GUSTAVO BOUZON

SENSORES EM SISTEMAS A EVENTOS DISCRETOS

FLORIANÓPOLIS
2004

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

SENSORES EM SISTEMAS A EVENTOS DISCRETOS

Dissertação submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a
obtenção do grau de Mestre em Engenharia Elétrica.

GUSTAVO BOUZON

Florianópolis, agosto de 2004.

SENSORES EM SISTEMAS A EVENTOS DISCRETOS

Gustavo Bouzon

‘Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica, Área de Concentração em *Controle e Automação*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.’

José Eduardo Ribeiro Cury, Dr. d’Etat
Orientador

Denizar Cruz Martins, Dr.
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

José Eduardo Ribeiro Cury, Dr. d’Etat
Presidente

Antonio Marcus Nogueira Lima, Dr.

Eduardo Camponogara, Dr.

Victor Juliano de Negri, Dr.

AGRADECIMENTOS

Ao orientador José Cury, pela paciência e sabedoria com que orientou o presente trabalho, bem como pela confiança depositada, desde o começo, no resultado do trabalho.

Aos colegas do Grupo de Sistemas a Eventos Discretos da UFSC, pelas discussões, sugestões, e pelos seus excelentes trabalhos, que representam uma excelente fonte de consulta e inspiração.

Aos professores do DAS, pela formação profissional e pessoal proporcionada ao longo destes sete anos de convivência.

Aos Professores Antônio Marcus Nogueira Lima (UFCG), Eduardo Camponogara (UFSC) e Victor Juliano De Negri (UFSC), por aceitarem gentilmente a tarefa de participar da banca de avaliação de mestrado, e pelas críticas e sugestões altamente pertinentes que realizaram.

Estendo também o agradecimento a todos os presentes na defesa pública da dissertação.

Aos colegas e funcionários do Programa de Pós-Graduação em Engenharia Elétrica.

Ao Grupo de Usuários de Software Livre da UFSC (GUFSC), bem como aos que se dedicaram para desenvolver o modelo em LaTeX para dissertações disponibilizado na página do GUFSC.

Ao contribuinte brasileiro, que mantém esta Universidade pública de qualidade, e que, pela agência CAPES, financiou a bolsa de estudos.

Aos meus pais, pelo amor incondicional e ensinamento incomensurável.

À Ana Paula, pela compreensão da importância dos dias que passamos distantes.

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para obtenção do grau de Mestre em Engenharia Elétrica.

SENSORES EM SISTEMAS A EVENTOS DISCRETOS

Gustavo Bouzon

Agosto/2004

Orientador: José Eduardo Ribeiro Cury
Área de Concentração: Controle e Automação
Palavras-chave: Controle Supervisório, Sistemas a Eventos Discretos, Controle Modular
Número de Páginas: xvii + 90

Este trabalho avalia em que medida a resolução de um problema de controle supervisório pode ser simplificada pela introdução de sensores na planta. Na abordagem apresentada, cada sensor corresponde a um novo evento, que deve ser adicionado ao modelo original da planta. Para tanto, definem-se cadeias de eventos geradas pela planta original que ativam o sensor e eventos responsáveis por desativá-lo. Com estas informações, constrói-se o modelo exato do comportamento do sensor, que, combinado com o modelo da planta, fornece o modelo da planta com sensor. Mostra-se que, em geral, uma especificação de comportamento do problema original que contém as cadeias que ativam o sensor pode ser reescrita como uma especificação com menor número de estados utilizando o evento associado ao sensor. No âmbito da metodologia de controle modular local, a redução das especificações, combinada com a aproximação do modelo do sensor, permite a redução tanto da complexidade computacional do cálculo de cada supervisor quanto do número de estados dos supervisores. No entanto, a solução global obtida não é, necessariamente, equivalente à solução do problema original. Na classe de sensores desativados por eventos controláveis, entretanto, mostra-se que a utilização do modelo exato do sensor como nova especificação (emulação do sensor) fornece uma solução com as mesmas vantagens apontadas, garantindo-se ainda a equivalência com o problema original. Quando há eventos de desativação não controláveis, a emulação é também possível se o evento do sensor for considerado forçável. Esta alternativa necessita, no entanto, de que os resultados preliminares apresentados sejam aprofundados.

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

SENSORS IN DISCRETE-EVENT SYSTEMS

Gustavo Bouzon

August/2004

Advisor: José Eduardo Ribeiro Cury
Area of Concentration: Control and Automation
Key words: Supervisory Control, Discrete-Event Systems, Modular Control
Number of Pages: xvii + 90

This work evaluates how the resolution of a supervisory control problem can be simplified by the insertion of sensors in the plant. In the presented approach, each sensor is assigned to a new event that must be added to the original plant model. For that purpose, we define which event strings generated by the original plant activate the sensor and which events deactivate it. Based on that, a model of the sensor's exact behavior is built, whose combination with the plant model gives rise to the model of plant with sensor. It is shown that, in general, a specification at the original problem containing sensor activation strings may be represented by a generator with a smaller number of states if the sensor event is used. In the context of the local modular methodology, reduced specifications, combined with a sensor model approximation, provide reduction both of the complexity of calculating each supervisor and of the number of states of the supervisors obtained. Nevertheless, the solution obtained is not assured to be equivalent to the solution of the original problem. When the sensor is only deactivated by controllable events, however, it is shown that the sensor model usage as a new specification (sensor emulation) provides a solution with the same benefits pointed out before, with the additional guarantee of equivalence to the original problem. When some of the deactivation events are uncontrollable, emulation is also possible if the sensor event is considered to be forcible. In this case, preliminary results presented in this document must be further developed.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Exemplos

Lista de Abreviaturas e Símbolos

SED	Sistema a Eventos Discretos
RSP	Representação por Sistemas Produto
PCS	Problema de Controle Supervisório
$\cup$	união de conjuntos
$\cap$	intersecção de conjuntos
$\dot{\cup}$	união de conjuntos disjuntos
Σ	alfabeto
Σ^n	conjunto de todas as cadeias de n elementos de Σ
Σ^*	conjunto de todas as cadeias finitas de elementos de Σ
$\emptyset$	conjunto vazio
ε	cadeia nula
s, t, u, v	cadeias de eventos
L	linguagem
$\overline{L}$	prefixo-fechamento de L
$\in$	pertence a
$\subseteq$	está contido em
$\preceq$	é prefixo de
$\prec$	é prefixo próprio de
$\Sigma_L(t)$	conjunto de eventos ativos de L após t
α, β, γ, a, r	eventos determinados
P_i	projeção natural
P_i^{-1}	projeção inversa
A	autômato
q, x	estado
q_0, x_0	estado inicial
e	evento
f, g, θ	funções de transição de estados
Q	conjunto de estados
Q_m	conjunto de estados marcados
G	gerador

