dois indíviduos, os pais, é mesclada para dar origem ao material genético do filho. Esse material pode ainda sofrer mutações, em que alterações aleatórias ocorrem durante a combinação dos pais. Ao longo de várias gerações, esse processo resulta em indíviduos evoluídos e melhor adaptados ao meio.

A adaptação desta estratégia evolucionária aos algoritmos de busca se inicia por transcrever as soluções em representações que possam ser adequadamente trabalhadas pelas operações de combinação e mutação. A representação mais comum, devido a sua simplicidade, é obtida pela codificação binária. Isso porque é fácil estabelecer uma correspondência entre os atributos das soluções e uma cadeia binária. A decodificação acontece somente por ocasião do cálculo do índice de adaptabilidade.

O índice de adaptabilidade é um parâmetro qualquer estabelecido como critério para expressar a adaptabilidade da solução. Ele deve ser escolhido de forma a melhor representar a característica que se deseja maximizar (ou minimizar) ao longo da busca, na figura da evolução. Neste trabalho é utilizada a medida de linguagem, seção 4.2.

Para o algoritmo neste trabalho, escolheu-se uma cadeia binária para representação dos supervisores. Ela representa um gene com tantos alelos quanto o número de módulos de distinguidor disponíveis. Para cada alelo, ou posição na cadeia, o número 1 significa que o módulo correspondente integra a linguagem do distinguidor aproximado e o número 0 que ele não integra.

O algoritmo 4.2 foi construído para ser embutido na função *Busca heurística* do algoritmo 4.1. Ele começa pela inicialização por meio da função *Inicializa* das variáveis *n_pop* e *pop*, que são respectivamente o número de indíviduos na população e a população, formada pelas representações binárias dos supervisores. A população é inicializada com genes em que somente um alelo é igual a 1 e de forma mais diversificada possível.

Algoritmo 4.2 - Algoritmo Genético

```
Inicializa (n_pop, pop);
Calcula_adaptabilidade(pop);
para j de 1 até n_pop faça
    Seleciona_pais(pop);
    Reproduz(filhos(j));
fim
pop := filhos;
retorna Maior_adaptabilidade(filhos);
```

Em seguida os índices dos supervisores são calculados pela função *Calcula_adaptabilidade* para se obter a adaptabilidade de cada indivíduo de *pop*. Os pais então são selecionados pela função *Seleciona_pais* de forma que aqueles com maior adaptabilidade possuam maior chance de contribuírem para a formação da nova geração. Os indivíduos que representam supervisores conflitantes são penalizados para que tenham menos chances de se reproduzir. Dentro da função *Reproduz*, o gene dos pais é combinado, de forma que o filho receba uma porção dos alelos do pai e outra da mãe. O ponto limite até o qual é feita a cópia de cada pai é escolhido aleatoriamente. A literatura (BROWNLEE, 2011) sugere uma probabilidade de mutação igual a $\frac{1}{l}$, em que l é o comprimento da cadeia. Se a mutação ocorre, o alelo a ser modificado é escolhido aleatoriamente.

A distribuição utilizada para gerar as probabilidades desde algoritmo é a uniforme. O algoritmo procede então ao cálculo da adaptabilidade de cada integrante da nova geração e o mais adaptado é retornado como melhor solução pela função *Maior_adaptabilidade*. A população atual é então substituída pelos filhos. Caso uma nova iteração ocorra, esta será a população que irá servir como pais da próxima geração.

4.1.2 BUSCA TABU

A Busca Tabu manipula três tipos de memória, cada uma com uma finalidade específica. A *memória de curto prazo* é responsável por armazenar as mudanças recentes e impedir que o processo de busca as desfaça num período curto de tempo. A *memória de médio prazo* é responsável por direcionar o processo de busca para áreas mais promissoras do espaço de busca, enquanto a *memória de longo prazo* promove diversidade, possibilitando que a busca escape de ótimos locais (PARDALOS et al., 2011).

A Busca Tabu é melhor aplicada a espaços discretos e cujos elementos possam ser representados por estruturas que podem ter partes adicionadas ou removidas. Cada adição e remoção é chamada de *movimento* e resulta em um novo elemento. A exploração do espaço de busca por esta heurística então é feita ao iterativamente modificar uma solução inicial por meio de movimentos que incluem adicionar e/ou remover partes a estas estruturas.

Os movimentos realizados são armazenados em uma tabela e durante um período de tempo chamado *tempo de tabu* eles não podem ser repetidos ou desfeitos. Qualquer nova solução obtida por meio de um movimento contido nesta tabela é chamada *tabu* e não é permitida. Com isso evita-se, ao mesmo tempo, que a mesma solução seja encontrada repetidamente e que uma solução recém-criada seja descartada ao se desfazer um movimento executado há poucas iterações (GLOVER e LAGUNA, 1997).

A Busca Tabu foi implementada pelo algoritmo 4.3, também para ser usado como heurística embutida na função *Busca heurística* do algoritmo 4.1. Inicializa-se, por intermédio da função *Inicializa*, as variáveis *n_iter* com o número de iterações, *tabu_tenure* com o tempo que um movimento deve permanecer na tabela de movimentos recentes, *sup* com um supervisor aproximado por um módulo de distinguidor aleatoriamente escolhido, melhor_solução com vazio e o vetor *melhor_dist* com um vetor vazio.

Três tipos de movimento são possíveis na evolução das soluções: adicionar, remover ou substituir um distinguidor, cada um com uma probabilidade diferente de ocorrer. A cada movimento selecionado, os distinguidores envolvidos são aleatoriamente escolhidos dentre aqueles disponíveis, ou seja, pertencente ao domínio em questão e que não conste como movimento tabu. A função *Decisão_aleatória* realiza esta tarefa e sintetiza o supervisor correspondente, atribuindo-o à variável *sup*. Se isto resultar em um supervisor com índice melhor que o do atual, a solução ótima é substituída pela atual e a busca prossegue a partir dela. A função *Indice* calcula o índice dos supervisores e é implementada pelos algoritmos 4.4 e 4.5 da seção 4.2.

Quando um movimento que adicione um módulo melhorar o índice da solução, o módulo de distinguidor adicionado é considerado chave e o mesmo é adicionado ao vetor *melhor_dist*. Se o movimento que remove um módulo resultar em um supervisor com índice inferior ao atual, esse módulo também é considerado chave, pois sua retirada degenera o indíce. Esse módulo é também adicionado ao vetor *melhor_dist*.

Ao final das iterações, a função *Supervisor_melhor_dist* retorna um supervisor sintetizado a partir da aproximação construída com os módulos de distinguidor contidos em *melhor_dist*.

Algoritmo 4.3 - Busca Tabu

```
Inicializa (n_iter, tabu_tenure, melhor_dist, melhor_solução, sup);
para j de 1 até n_iter faça
    Decisão_aleatoria(sup);
    se Indice(sup) > Indice(melhor_solução) então
        melhor_solução := sup;
        se modulo adicionado então
            melhor_dist:=melhor_dist + modulo;
        fim
    fim
    se Indice(sup) < Indice(melhor_solução) então
        se modulo removido então
            melhor_dist := melhor_dist + modulo;
        fim
    fim
fim
retorna supervisor_melhor_dist();
```

4.2 MEDIDA DE LINGUAGEM

Os métodos de busca necessitam comparar as soluções encontradas em cada iteração para orientar a construção da solução e direcionar a busca para o ótimo global ou local. Essa necessidade é mantida na busca por supervisores aproximados e um meio para compará-los é mister. A permissividade de um supervisor fornece meios para uma análise qualitativa, porém não quantitava, uma vez que a permissividade não é expressa por um índice.

A relação $\subseteq$ induz um ordenamento parcial no conjunto formado pelos diversos supervisores aproximados. Se o ordenamento induzido por $\subseteq$ fosse total, para quaisquer duas linguagens $L_1, L_2 \subset \Delta^*$, ter-se-ia $L_1 \subseteq L_2$ ou $L_2 \subseteq L_1$. No entanto, há linguagens

que não estão de forma alguma contidas uma na outra e não é possível estabelecer uma ordem entre elas.

Para preencher essa lacuna, WANG e RAY (2002) introduziram o conceito de medida de linguagem. Esta medida, chamada μ induz um ordenamento total no conjunto supracitado e permite a comparação entre os diversos supervisores pelo uso do índice μ.

4.2.1 OBTENÇÃO DA MEDIDA DE LINGUAGEM

Seja o autômato $G_i = (Q, \Sigma, \delta, q_i, Q_m)$, determinista e trim, como o definido em 2.1.

Em RAY e PHOHA (2003) e WANG e RAY (2004) é definida a σ-álgebra Θ como um conjunto das partes L_k de $L(G_i)$.

A partir disto, define-se que a função $\mu : \Theta \rightarrow \Re$ é uma medida para linguagens se duas condições são satisfeitas:

$1) \mu(\emptyset) = 0$; e

$2) \mu\left(\bigcup_{k=1}^{\infty} L_k\right) = \sum_{k=1}^{\infty} \mu(L_k)$, para cada uma das partes $L_k \in \Theta$.

Em seguida foi estabelecida uma forma de se obter a medida μ. Para isso é definida a linguagem regular $L(q_i, q) \subseteq L(G_i)$ formada por todas as cadeias que partem do estado inicial q_i e terminam no estado q. Pode-se provar que $L(G_i) = \bigcup_{q \in Q} L(q_i, q)$ e $L_m(G) = \bigcup_{q \in Q_m} L(q_i, q)$.

Também é definida a função característica $\chi : Q \rightarrow [-1, 1]$, por meio da qual é atribuído um peso a cada estado de G_i, dentro das seguintes condições:

$$\chi(q) \in \begin{cases} [-1, 1] - \{0\} & \text{se } q \in Q_m \\ \{0\} & \text{se } q \notin Q_m \end{cases} \tag{4.1}$$

Na proposta de WANG e RAY (2002), divide-se o conjunto de estados marcados em *bons estados* e *maus estados*, de acordo com o impacto positivo ou negativo que imprimem quando alcançados. Para o presente trabalho, considera-se que todos os estados marcados tem o peso igual a 1, pois o objetivo não é diferenciá-los e qualquer estado marcado é considerado desejável. As cadeias que não terminam em um estado marcado não são de interesse, pois não consistem em tarefas completas e devem ter peso atribuído igual a zero.

Considerando cada estado do autômato G_i, para cada evento em Σ cuja ocorrência é definida por σ há um custo relacionado a sua ocorrência. A atribuição deste custo é

definida pelo projetista e este pode se basear em qualquer característica do sistema para defini-lo.

Como o custo de cada evento pode variar de acordo com o estado em que ocorre, o custo é função do evento e do estado em questão, em outras palavras, o custo do evento depende da sucessão de eventos que o antecedem. A função custo de evento foi concebida de forma similar à probabilidade condicional, em que o custo total dos eventos em

A função custo de evento $\widetilde{\psi} : \Sigma \times Q \to [0, 1)$ tal que, $\forall q_j \in Q$ e $\forall \sigma, \sigma_k \in \Sigma$, é definida de modo que:

$$\begin{aligned} &1)\widetilde{\psi}[\sigma_k, q_j] \equiv \widetilde{\psi}_{jk} \in [0, 1); \\ &2)\sum_{j=1}^{m} \widetilde{\psi}_{jk} < 1; \ e \\ &3)\widetilde{\psi}[\sigma_k, q_j] = 0 \text{ se } \delta(q_j, \sigma_k) \text{ não é definida.} \end{aligned} \tag{4.2}$$

A função custo de cadeia surge como uma extensão do custo de evento e utiliza a mesma notação $\widetilde{\psi}$. A função modifica o domínio e o contra-domínio para $\widetilde{\psi} : \Sigma^* \times Q \to [0, 1]$ e é definida como:

$$\begin{aligned} &1)\widetilde{\psi}[\epsilon, q_k] = 1; \ e \\ &2)\widetilde{\psi}[\sigma s, q_j] = \widetilde{\psi}[\sigma, q_j]\widetilde{\psi}[s, \delta(q_j, \sigma)]. \end{aligned} \tag{4.3}$$

A restrição de que $\widetilde{\psi}_{jk} \in [0, 1)$ garante que $\widetilde{\psi}[s, q_j]$ seja limitado.

Estabelece-se então a medida de cada cadeia pertencente a uma linguagem para $s \in L(q_i, q)$:

$$\mu(\{s\}) \equiv \widetilde{\psi}(s, q_i)\chi(q). \tag{4.4}$$

Ao se extender essa aplicação para se obter a medida de uma linguagem, cria-se a figura do custo de sublinguagem para $K \subseteq L(G)$:

$$\upsilon(K) = \sum_{s \in K} \widetilde{\psi}(s, q_i). \tag{4.5}$$

A obtenção da medida de linguagem $\mu(K)$ é então dada por:

$$\mu(K) = \sum_{j \in Q} \upsilon(L(q_i, q_j) \cap K)\chi(q_j). \tag{4.6}$$

Para os critérios de atribuição de custo de evento acima estabelecidos, a convergência da

medida μ é demonstrada em (RAY e PHOHA, 2003).

Buscando-se facilitar o cômputo numérico da medida de linguagem μ, WANG e RAY (2004) desenvolvem uma equação matricial relacionando a medida μ, a função característica χ e o custo de evento $\widetilde{\psi}$. Antes porém, define-se a função custo de transição de estado e a matriz custo de transição de estado.

A função custo de transição de estado $\psi : Q \times Q \rightarrow [0, 1)$ é o custo para que um estado evolua para um outro qualquer imediatamente adjacente ao mesmo, ou seja, alcançável em uma transição apenas. Para todo $q_j, q_k \in Q$, o custo é obtido pela soma dos custos de todos os eventos $\sigma \in \Sigma$ tal que $\delta(q_j, \sigma) = q_k$, de modo que:

$$\begin{array}{l} 1)\psi(q_k, q_j) = \psi_{jk} = \sum_{\sigma \in \Sigma : \delta(q_j, \sigma) = q_k} \widetilde{\psi}(\sigma, q_j); \ e \\ 2)\psi_{jk} = 0 \text{ se } \{\sigma \in \Sigma \mid \delta(q_j, \sigma) = q_k\} = \emptyset. \end{array} \tag{4.7}$$

As entradas da matriz custo de transição de estado Ψ são os custo de transição de estado $\psi(q_k, q_j)$ para todos os pares de estados em $Q \times Q$. Abaixo segue a matriz Ψ:

$$\Psi = \begin{pmatrix} \psi_{11} & \psi_{12} & \cdots & \psi_{1n} \\ \psi_{21} & \psi_{22} & \cdots & \psi_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \psi_{n1} & \psi_{n2} & \cdots & \psi_{nn} \end{pmatrix}$$

O cômputo matricial para a medida de linguagem é então dado por:

$$\overline{\mu} = [I - \Psi]^{-1} \overline{\chi} \tag{4.8}$$

em que $\overline{\mu} \equiv [\mu_1 \, \mu_2 \ldots \mu_n]^T$ e $\overline{\chi} \equiv [\chi_1 \, \chi_2 \ldots \chi_n]^T$. A medida desejada $\mu(L(G_i) \equiv \mu_i$ é o *i-ésimo* elemento do vetor $\overline{\mu}$, que corresponde a medida obtida para um autômato em que q_i é o estado inicial.

Em (WANG e RAY, 2004) também é apresentado um método recursivo ideal para aplicações em tempo real, enquanto a solução matricial tem melhor aplicação para análise e síntese. Neste trabalho foi adotada a forma matricial pois há uma grande oferta de algoritmos otimizados para manipulação de matrizes, o que implica melhor desempenho para obtenção da medida em comparação ao método recursivo. A ordem de complexidade computacional de ambos é $O(n^3)$, em que n é o número de estados da planta. (SURANA e RAY, 2004).

Seja então agora o autômato $G = (Q_1, \Sigma, \delta_1, q_{11}, Q_{m1})$, como definido na seção 2.1, o autômato de uma planta não supervisionada, o autômato $Z = (Q_2, \Sigma, \delta_2, q_{21}, Q_{m2})$, de um supervisor para a planta, e suas linguagens marcadas $L_m(G)$ e $L_m(Z)$, respectivamente. A linguagem da planta sob supervisão de Z é dada por $L_m(G) \cap L_m(Z)$ e o autômato que aceita esta linguagem é $G_C = G \times Z$.

Sejam também $\widetilde{\psi_1}$ e χ_1 o custo de evento e a função característica relativos à planta G e $\widetilde{\psi}$ e χ o custo de evento e a função característica relativos a G_C. SURANA e RAY (2004) mostram que $\widetilde{\psi}[\sigma, (q_{1i}, q_{2j})] = \widetilde{\psi_1}[\sigma, q_{1i}]$ e $\chi((q_{1i}, q_{2j})) = \chi_1(q_{1i})\Upsilon(q_{2j})$ em que Υ é a função índice definida como:

$$\Upsilon(q) = \begin{cases} 1 & \text{se } q \in Q_{m2} \\ 0 & \text{se } q \notin Q_{m2} \end{cases}$$

A partir deste ponto, obtém-se a matriz Ψ custo de transição de estado para $G_C = G \times Z$ e a medida μ_i da linguagem $L_m(G) \cap L_m(Z)$.

A medida de linguagem pode ser intuitivamente relacionada a quão permissivo um supervisor é. Dado que a medida leva em consideração o número de cadeias terminando em um estado marcado, quanto mais cadeias terminando em estados marcados, maior a medida será, ou, em outras palavras, menos restritivo será o supervisor.

4.2.2 LIMITAÇÃO DA MEDIDA DE LINGUAGEM

A medida de linguagem possui uma importante limitação: para uma linguagem $L_m(G)$ formada apenas por cadeias muito longas, a medida desta linguagem, e a de $L(G)$, tende a 0. Isto significa que linguagens diferentes entre si, porém compostas apenas por cadeias muito longas, terão medidas de linguagem muito próximas e dependendo da precisão, iguais.

Este fato foi observado durante a aplicação da medida de linguagem a diferentes exemplos. Isso ocorre porque a medida é obtida do somatório do custo de todas as cadeias contidas na linguagem, equação 4.6, e o custo da cadeia é o produto dos custos dos eventos que a compõem, equação 4.3. Como o custo de um evento é sempre menor que 1, equação 4.2, o custo de cadeias longas tende a zero.

Ainda, pela equação 4.3, a medidas da linguagem $L(G)$, tal que $L_m(G)$ contenha somente cadeias muito longas e a cadeia vazia ε, terá suas medidas tendendo a 1, que é o custo da cadeia vazia ε.

O exemplo a seguir ilustra essa limitação. O autômato da figura 4.1 é sucessivamente modificado de forma que a menor cadeia de $L_m(G)$ seja cada vez maior.

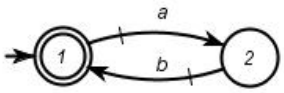


FIG. 4.1: Autômato $G1$ para exemplo de limitação da medida de linguagem.

Define-se o custo de evento $\widetilde{\psi}[a,1] = p$ e $\widetilde{\psi}[b,2] = p$, com $p < 1$. A matriz custo de transição de estado para o autômato $G1$ da figura 4.1 e vetor de pesos para os estados são:

$$\Psi_{G1} = \begin{pmatrix} 0 & p \\ p & 0 \end{pmatrix} \quad \overline{\chi}_{G1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

A medida de $L(G1)$ é então $\mu(L(G1)) = (1-p^2)^{-1}$. Por exemplo, para $p = 0,4$, $\mu(L(G1)) = 1,1905$.

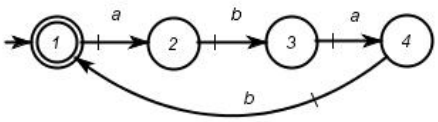


FIG. 4.2: Autômato $G2$ para exemplo de limitação da medida de linguagem.

Modificando o autômato $G1$ para o autômato $G2$ da figura 4.2, mantendo-se os custos de cada evento iguais a p, obtém-se novos vetor de pesos e matriz de custo de transição de estado:

$$\Psi_{G2} = \begin{pmatrix} 0 & p & 0 & 0 \\ 0 & 0 & p & 0 \\ 0 & 0 & 0 & p \\ p & 0 & 0 & 0 \end{pmatrix} \quad \overline{\chi}_{G2} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

A medida de $L(G2)$ calculada com χ_{G2} e Ψ_{G2} é $\mu(L(G2)) = (1-p^4)^{-1}$. Para o mesmo valor de $p = 0,4$, $\mu(L(G2)) = 1,0263$.

Seja agora o autômato $G2$ expandido para $G3$, figura 4.3, mantendo-se ainda o custo de cada evento igual aos exemplos anteriores.

Sua matriz de transição de estado correspondente e o vetor de pesos dos estados são, respectivamente:

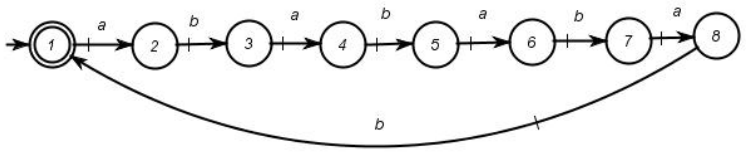


FIG. 4.3: Autômato $G3$ para exemplo de limitação da medida de linguagem.

$$\Psi_{G3} = \begin{pmatrix} 0 & p & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & p & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & p & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & p & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & p & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & p & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & p \\ p & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \overline{\chi}_{G3} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Com esses parâmetros, a medida $L(G3)$ se aproxima de 1, $\mu(L(G3)) = (1 - p^8)^{-1}$. Mais uma vez com $p = 0,4$, tem-se $\mu(L(G3)) = 1,0007$. Para um autômato que siga expandindo desta forma, a medida de sua linguagem se aproximará cada vez mais do valor 1.

4.2.3 APLICAÇÃO NO CONTROLE SUPERVISÓRIO DISTINGUIDO

Para se obter a medida de linguagem de um supervisor, utiliza-se o custo de evento atribuído à planta. Isso por dois motivos: primeiro porque é a planta que modela o comportamento livre do sistema e por isso também determina o custo de cada evento que ocorre, segundo porque é preciso estabelecer uma base comum para que se possa efetuar a comparação entre os diversos supervisores.