$L(G)$	linguagem gerada por G
$L_m(G)$	linguagem marcada por G
2^Q	conjuntos dos conjuntos de Q
$f(q,e)!$	f é definida para o par (q,e)
$\|\|$	produto síncrono de linguagens
$\|\|_{as}$	produto assíncrono de linguagens
Σ_c	conjunto de eventos controláveis
Σ_u	conjunto de eventos não controláveis
Σ_f	conjunto de eventos forçáveis
γ	entrada de controle
Γ	conjunto de entradas de controle admissíveis
Γ_f	conjunto de entradas de controle admissíveis, considerando eventos forçáveis
$cl(\Gamma)$	fechamento para união de Γ
h	função de supervisão
$h\|G$	G sob supervisão de h
S	representação de um supervisor como gerador
RS	supervisor reduzido
$S\|G$	G sob supervisão de S
Y	conjunto de estados da representação de supervisores como autômatos
ϕ	função que mapeia estados em entradas de controle
E,R	especificações genéricas
K	especificação global
$supC(M,G)$	máxima linguagem contida em M e controlável com relação a G
$supC_f(M,G)$	máxima linguagem $(L,cl(\Gamma_f))$-controlável contida em M
$O(.)$	complexidade computacional
I	conjunto de índices
$\geq$	é aproximação externa de
$P_{-\delta}$	projeção que retira eventos de Σ_δ
T	sensor
Σ_d	eventos dessensibilizadores
Σ_s	alfabeto do modelo do sensor
Σ_{ss}	alfabeto da linguagem sensibilizadora
L_{ss}	linguagem sensibilizadora
δ	evento do sensor
τ	evento auxiliar
L_{aux}	linguagem auxiliar
$L_{ss\tau}$	linguagem sensibilizadora genérica
G_q	planta sem sensor
G_s	modelo do sensor

$G_{s,ap}$	modelo aproximado do sensor
K_q	especificação sem evento do sensor
E_s	especificação genérica com evento do sensor
K_{qs}	especificação global com evento do sensor
$K_{qs,ap}$	especificação global com evento do sensor, considerando sensor com aproximação
M	máquina (elemento de processamento)
P	posição da mesa giratória
$supC_{\delta c}(M,G)$	$supC(M,G)$ para evento δ controlável
$supC_{\delta u}(M,G)$	$supC(M,G)$ para evento δ não controlável
$supC_{\delta f}(M,G)$	$supC(M,G)$ para evento δ forçável

Capítulo 1

Introdução

A teoria de controle moderno experimentou um desenvolvimento formidável no século XX, sustentado principalmente pelo desenvolvimento vertiginoso dos sistemas computacionais. Tradicionalmente, esta teoria lida com o comportamento dinâmico de processos cujas variáveis são numéricas e cuja evolução pode ser modelada por equações diferenciais ou a diferenças (Heymann, 1989).

Por outro lado, com o desenvolvimento e a difusão dos computadores, novos sistemas emergiram, com complexidade tal que não mais podem ser tratados pelos modelos convencionais. Também um número crescente de variáveis associadas aos novos processos não possuem representação numérica adequada, mas sim lógica (Heymann, 1989). Neste contexto, novas perspectivas de modelagem de sistemas foram desenvolvidas, dentre as quais destacam-se os modelos de sistemas a eventos discretos.

Um sistema a eventos discretos (SED) é um sistema dinâmico cujas mudanças de estado ocorrem em pontos discretos de tempo, dirigidas por eventos isolados e, em geral, assíncronos. Um modelo de sistemas a eventos discretos é, em geral, não determinístico, no sentido de que pode representar diversas transições de estados possíveis a partir de determinado estado, que são escolhidas e executadas por um mecanismo não necessariamente representado no modelo (Wonham, 2001).

Os princípios gerais que regem os sistemas a eventos discretos podem ser identificados em áreas diversas, tais como sistemas de manufatura, sistemas de tráfego, sistemas de gerenciamento de dados, protocolos de comunicação e sistemas logísticos (Wonham, 2001). Em função da variedade de áreas de aplicação e das especialidades que as suportam, diversas abordagens para modelagem, análise e controle de sistemas a eventos discretos foram desenvolvidas, enfocando diferentes tipos de problemas. Dentre estas, destacam-se hoje as abordagens de Cadeias de Markov (Çinlair, 1975), Redes de Petri (Cardoso e Valette, 1997), Redes de Filas (Kleinrock, 1975), Lógica Temporal (Manna e Pnueli, 1992) e Controle Supervisório (Ramadge e Wonham, 1987b; Wonham, 2001).

Dentre estas abordagens, o Controle Supervisório se destaca por fornecer uma metodologia para

projeto automático de sistemas de controle a partir de especificações de comportamento do sistema. Nesta abordagem, o sistema a ser controlado, denominado planta, é modelado como um gerador de eventos, dos quais alguns podem ser inibidos por ação de controle. A função do elemento de controle, denominado supervisor, é desabilitar um determinado conjunto de eventos, de forma a confinar o comportamento em malha fechada dentro de limites pré-estabelecidos. Na abordagem clássica desenvolvida por Ramadge e Wonham (1987b), o supervisor é projetado de forma a desabilitar um conjunto mínimo de eventos, baseado no comportamento passado da planta, de forma que se obtenha um comportamento ótimo, no sentido de ser o comportamento minimamente restritivo que se pode obter sem transgredir as especificações.

No estado atual da tecnologia de *software* para esta abordagem, tanto as especificações quanto a planta são modelados por máquinas de estados finitos (Hopcroft e Ullman, 1979), e os principais algoritmos para cálculo de soluções de controle apresentam complexidade polinomial no número de estados do modelo. Entretanto, sistemas complexos são modelados diretamente pela combinação de seus sistemas componentes. Por esta razão, a cada vez que um novo componente é adicionado, o número de estados do sistema cresce na razão do número de estados do novo componente. Assim, o tamanho do modelo cresce exponencialmente com o número de componentes, o que inviabiliza a obtenção de modelos de sistemas de grande porte e, portanto, a aplicação da abordagem.

Para tratar o problema de complexidade computacional inerente à abordagem clássica de Ramadge e Wonham, diversos refinamentos têm sido propostos. Alguns destes vão na direção de explorar regularidades internas da estrutura algébrica ou aritmética do sistema, como no caso da abordagem proposta por Eyzell e Cury (1998), que explora aspectos de simetria, e da abordagem BDD (Bryant, 1986). Seguindo outra estratégia, o refinamento da abordagem pode ser no sentido da proposição de arquiteturas mais refinadas, baseadas na modularidade do problema. Nas abordagens de controle hierárquico, (Zhong e Wonham, 1990; Wong e Wonham, 1992; Cunha e Cury, 2002; Torrico e Cury, 2002), o problema de controle global é decomposto verticalmente e resolvido em diferentes níveis de abstração. A modularidade do problema pode ser explorada também horizontalmente, a partir do aproveitamento de características modulares das especificações (Ramadge e Wonham, 1988), das plantas (Lin e Wonham, 1990), ou de ambas (Queiroz, 2000).

Na abordagem de controle modular, em vez de projetar um único supervisor responsável por restringir o comportamento da planta de acordo com múltiplas especificações, para cada especificação é projetado um supervisor modular. Neste caso, deseja-se que a ação conjunta dos supervisores sobre a planta garanta um comportamento, em malha fechada, equivalente ao comportamento obtido com o supervisor único (Ramadge e Wonham, 1988). Entretanto, assim como na abordagem clássica, nesta abordagem o cálculo dos supervisores exige que se obtenha um modelo único para o comportamento global do sistema sem controle.

A abordagem de controle modular local faz um refinamento do controle modular clássico, explorando também aspectos de modularidade da planta nos casos em que esta é composta por subsistemas

dirigidos por conjuntos disjuntos de eventos. Neste caso, para cada supervisor modular, o cálculo do comportamento ótimo é restrito ao conjunto de subsistemas da planta que são afetados pela especificação. Como resultado, obtêm-se uma estrutura de controle naturalmente descentralizada (Queiroz, 2000). No entanto, em diversos casos o comportamento desejado, modelado nas especificações, é desnecessariamente complexo, por englobar seqüências de eventos que contém uma semântica particular que poderia ser associada explicitamente a um novo evento da planta. Por analogia com um sistema de manufatura, este novo evento pode ser pensado como gerado por um *sensor* que identifica na planta a situação associada a uma determinada seqüência de eventos e reporta aos supervisores.

Por outro lado, propostas de generalização do modelo de Ramadge e Wonham na direção de maior realismo e flexibilidade de modelagem têm sido realizadas, com o objetivo de englobar novas classes de problemas que possam ser resolvidos formalmente. Dentre estas propostas, podem-se citar o controle com eventos forçáveis (Golaszewski e Ramadge, 1987), sistemas dinâmicos híbridos (Cury *et al.*, 1998), sistemas temporizados (Brandin e Wonham, 1994), e, mais recentemente, sistemas multitarefa (Queiroz *et al.*, 2004).

Em sistemas dinâmicos híbridos e sistemas temporizados, a necessidade de reduzir a complexidade do cálculo dos supervisores levou à introdução de problemas de controle supervisório aproximados (Cury *et al.*, 1998; Gohari e Wonham, 2000). Nestes casos, o cálculo de um supervisor para uma aproximação conservadora da planta garante que o comportamento do sistema em malha fechada é mantido sob limites estabelecidos pelas especificações, porém nem sempre garantindo que o comportamento obtido seja ótimo, quando comparado ao comportamento em malha fechada da planta sob o supervisor original.

1.1 Objetivos do Trabalho

O cerne da presente dissertação surgiu a partir do interesse na formalização de um mecanismo de comunicação entre supervisores modulares utilizado, em um exemplo particular, por Santos (2003). Após uma primeira análise, constatou-se que, naquele caso particular, este mecanismo de comunicação poderia ser adequadamente caracterizado como o comportamento de um sensor adicionado ao sistema.

Este trabalho objetiva discutir a modelagem de sensores em sistemas a eventos discretos, no contexto da abordagem modular local. Objetiva-se mostrar que, nos casos onde as especificações de comportamento contém seqüências de eventos com uma semântica particular associada, a abordagem modular local pode ser refinada tanto pela introdução explícita de sensores quanto pela emulação do comportamento de sensores na forma de especificações.

Na abordagem apresentada, o refinamento obtido com a introdução de sensores é expresso, quantitativamente, pela redução do número de estados das especificações originais, que, combinada com a

utilização de aproximação do comportamento dos sensores, permite que a complexidade computacional associada ao cálculo de cada supervisor modular local seja reduzida. O refinamento da arquitetura se traduz também na obtenção supervisores menores, bem como de uma solução modular local com menor número total de estados.

1.2 Organização do Trabalho

O capítulo 2 apresenta a fundamentação da teoria de sistemas a eventos discretos baseada em autômatos e linguagens. Neste capítulo, também são descritas três extensões desta teoria utilizadas nos capítulos seguintes: controle com eventos forçáveis, controle modular e controle modular local.

No capítulo 3, são apresentados resultados básicos sobre o uso de aproximação externa do comportamento de sistemas na teoria de controle de sistemas a eventos discretos. Para o caso particular no qual a aproximação é restrita apenas a alguns eventos do sistema, alguns resultados que serão utilizados nos capítulos seguintes são apresentados.

O capítulo 4 propõe uma abordagem para modelagem de sensores em sistemas a eventos discretos a partir da identificação das cadeias de eventos responsáveis pela ativação e desativação dos sensores. Também é discutida e ilustrada a necessidade de utilizar aproximações dos modelos dos sensores para que exista redução dos supervisores obtidos.

A partir do estabelecimento de condições sobre a modelagem dos sensores, o capítulo 5 mostra que, em determinadas situações, é possível obter solução equivalente àquela obtida com alocação de sensores reais em um sistema físico, por meio da emulação do comportamento dos sensores. Também é ilustrado que a emulação caracteriza uma situação adequada para a utilização da abordagem de eventos forçáveis.

Enfim, no capítulo 6 os principais resultados da dissertação são revisitados e são apontadas perspectivas de trabalhos futuros.

Capítulo 2

Sistemas a Eventos Discretos

Este capítulo tem, como eixo central, a apresentação da abordagem de autômatos e linguagens para a modelagem e controle de sistemas a eventos discretos, como introduzida por Ramadge e Wonham (1987b).