Em face disto, é necessário estipular a que planta em comum será referida a medida, uma vez que cada supervisor aproximado é sintetizado com base em uma aproximação diferente de G_d. A teoria apresentada no capítulo 3 mostra que se um supervisor aproximado $SupC(K_{aj}, G_{aj})$ é não conflitante com a linguagem L_D do distinguidor, então o supervisor $S_{aj} = SupC(K_{aj}, G_{aj}) \cap L_D$ é solução do PCS-D e é dirigido pelos eventos da planta exata $G_d = G_a \times H$, para $L_m(H) = L_D$ e $G_a = \Pi^{-1}(G)$. Ou seja, para $S_{aj} = SupC(K_{aj}, G_{aj}) \cap L_D$ é possível estabelecer a planta G_d como base comum na obtenção da medida.

Propõe-se aqui que a comparação entre os diversos supervisores aproximados será realizada mediante a medida obtida da linguagem $S_{aj} = SupC(K_{aj}, G_{aj}) \cap L_D$, pois assim o custo de evento utilizado para se obter a medida de cada um será igual para todos, garantindo a adequada comparação. A medida obtida para $S_{aj} = SupC(K_{aj}, G_{aj}) \cap L_D$ mede o desempenho da planta G_d sob supervisão deste supervisor.

A planta G_d, no entanto, em muitos casos possui um número muito elevado de estados, fruto da distinção dos eventos pelo distinguidor preditivo. A solução proposta pelo presente trabalho é a de se utilizar em seu lugar a planta aproximada $G_a = \Pi^{-1}(G)$. Ora, $L_m(G_d) = \Pi^{-1}(L_m(G)) \cap L_D \subseteq \Pi^{-1}(L_m(G)) = L_m(G_a)$, logo qualquer cadeia existente em $L_m(G_d)$ também existe em $L_m(G_a)$. A interseção de $L_m(G_a)$ com L_D apenas apaga de $L_m(G_a)$ algumas cadeias (eliminando a ambiguidade na ocorrência de refinamentos). Assim sendo, o cuso atribuído às cadeias em $L_m(G_a)$ não precisa ser alterado para $L_m(G_d)$. Como o custo de uma cadeia é obtido pelo produto do custo dos eventos que a compõem, tem-se que a matriz de custo de evento de G_a pode ser utilizada como base para a construção da matriz custo de evento de G_d.

Considerando o desenvolvimento da obtenção da medida de linguagem do supervisor em relação à planta, ao final da subseção 4.2.1, neste trabalho o autômato Z é tal que $L_m(Z) = SupC(K_{aj}, G_{aj}) \cap L_D$ e o autômato $G = G_a$. Ademais, se $SupC(K_{aj}, G_{aj})$ não é conflitante com a linguagem L_D do distinguidor, tem-se que:

$$\begin{aligned} &SupC(K_{aj}, G_{aj}) \cap L_D \subseteq SupC(K_d, G_d) \subseteq L_m(G_d) \subseteq L_m(G_a) \Rightarrow \\ &SupC(K_{aj}, G_{aj}) \cap L_D \cap L_m(G_a) \subseteq L_m(G_a) \Rightarrow \\ &SupC(K_{aj}, G_{aj}) \cap L_D \cap L_m(G_a) = SupC(K_{aj}, G_{aj}) \cap L_D. \end{aligned} \tag{4.9}$$

Portanto:

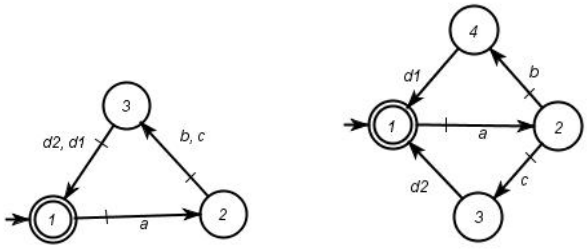
$$\mu(SupC(K_{aj}, G_{aj}) \cap L_D \cap L_m(G_a)) = \mu(SupC(K_{aj}, G_{aj}) \cap L_D). \tag{4.10}$$

A atribuição do custo de um evento pode estar correlacionada à probabilidade de sua ocorrência em um dado estado, obtida após simulações do sistema ou ensaios reais. Para este trabalho, considerou-se que todos os eventos em um dado estado são espontâneos e tem ocorrência equiprovável, independentemente de instanciarem um mesmo evento ou não, com o custo conjunto limitado a 0,8 para satisfazer os critérios de convergência, subseção 4.2.1.

Pode ser demonstrado que, para o conjunto de valores determinado para aplicação

neste trabalho, a medida de linguagem é um número real não-negativo. Desde que as linguagens $SupC(K_{aj}, G_{aj})$, para todas as plantas aproximadas G_{aj}, estãoo contidas na linguagem marcada da planta aproximada mais simples G_a, pode ser demonstrado que os valores para $\mu(SupC(K_{aj}, G_{aj}) \cap L_D)$ são limitados por $\mu(L_m(G_a))$, podendo este valor ser usado como fator de normalização das medidas.

Para ilustrar a aplicação da medida de linguagem no TCS-D, é apresentado a seguir um exemplo.



(a) G_a (b) G_d

FIG. 4.4: Plantas para o exemplo ilustrativo de medida de linguagem.

Seja a planta refinada $G_a = \Pi^{-1}(G)$, figura 4.4a, com os eventos $d1$ e $d2 \in \Delta^d$, ou seja, refinamentos de uma mesma máscara. Seja então a planta distinguida $G_d = \Pi^{-1}(G) \| H$, figura 4.4b, em que H, figura 4.5, é o distinguidor que determina que o refinamento $d1$ ocorre após o evento b e $d2$, após c.

FIG. 4.5: Distinguidor para o exemplo ilustrativo de medida de linguagem.

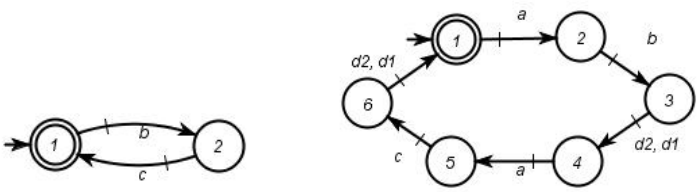
A matriz custo de evento $\widetilde{\Psi}$ de G_a, segundo os critérios adotados neste trabalho, é arbitrada como:

$$\widetilde{\Psi}_{G_a} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d1 & d2 \end{array} \\ \begin{pmatrix} 0,8 & 0 & 0 & 0 & 0 \\ 0 & 0,4 & 0,4 & 0 & 0 \\ 0 & 0 & 0 & 0,4 & 0,4 \end{pmatrix} \end{array}.$$

O vetor de pesos atribuídos aos estados de G_a é:

$\overline{\chi}_{G_a} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^T$

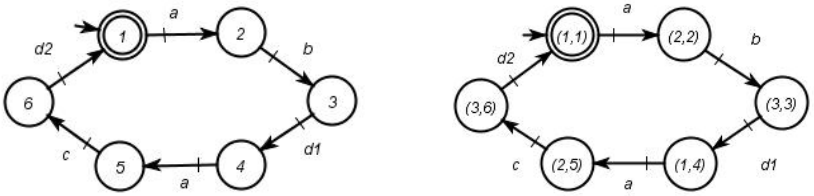
O autômato do supervisor Z_a, figura 4.6b, obtido a partir da planta aproximada G_a, é sintetizado a fim de que os eventos b e c se alternem, segundo a especificação



(a) E (b) Z_a

FIG. 4.6: Especificação e supervisor para o exemplo ilustrativo de medida de linguagem.

$E \subseteq \Delta^*$, figura 4.6a. A figura 4.7 apresenta os autômatos $Z_a \times H$ e $Z_a \times H \times G_a$. No autômato $Z_a \times H \times G_a$ os nomes dos estados são pares ordenados (a, b) em que a é o estado correspondente da planta G_a e b o estado de $Z_a \times H$.



(a) $Z_a \times H$ (b) $Z_a \times H \times G_a$

FIG. 4.7: Autômatos para obtenção da medida de linguagem do exemplo ilustrativo.

A matriz custo de transição de estado para $Za \times H$, obtida como explicado nesta seção é:

$$\Psi_{Za\times H} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 0 & 0,8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0,4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0,4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0,8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0,4 \\ 0,4 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

O vetor de pesos atribuídos aos estados de $Za \times H$ é:
$\overline{\chi}_{Za\times H} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}^T$

Ao aplicar-se a equação 4.8, a medida de linguagem para $Z_a \times H$ obtida é 1,0167.

A fim de mostrar que este é o mesmo resultado que se obtém ao considerar a planta base como G_d, a matriz de custo de transição de estado e o vetor de pesos atribuídos aos estados serão obtidas para $Za \times H$ a partir da matriz de custo de evento Ψ_{G_d}. Para isso

tem-se que:

$$\widetilde{\Psi}_{G_d} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d1 & d2 \end{array} \\ \begin{pmatrix} 0,8 & 0 & 0 & 0 & 0 \\ 0 & 0,4 & 0,4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0,4 \\ 0 & 0 & 0 & 0,4 & 0 \end{pmatrix} \end{array}$$

e

$$\overline{\chi}_{G_d} = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}^T$$

O autômato $Z_a \times H \times G_d$ é ilustrado na figura 4.8, em que os nomes dos estados são pares ordenados (a, b) em que a é o estado correspondente da planta G_d e b o estado de $Z_a \times H$.

A matriz de custo de estado $\Psi_{2\ Za\times H}$ de $Z_a \times H$ obtida a partir deste autômato e da matriz $\widetilde{\Psi}_{G_d}$ é:

$$\Psi_{2\ Za\times H} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 0 & 0,8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0,4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0,4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0,8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0,4 \\ 0,4 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

.

E o vetor $\chi_{2\ Za\times H}$ de pesos atribuídos aos estados de $Za \times H$ é:

$$\overline{\chi}_{2\ Za\times H} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}^T$$

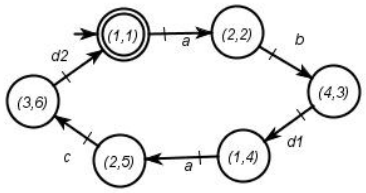


FIG. 4.8: Autômato $Z_a \times H \times G_d$ para o exemplo ilustrativo de medida de linagugem.

Como se pode notar $\Psi 2_{Za\times H} = \Psi_{Za\times H}$, o que implica que a medida de linguagem $\mu(Z_a \times H)$ obtida com a planta G_d como base é a mesma obtida para a planta G_a como base.

Para fins de comparação, as medidas de linguagem para a planta distinguida G_d e

aproximada G_a são 1,3441 e 2,0492 respectivamente.

4.2.4 ALGORITMO PARA CÔMPUTO DA MEDIDA DO SUPERVISOR

Para obter a matriz $\widetilde{\Psi}$ da planta Ga, considerou-se que, numa planta com comportamento livre, todos os eventos possuem as mesma probabilidade de ocorrer em um dado estado. Assim, a probabilidade de um evento ocorrer varia de acordo com o estado em que o sistema se encontra e de quantos eventos podem ocorrer neste estado.

O algoritmo então somente analisa estado a estado, quantos eventos podem ocorrer e atribui o custo de 0,8/n a cada um deles, em que n é o total de eventos que podem ocorrer. Aos eventos que não ocorrem no estado, é atribuída probabilidade zero. O algortimo que realiza esta tarefa é apresentado no algoritmo 4.4. Cabe ressaltar que o algoritmo trabalha com matrizes esparsas, pois na maior parte dos sistemas, uma pequena fração dos eventos ocorre em um dado estado, logo a matriz conterá diversos elementos igual a zero.

A variável *ga* é o autômato da planta aproximada G_a, *ne* é o número de estados do autômato G_a e *nl* é número de eventos contidos no alfabeto Σ. A saída do algoritmo 4.4 é $matriz\widetilde{\psi}$, variável que armazena a matriz $\widetilde{\Psi}$ de G_a.

Algoritmo 4.4 - Cômputo da matriz $\widetilde{\psi}$ de Ga

```
Inicializa (ga, ne, nl);
para i de 1 até ne faça
    nla := número de eventos ativos em ga no estado i;
    p := 0,8/nla;
    para j de 1 até nl faça
        se evento j ativo no estado i então
            matrizψ̃(i,j) := p;
        senão
            matrizψ̃(i,j) := 0;
        fim
    fim
fim
retorna matrizψ̃;
```

O algoritmo 4.5 calcula a medida de linguagem do autômato $Z_{aj} \times H$, de acordo com o estabelecido na subseção 4.2.3. Na função *Inicializa*, a variável $matriz\widetilde{\psi}$ recebe a matriz calculada pelo algoritmo 4.4, *ga* recebe o autômato G_a e *z* recebe o autômato $Z_{aj} \times H$.

A partir deste ponto, os pares de estados de z são percorridos um a um e a variável *eventos* armazena todos os eventos que produzem a transição entre o par de estados em questão. A variável *e_ga* recebe o estado correspondente na planta ao estado em questão no supervisor.

Em seguida, a função *Soma* recupera na matriz $matriz\widetilde{\psi}$ o custo dos eventos contidos na variável *eventos* para o estado correspondente da planta *e_ga* e obtém o somatório dos custos de evento, armazenando o custo de transição de estado para este par na variável $matriz\psi$ que é a matriz custo de transição de estado de $Z_{aj} \times H$.

Por fim, o algoritmo obtém o vetor $\overline{\chi}$ ao verificar a marcação de cada estado em $G_a \times Z_{aj} \times H$ e utiliza a equação 4.8 para calcular o vetor *medida*, retornando o primeiro elemento deste vetor, $medida(1)$.

Algoritmo 4.5 - Cômputo da medida de linguagem de um supervisor

```
Inicializa (matrizψ̃, ga, z);
ne := número de estados de z;
para i de 1 até ne faça
    e_ga := estado de Ga no estado i de z;
    para j de 1 até ne faça
        eventos := eventos de transição do estado i para j de z;
        se eventos>0 então
            matrizψ(i,j) := Soma(matrizψ̃(e_ga,eventos));
        senão
            matrizψ(i,j) := 0;
        fim
    fim
    se estado i de z é marcado então
        χ(i) := 1;
    senão
        χ(i) := 0;
    fim
fim
I := matriz identidade de ordem ne;
medida := (I - matrizψ)^-1 · χ;
retorna medida(1);
```

4.3 DISTINGUIDORES MODULARES POR INSTÂNCIA

O capítulo 3 apresentou o distinguidor como um mapeamento que distingue, ainda que parcialmente, as instâncias de um mesmo evento. Aos distinguidores que associam a ocorrência de um evento a uma única instância sua, deu-se o nome de distinguidor

preditivo, cuja linguagem é L_D e o autômato que marca essa linguagem é H, de tal forma que $L_D = L_m(H) = L(H)$.

No exemplo ilustrativo deste capítulo foi mostrado como obter o autômato H a partir da composição de vários módulos H_x, em que cada módulo é responsável por distinguir as instâncias de uma mesma máscara, e seu uso para obtenção de supervisores maximamente permissivos. A seguir foi introduzida a vantagem de se compor um distinguidor não preditivo, ou aproximado, H_a utilizando-se apenas alguns módulos, que implica uma síntese com menor complexidade e que pode ou não resultar em um supervisor tão permissivo quanto o obtido da síntese com um distinguidor preditivo.

Visando reduzir ainda mais a complexidade dos modelos aproximados, é proposta agora a modularização do distinguidor por módulos que, em vez de distinguirem todas as instâncias de uma máscara entre si, distinguem somente uma instância das demais. Dessa forma, cada módulo que distingue um evento refinado será formado pela composição de tantos módulos quanto forem as instâncias deste evento. Esse outro nível de modularização torna os modelos menos detalhados, com menos informação, mas dos quais ainda é possível inferir algum sentido no que tange a como o sistema é enxergado pelo supervisor.

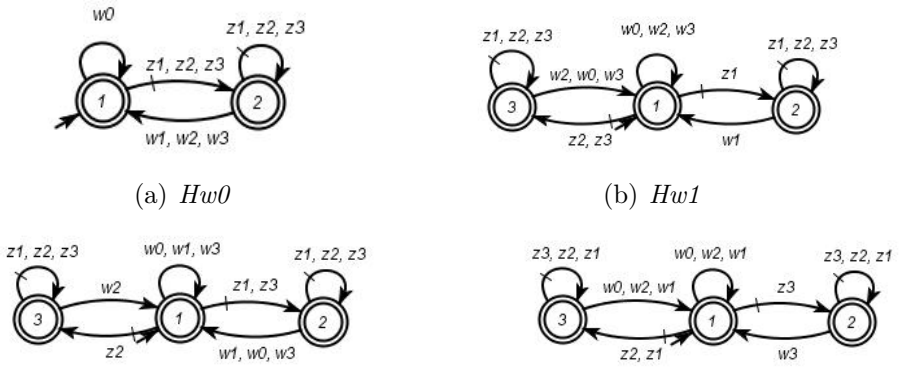
Estes módulos são os *distinguidores modulares por instância*, ou *distinguidores de instância*, e cada um deles distingue somente uma instância das demais de uma mesma máscara.

Para ilustrar o uso dos módulos por instância, emprega-se o exemplo do sistema de manufatura com retrabalho do capítulo 3, repartindo o módulo *Hw*, figura 3.8, em módulos distinguidores de instância para o evento *w*. Cada um dos módulos está exposto na figura 4.9. O módulo *Hw0* determina que a instância padrão para *w* é *w0*, ou seja, caso nenhuma das outras sequências de eventos ocorram, a instância de *w* deve ser identificada como *w0*. Para qualquer outra sequência, qualquer uma das instâncias restantes para mesma máscara pode ser correlacionada.

O módulo *Hw1*, por sua vez, determina que após a primeira ocorrência do evento *z1*, a instância correspondente de *w* deve ser identificada como *w1*. Caso *z1* não ocorra ou ocorra depois de *z2* ou *z3*, pode ser atribuída qualquer outra instância ao evento *w*. Os módulos distinguidores de instância para *w2* e *w3* seguem o mesmo modelo.

A composição síncrona dos quatro módulos apresentados na figura 4.9 resulta no autômato de *Hw*, apresentado na figura 3.8.

Dos resultados apresentados na tabela 3.2 do capítulo 3, concluiu-se que a partir da aproximação obtida somente com o módulo *Hw*, obtém-se um supervisor aproximado



(a) *Hw0* (b) *Hw1*

(c) *Hw2* (d) *Hw3*

FIG. 4.9: Módulos de instância para Hw

TAB. 4.1: Resultados da síntese utilizando módulos por instância

	G	R	Z	Z‖H
H	2736	51960	492	492
Ha	72	4320	540	468
Hw	180	10800	870	492
Hw0	108	6480	606	468
Hw1	144	8640	672	468
Hw2	144	8640	672	468
Hw3	144	8640	804	492

não conflitante com a linguagem do supervisor preditivo L_D e maximamente permissivo quando implementado concorrentemente ao distinguidor preditivo. Na tabela 4.1, apresenta-se os resultados para as sínteses aproximadas utilizando, cada uma, apenas um dos módulos *Hw0*, *Hw1*, *Hw2* e *Hw3*. Os resultados para as sínteses exata, aproximada por nenhum módulo distinguidor e aproximada pelo módulo *Hw* foram adicionados à tabela 4.1 para efeitos de comparação.

A análise dos resultados obtidos comprova que o supervisor sintetizado a partir da aproximação pelo módulo *Hw3* é não conflitante com a linguagem do distinguidor preditivo L_D e é tão permissivo quanto o supervisor ótimo quando implementado concorrentemente ao distinguidor preditivo para o sistema. Deve ser notado que todas as aproximações por módulos distinguidores de instância implicaram uma redução da complexidade da síntese, não só em relação à síntese exata, mas também em relação à síntese com a aproximação utilizando somente o módulo *Hw* e em todos os casos obteve-se um supervisor não conflitante com a linguagem do distinguidor L_D.

Uma propriedade dos módulos distinguidores por instância é a complementaridade entre eles, ou seja, se a linguagem desejada visa diferenciar qualquer grupo de instâncias

TAB. 4.2: Resultados da síntese utilizando módulos por instância

	G	R	Z	Z‖H
Hw013	180	10800	870	492

de um outro complementar em relação à mesma máscara, é indiferente utilizar um distinguidor aproximado composto pelos módulos de um grupo ou composto pelos do outro.

Seja por exemplo a máscara $\sigma \in \Sigma$ e seus refinamentos em Δ^*, $\Delta^\sigma = \{\sigma_a, \sigma_b, \sigma_c, \sigma_d\}$. Para se distinguiguir a instância σ_a das demais pode ser utilizado somente o módulo distinguidor para a instância σ_a ou alternativamente a composição dos módulos do grupo complementar $\Delta^\sigma - \{\sigma_a\} = \{\sigma_b, \sigma_c, \sigma_d\}$.

Como ilustração, seja ainda o exemplo do sistema de manufatura com retrabalho do capítulo 3 e os resultados da tabela 4.1 para as sínteses aproximadas. Os resultados para a síntese a partir da aproximação com o complemento de *Hw3*, constantes na tabela 4.2, são iguais aos resultados da síntese para a aproximação com o módulo *Hw*.

De fato, ao se compor os módulos de *Hw0*, *Hw1* e *Hw2*, todas as instâncias do evento w ficam distinguidas entre si, pois cada módulo distingue sua instância correspondente com base em sequências específicas de eventos e, por exclusão, a instância restante deve ser identificada como *w3*, necessária para a síntese do supervisor ótimo.

Outra vantagem indireta advinda desta nova modularização é ganhar um melhor entendimento do sistema, pois é possível compreender não só o evento crítico para o sistema, mas também qual a instância crítica deste evento. Entende-se por evento crítico aquele cujo refinamento provê informação fundamental para que o supervisor do sistema permita que este opere no nível ótimo e sem bloqueio. No exemplo acima, esta instância seria *w3*, pelos motivos já explicitados na seção 3.3.

4.4 CONJECTURA PARA NOVA SÍNTESE DE SUPERVISOR

Na seção 3.3 foi exposto que os supervisores aproximados $SupC(K_{aj}, G_{aj})$, obtidos a partir de plantas aproximadas $L_m(G_{aj}) = \Pi^{-1}(L_m(G)) \cap L_{Daj}$, são soluções do PCS-D somente se as linguagens do supervisor $SupC(K_{aj}, G_{aj})$ e do distinguidor preditivo L_D são não conflitantes.