Nesta abordagem, um sistema é modelado por uma representação na qual as mudanças de estado ocorrem apenas em instantes discretos, em decorrência de eventos específicos e possivelmente assíncronos. Em contraste com as teorias de controle contínuo e discretizado, aqui a dinâmica do sistema não pode ser representada por equações regidas pelo tempo. Fundamental, nesta abordagem, é a representação do ordenamento dos eventos que geram as transições de estado, quer seja sob a forma de conjuntos de cadeias de eventos, quer sob a forma de máquinas de estado que reconhecem as sequências de eventos. É importante, também, destacar que nesta abordagem assume-se que dois eventos não ocorrem simultaneamente.

A seção 2.1 apresenta a fundamentação matemática básica para modelagem de sistemas a eventos discretos por linguagens e, equivalentemente, por autômatos de estados finitos. Em seguida, a seção 2.2 descreve a abordagem de controle monolítico de sistemas a eventos discretos (Ramadge e Wonham, 1987b), bem como três de suas principais extensões: controle com eventos forçáveis, controle modular, e controle modular local (Golaszewski e Ramadge, 1987; Ramadge e Wonham, 1988; Queiroz, 2000), fundamentais para o desenvolvimento posterior tratado neste trabalho.

2.1 Modelagem de Sistemas a Eventos Discretos

Sistemas físicos podem ser modelados matematicamente pelas seqüências de eventos que representam mudanças de estados lógicos do sistema. São exemplos típicos de eventos o início e o fim do processamento em uma máquina, uma mudança significativa em uma leitura de um sensor, a parada faltosa (quebra) de um equipamento, a conclusão do reparo de um equipamento, entre outros.

É interessante destacar que, embora eventos como a mudança do valor lido por um sensor possam occorer de forma contínua no sistema físico, para a modelagem adotada neste trabalho esta mudança é representada como lógica, podendo corresponder, por exemplo, ao cruzamento de um patamar pré-definido. Assim, a modelagem de um sistema físico por eventos discretos corresponde, em geral, a uma maior abstração do sistema do que quando o mesmo é modelado como sistema contínuo.

Na modelagem de sistema a eventos discretos, para cada evento do sistema é atribuído um símbolo. As possíveis trajetórias do sistema são, então, representadas por um conjunto de cadeias de símbolos denominada linguagem. O conceito de linguagens é, portanto, de importância fundamental para a teoria aqui introduzida.

2.1.1 Linguagens

Seja o *alfabeto* Σ, o conjunto de símbolos que representam eventos de um SED. Os elementos de Σ serão também, para efeito de simplificação, denominados *eventos*. Uma seqüência de transições de estado de um SED pode ser representada por uma *cadeia* de eventos de Σ - por exemplo, uma cadeia $s \in \Sigma^n$, $n \in \mathbb{N}$ representa uma seqüência de n transições de estado. O comprimento $n = |s|$ é denominado *cardinalidade* de s.

Seja Σ^* o conjunto de todas as cadeias finitas de eventos de Σ, incluindo a cadeia com zero eventos, representada por ε. Assim, uma *linguagem* definida sobre o alfabeto Σ é um conjunto de cadeias de eventos de Σ, ou seja, um subconjunto de Σ^*. O conjunto de linguagens finitas definidos sobre um alfabeto Σ é fechado para união, com elemento máximo Σ^*, bem como fechado para intersecção, com elemento mínimo $\emptyset$.

Dadas duas cadeias $s,t \in \Sigma^*$, $s \in \Sigma^n$ e $t \in \Sigma^m$, $n,m \in \mathbb{N}$, a *concatenação* de s e t, representada por st, é o elemento de Σ^{n+m} equivalente à seqüência de eventos s seguida por uma seqüência de eventos t.

O conceito de concatenação pode ser estendido também para linguagens: dadas linguagens $L_1 \subseteq \Sigma_1^*$ e $L_2 \subseteq \Sigma_2^*$, a concatenação de L_1 com L_2, é a operação que gera uma linguagem definida em $\Sigma_1 \cup \Sigma_2$, formada pelas concatenações de cadeias de L_1 com cadeias de L_2: $L_1L_2 = \{st \in (\Sigma_1 \cup \Sigma_2)^* | s \in L_1 \text{ e } t \in L_2\}$. Como caso particular, dados uma linguagem $L \subseteq \Sigma_i^*$ e um alfabeto Σ_j, utiliza-se a notação $L\Sigma_j$ para representar a concatenação de L com os eventos de Σ_j, ou seja, com cadeias de Σ_j^1: $L\Sigma_j = \{se \in (\Sigma_i \cup \Sigma_j)^* | s \in L \text{ e } e \in \Sigma_j\}$.

Uma cadeia s é *prefixo* de v, denotado por $s \preceq v$, se existe uma cadeia t tal que $st = v$. Ainda, se $t \neq \varepsilon$, s é um *prefixo próprio* de v, denotado por $s \prec v$.

Dada uma linguagem $L \subseteq \Sigma^*$, o *prefixo-fechamento* de L, ou apenas *fechamento*, denotado por $\overline{L}$, é a linguagem definida pelo conjunto de todos os prefixos das cadeias de L:

$\overline{L} = \{s \in \Sigma^* | \exists t \in \Sigma^* \text{ tal que } st \in L\}$.

Uma linguagem L é dita *prefixo-fechada* se $\overline{L} = L$.

Dadas uma linguagem $L \subseteq \Sigma^*$ e uma cadeia $t \in \overline{L}$, o conjunto de *eventos ativos* de L após t, representado por $\Sigma_L(t)$, é o subconjunto de Σ composto por eventos que, concatenados com t, geram prefixos de L: $\Sigma_L(t) = \{e \in \Sigma | te \in \overline{L}\}$. Note que, pela definição, se t não é um prefixo de uma cadeia de L, então $\Sigma_L(t) = \emptyset$.

Exemplo 2.1 (Linguagens e operações sobre cadeias)
Seja um alfabeto $\Sigma = \{\alpha, \beta, \gamma\}$. $L_1 = \{\varepsilon, \gamma\gamma\}$ e $L_2 = \{\alpha, \beta\beta\}$ são linguagens definidas sobre Σ. A concatenação de L_1 com L_2 é dada por $L_1L_2 = \{\alpha, \beta\beta, \gamma\gamma\alpha, \gamma\gamma\beta\beta\}$. O prefixo fechamento de L_1 é a linguagem $\overline{L_1} = \{\varepsilon, \gamma, \gamma\gamma\}$, enquanto para L_2 o prefixo fechamento é dado por $\overline{L_2} = \{\varepsilon, \alpha, \beta, \beta\beta\}$. Para L_2, os eventos ativos após ε são dados por $\Sigma_{L_2}(\varepsilon) = \{\alpha, \beta\}$, enquanto após a cadeia $\beta\beta$ são iguais a $\Sigma_{L_2}(\beta\beta) = \emptyset$. □

2.1.1.1 Projeções

Dados dois alfabetos Σ e Σ_i tais que $\Sigma_i \subseteq \Sigma$, a *projeção natural* de uma cadeia $s \in \Sigma^*$ em Σ_i é a operação, representada por $P_i(\cdot)$, que apaga da cadeia s todos os eventos que não pertencem a Σ_i. P_i é definida recursivamente por:

$P_i(\varepsilon) = \varepsilon$;

$$P_i(e) = \begin{cases} e, & \text{se} \quad e \in \Sigma_i \\ \varepsilon & \text{se} \quad e \notin \Sigma_i \end{cases} \quad \text{para todo } e \in \Sigma;$$

$P_i(se) = P_i(s)P_i(e)$, para todos $s \in \Sigma^*$, $e \in \Sigma$.

A definição de projeção pode ser estendida, também, para linguagens definidas em Σ. Dada uma linguagem $L \subseteq \Sigma^*$, a projeção de L em Σ_i é definida por:

$P_i(L) = \{s_i \in \Sigma_i^* | \exists s \in L \text{ tal que } P_i(s) = s_i\}$

Também define-se a *projeção inversa* de $L_i \subseteq \Sigma_i^*$ em Σ por:

$P_i^{-1}(L_i) = \{s \in \Sigma^* | P_i(s) \in L_i\}$

O subscrito em P_i será omitido quando a identificação do subalfabeto Σ_i for evidente no contexto.

A seguir, são apresentadas propriedades de projeções de linguagens que serão consideradas válidas ao longo desta dissertação. A demonstração destas propriedades pode ser obtidas em Queiroz (2000).

Propriedade 2.1 (Projeção e concatenação) *Para alfabetos Σ_i, Σ tais que $\Sigma_i \subseteq \Sigma$, sejam cadeias $s,t \in \Sigma^*$, e seja a projeção $P_i : \Sigma \to \Sigma_i$. Então $P_i(st) = P_i(s)P_i(t)$.*

Propriedade 2.2 (Projeção e prefixo-fechamento) *Para alfabetos Σ_i, Σ tais que $\Sigma_i \subseteq \Sigma$, sejam $L \subseteq \Sigma^*$ e $P_i : \Sigma \to \Sigma_i$. Então $\overline{P_i(L)} = P_i(\overline{L})$.*

Propriedade 2.3 (Projeção inversa e concatenação) *Para alfabetos Σ_i, Σ tais que $\Sigma_i \subseteq \Sigma$, sejam cadeias $s,t \in \Sigma_i^*$, e seja a projeção $P_i : \Sigma \to \Sigma_i$. Então $P_i^{-1}(st) = P_i^{-1}(s)P_i^{-1}(t)$.*

Propriedade 2.4 (Projeção inversa e prefixo-fechamento) *Para alfabetos Σ_i, Σ tais que $\Sigma_i \subseteq \Sigma$, sejam $L_i \subseteq \Sigma_i^*$ e $P_i : \Sigma \to \Sigma_i$. Então $\overline{P_i^{-1}(L_i)} = P_i^{-1}(\overline{L_i})$.*

Propriedade 2.5 (Projeção inversa e intersecção) *Para alfabetos Σ_i, Σ tais que $\Sigma_i \subseteq \Sigma$, sejam as linguagens $L_1, L_2 \subseteq \Sigma_i^*$ e a projeção $P_i : \Sigma \to \Sigma_i$. Então, $P_i^{-1}(L_1) \cap P_i^{-1}(L_2) = P_i^{-1}(L_1 \cap L_2)$.*

2.1.2 Autômatos de Estados Finitos

Embora linguagens sejam compostas por cadeias finitas, uma linguagem pode conter um número infinito de cadeias. No entanto, para que computações envolvendo linguagens possam ser realizadas, representações finitas são necessárias. Uma alternativa para isso é a representação de linguagens por expressões regulares, que são representações compactas das cadeias possíveis nas linguagens através de eventos e operações como prefixo-fechamento e concatenação. Uma linguagem que pode ser representada por uma expressão regular é denominada linguagem regular (Hopcroft e Ullman, 1979).

Quando um sistema a eventos discretos pode ser descrito por uma linguagem regular, é possível obter uma representação do sistema por um diagrama de transição de estados finitos, cuja representação matemática é denominada autômato. Neste caso, cada transição de estado é associada a um evento, e as possíveis seqüências de eventos geradas pelo diagrama correspondem à linguagem do SED.

2.1.2.1 Autômatos

Um sistema a eventos discretos pode ser representado por um *autômato A*, definido como uma quíntupla $A = (Q, \Sigma, q_0, f, Q_m)$, onde Q é o conjunto de estados do sistema, Σ é o conjunto de eventos que etiquetam as transições de estado, $q_0 \in Q$ é o estado inicial do sistema, $f : Q \times \Sigma \to Q$ é a função de transição (etiquetada) de estados e $Q_m \subseteq Q$ é o conjunto de estados marcados. Neste trabalho, o interesse é focado sobre autômatos de estados finitos, para os quais os conjuntos Q e Σ são finitos.

A função de transição associa, a cada par $(q_i, e) \in Q \times \Sigma$, um estado $q_j = f(q_i, e)$. A atribuição $f(q_i, e) = q_j$ indica que, a partir do estado q_i, o sistema pode evoluir para o estado q_j pela ocorrência da transição etiquetada pelo evento e. Ainda, a função de transição de estados pode ser estendida recursivamente para qualquer cadeia definida em Σ^*:

$$\begin{aligned} f(q, \varepsilon) &= q; \\ f(q, se) &= f(f(q, s), e). \end{aligned}$$

Assim, para $s \in \Sigma^*$, $f(q, s)$ representa o estado atingido quando, a partir do estado q, é executada a seqüência de transições de estados indicada pelos eventos de s.