A partir da inspeção de exemplos desenvolvidos durante a execução dos trabalhos referentes a esta dissertação, verificou-se que o conflito entre L_D e $SupC(K_{aj}, G_{aj})$ poderia surgir quando, em um dado estado, um refinamento de uma mascára era impedido de

ocorrer enquanto os outros refinamentos da mesma máscara permaneciam habiblitados, o que significa que a máscara em Σ continuava habilitada.

A figura 4.10 mostra um exemplo típico deste conflito. No exemplo, os eventos *d0, d1* e *d2* são refinamentos de um mesmo evento, isto é, pertencem à mesma máscara *d*. Na figura pode-se observar que o distinguidor H, cuja linguagem é dada por L_D, é responsável por eliminar de G_a o evento *d2* no estado 4, resultando na planta distinguida G_d.

Ora, a especificação determina que o evento não controlável e não pode ocorrer após duas ocorrências de *d1* o que leva ao supervisores Z_d e Z_a. Como na planta aproximada G_a ainda há o caminho que contém *d2*, o supervisor permite que se chegue ao estado 6. Porém, o distinguidor determina que o evento *d2* não ocorrerá após uma ocorrência de *d1*, fato que leva ao conflito entre as duas linguagens. O conflito é evidenciado pela diferença entre as linguagem de $\overline{SupC(K_a, G_a)} \cap L_D$ e $\overline{SupC(K_a, G_a) \cap L_D}$, que deveriam ser iguais para que $SupC(K_a, G_a)$ e L_D fossem não conflitantes. Na figura 4.10, o conflito é ilustrado pelo autômato não $Trim\ Za \times H$.

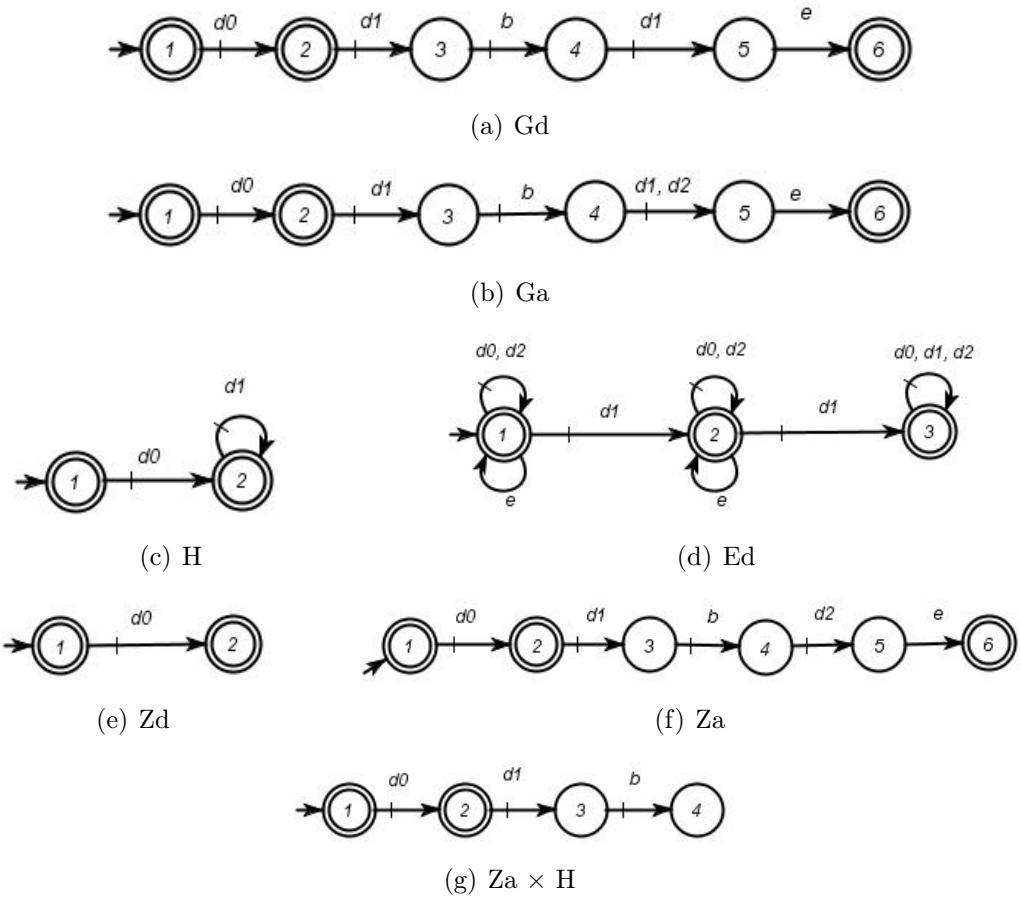


FIG. 4.10: Exemplo de bloqueio entre supervisor aproximado e distinguidor

Para que este potencial conflito entre supervisor e distinguidor fosse eliminado, foi proposto que o supervisor, além da controlabilidade e.r.a G_{aj}, possuísse uma consistência em relação a habilitação e a desabilitação de eventos de uma mesma máscara. Desta forma, propõe-se a definição de linguagem consistente em relação a um conjunto de máscaras Δ, conforme a seguir.

Definição 4.1. *(Linguagem consistente)*
Uma linguagem $K_{aj} \subseteq \Delta^*$ é consistente em relação a Δ se:
$\forall s \in \overline{K_{aj}}$ *e* $\forall \sigma \in \Sigma$,
se $\exists \delta \in \Delta^\sigma$ *tal que* $s\delta \in \overline{K_{aj}} \Rightarrow (s\Delta^\sigma \cap \overline{K_{aj}} \subseteq \overline{K_{aj}})$.

Em outras palavras, se após $s \in \overline{K_{aj}}$, K_{aj} habilita um evento $\delta \in \Delta^\sigma$, então todos os eventos refinados de Δ^σ que são possíveis em $\overline{K_{aj}}$ após s devem também ser habilitados em K_{aj}.

Deste modo, o supervisor seria obtido do supremo de um conjunto formado pelas linguagens contidas em K_{aj} que são controláveis e.r.a G_{aj} e também consistentes. Esse conjunto é definido como:

$$C'(K_{aj}, G_{aj}) = \{B \subset K_{aj} \mid B \text{ controlável e.r.a } L(G_{aj}) \text{ e } B \text{ é consistente}\}. \quad (4.11)$$

Proposição 4.1. *O conjunto $C'(K_{aj}, G_{aj})$ é fechado para união.*

Demonstração.

Seja $J_1, J_2 \in C'(K_{aj}, G_{aj})$

(1)$J_1 \cup J_2$ *é controlável e.r.a G_a, pela definição de controlabilidade.*

(2)Seja $s \in \overline{J_1 \cup J_2}$ *e* $\sigma \in \Sigma$
$s \in \overline{J_1 \cup J_2} \Rightarrow s \in \overline{J_1} \cup \overline{J_2} \Rightarrow s \in \overline{J_1}$ *ou* $s \in \overline{J_2}$
$s \in \overline{J_1}$ *e* J_1 *consistente* $\Rightarrow (s\Delta^\sigma \cup \overline{J_1} \subseteq \overline{J_1})$
$s \in \overline{J_2}$ *e* J_2 *consistente* $\Rightarrow (s\Delta^\sigma \cup \overline{J_2} \subseteq \overline{J_2})$
$(s \in \Delta^\sigma \cap \overline{J_1}) \cup (s \in \Delta^\sigma \cap \overline{J_2}) \subseteq \overline{J_1} \cup \overline{J_2}$
$s \in \Delta^\sigma \cap (\overline{J_1} \cup \overline{J_2}) \subseteq \overline{J_1} \cup \overline{J_2}$
$s \in \Delta^\sigma \cap \overline{J_1 \cup J_2} \subseteq \overline{J_1 \cup J_2}$
$J_1 \cup J_2$*é consistente*

De (1) e (2), $C'(K_{aj}, G_{aj})$ é fechado para união. □

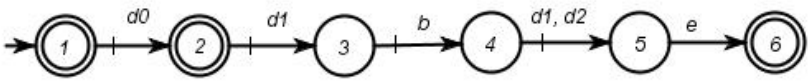
Como resultado da proposição 4.1, o conjunto $C'(K_a, G_a)$ é fechado para a união e portanto possui um elemento supremo:

$$SupC'(K_{aj}, G_{aj}) = \bigcup_{B \in C'(K_{aj}, G_{aj})} B. \tag{4.12}$$

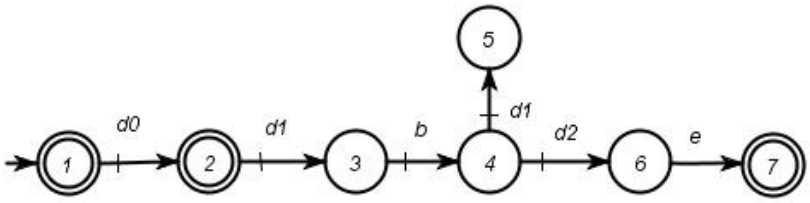
A linguagem resultante $SupC'(K_{aj}, G_{aj})$ é a máxima linguagem controlável e.r.a $L(G_a)$ e consistente contida em K_{aj}. O autômato Z'_{aj}, tal que $L_m(Z'_{aj}) = SupC'(K_{aj}, G_{aj})$, implementa um supervisor S' para a planta G_{aj}. O comportamento da planta G_{aj} em malha fechada é então dado por:

$$\begin{aligned} &L_m(S'/G) = L_m(Z'_{aj}) = SupC'(K_{aj}, G_{aj});\ e \\ &L(S'/G) = L(Z'_{aj}). \end{aligned} \tag{4.13}$$

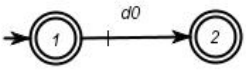
Utilizando esta nova síntese para supervisor, o mesmo exemplo é resolvido sintetizando um supervisor para a planta aproximada G_a e a especificação E. O resultado, Z'_a, mostrado na figura 4.11, é um supervisor não conflitante com a linguagem do distinguidor H.



(a) Ga



(b) Ka

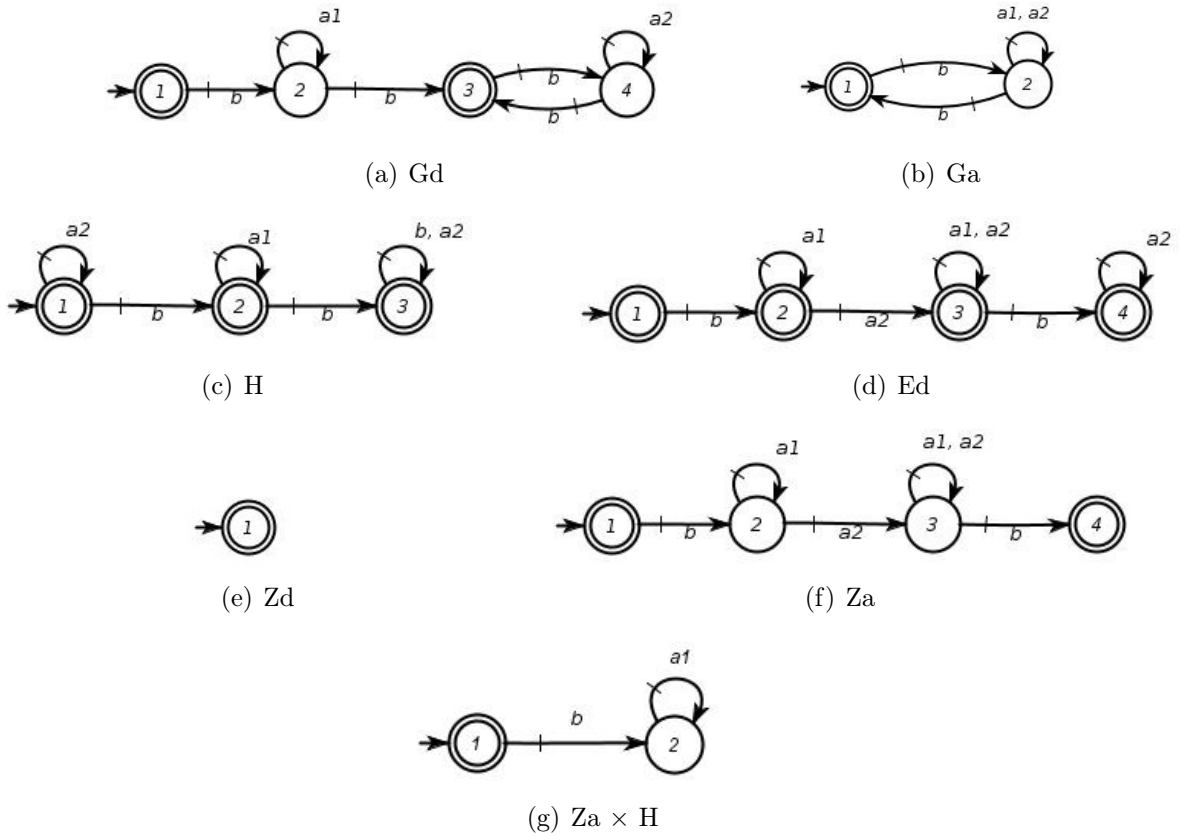


(c) Z'a

FIG. 4.11: Exemplo de supervisor não conflitante com distinguidor obtido pela nova síntese

Com o intuito de consolidar a hipótese de que o novo supervisor proposto eliminaria o potencial conflito com a linguagem do distinguidor, foram empregados esforços para provar tal hipótese. No entanto, após sucessivas tentativas, o objetivo foi frustrado pela construção de um novo exemplo, apresentado na figura 4.12.

Neste novo exemplo, os eventos *a1* e *a2* são refinamentos do evento *a*, logo pertencem à mesma máscara. O distinguidor *H* expressa que após a primeira ocorrência do evento *b*, a instância do evento *a* que ocorre é *a1* até acontecer novamente *b*, quando então todas as outras ocorrências de *a* são da instância *a2*. A especificação E condiciona a segunda ocorrência de *b* a uma ocorrência prévia da instância *a2*, fato que jamais se concretizará de acordo com o distinguidor. O resultado é o conflito entre o supervisor proposto aproximado e o distinguidor, provando que o novo supervisor não elimina o conflito em todas as situações.



(a) Gd (b) Ga

(c) H (d) Ed

(e) Zd (f) Za

(g) Za × H

FIG. 4.12: Exemplo de bloqueio entre supervisor obtido pela nova síntese e distinguidor

Este exemplo porém, traz nova perspectiva sobre possíveis propriedades da especificação que levam um supervisor a ser conflitante com o distinguidor e um estudo continuado está sendo realizado sobre este tópico.

5 RESULTADOS

5.1 SISTEMAS PARA VALIDAÇÃO

A validação do algoritmo foi realizada por meio de testes em quatro exemplos de sistemas de manufatura, um deles com mais de uma configuração. Os sistemas de manufatura foram escolhidos porque são simples de compreender e modelar e podem ser facilmente extendidos, ao se criar novas especificações tão complexas quanto o desejado.

O critério aplicado para escolha dos sistemas exigiu que eles permitessem o refinamento de eventos e que o uso de aproximações na linguagem do distinguidor resultasse em supervisores mais restritivos e/ou bloqueantes, para que se verificasse se o algoritmo tratava adequadamente estas soluções. Para garantir estas condições, os sistemas escolhidos envolvem realimentação ou restrições na sequência de produção, com execeção do sistema de manufatura concorrente, que não satisfaz as condições e foi utilizado somente para verificação do código.

Nas subeseções seguintes são descritos os sistemas, com suas respectivas modelagens e sínteses de supervisores aproximados e exatos seguidos de uma breve apreciação dos resultados. Somente os autômatos com eventos refinados são apresentados, pois permitem o entendimento sobre a modelagem e ilustram os refinamentos ao mesmo tempo.

5.1.1 SISTEMA DE MANUFATURA CONCORRENTE

O sistema de manufatura concorrente, ilustrado pela figura 5.1, envolve quatro máquinas. Duas máquinas, M1 e M2, com funcionamento paralelo, produzem, cada uma, um tipo de peça que ao final da operação são armazenadas no *buffer* B1. Este *buffer* possui capacidade para duas peças e abastece a máquina M3, uma por vez. A máquina M3 deve receber a peça na mesma ordem em que foram armazenadas, ou seja, a peça há mais tempo no *buffer* é a primeira a ser processada. As peças processadas em M3 são armazenadas no *buffer* B2, com capacidade para uma peça somente. A quarta máquina é um robô que coleta as peças do *buffer* B2 e as deposita em uma esteira de saída de duas possíveis, respeitando a regra de que peças oriundas de M1 devem ser destinadas à esteira E1, e de M2 para E2.

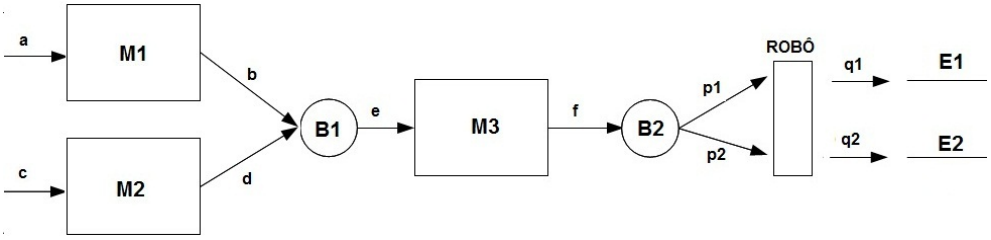
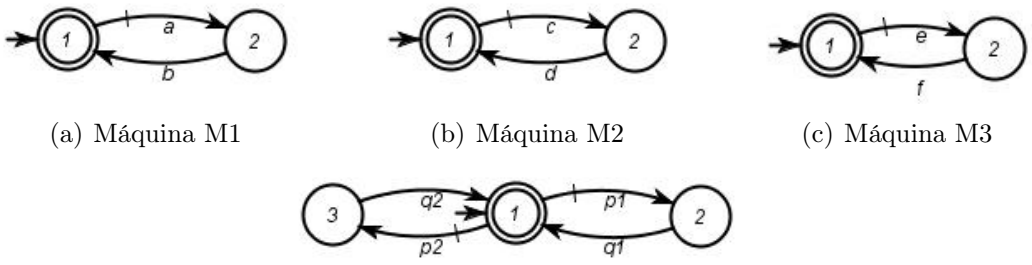


FIG. 5.1: Sistema de manufatura concorrente.

A. Plantas

Na figura 5.1, juntamente aos elementos integrantes do sistema, nota-se a distribuição e notação dos eventos associados ao funcionamento de cada máquina.

O funcionamento das máquinas M1, M2 e M3 pode ser representado por apenas dois eventos: início e término de operação. Para M1 o par é representado por a e b, assim como para M2 eles são c e d, e para M3, e e f. As plantas são apresentadas na figura 5.2. Os modelos possuem 2 estados cada, com os estados inicial e final representando máquina em repouso e máquina em funcionamento, respectivamente.



(a) Máquina M1 (b) Máquina M2 (c) Máquina M3

(d) Robô

FIG. 5.2: Plantas do sistema de manufatura concorrente.

O funcionamento do robô envolve os eventos de início e fim de operação de descarga na esteira E1, respectivamente *p1* e *q1*, e ínicio e fim de descarga na esteira E2, *p2* e *q2*. O estado inicial e final indica o robô em repouso e os outros dois indicam que ele está em movimento. O esquema de sua planta encontra-se também na figura 5.2.

A simplificação da modelagem das especificações motiva o refinamento de eventos do sistema. O refinamento proposto é ilustrado na figura 5.3.

O único elemento cujo refinamento de eventos contribuirá para a simplificação do problema é a máquina M3. Como a máquina pode operar sobre duas peças diferentes, o ideal é refinar o evento de início de operação em dois eventos distintos, ou seja, de e para *e1* e *e2*. O mesmo vale para o evento de término, que refina o f para *f1* e *f2*.

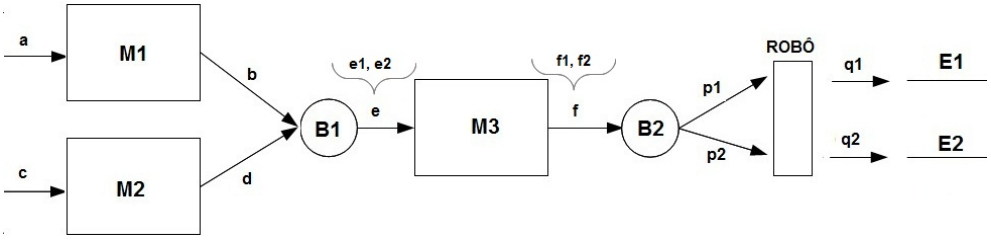


FIG. 5.3: Sistema de manufatura concorrente com eventos refinados.

Como a única planta a ser refinada é M3, apenas ela sofre alteração do seu autômato, reapresentado na figura 5.4.

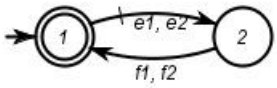


FIG. 5.4: Máquina M3 do sistema de manufatura concorrente com eventos refinados.

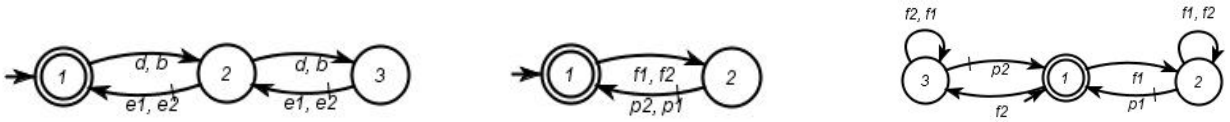
B. Especificações

As especificações para este sistema devem tratar as seguintes restrições e condições de funcionamento:

- O *buffer* B1 armazena somente duas peças;
- O *buffer* B2 armazena somente uma peça;
- A peça oriunda de M1 deve sair em E1 e a de M2 em E2;

A primeira especificação pode ser escrita como um autômato de três estados em que o primeiro indica o *buffer* vazio, o segundo, *buffer* com uma peça e o terceiro com duas. O evento que marca o fim de operação das máquinas M1 e M2 marca a transição que avança os estados do primeiro ao terceiro. A cada início de operação de M3 uma peça é retirada do *buffer* equivalendo ao retrocesso de um estado. A segunda especificação é análoga, porém tendo apenas 2 estados, vazio e com uma peça. O esquema pode ser conferido na figura 5.5.

A terceira especificação, de roteamento, deve acompanhar o funcionamento do sistema para que o robô, ao retirar uma peça de B2, seja informado de qual o destino adequado para ela.



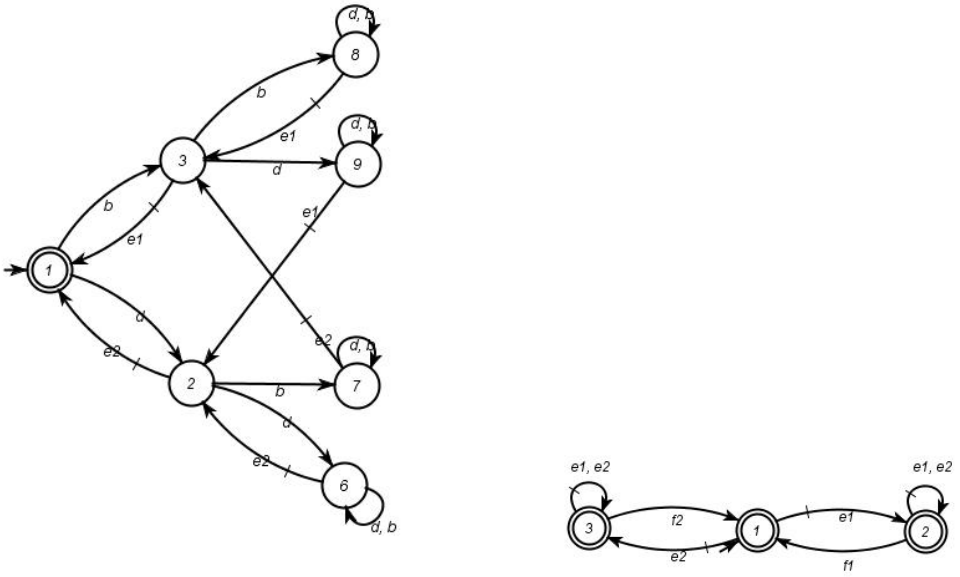
(a) *Buffer* 1 (b) *Buffer* 2 (c) Roteamento

FIG. 5.5: Especificações do sistema de manufatura concorrente.

C. Distinguidores

Em seguida, devem ser modelados os distinguidores dos eventos acima refinados. O objetivo é distinguir a funcionamento da máquina M3 para uma peça do tipo 1, que provém de M1 de uma do tipo 2, oriunda de M2. Assim, o evento para início de operação de M3 será *e1* se a peça provém de M1 e será *e2* caso contrário. O evento que marca o fim de operação será *f1* para processamento da peça tipo 1 e *f2* para peça do tipo 2.

Tem-se então os distinguidores modulares, *He* para os eventos *e1* e *e2*, e *Hf* para distinguir os eventos *f1* e *f2*, expostos na figura 5.6.Cabe ressaltar que para o distinguidor *He*, deve ser levado em consideração o funcionamento do *buffer* B1, pois a ordem em que serão armazenadas as peças influi no abastecimento de M3. Para este sistema considerou-se que o *buffer* opera como um sistema *FIFO*, do inglês *First In, First Out*, denominação para sistemas em que a primeira peça ou solicitação a entrar no sistema é a primeira a ser processada ou atendida.



(a) *He* (b) *Hf*

FIG. 5.6: Distinguidores do sistema de manufatura concorrente.

D. Síntese

Foram realizadas quatro sínteses para este sistema: aproximada sem nenhum distinguidor, aproximada pelo distingidor do evento e, aproximada pelo distinguidor do evento f e distinguido. Os resultados das síntese são apresentados na tabela 5.1. O autômato para a especificação global possui 9 estados.

TAB. 5.1: Resultados da síntese do sistema de manufatura concorrente

	H	G	R	Z	Z‖H
Ha	0	24	216	96	**210**
Hde	7	168	504	168	**210**
Hdf	3	36	324	120	**210**
Hd	21	252	756	210	**210**

A simples inspeção da tabela 5.1 é suficiente para concluir que somente o refinamento se faz necessário para implementar um supervisor ótimo para o problema proposto. Além disso, a condição do corolário 1 é válida para todas as sínteses apresentadas. Neste caso, o uso de distinguidores pode ser dispensado para a síntese do supervisor. Isso pode ser explicado pela natureza do problema, que relaciona de forma simples apenas a sucessão dos eventos.

5.1.2 SISTEMA DE MANUFATURA COM RETRABALHO

O sistema de manufatura com retrabalho, figura 5.7, foi retirado de TEIXEIRA (2011.). Este sistema de manufatura é composto por duas máquinas de produção, M1 e M2, uma unidade de teste, MT, um robô, RB e quatro *buffers* de capacidade unitária. Nesta linha de produção a máquina M1 produz uma peça e a entrega ao *buffer* B1, que a armazena até ser solicitada para tratamento pela máquina M2. Ao terminar, a máquina M2 deposita a peça no *buffer* B2 até que ela seja testada por MT.

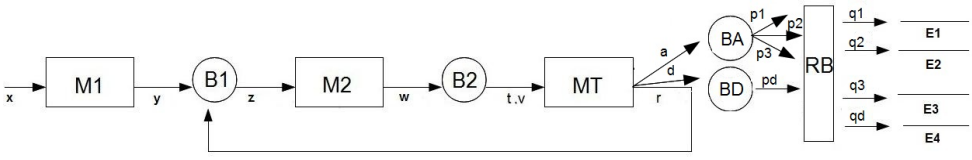


FIG. 5.7: Sistema de manufatura com retrabalho.

A verificação da peça pode exigir o seu retrabalho em M2, pode determinar o descarte da peça ou aprová-la. No caso de retrabalho a peça é encaminhada ao *buffer* B2 para futuro tratamento em M2 e em caso de aprovação ou descarte, segue para o *buffer* BA e

o *buffer* BD, respectivamente, de RB a recolherá e encaminhará para uma das esteiras de saída. É desejado que uma mesma peça seja retrabalhada no máximo duas vezes, após o que deve ser ou aprovada ou descartada.

A. Plantas

O sistema com eventos refinados, está ilustrado na figura 5.8. A modelagem de cada subsistema, a seguir, descreve também os refinamentos de seus respectivos eventos.

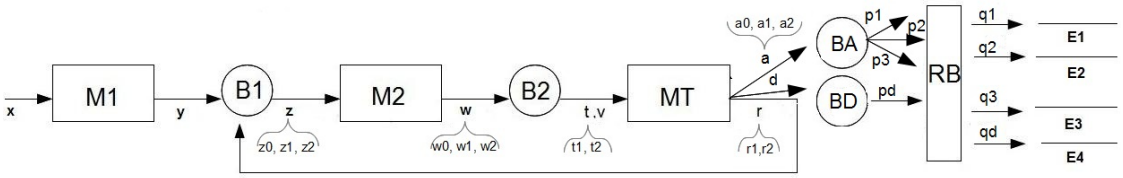


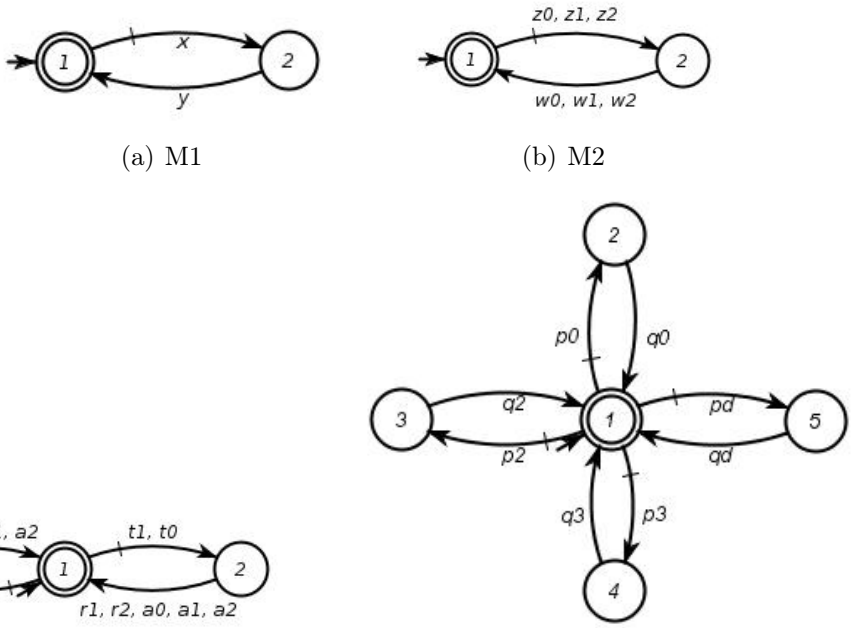
FIG. 5.8: Sistema de manufatura com retrabalho com eventos refinados.

O início de operação da máquina M1 é indicado pelo evento x e o término pelo evento y. Um autômato de dois estados representa os estados de máquina ociosa e máquina em funcionamento. O funcionamento de M2 é análogo, exceto que os eventos de início e término de operação são refinados para indicar o número de vezes que a peça tratada foi indicada para retrabalho. Os índices *0,1* e *2* fazem esta indicação nos eventos z e w de início e fim de operação de M2.

As plantas refinadas para os elementos do sistema são apresentadas na figura 5.9.

Para a máquina de teste, MT, dois tipos de operação são possíveis: teste e validação. Esta distinção se faz necessária para que ao fim de um segundo retrabalho, a peça somente possa ser aprovada ou descartada e não mais retrabalhada. Quando a operação de MT tratar-se de um teste, o evento correspondente é t e para validação é v. O teste pode ocorrer duas vezes e por isso ele é refinado em *t0* e *t1*, de acordo com o número de vezes que a peça foi retrabalhada. Como resultado do teste, a peça pode ser aprovada sem retrabalho, evento *a0*, com um retrabalho, *a1*, necessitar retrabalho pela primeira vez, *r1* ou pela segunda vez, *r2*. Caso a operação de MT seja a validação, os resultados podem ser peça aprovada depois de dois retrabalhos, representado pelo evento *a2* ou descartada, evento *d*.

O robô RB opera de quatro formas, cada uma representando o início e fim de movimento de transporte de uma peça de B3 a uma das esteiras. Os eventos de início e fim são p e q, respectivamente, seguidos do índice da esteira. O ínicio do movimento para a esteira 2 é *p2*, por exemplo.



(a) M1 (b) M2

(c) MT (d) RB

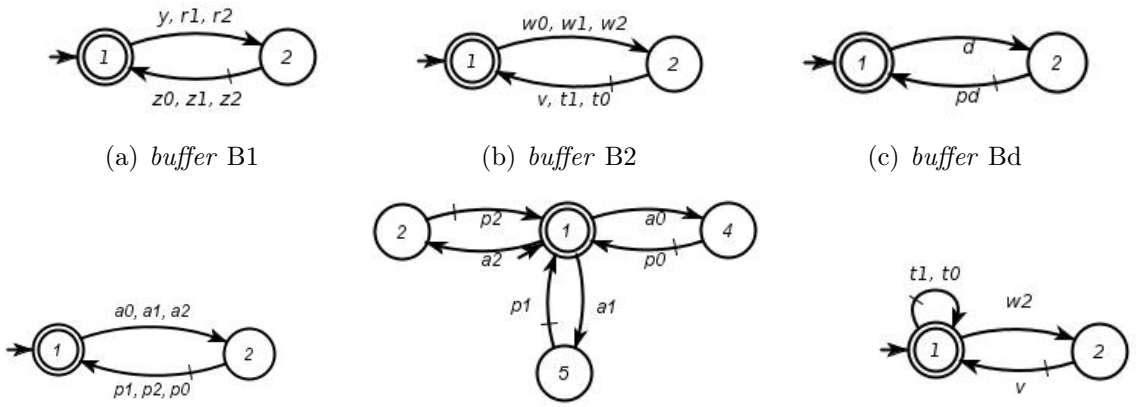
FIG. 5.9: Plantas dos elementos do sistema de manufatura com retrabalho.

B. Especificações

Para o funcionamento correto deste sistema, são necessárias seis especificações, ilustradas na figura 5.10. Quatro das especificações tratam de impedir que os *buffers* excedam sua capacidadede armazenamento e que uma peça seja retirada quando não há mais peças no *buffer*. Como todos os *buffers* do sistema têm capacidade para uma peça somente, suas configurações são iguais, diferindo apenas nos eventos que fazem o autômato transitar para o estado que representa *buffer* cheio.

O modelo das especificações e seus eventos podem ser conferidos nas figuras do *buffer* B1, B2, Bd e Ba da figura 5.10. Atenção especial deve ser dado ao *buffer* B2, pois tantos os eventos de fim de operação de M1, quanto os de exigência de retrabalho de MT, contribuem para preencher B2.

Para que as peças sejam encaminhadas corretamente a suas esteiras de saída, também se faz necessário uma especificação para o roteamento de peças. As peças aprovadas sem retrabalho, evento *a0*, devem sair pela esteira E1, relacionado ao movimento 1 de RB, ou seja, à ocorrência de *a0* deve-se seguir o evento *p1*, início do movimento 1 de RB. O mesmo ocorre para os pares *a1-p2* e *a2-p3*. A especificação do *buffer* Bd além de impedir que a capacidade de Bd seja extrapolada, também relaciona que a peça ao ser descartada, evento *d*, é direcionada à esteira E4, por meio do movimento d de RB, *pd-qd*.



(a) *buffer* B1 (b) *buffer* B2 (c) *buffer* Bd

(d) *buffer* Ba (e) roteamento (f) ciclos de retrabalho

FIG. 5.10: Especificações do sistema de manufatura com retrabalho.

A última especificação trata do número máximo de ciclos de retrabalho a que uma mesma peça pode ser submetida, no caso desejado, dois. Como uma peça só é encaminhada ao retrabalho após uma verificação em MT do tipo teste, basta que após o segundo retrabalho, o teste seja impedido e somente possa ocorrer validação. Este objetivo é atingido inibindo-se *t1* e *t2* de ocorrer após o evento *w2*, que indica que uma peça no segundo retrabalho terminou de ser produzida em M2. Este autômato encontra-se na figura 5.10 - ciclos de retrabalho.

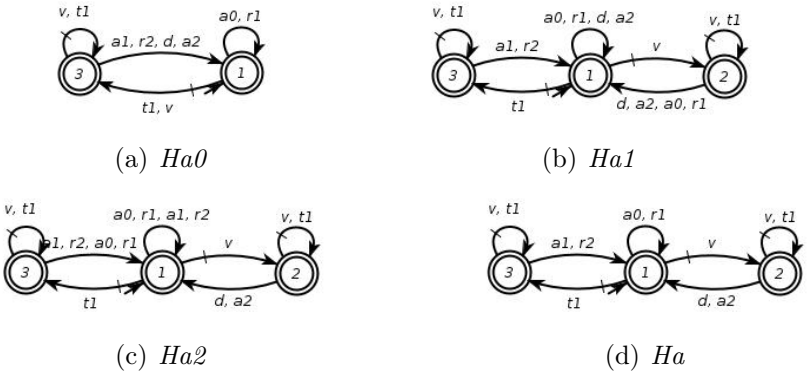
C. Distinguidores

Conforme mostrado pela figura 5.8, há 5 eventos que foram refinados com o objetivo de facilitar a modelagem das especificações. Tais refinamentos devem ser então distinguidos entre si. Tal como explicado no capítulo 3, os distinguidores podem ser modelados para distinguir somente uma instância das demais ou diferenciar todas entre si.

Para este trabalho foi modelado o distinguidor modular por instância e o distinguidor modular por evento foi obtido da composição entre os primeiros. O distinguidores modulares por instância são apresentados somente para os eventos a e r, junto com seu distinguidor por evento. Para os outros eventos, somente o distinguidor por evento é apresentado.

Os eventos a e r, da máquina de teste MT, que denotam aprovação e retrabalho foram refinados e variam de acordo com o número de vezes que a peça foi testada, por isso, o mesmo módulo de distinguidor pode ser usado para distinguir as instâncias dois eventos. Assim, quando a peça não foi testada nenhuma vez, as instâncias correspondentes são *a0* e *r1* e as outras instâncias seguem à ocorrência do segundo teste ou verificação, conforme

figura 5.11 para *Ha0*. O distinguidor de *a1* e *r2* determina que estes ocorrem somente após o segundo teste, *t1* e o distinguidor de *a2* determina que este só ocorre após a verificação, *v*. A composição dos três distinguidores resulta no distinguidor de evento *Ha* na figura 5.11.

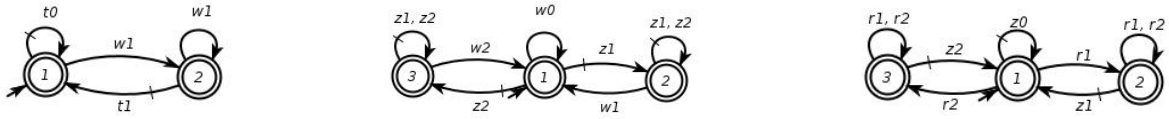


(a) *Ha0* (b) *Ha1*

(c) *Ha2* (d) *Ha*

FIG. 5.11: Distinguidores de instância e de evento dos eventos *a* e *r*.

Os distinguindores para o evento *t*,*z* e *w* podem ser conferidos na figura 5.12. Para o distinguidor do evento *t*, teste da peça em MT, tem-se que *t0* ocorre primeiro e *t1* ocorre somente após *w1*, que indica que a peça acabou de ser retrabalhada pela primeira vez e será testada pela segunda vez.



(a) *Ht* (b) *Hw* (c) *Hz*

FIG. 5.12: Distinguidores dos eventos *t*, *w* e *z*.

O distinguidor de *z*, ínicio de operação da máquina M2, determina que *z1* ocorre logo após à primeira ordem de retrabalho, *z2* após a segunda ordem para retrabalho e *z0* ocorre por padrão.

Analogamente, o distinguidor de *w* determina que sua segunda instância, *w1* ocorre após *z1*, sua segunda, após *z2* e *w0*, primeira instância ocorre por padrão.

D. Síntese

Para este sistema foram realizadas seis sínteses de supervisores, uma distinguida, cinco aproximadas por módulos distinguidores de evento e quatro aproximadas por módulos distinguindores por instância. A síntese distinguida é apresentada na tabela 5.2

na linha *Hd*. Ainda na mesma tabela, *Ha* apresenta os resultados para a síntese sem distinguidores, *Hd1* para aproximação com os distinguidores dos eventos r e a, *Hd2* para distinguidores de t, *Hd3* para w e *Hd4* para aproximação com os distinguidores de z.

As aproximações por distinguidores modulares por instância são *Hd5* para o módulo *w0*, *Hd6* para o módulo *w2*, *Hd7* para *z2* e Hd8 para *a1*. O autômato para a especificação global possui 48 estados.

TAB. 5.2: Resultados da síntese do sistema de manufatura com retrabalho

	H	G	R	Z	Z‖H
Ha	0	60	2880	370	300
Hd	50	840	16160	320	**320**
Hd1	3	80	3840	340	300
Hd2	2	120	3840	415	300
Hd3	3	120	5760	550	**320**
Hd4	3	180	5120	390	300
Hd5	2	90	4320	415	300
Hd6	3	120	5760	550	**320**
Hd7	3	180	5120	390	300
Hd8	3	80	3840	380	300

A tabela 5.2 evidencia o grande ganho, em termos computacionais, que uma síntese aproximada traz. De 16160 estados para a linguagem-alvo distinguida para 2880 da aproximada, uma redução de mais de 80%.

Foi possível verificar que os supervisores sintetizados pelas aproximações com o distinguidor modular do evento w e o módulo distinguidor de instância *w2*, quando implementados concorrentemente ao distinguidor total do sistema, levam à mesma solução de controle que o supervisor obtido da síntese distinguida, que é o supervisor mais permissivo possível, ou seja, satisfazem ao corolário 1. Os outros supervisores, embora mais restritivos, não são conflitantes com a linguagem do distinguidor, L_D.

5.1.3 SISTEMA DE EMPILHAMENTO

O sistema de empilhamento, ilustrado na figura 5.13, consiste em sobrepor, na máquina M3, duas peças de origem das máquinas M1 e M2. As máquinas M1 e M2 operam paralelamente, sendo que a primeira dá origem a peças do tipo B e a segunda a peças do tipo D. Ambas as máquinas depositam suas peças no *buffer* B1 de capacidade para duas peças e que alimenta a máquina M3, duas peças por vez. A ordem em que as peças são depositadas neste *buffer* deve ser preservada e de acordo com esta ordem quatro tipos de peça serão produzidas por M3: peças BB, BD, DB e DD. Ao fim da operação

de M3, as peças são depositadas no *buffer* B2 de capacidade unitária. O robô RB retira as peças de B2 e as destina a uma das esteiras de saída respeitando a convenção de que peças do tipo BB devem ser colocadas na esteira E1, BD em E2, DB em E3 e DD em E4.