Os estados marcados servem para distinguir as cadeias que tem algum significado especial, associado ao completamento de uma tarefa, como por exemplo o completamento de um ciclo de trabalho ou de uma seqüência de ciclos.

Um autômato A, como definido anteriormente, é dito *determinista*, pois cada estado apresenta apenas uma transição de saída para cada etiqueta. Alternativamente, pode-se definir uma função de transição generalizada $f' : Q \times \Sigma \to 2^Q$, cujo contradomínio é o conjunto das partes de Q. Neste caso, um estado q do autômato pode possuir a etiqueta e associada a um conjunto de estados $f'(q, e)$, ou seja, associada a mais de uma transição de saída. A estrutura $A' = (Q, \Sigma, q_0, f', Q_m)$ é denominada *autômato não determinista*.

2.1.2.2 Geradores

Na representação de um sistema a eventos discretos, pode não ser necessário que todos os eventos sejam associados a transições de saída de cada estado, pois a ocorrência de determinados eventos em alguns estados não faz sentido. Por exemplo, em um estado que represente uma máquina ociosa, o evento que representa o término do processamento nesta máquina não pode ocorrer.

Para modelar sistemas com esta propriedade, pode-se fazer uso de uma generalização de autômatos denominada *gerador*. Um gerador é uma estrutura de dados $G = (Q, \Sigma, q_0, f, Q_m)$, onde todos os elementos de G são definidos como para autômatos, porém f é uma função parcial. A notação $f(q, e)!$ é utilizada para indicar que a função f está definida para o par (q, e), ou seja, que o evento e pode ocorrer no estado q.

É interessante destacar que G pode ser representado como um autômato não determinista. Para tanto, basta que a função de transição $f' : Q \times \Sigma \to 2^Q$ seja definida pela associação, a cada par (q, e), de um conjunto com zero ou um estado.

A função de transição de estados de um gerador pode ser, recursivamente, estendida para qualquer cadeia definida em Σ^*:

$$f(q,\varepsilon) = q;$$
$$f(q,se) = f(f(q,s),e), \quad \text{se e somente se} \quad q' = f(q,s)! \quad \text{e} \quad f(q',e)!.$$

Se todas as transições de estado indicadas por uma cadeia s estão definidas, ou seja, $f(q,s)!$, diz-se que o gerador G *reconhece*, ou *gera*, a cadeia s.

Um gerador G pode ser representado por um grafo direcionado, denominado *diagrama de transição de estados*, no qual os nós representam os estados Q e os arcos representam as transições de estado, etiquetadas de acordo com a função f. Os estados marcados são identificados por linhas duplas em seu desenho, enquanto o estado inicial é identificado por uma seta curta.

Dados $q_i \in Q$ e $e_j \in \Sigma$, diz-se que e_j está em *self-loop* (auto-laço) no estado q_i quando $f(q_i,e_j) = q_i$. Esta situação é representada por uma transição que retorna ao estado q_i, etiquetada por e_j.

Exemplo 2.2 (Autômatos e Geradores)
Seja o autômato $A = (Q,\Sigma,q_0,f,Q_m)$, onde $Q = \{x_0,x_1,x_2\}$, $q_0 = x_0$, $\Sigma = \{\alpha,\beta,\gamma\}$, $Q_m = \{x_1\}$ e f é dada por: $f(x_0,\alpha) = x_0$; $f(x_1,\alpha) = x_1$; $f(x_2,\alpha) = x_2$; $f(x_0,\beta) = x_1$; $f(x_1,\beta) = x_2$; $f(x_2,\beta) = x_0$; $f(x_0,\gamma) = x_2$; $f(x_1,\gamma) = x_0$; $f(x_2,\gamma) = x_1$. A figura 2.1 apresenta o diagrama de transição de estados de A.

Seja agora um gerador $G = (Q,\Sigma,q_0,g,Q_m)$, onde Q, q_0, Σ, e Q_m são como definidos anteriormente, e a função g é definida como uma função parcial: $g(x_2,\alpha) = x_2$; $g(x_0,\beta) = x_1$; $g(x_1,\beta) = x_2$; $g(x_0,\gamma) = x_2$; $g(x_1,\gamma) = x_0$; $g(x_2,\gamma) = x_1$. O gerador $G = (Q,\Sigma,q_0,g,Q_m)$ é ilustrado na figura 2.2. □

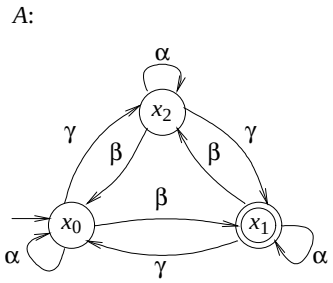

Figura 2.1: Diagrama de transição de estados para o autômato A.

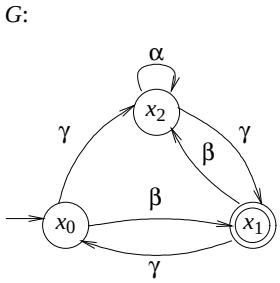

Figura 2.2: Diagrama de transição de estados para o gerador G.

2.1.2.3 Linguagens Associadas a Geradores

Um gerador G corresponde a uma descrição estrutural de um SED, centrada nos estados. De um ponto de vista externo, ou comportamental, o mesmo sistema pode ser descrito pelas diferentes cadeias de eventos que etiquetam as transições de estados. Assim, um SED pode ser expresso por duas linguagens:

- *Linguagem Gerada*, $L(G) : \{s \in \Sigma^* | f(q_0,s)!\}$; e

- *Linguagem Marcada*, $L_m(G) : \{s \in \Sigma^* | f(q_0, s) \in Q_m\}$.

A linguagem gerada representa todas as seqüências de eventos que podem ocorrer no SED, a partir do estado inicial. Por sua vez, a linguagem marcada é interpretada como o conjunto de cadeias da linguagem gerada que correspondem a tarefas completas.

Considerando ainda a definição de eventos ativos após uma cadeia, para a linguagem gerada $L(G)$, $\Sigma_{L(G)}(t)$ é o conjunto de eventos que podem ocorrer no estado $f(q_0,t)$ de G. Assim, $\Sigma_{L(G)}(t) = \{e \in \Sigma | f(q_0,te)!\}$.