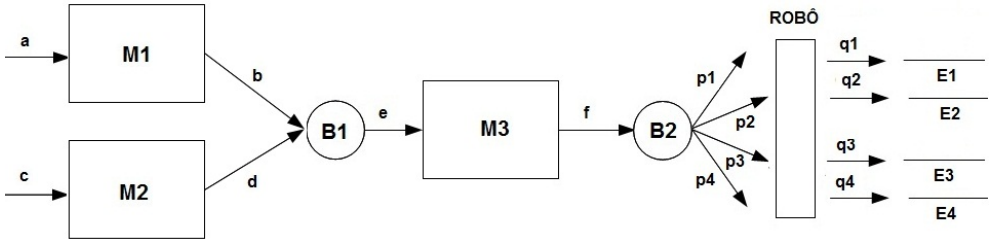
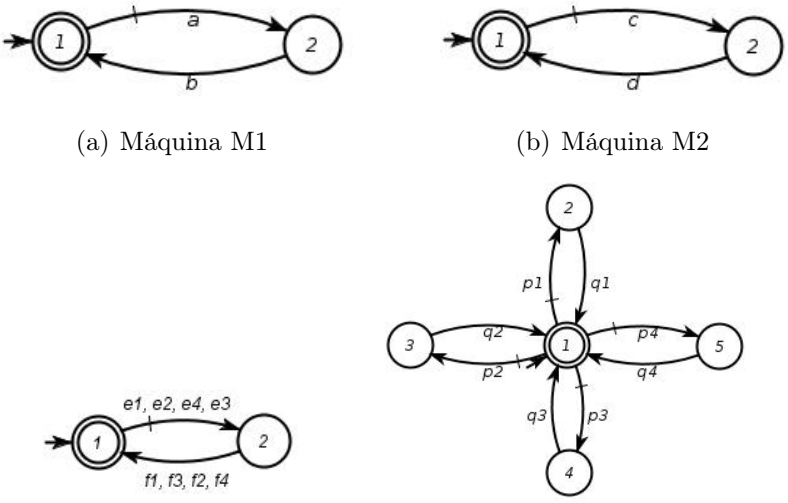


FIG. 5.13: Sistema de empilhamento.

A. Plantas

As máquinas M1, M2 e M3 podem ser modeladas segundo seus eventos de início e término de operação. Para a máquina M1 estes eventos são respectivamente a e b, para M2 são c e d e para M3, e e f. Os estados inicial e final são o estado que representa a máquina em descanso, ou seja, sem operar nenhuma peça. Ao iniciar operação o autômato passa ao estado de operação e retorna ao estado inicial quando ocorre o evento de término de operação conforme pode ser verificado para as máquina M1 e M2 na figura 5.14.



(a) Máquina M1 (b) Máquina M2

(c) Máquina M3 (d) Robô

FIG. 5.14: Plantas do sistema de empilhamento.

A tarefa de modelagem das especificações no entanto, pode ser facilitada pelo refinamento dos eventos de M3, atribuindo cada refinamento a um tipo de peça processada pela máquina. Assim o evento e será refinado em *e1, e2, e3* e *e4* e o evento f em *f1,f2,f3*

e *f4*, resultando na planta apresentada na figura 5.14. O sistema com eventos refinados é apresentado na figura 5.15. A atribuição de cada refinamento a respectiva peça será feita pelos distinguidores a serem modelados na subseção correspondente.

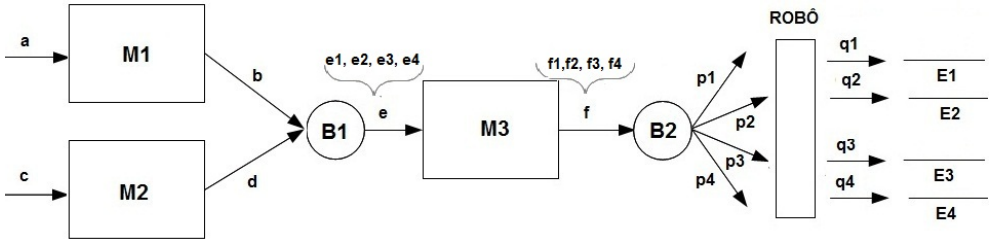


FIG. 5.15: Sistema de empilhamento com eventos refinados.

O último elemento a ser modelado é o robô RB. Este robô executa quatro movimentos diferentes que se traduzem em pegar uma peça em B2 e depositar em uma esteira de saída. Para colocar a peça em E1, o par de eventos é *p1* para pegar a peça e *q1* para colocar na esteira. O mesmo vale para as esteiras E2, E3 e E4 alterando-se os índices de p e q para 2,3 e 4 respectivamente. A planta resultante é visualizada na figura 5.14.

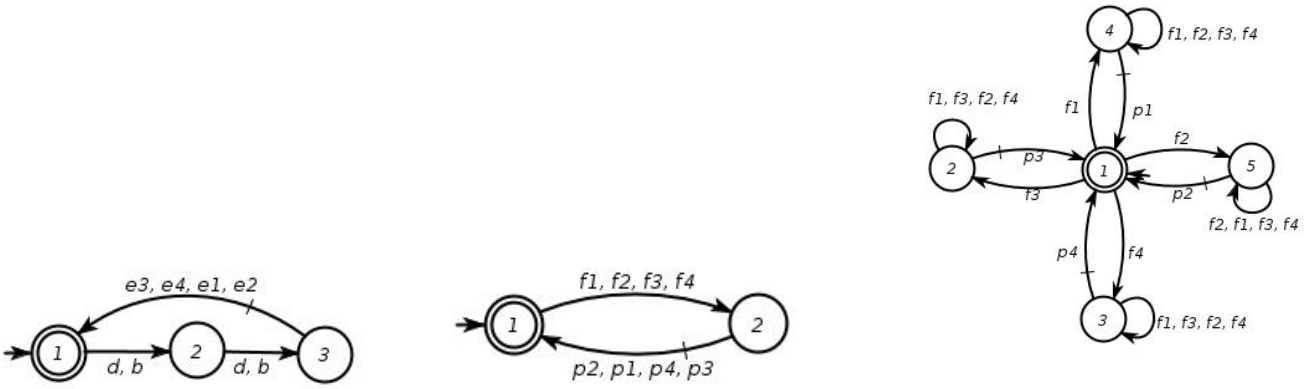
B. Especificações

As duas primeiras especificações que devem ser modeladas são as que asseguram que a capacidade dos *buffers* não seja extrapolada, ou seja, devem impedir que uma máquina ou robô despeje uma peça em um *buffer* cheio ou tente retirar uma peça de um *buffer* vazio.

Como o *buffer* B1 tem capacidade para duas peças, as máquinas M1 e M2 podem depositar juntas até duas peças para o completo preenchimento de B1. M3 ao retirar peças, duas por vez, esvazia totalmente o *buffer*. Enquanto para o *buffer* B2, RB não pode retirar uma peça até que ocorra um depósito por M3 e, até que RB retire uma peça, M3 não pode depositar outra peça. Ambas as especificações estão ilustradas na figura 5.16.

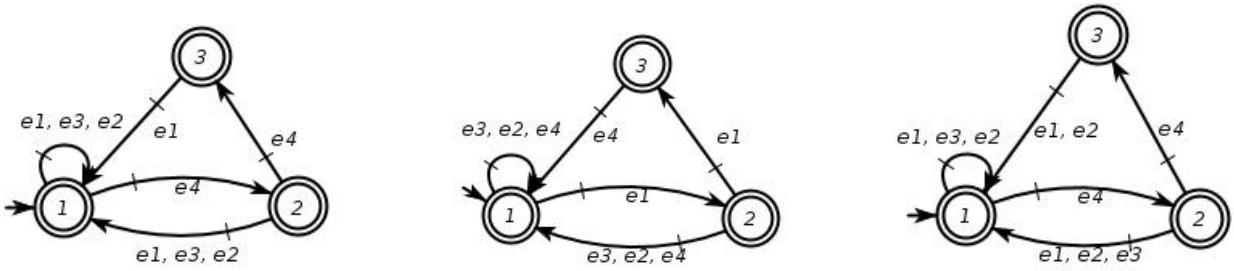
A especificação de roteamento, figura 5.16, deve determinar o destino de cada tipo de peça, assegurando que sejam despachadas pela esteira correta, conforme determinado anteriormente. Assim quando uma peça do tipo BB é produzida, evento *f1*, o robô, para depositá-la na esteira E1, deve inciar o movimento 1, ou seja, o evento *p1* deve se seguir. O mesmo é válido para os pares *f2-p2*, *f3-p3* e *f4-p4*.

Para evitar desgaste dos elementos pode ser necessário impor restrições na sequência de peças produzidas. Foram concebidos três tipos de restrição de sequência de peças. A



(a) *buffer* B1 (b) *buffer* B2 (c) roteamento

FIG. 5.16: Especificações de funcionamento do sistema de empilhamento.



(a) restrição 1 (b) restrição 2 (c) restrição 3

FIG. 5.17: Especificações de restrição de sequência de peças.

primeira, restrição 1, determina que uma vez que duas peças do tipo DD sejam produzidas sequencialmente, obrigatoriamente uma peça do tipo BB deve ser produzida. A segunda trata o mesmo problema, porém invertendo o papel das peças BB e DD.

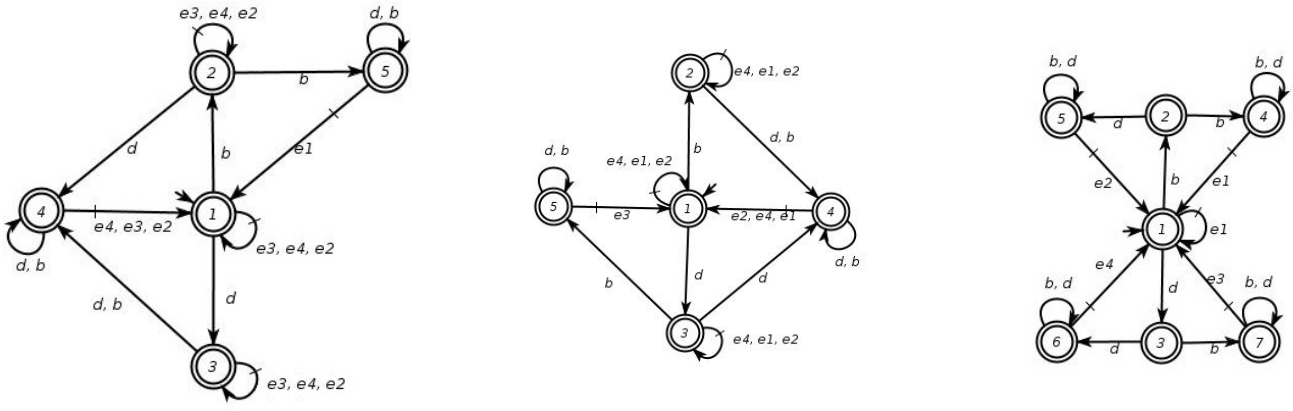
A restrição 3 determina que se duas peças do tipo DD são produzidas em sequência, então obrigatoriamente deve ser produzida uma peça do tipo BB ou BD. Os autômatos que representam estas restrições são apresentados na figura 5.17.

C. Distinguidores

O sistema possui somente um elemento com eventos refinados, a máquina M3 e, portanto, somente os eventos de M3 demandam distinguidores. Os eventos e e f, de M3, foram refinados em quatro instâncias cada e um distinguidor modular de instância foi modelado para cada uma delas. O distinguidor modular total para um evento é então obtido da composição síncrona dos módulos de suas instâncias.

O evento e, ínicio de operação de M3, somente ocorre após haver dois depósitos de

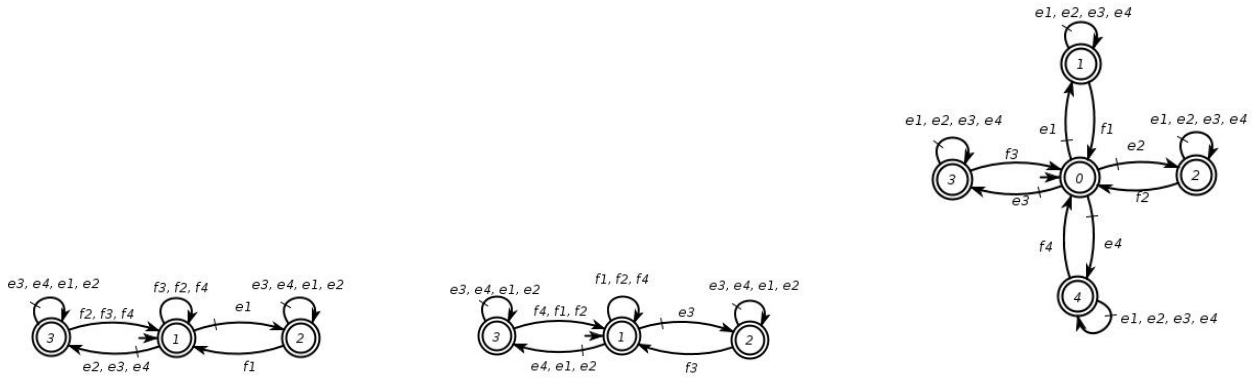
peças em B1 por M1 e M2 e a instância é determinada pelo tipo e ordem de peças retiradas do *buffer*. O distinguidor de *e1* determina que se duas peças do tipo B são depositadas, então a instância correspondente é *e1*. Para o distinguidor de *e3*, *e3* ocorre somente quando as peças D e B são depositadas nesta ordem. O mesmo para os distinguidores de *e2* e *e4*, para peças B e D e duas do tipo D, respectivamente. Os distinguidores de instância e de evento para *e* estão expostos na figura 5.18.



(a) *He1* (b) *He3* (c) *He*

FIG. 5.18: Distinguidores de instância e evento do evento e do sistema de empilhamento.

Os distinguidores de instância de f são mais simples e só determinam que uma dada instância segue a ocorrência do evento e de mesmo índice. Os distinguidores das instâncias *f1* e *f3* e o distinguidor de evento estão ilustrados na figura 5.19.



(a) *Hf1* (b) *Hf3* (c) *Hf*

FIG. 5.19: Distinguidores de instância e evento do evento f do sistema de empilhamento.

D. Síntese

Baseada nas diferentes especificações de restrição de sequência de peças, cinco sistemas diferentes foram criados. Os sistemas são modelados pelas mesmas plantas, distinguidores e especificações de roteamento e preservação de capacidade dos *buffers*, tais quais apresentados, diferindo apenas quanto a restrição de sequência de peças aplicada.

Os sistemas denominados A1, A2 e A3 utilizam, cada um, somente uma das restrições de sequência de peças da figura 5.17, respectivamente as restrições 1, 2 e 3. O sistema A4 utiliza as restrições 1 e 2 e o sistema A5 usa uma combinação das restrições 2 e 3. A tabela 5.3 mostra o número total de estados da especificação para cada um dos sistemas.

TAB. 5.3: Especificações para síntese no sistema de empilhamento

A1	A2	A3	A4	A5
45	45	45	75	75

Para cada sistema serão sintetizados supervisores obtidos a partir de diversas plantas aproximadas e as plantas são as mesmas para todos os sistemas. A tabela 5.4 mostra o número de estados para cada aproximação da planta. A planta Gde e Gdf foram aproximadas utilizando-se somente os módulos *He* e *Hf*, respectivamente. As plantas aproximadas por somente um módulo distinguidor de instância do evento e possuem o mesmo número de estados, independente da instância, e estão agrupadas na coluna indicada por Gde$\sharp$.

TAB. 5.4: Plantas aproximadas para síntese no sistema de empilhamento

Ga	Gd	Gde	Gde$\sharp$	Gdf
40	700	280	200	100

Os resultados das sínteses são apresentados nas tabelas 5.5, 5.6 e 5.7 abaixo. A tabela 5.5 mostra o número de estados para os autômatos R sendo $K = L_m(R) = L_m(G) \cap E$.

TAB. 5.5: Linguagem alvo do sistema de empilhamento

	Ra	Rd	Rde	Rde1	Rde2	Rde3	Rde4	Rdf
A1	1800	3396	3400	2600	2800	2800	2800	1524
A2	1800	3396	3400	2800	2800	2800	2600	1524
A3	1800	3464	3600	3000	3000	2800	2800	1536
A4	3000	3292	5400	4400	4600	4600	4400	1548
A5	3000	3360	5600	4800	4800	4600	4400	1560

As tabelas 5.6 e 5.7 expõem os resultados para os supervisores sintetizados e a interseção dos autômatos dos supervisores com o distinguidor exato. Pode-se verificar na tabela 5.6 que, para todos os sistemas, os supervisores obtidos a partir de aproximações com o distinguidor do evento e, quando implementados concorrentemente com o distinguidor, levam às soluções mais permissivas, ou seja, as mesmas obtidas com a planta distuinguida e satisfazem ao corolário 1.

TAB. 5.6: Supervisores do sistema de empilhamento

	Za	Za‖H	Zd	Zde	Zde‖H	Zdf	Zdf‖H
A1	720	700	**646**	990	**646**	400	700
A3	720	700	**646**	990	**646**	400	700
A3	720	700	**658**	1050	**658**	400	700
A4	1200	770	**662**	1560	**662**	440	770
A5	1200	770	**674**	1620	**674**	440	770

Para os distinguidores de instância, a mesma observação é válida para os sistemas A1 e A2, respectivamente para distinguindores de instância *e1* e *e4*, conforme pode ser observado na tabela 5.7.

Qualquer outra aproximação que não as supracitadas, resulta em supervisores conflitantes com a linguagem L_D do distinguidor e não levam a um supervisor que seja solução para o PCS-D. Estes resultados estão ressaltados na tabela pelo fundo de cor cinza da célula.

TAB. 5.7: Supervisores do sistema de empilhamento

	zd	zde1	zde1‖H	zde2	zde2‖H	zde3	zde3‖H	zde4	zde4‖H
A1	**646**	870	**646**	990	682	990	682	1020	688
A2	**646**	1020	688	990	682	990	682	870	**646**
A3	**658**	1080	700	1080	700	990	682	1020	688
A4	**662**	1530	704	1620	734	1620	734	1530	704
A5	**674**	1740	758	1710	752	1620	734	1530	704

No caso dos sistemas A3, A4 e A5, embora não explicitado na tabela 5.7, há uma combinação de módulos de distinguindores de instância de e, inferior a total, que levam a este resultado. Os módulos distinguidores de instância necessários para a síntese de um supervisor aproximado que leve a uma solução para o PCS-D para cada sistema são apresentados na tabela 5.8.

Pode-se observar que há uma estreita relação entre a especificação de restrição de sequência de peças e o módulo distinguidor necessário para uma solução ótima. Os módulos necessários correspondem exatamente às instâncias que devem ocorrer após a

sequência repetida, por exemplo as instâncias *e1* e *e2* para a sequência repetida de eventos *e4* no sistema A3.

TAB. 5.8: Módulos distinguidores de instância necessários para supervisores aproximados ótimos

A1	He1
A2	He4
A3	He1 e He2
A4	He1 e He4
A5	He1, He2 e He4

5.1.4 SISTEMA DE MANUFATURA DE TINTAS

O sistema de manufatura de tintas, figura 5.20, possui três máquinas, M1, M2 e M3, um robô, RB e três *buffers* de capacidade unitária, B1, B2 e B3. Ao iniciar-se a manufatura de uma lata de tinta em M1, é determinada a cor final da tinta, entre vermelha, azul ou roxa. Da máquina M1, as latas podem ser encaminhadas para a máquina M2 ou M3, antecedidas pelos *buffers* B1 e B2, respectivamente. A máquina M3 preenche a lata com tinta vermelha e M2 com tinta azul. Se a lata deve ser de tinta roxa, ela passar por M2 e receber tinta azul para em seguida receber tinta vermelha em M3. O robô RB é responsável por reconduzir uma lata vinda de M2 para M3. Quando uma lata é preenchida em M2 ou M3, ela segue para o *buffer* B3 de onde um robô a distribui para esteiras de saída de acordo com a cor da tinta de cada lata ou de volta a M3.

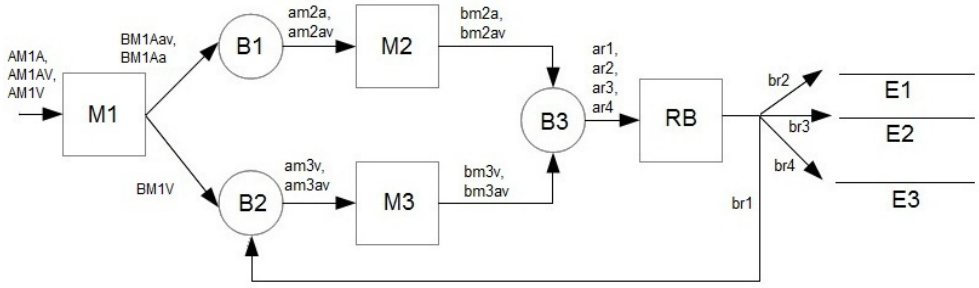


FIG. 5.20: Sistema de manufatura de tintas com eventos refinados.

A. Plantas

O funcionamento das máquinas M1, M2 e M3 consiste em início e término de operação. As plantas são apresentadas na figura 5.21. Para M2 e M3 este funcionamento é modelado por dois estados, o inicial, e também final, representando a máquina em

repouso e o outro a máquina em funcionamento. Os eventos que marcam o ínicio e término de operação são *am2* e *bm2* para M2 e *am3* e *bm3* para M3.

Cada um destes eventos é refinado em um conjunto que indica a cor de tinta que será obtida ao fim do processo. A nomenclatura dos refinamentos é o nome do evento que refina seguido do sufixo indicativo da cor da tinta, *a* para azul, *v*, para vermelha e *av* para roxa. Como a máquina M2 está envolvida no processamento das tintas azul e roxa, os refinamentos para *am2* são *am2a* e *am2av*, por exemplo.

A cor final da tinta contida em cada lata é determinada no início de operação da máquina M1. Para cada cor há um evento de início de operação de M1 correspondente. Esses eventos são expressos em letras maiúsculas para indicar que não são refinamentos de um mesmo evento. Para início de operação de uma lata que conterá tinta azul o evento é *AM1A*, para latas de tinta vermelha é *AM1V* e para roxa é *AM1AV*. O fim de operação é denotado pelos eventos *BM1V* e *BM1A*, conforme as latas devam seguir para receber tinta vermelha ou azul.

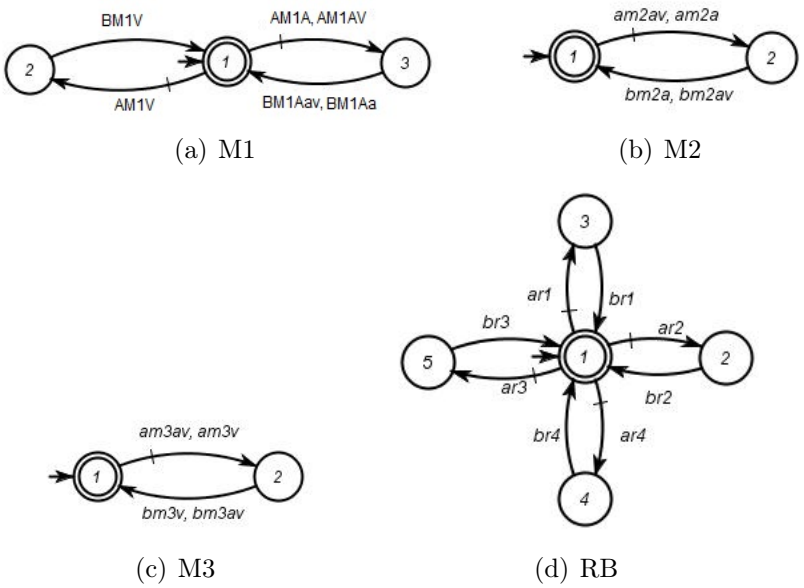
No autômato que modela M1 o estado inicial, que também é o único marcado, representa a máquina em repouso. Os eventos *AM1AV* e *AM1A* marcam a transição para um segundo estado, que representa máquina em operação para latas que seguirão para M2 e o evento *BM1A* marca o retorno ao estado inicial. A ocorrência do evento *AM1V* marca a transição para um terceiro estado, que representa máquina em operação para latas que seguirão para M3 e o evento *BM1V* marca o retorno ao estado inicial.

O evento *BM1A* é refinado em *BM1Aa* e *BM1Aav*, para diferenciar o término de processamento de latas que conterão tinta azul ou roxa.

O robô RB executa quatro tipos de movimentos, cada um compreendendo a retirada de uma peça em B3 e o depósito em uma das três esteiras de saída ou no *buffer* B3. Cada movimento é associado a um par de eventos que marcam o início e o fim do movimento. Os eventos são *ar1*, *ar2*, *ar3* e *ar4* para início do movimento e *br1*, *br2*, *br3* e *br4* para fim do movimento. Na planta de RB, figura 5.21, o estado inicial indica RB em repouso e cada um dos outros estados indica que RB está executando um movimento de transferência de peça para uma esteira.

B. Especificações

Para o *buffer* B1, o *buffer* é preenchido após os eventos de fim de operação de M1 refinados de *BM1A* e é esvaziado na ocorrência de qualquer evento de início de operação de M2. Assim, para evitar *overflow* e *underflow* do *buffer*, deve-se alternar o preenchimento



(a) M1 (b) M2

(c) M3 (d) RB

FIG. 5.21: Plantas dos elementos do sistema de manufatura de tintas.

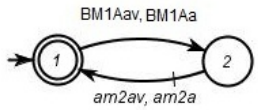
e a retirada de peças, iniciando-se pelo primeiro, e o estado inicial representa o *buffer* vazio.

O funcionamento de B2 e B3 são análogos. Para B2 os eventos de entrada de peças são o fim de operação de M1 *BM1V* e o direcionamento de peça feito por RB, *br1* e os eventos de retirada são os refinados do início de operação de M3. Para B3, a entrada de peças ocorre no fim de operação de M2 e M3 e a retitada pelo início de operação de RB.

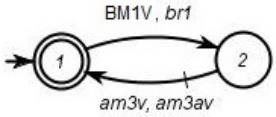
A especificação de roteamento determina o destino das peças manufaturadas por M2 e M3. Caso e peça tenha sido manufaturada por M2 e ainda deva passar por M3, evento *bm2av*, RB deve iniciar a operação por *ar1*. Caso as peças tenham por destino as esteiras de saída, as latas de tinta azul, cujo término de manufatura é marcado pela ocorrência de *bm2a*, irão para E1, o evento de início de operação de RB é *ar2*, para tinta roxa, *bm3av*, esteira E2, *ar3* e para tinta vermelha, *bm3v*, esteira E3, *ar4*.

C. Distinguidores

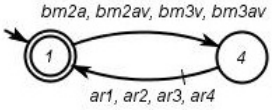
O distinguidor do evento de início de operação de M2, *am2*, determina que o evento refinado *am2a* é o padrão, e caso ocorra *BM1Aav*, o evento é distinguido em *am2av*, figura 5.23. Esta regra é seguida pelos distinguidores de *bm2, am3* e *bm3*, em que cada um tem seu evento refinado padrão e para a ocorrência de um evento específico, determina outro evento como distinção. Esses distinguidores também podem ser observados na figura 5.23.



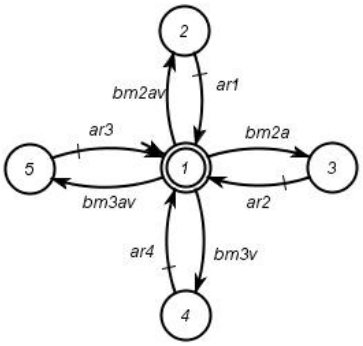
(a) *overflow/underflow* de B1



(b) *overflow/underflow* de B2



(c) *overflow/underflow* de B3



(d) Roteamento de peças pelo robô

FIG. 5.22: Especificações do sistema de manufatura de tintas.

Na máquina M1, somente o evento *BM1A* é refinado em mais de uma instância e portanto só é confeccionado um distinguidor para este evento. O padrão é a distinção pelo evento *BM1Aav*, fim de operação para tinta roxa, porém se ocorrer o evento *AM1A*, então o evento será distinguido para *BM1Aa*, fim de operação para tinta azul, figura 5.23.

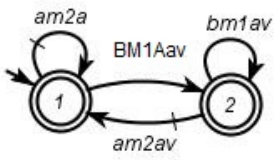
D. Síntese

Foram sintetizados oito supervisores para o sistema de manufatuira de tintas, sendo sete obtidos de plantas aproximadas. A tabela 5.9 mostra os distinguidores utilizados em cada aproximação.

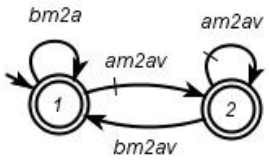
TAB. 5.9: Composição dos distinguidores aproximados para sistema de manufatura de tintas

Ha	0
Hd	ham2, ham3, hbm1, hbm2 e hbm3
Hd1	hbm1, hbm2 e hbm3
Hd2	ham2 e ham3
Hd3	ham3, hbm1 hbm3
Hd4	ham2, hbm1 e hbm2
Hd5	ham2 e hbm2
Hd6	ham3 e hbm3
Hd7	hbm2

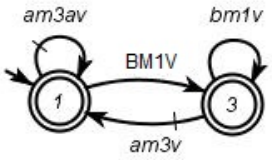
O autômato para a especificação global possui 20 estados. Os resultados das sínteses



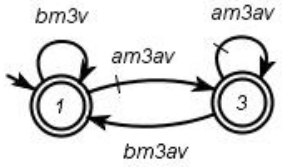
(a) início de operação M2



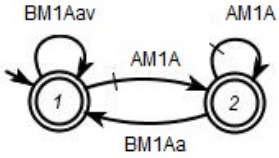
(b) fim de operação M2



(c) início de operação M3



(d) fim de operação M3



(e) fim de operação de M1

FIG. 5.23: Distinguidores dos eventos refinados do sistema de manufatura de tintas.

estão na tabela 5.10.

TAB. 5.10: Resultados da síntese para sistema de manufatura de tintas

	H	G	R	Z	Z‖H
Ha	0	60	1200	187	304
Hd	32	720	7524	397	**397**
Hd1	8	180	3600	324	**397**
Hd2	4	240	2556	260	304
Hd3	8	240	3344	220	304
Hd4	8	240	3600	367	**397**
Hd5	4	180	2700	308	**397**
Hd6	4	180	2508	174	304
Hd7	1	90	1800	225	**397**

Foi verificado que as linguagens de $Zd1\|H$, $Zd4\|H$ e $Zd5\|H$ satisfazem ao corolário 1 e são a mesma de Zd, o supervisor mais permissivo para o sistema. Como o único módulo distiguidor contido nas três aproximações é o módulo *hbm2*, foi realizada a síntese para a aproximação *Hd7* utilizando somente este módulo. O resultado mostra que este supervisor aproximado também implementa com H um supervisor maximamente permissivo e não conflitante com a linguagem do distinguidor L_D. Os outros supervisores aproximados, embora mais restritivos, não são conflitantes com a linguagem do distinguidor, L_D.

5.2 TESTES DE VALIDAÇÃO

Os testes de validação foram executados dez vezes para cada sistema. O programa que implementa o algoritmo de busca retorna como resultado o supervisor Z encontrado, sua medida de linagugem, conforme definida na seção 4.2, e os módulos distinguidores utilizados na aproximação a partir da qual o supervisor foi sintetizado.

Todos os testes e execuções do programa de busca foram realizados em uma máquina operando com Windows 7 Home Premium 64bits versão 6.1.7601, com processador AMD A6-3420M e memória DDR3 4,00GB.

5.2.1 RESULTADOS PARA BUSCA TABU

Os resultados para a busca por supervisores aproximados, utilizando a Busca Tabu como heurística, estão expostos nas tabelas 5.11, 5.12, 5.13, 5.14, 5.15, 5.16, 5.17 e 5.18, mostrando o número de soluções pesquisadas até o término do algoritmo, coluna *Zo*, e os módulos distinguidores envolvidos na síntese, coluna *H*.

A coluna intitulada *condição* expõe se o supervisor retornado é conflitante com L_D, indicado por *cflt*, se ele pode ser implementado concorrentemente a H como solução para o PCS-D, indicado por *ótimo*, ou, quando não ocorrer nenhum dos dois casos, a coluna contém a medida da linguagem computada pelos algoritmos apresentados na subseção 4.2.4. Quando o algoritmo alcançar o número máximo de iterações sem encontrar uma supervisor não conflitante, a coluna *Zo* conterá a indicação *max*.

Ao se analisar os resultados contidos nas tabelas, deve-se observar que a cada iteração o algoritmo de Busca Tabu pode adicionar um novo módulo distinguidor, retirar um existente ou trocar um módulo existente por outro aleatório. Por isso, o número de módulos distinguidores utilizados na síntese de um supervisor não é função do número de soluções que a precederam, ou seja, a aproximação para um supervisor obtido na quarta iteração pode conter mais módulos que a aproximação para um supervisor obtido na décima iteração.

A tabela 5.11 expõe os resultados para a aplicação do algoritmo para o sistema de manufatura concorrente. Todas as dez vezes que o algoritmo foi aplicado o supervisor que pode ser implementado como ótimo foi retornado. Os resultados são consistentes com os apresentados na subseção 5.1.1, de que o supervisor obtido a partir de qualquer aproximação da planta pode ser implementado como ótimo.

A tabela 5.12 expõe os resultados da aplicação do algoritmo de busca com Busca

TAB. 5.11: Resultados do algoritmo de busca com Busca Tabu para sistema de manufatura concorrente

Teste	Zo	H	condição	Teste	Zo	H	condição
1	1	2	ótimo	6	1	2	ótimo
2	1	1	ótimo	7	1	0	ótimo
3	1	2	ótimo	8	1	0	ótimo
4	1	2	ótimo	9	1	2	ótimo
5	1	1	ótimo	10	1	2	ótimo

Tabu para o sistema de manufatura com retrabalho. Todas as dez aplicações obtiveram o supervisor aproximado que pode ser implementado como solução ótima. Os distinguidores utilizados para cada síntese são consistentes com os resultados apresentados na subseção 5.1.2, em que apenas o módulo distinguidor para a instância *w2* é necessária para a síntese do supervisor que implementado é ótimo. Esse módulo distinguidor corresponde ao módulo 7 para a busca.

TAB. 5.12: Resultados do algoritmo de busca com Busca Tabu para sistema de manufatura com retrabalho

Teste	Zo	H	condição	Teste	Zo	H	condição
1	3	2,7,8	ótimo	6	8	1,2,7,8,9,10	ótimo
2	8	1,7,8,9,10	ótimo	7	13	2,3,4,5,7,9,10	ótimo
3	13	2,3,4,5,7,9,10	ótimo	8	6	1,3,4,7	ótimo
4	3	2,7,8	ótimo	9	3	2,7,8	ótimo
5	13	2,3,4,5,7,9,10	ótimo	10	8	1,2,7,8,9,10	ótimo

Os resultados para o sistema de empilhamento tipo A1 estão na tabela 5.13. Os resultados são consistentes com os apresentados na subseção 5.1.3, de que somente o módulo distinguidor de instância para *e1*, módulo 1, é necessário para a síntese do supervisor que pode ser utilizado como solução do PCS-D.

TAB. 5.13: Resultados do algoritmo de busca com Busca Tabu para sistema de empilhamento A1

Teste	Zo	H	condição	Teste	Zo	H	condição
1	1	1	ótimo	6	9	1,2,4,5,6,7	ótimo
2	9	2,3,4,6,7,8	ótimo	7	4	1,2,5	ótimo
3	1	1	ótimo	8	4	1,2	ótimo
4	3	2,3,4	ótimo	9	3	2,3,4	ótimo
5	9	1,2,4,6,7,8	ótimo	10	max	0	cflt

Para os testes de número 2 e 9, o módulo 1 é não foi necessário, pois os módulos 2, 3 e 4 estavam presentes, conduzindo à linguagem complementar do módulo 1, conforme

explicado na seção 4.3.

Para o teste de número 10 a busca não convergiu para o supervisor ótimo e como não continha o módulo 4, o supervisor sintetizado é conflitante com L_D e portanto não pode ser implementado como supervisor para o presente sistema.

Os resultados retornados para o algoritmo de busca para o sistema de empilhamento tipo A2, tabela 5.14, são supervisores que implementam a solução ótima para nove de dez testes. Conforme a subseção 5.1.3, para este sistema o módulo 4, módulo distinguidor da instância *e4*, é o único necessário para uma aproximação conduzir à solução ótima. Os testes de número 3 e 10, no entanto, também obtiveram a solução ótima sem conter o módulo 4, pois continham os módulos 1, 2 e 3, que combinados geram o complemento do módulo 4.

TAB. 5.14: Resultados do algoritmo de busca com Busca Tabu para sistema de empilhamento A2

Teste	Zo	H	condição	Teste	Zo	H	condição
1	1	4	ótimo	6	1	4	ótimo
2	2	4	ótimo	7	8	1,2,4,5,6,7	ótimo
3	6	1,2,3,8	ótimo	8	7	1,4,6,8	ótimo
4	max	0	cflt	9	5	1,3,4	ótimo
5	1	4	ótimo	10	4	1,2,3,5	ótimo

O teste de número 4 incorreu na mesma situação que o teste número 10 para o sistema anterior, não encontrando um supervisor não conflitante com a linguagem do distinuidor L_D.

Os resultados para o sistema de empilhamento tipo A3 constam na tabela 5.15. Para o sistema de empilhamento tipo A3, a solução ótima deveria ser sintetizada a partir de aproximações contendo os módulos 1 e 2, subseção 5.1.3, mas, segundo apresentado na seção 4.3, as aproximações contendo os módulos complementares 3 e 4 também levam à solução ótima, caso dos testes de número 3, 7 e 10. Os testes de número 4 e 6 não encontram um supervisor que possa ser implementado, ou seja, um supervisor que não seja bloqueante quando atuando paralelamente ao distinguidor.

Para o sistema de empilhamento tipo A4, qualquer aproximação contendo 3 módulos dentre 1, 2, 3 e 4 será solução, devido à complementaridade, exposta na seção 4.3. Além disso, as soluções aproximadas contendo os módulos 1 e 4, ou 2 e 3, também conduzirão à solução ótima, conforme resultados da subseção 5.1.3. Somente um dos dez testes retornou a solução vazia, ou seja, não foi encontrado um supervisor que, implementado com o autômato do distinguidor, conduza o sistema a um comportamento livre de estados

TAB. 5.15: Resultados do algoritmo de busca com Busca Tabu para sistema de empilhamento A3

Teste	Zo	H	condição	Teste	Zo	H	condição
1	13	1,2,3,5,6,8	ótimo	6	max	0	cflt
2	8	1,2,6,7,8	ótimo	7	5	3,4,5,6,8	ótimo
3	10	3,4,5,8	ótimo	8	9	3,4,6,7,8	ótimo
4	max	0	cflt	9	9	1,2,3,6,8	ótimo
5	4	1,2,3,7	ótimo	10	3	3,4,7	ótimo

bloqueantes. Para todos os outros testes a solução encontrada foi equivalente à ótima.

O único teste que retornou a solução ótima contendo apenas os módulos 1 e 4, dentre o o conjunto 1, 2, 3, e 3, foi o teste de número 8. Os testes restantes continham três dos quatro módulos distinguidores de instância do evento e.

TAB. 5.16: Resultados do algoritmo de busca com Busca Tabu para sistema de empilhamento A4

Teste	Zo	H	condição	Teste	Zo	H	condição
1	3	2,3,4	ótimo	6	14	1,2,4,7,8	ótimo
2	max	0	cflt	7	7	1,2,3,8	ótimo
3	13	2,3,4,8	ótimo	8	7	1,4,6,7,8	ótimo
4	5	1,3,4,5,8	ótimo	9	10	2,3,4,7	ótimo
5	9	1,2,4,5,6,7	ótimo	10	9	1,2,4,5,8	ótimo

Os supervisores aproximados para o sistema de empilhamento tipo A5 devem ser sintetizados a partir de aproximações contendo os módulos 1, 2 e 4, porém qualquer combinação de três dos quatro módulos distinguidores de instância do evento e também conduzem à solução ótima, devido à complementaridade, apresentada na seção 4.3. Como a aproximação que leva à solução ótima demanda mais módulos, ela é mais improvável de ocorrer, por isso para o sistema A5 o algoritmo encontra a solução ótima em um número menor de testes do que para outros sistemas.

TAB. 5.17: Resultados do algoritmo de busca com Busca Tabu para sistema de empilhamento A5

Teste	Zo	H	condição	Teste	Zo	H	condição
1	16	2,3,4,5,7,8	ótimo	6	8	1,2,3,7,8	ótimo
2	max	0	cflt	7	7	1,2,4,6	ótimo
3	5	1,2,3,8	ótimo	8	12	1,2,4,5,7	ótimo
4	max	0	cflt	9	max	0	cflt
5	7	1,3,4,7,8	ótimo	10	3	1,2,3	ótimo

A tabela 5.18 apresenta os resultados para o sistema de manufatura de tintas. Os

supervisores sintetizados na seção 5.1.4 mostram que somente o módulo 4 é necessário para se obter o supervisor aproximado que, com L_D, pode ser usado como solução do PCS-D, implicando que os resultados do algoritmo são consistentes, pois todo supervisor sintetizado a partir de uma aproximação contendo este módulo foi retornado como ótimo.

Apenas em dois casos, testes 8 e 9, o algoritmo não retornou a solução ótima. Nestes testes o resultado foi o supervisor sintetizado a partir da aproximação contendo os módulos distinguidores que melhoraram o indíce das soluções. No caso, para ambos os testes, nenhum módulo foi adicionado e o supervisor com maior medida de linguagem é o obtido a partir da aproximação sem nenhum módulo distinguidor, cuja medida é 1,000000472499455.

TAB. 5.18: Resultados do algoritmo de busca com Busca Tabu para sistema de manufatura de tintas

Teste	Zo	H	condição	Teste	Zo	H	condição
1	3	1,4	ótimo	6	5	2,4	ótimo
2	5	3,4,5	ótimo	7	2	1,4	ótimo
3	1	4	ótimo	8	1	0	1,000000472499455
4	2	4,5	ótimo	9	1	0	1,000000472499455
5	6	2,3,4	ótimo	10	3	1,4,5	ótimo

5.2.2 RESULTADOS PARA ALGORITMO GENÉTICO

Os resultados para a busca por supervisores aproximados utilzando o Algoritmo Genético como heurística estão expostos nas tabelas 5.19, 5.20, 5.21, 5.22, 5.23, 5.24, 5.25 e 5.26. A forma de exposição dos resultados é análoga à para a Busca Tabu, em que a coluna *Zo* mostra o número de soluções pesquisadas até o término do algoritmo, a coluna H exibe os módulos distinguidores envolvidos na síntese e a coluna *condição* exibe se o supervisor é conflitante com L_D, indicado por *cflt*, se ele pode ser implementado concorrentemente a H como solução para o PCS-D, indicado por *ótimo* ou para nenhum dos dois casos, a coluna contém a medida da linguagem do supervisor. A indicação *max* é utilizada na coluna *Zo* quando o algoritmo alcançar o número máximo de iterações sem encontrar um supervisor não conflitante com L_D.

A tabela 5.19 mostra os resultados para o sistema de manufatura concorrente. Como qualquer aproximação resulta em um supervisor ótimo para este sistema, subseção 5.1.1, o supervisor codificado pelo primeiro indivíduo da população inicial é retornado como solução em todos os casos, independente da aproximação que este indivíduo representa.

TAB. 5.19: Resultados do algoritmo de busca com Algoritmo Genético para sistema de manufatura concorrente

Teste	Zo	H	condição	Teste	Zo	H	condição
1	1	0	ótimo	6	1	0	ótimo
2	1	0	ótimo	7	1	1	ótimo
3	1	2	ótimo	8	1	0	ótimo
4	1	2	ótimo	9	1	1	ótimo
5	1	1	ótimo	10	1	1	ótimo

Os resultados do algoritmo para o sistema de manufatura com retrabalho estão expostos na tabela 5.20. Três dos dez testes, números 1, 2 e 4, não retornaram o supervisor que pode ser implementado como ótimo. Em conformidade com o verificado na seção 5.1.2, os supervisores que não foram retornados como ótimos não são conflitantes com a linguagem L_D do distinguidor e suas medidas foram as maiores calculadas dentre as dos outros supervisores não ótimos sintetizados.

TAB. 5.20: Resultados do algoritmo de busca com Algoritmo Genético para sistema de manufatura com retrabalho

Teste	Zo	H	condição	Teste	Zo	H	condição
1	19	6	1,000000000910600	6	10	7	ótimo
2	13	3	1,000000000910600	7	2	7	ótimo
3	1	7	ótimo	8	5	7	ótimo
4	13	1,5,8	1,000000000910600	9	1	7	ótimo
5	3	7	ótimo	10	1	7	ótimo

Para o sistema de empilhamento tipo A1, os testes retornaram supervisores ótimos em seis dos dez testes e para os quatro restantes, testes número 2, 3, 5 e 10, não foi encontrada uma solução para o problema de controle, uma vez que todos os supervisores encontrados eram conflitantes com a linguagem do distinguidor L_D. Os resultados estão expostos na tabela 5.21.