Alternativamente, se um SED é descrito inicialmente pelas linguagens gerada e marcada, e se estas linguagens são regulares, é possível obter uma representação do SED por um gerador. Embora esta representação não seja única, existe uma representação para o SED com número mínimo de estados, que pode ser obtida através do procedimento de *minimização de geradores* (Hopcroft e Ullman, 1979). Ao longo deste documento, é assumido que os geradores associados às linguagens são mínimos.

2.1.2.4 Acessibilidade e Coacessibilidade

Dado um gerador $G = (Q, \Sigma, q_0, g, Q_m)$, um estado $q \in Q$ é dito *acessível* se existe uma cadeia s definida em Σ tal que $f(q_0, s) = q$. Caso todos os estados de um gerador sejam acessíveis, o gerador é dito acessível. O subconjunto de estados acessíveis de Q representa os estados que podem ser alcançados a partir do estado inicial.

Além disto, um estado $q \in Q$ é dito *coacessível* se existe uma cadeia s definida em Σ tal que $f(q,s) \in Q_m$. A coacessibilidade de um estado significa que, a partir dele, é possível atingir um estado marcado.

Caso todos os estados de um gerador sejam coacessíveis, o gerador é dito coacessível. Um gerador acessível e coacessível é dito aparado, ou *trim*.

Propriedade 2.6 (Coacessibilidade e linguagem marcada) *Um gerador G acessível é coacessível se e somente se* $\overline{L_m(G)} = L(G)$.

Exemplo 2.3 (Acessibilidade e Coacessibilidade)
Seja o gerador $G = (Q, \Sigma, q_0, f, Q_m)$, representado na figura 2.3. O estado x_3 não é acessível, enquanto os estados x_2 e x_4 não são co-acessíveis. As figuras 2.4, 2.5 e 2.6 representam, respectivamente, as componentes coacessível G_{coac}, acessível G_{ac} e *trim* G_{trim} de G, obtidas pela eliminação dos estados não acessíveis e não coacessíveis. □

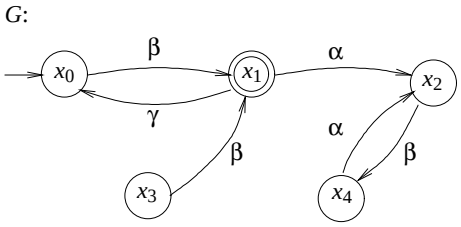

Figura 2.3: Gerador não acessível e não coacessível.

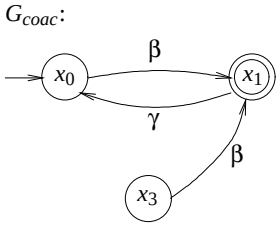

Figura 2.4: Componente coacessível de G.

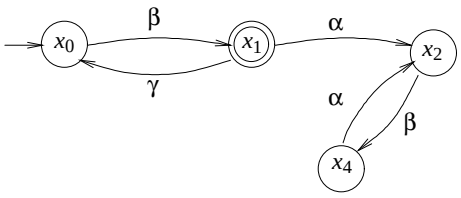

Figura 2.5: Componente acessível de G.

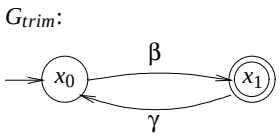

Figura 2.6: Componente *trim* de G.

2.1.3 Composição de Sistemas

A representação de SEDs por geradores permite que uma representação finita de um sistema seja obtida através da composição matemática das representações de seus subsistemas. Esta composição é obtida por uma operação, descrita a seguir, que preserva o sincronismo entre eventos comuns aos alfabetos.

Sejam duas linguagens $L_1 \subseteq \Sigma_1^*$ e $L_2 \subseteq \Sigma_2^*$, e seja o alfabeto $\Sigma = \Sigma_1 \cup \Sigma_2$. Sejam também as projeções $P_i : \Sigma \to \Sigma_i$, $i = 1,2$. O *produto síncrono* de L_1 e L_2, representado por $L_1||L_2$, é definido por $L_1||L_2 = P_1^{-1}(L_1) \cap P_2^{-1}(L_2) \subseteq (\Sigma_1 \cup \Sigma_2)^*$.

É possível também definir o produto síncrono como uma operação de composição de geradores. Dados dois geradores $G_i = (Q_i, \Sigma_i, q_{0,i}, f_i, Q_{m,i})$, i=1,2, a composição síncrona de G_1 e G_2 é definida por $G_1||G_2 = \{Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, (q_{0,1} \times q_{0,2}), f, Q_{m,1} \times Q_{m,2}\}$. A função de transição f é definida por

$$f((x,y),e) = \begin{cases} (f_1(x,e),y), & \text{se} \quad e \in \Sigma_1 - \Sigma_2 \quad \text{e} \quad f_1(x,e)! \\ (x, f_2(y,e)), & \text{se} \quad e \in \Sigma_2 - \Sigma_1 \quad \text{e} \quad f_2(x,e)! \\ (f_1(x,e), f_2(y,e)), & \text{se} \quad e \in \Sigma_1 \cap \Sigma_2 \quad , \quad f_1(x,e)! \quad \text{e} \quad f_2(x,e)! \\ \text{não definida}, & \text{para os outros casos.} \end{cases}$$

para todos $x \in Q_1$, $y \in Q_2$ e $e \in \Sigma_1 \cup \Sigma_2$.

Assim, é possível obter a representação de um sistema G através da composição síncrona de seus componentes. É importante notar que $G = G_1||G_2$ se e somente se $L(G) = L(G_1)||L(G_2)$ e $L_m(G) = L_m(G_1)||L_m(G_2)$.

Para o caso particular $\Sigma_1 \cap \Sigma_2 = \emptyset$, $G = G_1||G_2$ é denominada *composição assíncrona* de G_1 e G_2, podendo ser denotada também por $G_1||_{as}G_2$. Equivalentemente, o *produto assíncrono* de duas

linguagens definidas em alfabetos disjuntos pode ser denotado por $L_1 \|_{as} L_2$. Plantas $G_i \subseteq \Sigma_i^*$, $i = 1, \ldots, n$ são ditas assíncronas se $\forall j, k = 1, \ldots, n$, $\Sigma_j \cap \Sigma_k = \emptyset$.

A seguir, são apresentadas propriedades do produto síncrono de linguagens, extraídas de Wonham (2001) e Queiroz (2000), que serão consideradas válidas ao longo deste trabalho.