Os resultados do teste retornam um supervisor ótimo sempre que o módulo 1, distinguidor modular da instância *e1*, está presente e um supervisor conflitante caso contrário, tais quais os resultados da seção 5.1.3.

Na seção 5.1.3 foi verificado que, para o sistema de empilhamento do tipo A2, os supervisores sintetizados a partir de aproximações do distinguidor que continham o módulo distinguidor de instância *e4* eram implementados como ótimos e aqueles sintetizados a partir de aproximações sem esse módulo eram conflitantes com a linguagem do distinguidor.

TAB. 5.21: Resultados do algoritmo de busca com Algoritmo Genético para sistema de empilhamento A1

Teste	Zo	H	condição	Teste	Zo	H	condição
1	3	1	ótimo	6	2	1	ótimo
2	max	2,4	cflt	7	4	1	ótimo
3	max	3	cflt	8	4	1	ótimo
4	2	1	ótimo	9	4	1	ótimo
5	max	0	cflt	10	max	8	cflt

Os resultados dos testes de busca utilizando Algoritmo Genético para este sistema estão de acordo com o observado para o sistema e retornou o supervisor ótimo, nos dez testes, somente quando a aproximação continha o módulo 4, que corresponde ao módulo distinguidor da instância *e4*.

TAB. 5.22: Resultados do algoritmo de busca com Algoritmo Genético para sistema de empilhamento A2

Teste	Zo	H	condição	Teste	Zo	H	condição
1	1	4	ótimo	6	6	4	ótimo
2	5	4	ótimo	7	4	4	ótimo
3	1	4	ótimo	8	3	4	ótimo
4	4	4	ótimo	9	5	4	ótimo
5	1	4	ótimo	10	16	3,4	ótimo

As aproximações para os sistemas de empilhamento A3 e A4 devem conter dois módulos distinguidores para resultar em supervisores ótimos, respectivamente os pares de módulos 1, 2 e 1, 4, conforme subseção 5.1.3.

Como pode ser observado na tabela 5.23, resultados para o sistema de empilhamento tipo A3, os testes de número 2, 3, 8 e 9 não retornaram supervisores não conflitantes com a linguagem do distinguidor preditivo. Os testes de número 1, 4 e 10 retornaram a solução ótima para aproximações contendo os módulos 1 e 2, enquanto os de número 5, 6 e 7 retornaram a solução ótima para aproximações que não contêm o módulo 1, porém contêm os módulos complementares necessários, e estão de acordo com o exposto na seção 4.3.

A análise dos resultados dos testes para o sistema de empilhamento tipo A4, tabela 5.24, é semelhante a do sistema tipo A3. Os testes números 1, 6 e 7 não retornaram solução, uma vez que todos os supervisores pesquisados eram conflitantes com o distinguidor preditivo, os testes números 3, 4, 5, 8 e 10 obtiveram o supervisor ótimo sintetizados a partir de aproximações que continham o par de módulos distinguidores 1,4 e para os testes números 2 e 9, apesar das aproximações não conterem o módulo

TAB. 5.23: Resultados do algoritmo de busca com Algoritmo Genético para sistema de empilhamento A3

Teste	Zo	H	condição	Teste	Zo	H	condição
1	24	1,2	ótimo	6	9	2,3,4,7	ótimo
2	max	2,5	cflt	7	9	3,4,5,7	ótimo
3	max	2,4,5,6	cflt	8	max	7	cflt
4	9	1,2,5,6	ótimo	9	max	3	cflt
5	19	3,4,6	ótimo	10	14	1,2,7	ótimo

1, elas continham os módulos complementares para que a distinção da instância *e1* em relação às demais fosse realizada, conduzindo à solução ótima.

TAB. 5.24: Resultados do algoritmo de busca com Algoritmo Genético para sistema de empilhamento A4

Teste	Zo	H	condição	Teste	Zo	H	condição
1	max	3,5	cflt	6	max	5	cflt
2	11	2,3,4,5	ótimo	7	max	0	cflt
3	10	1,4,8	ótimo	8	23	1,4,7	ótimo
4	24	1,3,4,5	ótimo	9	15	2,3,4,7,8	ótimo
5	13	1,2,4,8	ótimo	10	18	1,2,4,7	ótimo

Para o sistema de empilhamento A5, os testes retornaram o supervisor ótimo em seis dos dez testes. Os resultados são apresentados na tabela 5.25. Na subseção 5.1.3 verificou-se que os supervisores obtidos a partir de aproximações contendo os módulos distinguidores correspondentes aos módulos 3 e 4 são ótimos.

Os testes de números de 1, 3, 4 e 7 estão de acordo com este resultado e retornaram corretamente o supervisor ótimo. Os teste de número 5 e 7, no entanto, retornaram o supervisor ótimo para aproximações construídas com os módulos 1, 2 e 3, que de acordo com a seção 4.3, também distinguem o evento *e4*, logo, é consistente com os resultados da seção 5.1.3.

TAB. 5.25: Resultados do algoritmo de busca com Algoritmo Genético para sistema de empilhamento A5

Teste	Zo	H	condição	Teste	Zo	H	condição
1	21	3,4,6,7	ótimo	6	max	3	cflt
2	max	4,7	cflt	7	21	1,2,3	ótimo
3	12	3,4,6	ótimo	8	max	1	cflt
4	14	3,4	ótimo	9	7	2,3,4,7	ótimo
5	12	1,2,3	ótimo	10	max	5	cflt

O algoritmo aplicado ao sistema de manufatura de tintas, cujos resultados estão

expostos na tabela 5.26, obtiveram o supervisor ótimo em sete dos dez testes. Os resultados são consistentes com os obtidos na subseção 5.1.4, de que o único módulo necessário para a síntese do supervisor ótimo é o módulo 4 correspondente ao módulo distinguidor do evento *bm2*.

TAB. 5.26: Resultados do algoritmo de busca com Algoritmo Genético para sistema de manufatura de tintas

Teste	Zo	H	condição	Teste	Zo	H	condição
1	2	4	ótimo	6	2	4	ótimo
2	17	1	1.000000472499455	7	3	4	ótimo
3	17	2	1.000000472499455	8	1	4	ótimo
4	2	4	ótimo	9	1	4	ótimo
5	17	1	1.000000472499455	10	1	4	ótimo

5.3 CONCLUSÃO

Os resultados dos testes para os algoritmos baseados em Busca Tabu e Algoritmo Genético evidenciaram o exposto na seção 4.3, de que um grupo de módulos complementares também pode ser utilizado para se obter o supervisor que implementado concorrentemente ao distinguidor é solução para o PCS-D.

O algoritmo baseado na Busca Tabu, apesar de ser orientado a retornar o supervisor sintetizado a partir de aproximações contendo somente os módulos que afetam a solução, muitas vezes retornou o supervisor obtido a partir de aproximações que continham outros módulos aleatórios além do módulo necessário. Isso decorre da condição de parada imposta ao algoritmo de busca de que para $SupC(K_{aj}, G_{aj}) = Sup\Sigma_u(K_{aj}, G_{aj})$, este supervisor é retornado. Com a condição de parada satisfeita, o algoritmo não tem a oportunidade de proceder à investigação de que somente um grupo determinado de módulos melhora o índice da solução, o que faria com que o supervisor retornado fosse obtido somente com este grupo.

Outra limitação do algoritmo da Busca Tabu, que contribuiu para que em alguns testes não fosse encontrado o supervisor não-conflitante, é o fato de módulos que não melhoram a solução serem automaticamente considerados como não chave. Para os sistemas de empilhamento A3, A4 e A5, mais de um módulo era necssário para eliminar o não conflito, logo, individualmente, cada módulo foi descartado como chave.

Para os sistemas que necessitavam de apenas um módulo distinguidor para a síntese de um supervisor ótimo, o algoritmo baseado em Algoritmo Genético não foi capaz de

encontrar a solução ótima em todos os testes. Isso se deve ao fato de que, uma vez que o alelo correspondente a este módulo não é encontrado em nenhum gene da população inicial, a única forma deste alelo integrar algum gene é por meio da mutação.

Nos casos em que o número de indíviduos na população inicial excede ao número de módulos distinguidores, cada módulo estará presente em pelo menos um gene, implicando que todos os testes retornarão um supervisor ótimo. Ao se considerar que o espaço contém 2^{n_dist} aproximações, em que n_dist é o número de módulos distinguidores para o sistema, uma população inicial com n_dist representa menos de 1% do total de aproximações possíveis para $n\_dist = 10$ e essa porcentagem diminui exponencialmente.

Recomenda-se então que para sistemas com número de módulos distinguidores superior a dez, a população inicial contenha tantos indíviduos quanto módulos distinguidores.

6 ESTUDO DE CASO PARA UM SISTEMA MULTIVEICULAR

O emprego de robôs é aconselhado sempre que a atividade a ser desenvolvida é demasiadamente repetitiva ou oferece risco ao ser humano. A substituição resulta também em aumento da produtividade e confiabilidade do sistema, uma vez que os robôs não são propensos a erros causados por desvio de atenção ou subjetividade e podem desempenhar a mesma tarefa continuamente sem ter seu desempenho alterado. Um sistema que emprega simultaneamente dois ou mais veículos autônomos é denominado Sistema Multiveicular. A utilização de múltiplos agentes melhora sobremaneira a performance do sistema, uma vez que pode haver cooperação, isto é, um ou mais agentes atuando em conjunto para executar uma tarefa a qual não poderiam terminar sozinhos ou simplesmente a atuação conjunta resulta em aumento da eficiência (IOCCHI et al., 2001).

Uma das aplicações mais comuns para estes veículos está no sistema fabril, no qual são chamados veículos auto-guiados ou AGV, do inglês *automated guided vehicle* e sua função é o transporte de peças. Embora um sistema composto por um AGV seja fácil de coordenar, ele é pouco eficiente. Em contrapartida, a inserção de mais elementos implica um aumento de complexidade no controle do sistema (VIS, 2006). Não são raros os casos em que, visando a assegurar a segurança do sistema com a garantia de funcionamento, sacrifica-se o desempenho ao se conceber um modelo simplista do sistema, que exarceba as restrições de funcionamento, pois a modelagem da especificação requerida exige níveis de abstração nem sempre possíveis de serem alcançados. Soma-se a isto a explosão combinatória de estados do sistema, que inviabiliza a síntese de um supervisor ótimo. Por estes motivos, muitos sistemas têm rendimento muito aquém de sua capacidade.

Na área de defesa são várias as possíveis aplicações para sistemas com múltiplos veículos coordenados, como o resgate de feridos em campo de combate, mapeamento de minas terrestres e até mesmo missões de espionagem. Soma-se a este conjunto de possibilidades, as atividades desempenhadas por veículos aéreos não tripulados, VANTs, que incluem de missões de defesa passiva, como as de monitoramento remoto e vigilância a missões de reconhecimento e destruição de alvos. O Exército Brasileiro tem direcionado parte de seus esforços na área tencnológica e acadêmica para programas de pesquisa na área de VANT visando ao desenvolvimento de veículos nacionais. Estes esforços buscam

cumprir os objetivos constantes do Plano Básico de Ciência e Tecnologia (PBCT), que criou um grupo finalístico dedicado a este fim.

Os sistemas compostos por mútiplos AGVs podem ser facilmente extrapolados para um sistema com múltiplos VANTs atuando em conjunto para atingir um determinado propósito, uma vez que guardam em comum as mesmas especificações e as plantas de ambos são intercambiáveis. Por especificações em comum, entenda-se as especificações inerentes para se evitar colisões, garantir rotas ótimas e assegurar a cooperação entre os elementos.

6.1 SISTEMA MULTIVEICULAR COM DOIS AGVS

O sistema concebido trata de recolher peças de um ponto de origem, transportá-las até um local de triagem, onde são testadas e direcioná-las, de acordo com o resultado da triagem, ao destino mais adequado. As peças são transferidas entre as diversas estações por AGVs. Uma das possíveis aplicações deste tipo de sistema na área de defesa é o transporte de feridos entre estações de primeiros-socorros e triagem, posteriormente os encaminhando para a estação de tratamento mais adequada.

O sistema é composto por uma esteira de entrada E, duas esteiras de saída P e Q e uma unidade de teste T, chamadas estações de trabalho. Os AGVs, idênticos para todos os fins, se movimentam guiando-se por trilhas com sentido pré-estabelecido. As estações, de onde os AGVs retiram ou depositam peças, encontram-se à margem da trilha. O esquema da disposição das trilhas e das estações encontra-se na figura 6.1. Em cada estação há um sensor que indica a presença de um AGV.

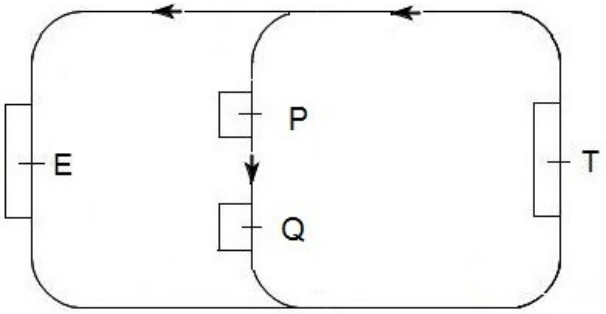


FIG. 6.1: Trilha do sistema multiveicular.

Para este sistema foram empregados dois AGVs, que circulam livremente pelas trilhas. O controle pretendido é centralizado e os dois robôs não se comunicam entre si.

Uma esteira rolante na estação de entrada E provém peças para o sistema, garantindo que a qualquer momento haja uma peça disponível. A peça é recolhida desta estação e

encaminhada para teste na estação de testes T. O resultado do teste determina que a peça está aprovada ou descartada. As peças aprovadas e descartadas são então encaminhadas às estações de saída P e Q, respectivamente.

As estações de saída P e Q consistem em esteiras que sempre estão aptas para receber peças e a estação T têm capacidade para uma peça por vez, assim como os AGVs, que podem transportar somente uma peça.

A. Plantas

A estação de entrada E não é modelada pois é considerado que sempre há uma peça disponível para ser retirada, logo não é necessário que uma planta modele este comportamento. Pela mesma razão, as estações de saída P e Q também não são modeladas, pois considera-se que sempre é possível depositar uma peça em quaisquer das esteiras.

Os eventos relativos à estação de testes T são *vldr*, que indica o ínico de um teste, *aprv*, que indica fim do teste com aprovação da peça e *desc*, que indica fim do teste com reprovação da peça. A figura que ilustra o modelo dessa planta é apresentada em 6.2.

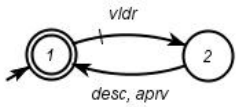


FIG. 6.2: Planta da unidade de teste T do sistema multiveicular.

O comportamento do AGV demanda que dois aspectos sejam modelados: a movimentação nas trilhas e a operação de carga/descarga. Para isso, duas plantas foram modeladas separadamente e para diferenciar os movimentos entre os AGVs, tanto no deslocamento, quanto nas operações de carga, os sufixos x e y são adicionados aos nomes dos eventos. O deslocamento de um AGV pelas trilhas é sinalizado por eventos controláveis que denotam o trecho no qual o AGV está circulando e eventos não-controláveis que indicam a que estação o AGV chegou. O estado inicial da planta deste deslocamento, figura 6.3, é o estado que corresponde ao trecho no qual o AGV se encontra no início de operação da trilha e é também um estado final, pois é desejado que ao fim da operação do sistema o AGV possa retorna ao seu local de origem. Cada um dos estados que representa que o AGV chegou a uma estação também é um estado marcado. A planta mapeia os deslocamentos que o AGV pode executar na trilha.

Quando o AGV está prestes a iniciar o seu movimento por um trecho, o desloca-

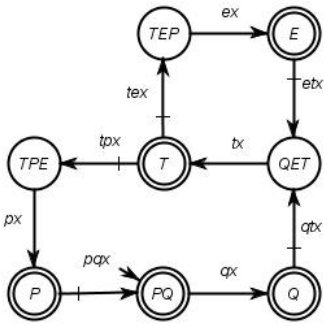
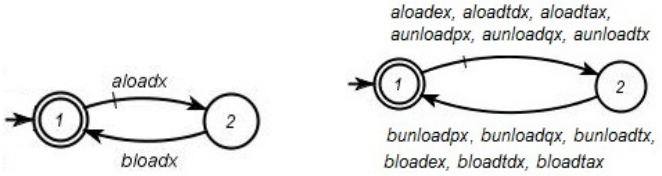


FIG. 6.3: Plantas do sistema multiveicular.

mento pode ser interrompido, desabilitando o evento controlável correspondente, com a finalidade de controlar o fluxo pelas trilhas. Uma vez iniciado o deslocamento, ele só é interrompido ao se adentrar um novo trecho, motivo pelo qual a chegada a uma estação é um evento não controlável.

Com respeito a carga e descarga de peças pelo AGV, os eventos são aload e bload para início e fim da manipulação do compartimento de carga do AGV. A planta está ilustrada na figura 6.4.a. Os refinamentos dos eventos da planta de carga, figura 6.4.b, embora numerosos, seguem uma regra simples. Os refinamentos da planta de carga referem-se primeiramente a se o AGV está sendo carregado ou descarregado, adicionando a subcadeia *un* em *aload* e *bload* quando for operação de descarga, refinando os eventos para *aunload* e *bunload*. Em seguida adiciona-se o nome da estação onde ocorre a operação. Para peças carregadas em T, adiciona-se ainda um sufixo que indica o resultado do teste, sendo a para aprovada e d para reprovada. Por exemplo, se uma peça aprovada está sendo carregada em T, o evento refinado é *aloadtax* ou *aloadtay*, dependendo do AGV e para uma peça sendo descarregada em Q pelo AGV y, o evento correspondente é *aunloadqy*.



(a) Planta não refinada (b) Planta refinada

FIG. 6.4: Planta da carga do AGV x do sistema multiveicular.

A planta refinada do sistema, obtida pela composição síncrona das plantas da estação de testes T, deslocamento e carga dos AGV x e y possui o total de 1800 estados.

B. Especificações

Para que o sistema funcione sem que haja situações bloqueantes, ou seja, situações em que o estado do sistema não pode ser alterado, é necesário estabelecer uma série de especificações. Elas são:

- Só pode haver uma peça por vez em T;
- Só pode haver uma peça por vez no AGV;
- O AGV só pode iniciar a carga/descarga se houver peça disponível na estação;
- O AGV só pode iniciar a carga/descarga quando posicionado adequadamente nas estações;
- Uma vez iniciada a carga/descarga, o AGV só pode se mover após o término da operação;
- Para evitar colisões, somente um AGV por vez pode circular em um dado trecho de trilha;
- O AGV carregado deve se direcionar para a estação de descarga adequada;
- O AGV deve obrigatoriamente descarregar na estação de destino caso esta esteja apta a receber a peça;
- A peça deve seguir a ordem correta dentro do sistema.

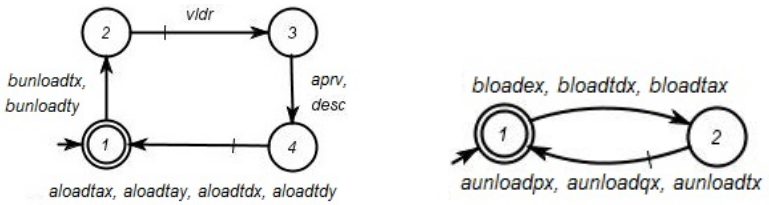
Além disso, deve ser assegurado que:

- uma vez carregados, os AGVs sempre possam descarregar;
- que as peças na estação T sempre possam ser coletadas e
- que cada AGV possa retornar a seu ponto de partida inicial.

Algumas dessas especificações podem ser modeladas em conjunto pelo mesmo autômato e outras serão asseguradas uma vez que as outras sejam respeitadas.

As especificações $EbufT$ modela a restrição de capacidade de uma peça para T. Partindo do estado inicial, que representa a estação vazia, um AGV, x ou y, pode descarregar uma peça na estação. Após a mesma processar a peça, um AGV, não necessariamente o mesmo que descarregou a peça, pode recolhê-la, quando o autômato

retorna ao estado inicial. Esta especificação ao mesmo tempo que impede que mais de uma peça seja depositada na estação, também impede que uma peça seja retirada sem que haja peça disponível para tal, figura 6.5.



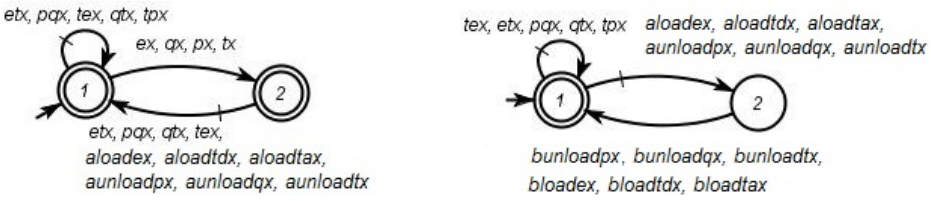
(a) EbufTU (b) EbufAGV

FIG. 6.5: Especificação de capacidade dos elementos do sistema multiveicular.

Para impedir que o AGV tente realizar operação de descarga quando não está carregado ou tente carregar quando já está transportando um peça, modela-se a especificação EbufAGV, exposta na figura 6.5.

A especificação *Estart*, figura 6.6, é responsável por asssegurar que um AGV só inicie a operação de carga ou descarga uma vez que esteja posicionado corretamente na estação, o que é determinado pelo sensor de presença na estação. Esta especificação determina que os eventos de início de carga e início de descarga só ocorrem após a chegada do respectivo AGV a uma das estações do sistema.

A especificação *Efinish*, figura 6.6, determina que uma vez iniciada uma operação de carga ou descarga, denotado pelos refinamentos do evento *aload*, o AGV só pode se deslocar após a ocorrência dos refinamentos de *bload*, que representam o término da operação de carga e descarga.



(a) Estart AGV x (b) Efinish AGV x

FIG. 6.6: Especificações para início e término de carga dos AGV do sistema multiveicular.

Com a finalidade de garantir que não haja colisão entre os AGVs durante a circulação pelas trilhas, determina-se que dois AGVs não podem trafegar em dado trecho simultaneamente. Para isso, modela-se especificações de exclusão mútua para cada trecho da trilha. O autômato consiste em dois estados, um representando o trecho livre e outro

que o trecho está ocupado. Do estado livre, se ocorre um evento que denota que um AGV iniciou o deslocamento para aquele trecho, passa-se ao estado de trecho ocupado e todos os eventos de início de deslocamento naquele trecho ficam impedidos de ocorrer até que um evento de marca a saída de um AGV daquele trecho ocorra.

A figura 6.7 mostra o exemplo para o trecho PQ e para a bifurcação QET. Para as bifurcações, como QET, há um trecho em comum para dois destinos diferentes e portanto, tanto os deslocamentos no trecho QT, quanto em ET, ocupam a bifurcação. A bifurcação QET é liberada tão logo o AGV siga para o trecho TE ou TP. Deve ser notado que quando o trecho é o ponto de partida para um AGV, o estado inicial da especificação que modela este trecho representa o estado ocupado, caso do trecho PQ por exemplo.

(a) trecho PQ (b) bifurcação QET

FIG. 6.7: Especificação de exclusão mútua do sistema multiveicular.

A especificação *Erot*, uma por AGV, desempenha a dupla função de garantir a ordem correta do processamento das peças no sistema e de determinar a rota que o AGV deve seguir entre duas estações uma vez que esteja carregado. O AGV vazio, representado pelo estado inicial, pode trafegar por qualquer trecho. De acordo com a estação e peça que ele carrega, seu destino será conforme apresentado na tabela 6.1.

TAB. 6.1: Roteamento dos AGVs.

estação de carga	estação de descarga	rota
E	T	E-T
T peça descartada	Q	T-P-Q
TU peça aprovada	P	T-P

A figura 6.8 mostra o autômato da especificação de roteamento para o AGV x.

A composição síncrona de todas as especificações resulta em um autômato de 10224 estados.

Ressalta-se que para os modelos das especificações $EbufT$, $EbufAGV_x$, $EbufAGV_y$, e as de roteamento de AGV, o refinamento do alfabeto simplicou sobremaneira a tarefa de modelagem, assim como os modelos em si.

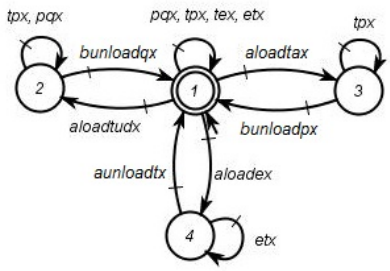
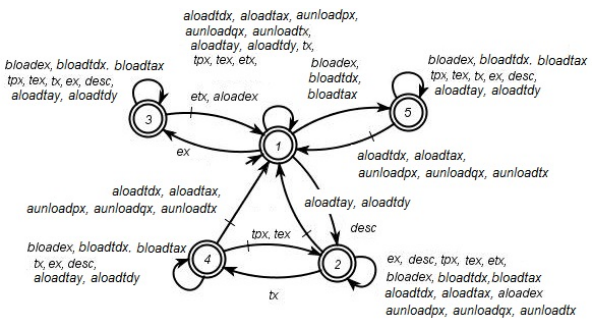


FIG. 6.8: Especificação de roteamento do AGV x do sistema multiveicular.

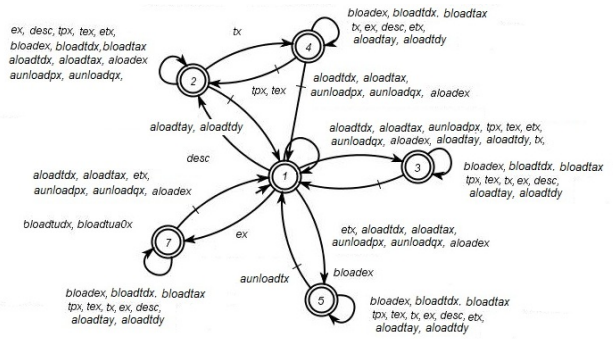
C. Distinguidores

Para os distinguir os eventos refinados do sistema multiveicular, foram modelados, sempre que possível, distinguidores de instância. Na figura 6.9 são apresentadas as ilustrações de dois distinguidores de instância para o evento *aloadx*, carga em E e descarga em T. O módulo distinguidor para as instâncias de *aloadx*, análogo para *aloady*, deve diferenciar uma instância de outra, estabelecendo a sequência de eventos para a qual cada uma se aplica quando da ocorrência de *aloadx*. Assim, após o AGV x chegar a estação E, se ocorrer o evento *aloadx*, a instância correspondente é *aloadex*. Se uma peça for reprovada em T e o AGV x chegar a esta estação antes que a peça seja retirada, a instância correspondente será *aloadtdx*. A instância padrão é *aloadtax*.

O distinguidor modular de instância para *aunloadtx* determina que após o AGV x ser carregado em E, evento *bloadex*, a única instância de *aloadx* que pode ocorrer é *aunloadtx* e para qualquer outra sequência, qualquer outra instância pode ser designada. Seguindo esta lógica, por exemplo, após o AGV x ser carregado em T, evento *bloadtax*, a instância que deve ocorrer é *aunloadpx*, o que é garantido pelo distinguidor de instância de *aunloadpx*.



(a) instância carga em E de *aloadx*

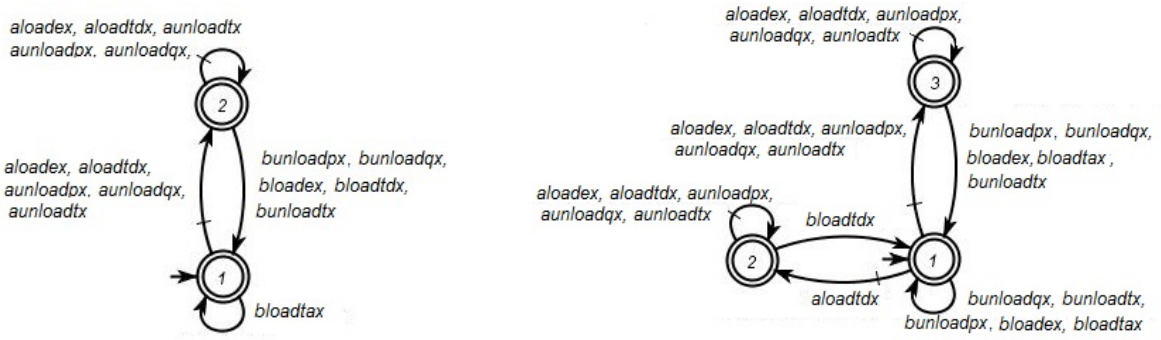


(b) instância de descarga em T de *aloadx*

FIG. 6.9: Distinguidores de instância do evento *aloadx* do sistema multiveicular.

Para cada instância de *bload* o distinguidor modular por instância determina que

a instância que ocorre é a de mesmo índice do evento *aload* que o antecede, ou seja, após o evento *aloadtax*, o refinamento correspondente de *bloadx* é *bloadtax*. A figura 6.10 apresenta dois módulos distinguidores de instância para *bload*, para carga em T de peça aprovada e para carga em T de peça reprovada.



(a) instância carga em T de peça aprovada de *bloadx*

(b) instância carga em T de peça reprovada de *bloadx*

FIG. 6.10: Distinguidores de instância do evento *bloadx* do sistema multiveicular.

Ao todo, foram modelados 24 módulos de distinguidores de instância, seis para cada evento refinado. O distinguidor de todos os eventos refinados do sistema, obtidos da composição síncrona de todos os módulos distinguidores possui 729 estados

6.2 RESULTADOS

De posse das plantas, especificações e distinguidores, as rotinas de busca propostas foram utilizadas para encontrar um supervisor para o sistema multiveicular. Devido ao grande número de estados da especificação global, foi empregado o Controle Modular, apresentado na seção 2.2.2, ou seja, foi sintetizado um supervisor para cada especificação elementar separadamente.

Cada um dos dois métodos de busca foi utilizado cinco vezes para obtenção do supervisor, com resultados independentes entre si. Os resultados para a busca são apresentados em tabelas indicando a instância, dentro do algoritmo de busca, a que corresponde a solução retornada, o número de estados do supervisor obtido, os módulos distinguidores utilizados na aproximação a partir da qual este supervisor foi sintetizado e se o supervisor é ótimo, conflitante ou, para nenhum dos dois casos, sua medida de linguagem.

A especificação trabalhada foi a de exclusão mútua para o trecho QET. Os resultados para a Busca Tabu estão expostos na tabela 6.2. O supervisor que pode ser implementado paralelamente ao distinguidor como ótimo foi retornado em todos os cinco testes.

Para cada um dos testes, um módulo distinguidor diferente foi utilizado na síntese do supervisor.

TAB. 6.2: Supervisores para o sistema multiveicular obtidos por Busca Tabu

Teste	**Instância**	**Estados**	**Distinguidores**	**Condição**
1	1	960	18	ótimo
2	1	1552	9	ótimo
3	1	720	16	ótimo
4	1	1312	4	ótimo
5	1	960	17	ótimo

A tabela 6.3 mostra os resultados para a busca baseada em Algoritmo Genético, também para supervisores para a especificação de exculão mútua do trecho QET. Assim como para a Busca Tabu, este algortimo retornou o supervisor que pode ser implementado como solução ótima em todos os cinco testes, também com módulos distinguidores diferentes em cada teste.

A análise dos resultados revela que, uma vez que módulos diferentes e únicos conduziram à síntese do supervisor ótimo, qualquer aproximação conduzirá ao mesmo resultado, assim como a aproximação sem módulos distinguidores, distinguidor H_a.

TAB. 6.3: Supervisores para o sistema multiveicular obtidos por Algoritmo Genético

Teste	**Instância**	**Estados**	**Distinguidores**	**Condição**
1	1	1920	2	ótimo
2	1	960	20	ótimo
3	1	960	17	ótimo
4	1	1552	7	ótimo
5	1	960	23	ótimo

6.3 CONCLUSÃO

Os algoritmos de busca com heurísticas de Busca Tabu e de Algoritmo Genético apresentaram resultados semelhantes para o sistema multiveicular com dois AGVs. Isso se deve ao fato de que, para a especificação escolhida, a partir de qualquer aproximação da linguagem do distinguidor é possível sintetizar um supervisor que pode ser implementado como ótimo, o que implica que a primeira aproximação produzida por qualquer um dos algoritmos é também a única.

7 CONCLUSÃO

Esta dissertação apresentou um método para se obter um supervisor o menos restritivo possível para sistemas de grande porte, por intermédio de busca heurística no espaço de supervisores aproximados por distinguidores. A motivação para este trabalho surge do fato de que para sistemas de grande porte, há uma explosão combinatória do número de estados que inviabiliza a obtenção de supervisores ótimos a partir da Teoria de Controle Supervisório existente. As alternativas envolvem aproximações e escolheu-se as aproximações por distinguidores para dar seguimento ao trabalho.

Os distinguidores introduzem o conceito de refinamento de eventos em instâncias, e ao mesmo tempo que simplificam a escrita de especificações elaboradas, permitem a redução da complexidade dos autômatos ao trabalharem as aproximações. As aproximações surgem ao se inserir apenas parte da linguagem do distinguidor, responsável por diferenciar as diversas instância de um evento. A busca heurística foi sugerida para explorar o amplo espaço formado pelos supervisores aproximados, obtendo dentre as possibilidades, aquele que apresentasse o melhor índice.

O índice utilizado foi a medida de linguagem, adaptada para aplicação a supervisores aproximados. Os distinguidores foram decompostos em módulos que distinguiam apenas uma instância em particular das demais, resultando em mais módulos disponíveis, uma síntese com menor complexidade e um maior entendimento de quais instâncias devem obrigatoriamente ser identificadas para que o supervisor tenha a informação necessária para que atue de forma mais permissiva possível.

As heurísticas escolhidas, e implementadas integradas ao algoritmo de busca, foram a Busca Tabu e o Algoritmo Genético. Os testes dos algoritmos com os sistemas criados demonstraram que os algoritmos obtiveram resultados sempre consistentes com os obtidos por ferramentas consolidadas para a área de Controle Supervisório. Além disso, examinando apenas uma pequena fração do espaço, os algoritmos alcançaram boa taxa de obtenção do supervisor ótimo.

Para o sistema multiveicular, devido ao grande número de estado obtidos para a especificação global, foi feita a opção pela síntese modular dos supervisores. O emprego dos algoritmos para solucionar um sistema de AGVs, que é naturalmente complexo, se mostrou satisfatótio, encontrando soluções ótimas para o problema. Por meio da análise

dos resultados, pôde-se ainda inferir características do sistema em relação à síntese, como, por exemplo, que para a síntese do supervisor modular para a exclusão mútua de um trecho do sistema não é necessário nenhum módulo distinguidor para obtenção da solução ótima.

7.1 PRINCIPAIS CONTRIBUIÇÕES

A principal contribuição deste trabalho é construção de um método para obter a melhor solução possível para sistemas de grande porte, para os quais a síntese de um supervisor monolítico ótimo é proibitiva devido a explosão combinatória do número de estados.

Também constitui contribuição o emprego de medida de linguagem para a comparação entre supervisores obtidos a partir de plantas aproximadas por distinguidores distintos, uma vez que eles nem sempre guardam relação de continência entre si. O foco da contribuição é a adaptação da teoria existente às particulariedades da Teoria de Controle Supervisório Distinguido, como a planta a ser utilizada como base para medida e a determinação de que a comparação deve utilizar o supervisor composto com a linguagem do distinguidor.

Outra contribuição está no estabelecimento de distinguidores modulares por instância e a sistematização da criação de um distinguidor. A vantagem do uso destes reside no fato de que tais distinguidores resultam em plantas menos complexas enquanto também levam a supervisores minimamente restritivos. Isto devido ao fato de que mais do que identificar corretamente as instâncias de um eventos, às vezes é suficiente diferenciar apenas uma instância das outras possíveis. Ademais, eles permitem que se compreenda mais a fundo a relacão entre especificação e distinguidores, revelando mais sobre a dependência entre os dois.

Por fim, a conjectura para nova síntese de supervisor, embora não tenha resultado numa solução efetiva para eliminar o conflito em todos os casos, representa um passo no sentido de compreender melhor as causas do conflito entre as linguagens dos supervisores aproximados e do distinguidor.

7.2 LIMITAÇÕES

A principal dificuldade encontrada na realização deste trabalho diz respeito a convergência numérica da medida de linguagem. A restrição de manter o limite superior

igual a 1 para o custo de evento, de modo que a medida de linguagem possa convergir, implica que, para cadeias longas, o custo total tende a zero. Como este trabalho trata de sistemas com grande número de estados, é comum que muitas cadeias que formam a linguagem dos supervisores sejam muito longas. A implicação desta tendência é que linguagens de supervisores com permissividades diferentes podem ter a mesma medida, o que afeta sobremaneira o processo de busca heurística.

Outra dificuldade encontrada na realização deste trabalho foi a escassez de sistemas disponíveis para validação dos algoritmos. Muitos sistemas se beneficiam da utilização dos distinguidores para simplificação da escrita das especificações e a aproximação pode ser utilizada para reduzir a complexidade da síntese de um supervisor. Porém, duas situações adversas restrigem o uso de tais sistemas neste trabalho. A primeira é o fato de a aproximação resultar em um supervisor conflitante com a linguagem do distinguidor, o que significa que tal supervisor não pode ser considerado solução. A segunda reside no fato de que as aproximações de muitos sistemas se agrupam em apenas dois pólos, em que as soluções ou são maximamente permissivas ou são iguais à solução com nenhum módulo distinguidor. Esta propriedade impede a análise da melhora gradual da solução pelos algoritmos de busca heurística.

7.3 TRABALHOS FUTUROS

Durante a execução dos trabalhos que concernem a este documento, várias questões surgiram e, devido a exiguidade de tempo, foram deixadas sem resposta. Algumas destas questões surgiram de limitações encontradas para a implementação do trabalho e outras, de investigações realizadas, porém não concluídas.

A primeira sugestão para investigação futura trata dos distinguidores necessários para se obter uma solução minimamente restritiva. Questiona-se que a existência de uma estreita relação entre a especificação e os distinguidores que levam à uma solução ótima e que por meio de uma análise prévia das especificações é possível prever os distinguidores fundamentais ao sistema. Essa análise pode inclusive auxiliar no estudo do conflito entre linguagem do distinguidor e dos supervisores aproximados.

Acredita-se que a causa do conflito entre supervisor e distinguidor encontra-se na especificação do sistema. Ao identificar-se o tipo de estrutura contida nessas especificações, por meio do confronto com o distinguidor, será possível prever o conflito antes de se efetuar a síntese do supervisor. De posse desta informação, vislumbra-se a possibilidade de se propor uma nova síntese de supervisor para SEDs. Este novo método então eli-

minaria sistematicamente as estruturas da linguagem-alvo que sabidamente interfeririam na obtenção de um supervisor não conflitante.

Sugere-se também investigar outros métodos para comparação entre supervisores aproximados. Este campo carece de métodos eficazes para comparação de linguagens e a medida aqui empregada é uma das propostas para preencher essa lacuna. Em relação a medida utilizada, também sugere-se inspecionar formas de melhor adaptá-la ao caso de linguagens com cadeias muito longas, evitando assim a tendência do custo de tais cadeias ser zero.

8 REFERÊNCIAS BIBLIOGRÁFICAS

BOUZON, G., QUEIROZ, M. e CURY, J. **Controle Supervisório de Sistemas a Eventos Discretos com Sensores Distinguidores**. *Anais do XVII Congresso Brasileiro de Automática*, 2008a.

BOUZON, G., QUEIROZ, M. e CURY, J. **Supervisory Control of DES with Distinguishing Sensors**. *International Worshop on Discret Event Systems*, págs. 22–27, 2008b.

BOUZON, G., QUEIROZ, M. e CURY, J. **Exploiting Distinguishing Sensors in Supervisory Control of DES**. *IEEE International Conference on Control and Automation*, págs. 442–447, Dezembro 2009.

BROWNLEE, J. ***Clever Algorithms: Nature-Inspired Programming Recipes***. LuLu, USA, first edition, 2011.

CASSANDRAS, C. e LAFORTUNE, S. ***Introduction to Discrete Event Systems***. LLC, USA, second edition, 2008.

CLAVIJO, L. ***Aspectos computacionais associados à implementação de algoritmos para Sistemas a Eventos Discretos***. Qualificação de tese de doutorado, COPPE/UFRJ, 2012.

CURY, J. **Teoria de Controle Supervisório de Sistemas a Eventos Discretos**. V Simpósio Brasileiro de Automação Inteligente, Novembro 2001.

CURY, J., QUEIROZ, M., BOUZON, G. e TEIXEIRA, M. **Supervisory Control of Discrete Event Systems with Distinguishers**. 2013. Artigo submetido a *Automatica*.

GLOVER, F. e LAGUNA, M. ***Tabu Search***. Kluwer Academic Publishers, Boston, 1997.

IOCCHI, L., NARDI, D. e SALERNO, M. **Reactivity and Deliberation: a survey on Multi-Robot System**. *Lectures Notes in Computer Science*, 2103:09–32, 2001.

JANG, J., SUN, C. e MIZUTANI, E. ***Neuro-Fuzzy and Soft Computing : A Computational Approach to Learning and Machine Intelligence***. Prentice-Hall,Inc., USA, 1997.

KEPHART, D. ***Topology, Morphisms, and Randomness in the Space of Formal Languages***. Tese de doutorado, University of South Florida, 2005.

PARDALOS, P., DU, D. e GRAHAM, R. ***Handbook of Combinatorial Optimization***. Springer, USA, first edition, 2011.

QUEIROZ, M. e CURY, J. **Modular supervisory control of large scale discrete-event systems**. *Discrete Event Systems: Analysis and Control*, págs. 103–110, 2000.

QUEIROZ, M. e CURY, J. **Controle Supervisório Modular de Sistemas de Manufatura**. *Revista de Controle & Automação*, 13(2):123–133, Maio, Junho, Julho, Agosto 2002a.

QUEIROZ, M. e CURY, J. **Synthesis and Implementation of Local Modular Supervisory Control for a Manufacturing Cell**. *Proceedings of the Sixth International Workshop on Discrete Event Systems (WODES'02)*, 2002b.

RAMADGE, P. e WONHAM, W. **Supervisory control of a class of discrete event process**. *SIAM Journal of Control and Optimization*, 25(1):206–230, 1987.

RAMADGE, P. e WONHAM, W. **The control of discrete event systems**. *Proceedings of IEEE, Special Issue on Discrete Event Dynamic Systems*, 77(1):81–98, Janeiro 1989.

RAY, A. e PHOHA, S. **Signed real measure of regular languages for discrete-event automata**. *International Journal of Control*, 76(18):1800–1808, 2003.

SURANA, A. e RAY, A. **Measure of Regular Languages**. *Demonstratio Mathematica*, XXXVII(2):485–503, 2004.

TEIXEIRA, M. ***Usando Distinguidores no Controle Supervisório Modular Local de Sistemas a Eventos Discretos***. Qualificação de tese de doutorado, UFSC, 2011.

TEIXEIRA, M., CURY, J. e QUEIROZ, M. **Local Modular Supervisory Control of DES with Distinguishers**. *IEEE International Conference on Emerging Technologies and Factory Automation*, 2011.

VIS, I. **Survey of research in the design and control of automated guided vehicle systems**. *European Journal of Operational Research*, 170(3):677–709, Maio 2006.

WANG, X. e RAY, A. **Signed real measure of regular languages**. *Proceedings of the American Control Conference*, 5(18):3937–3942, 2002.

WANG, X. e RAY, A. **A language measure for performance evaluation of discrete-event supervisory control systems**. *Applied Mathematical Modelling*, 28(9):817–833, Setembro 2004.

WONHAM, W. **SUPERVISORY CONTROL OF DISCRETE-EVENT SYSTEMS** , howpublished = Dept. of Electrical & Computer Engineering, University of Toronto, Julho 2011.

WONHAM, W. e RAMADGE, P. **Modular Supervisory Control of Discrete Event Systems**. *Math Control Signals Systems*, 1(1):13–30, 1988.