Propriedade 2.7 *Para alfabetos $\Sigma = \Sigma_1 \cup \Sigma_2$, sejam a linguagem $L_1 \subseteq \Sigma_1^*$ e a projeção $P_1 : \Sigma \to \Sigma_1$. Então $P_1^{-1}(L_1) = L_1 \| \Sigma^* = L_1 \| (\Sigma - \Sigma_1)^* = L_1 \| \Sigma_2^* = L_1 \| (\Sigma_2 - \Sigma_1)^*$.*

Propriedade 2.8 *Para alfabetos $\Sigma = \Sigma_1 \cup \Sigma_2$, sejam as linguagens $L_1 \subseteq \Sigma_1^*$ e $L_2 \subseteq \Sigma_2^*$. Então $L_1 \| L_2 = (L_1 \| (\Sigma_2 - \Sigma_1)^*) \cap (L_2 \| (\Sigma_1 - \Sigma_2)^*)$.*

Propriedade 2.9 (Associatividade do produto síncrono) *Para alfabetos $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3$, sejam linguagens $L_1 \subseteq \Sigma_1^*$, $L_2 \subseteq \Sigma_2^*$ e $L_3 \subseteq \Sigma_3^*$. Então $L_1 \| (L_2 \| L_3) = (L_1 \| L_2) \| L_3$.*

Propriedade 2.10 (Projeção de um produto síncrono) *Para alfabetos $\Sigma_0 \subseteq \Sigma_1 \cup \Sigma_2 = \Sigma$, sejam linguagens $L_1 \subseteq \Sigma_1^*$ e $L_2 \subseteq \Sigma_2^*$. Então $P_0(L_1 \| L_2) \subseteq P_0(L_1) \| P_0(L_2)$. Além disso, se $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma_0$, então $P_0(L_1 \| L_2) = P_0(L_1) \| P_0(L_2)$.*

Propriedade 2.11 (Modularidade de linguagens assíncronas) *Para alfabetos $\Sigma_1 \cup \Sigma_2 = \Sigma$, sejam linguagens $L_1 \subseteq \Sigma_1^*$ e $L_2 \subseteq \Sigma_2^*$. Então $\overline{L_1 \| L_2} \subseteq \overline{L_1} \| \overline{L_2}$. Além disso, se $\Sigma_1 \cap \Sigma_2 = \emptyset$, então $\overline{L_1 \| L_2} = \overline{L_1} \| \overline{L_2}$.*

2.1.3.1 Representação por Sistemas Produto

A obtenção da representação de sistemas através da composição de seus subsistemas por produto síncrono apresenta, como principal desvantagem, o crescimento exponencial do número de estados do sistema composto. Em uma análise de pior caso, o gerador que representa o sistema composto poderá apresentar um número de estados igual ao produto dos números de estados dos geradores dos subsistemas. Para sistemas de grande porte, esta explosão no número de estados pode inviabilizar a adoção de abordagens que dependam da obtenção da representação por autômatos do sistema como um todo, como por exemplo a abordagem para síntese de lógica de controle de Ramadge e Wonham (1987b).

Seja um sistema composto por subplantas modeladas como geradores G_i', $i = 1, \ldots, n'$, possivelmente síncronos. A representação deste sistema por um conjunto de plantas assíncronas, denominada *Representação por Sistemas Produto* (RSP) (Ramadge, 1989; Ramadge e Wonham, 1989), pode ser obtida a partir da agregação de plantas G_i' em subconjuntos assíncronos. Assim, uma RSP é um conjunto $\{G_i \subseteq \Sigma_i^* | G_i = \|_{i \in 1, \ldots, n'} G_i' \text{ e } j \neq i \Rightarrow \Sigma_j \cap \Sigma_i = \emptyset\}$. Queiroz (2000) ilustra que, embora esta

representação não seja necessariamente única, o conjunto de todas as RSPs possíveis para o sistema possui um único elemento de maior cardinalidade (maior número de subsistemas assíncronos), que corresponde à RSP mais refinada.

Exemplo 2.4 (Representação por sistemas produto)
Sejam as plantas: G'_i, i=1,...,5, com alfabetos Σ'_i definidos conforme o diagrama da figura 2.7. Neste caso, a RSP mais refinada possui quatro elementos, G_i, $i = 1,\ldots,4$, definidos por: $G_1 = G'_1$, $G_2 = G'_2 \| G'_3$, $G_3 = G'_4$, $G_4 = G'_5$.

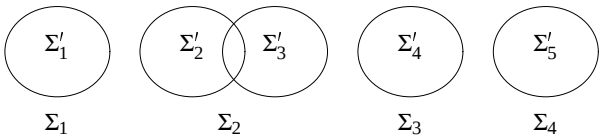

Figura 2.7: Alfabetos da representação original e da representação por sistemas produto mais refinada.

Note que também $\{G'_1, G'_2 \| G'_3, G'_4 \| G'_5\}$ ou $\{G'_1 \| G'_2 \| G'_3, G'_4, G'_5\}$ poderiam ser adotadas como RSPs para o sistema apresentado, porém ambas possuem um menor número de subplantas. □

2.2 Controle de Sistemas a Eventos Discretos

2.2.1 Abordagem Clássica

A teoria de controle de Sistemas a Eventos Discretos baseada em linguagens controláveis foi introduzida por Ramadge e Wonham (1987b). Nesta abordagem, o sistema a ser controlado, denominado *planta*, gera espontaneamente eventos, que são registrados por um elemento de controle denominado *supervisor*. Baseado na observação das seqüências de eventos geradas pela planta, o supervisor tem a função de desabilitar a ocorrência de eventos na planta, de forma a manter o comportamento controlado da planta dentro de limites pré-estabelecidos. Este tipo de controle é dito permissivo, no sentido de que os eventos inibidos não podem ocorrer e os habilitados não ocorrem obrigatoriamente. O funcionamento do sistema sob supervisão é ilustrado na figura 2.8.

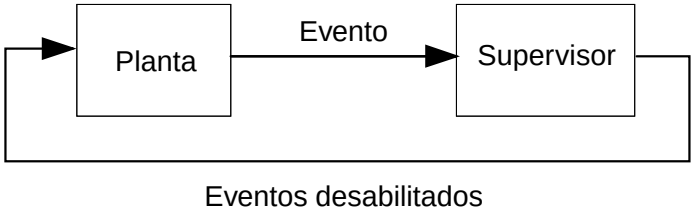

Figura 2.8: Ação de controle de um supervisor sobre uma planta.

Nesta abordagem, considera-se que apenas parte dos eventos definidos no alfabeto da planta pode ser desabilitada externamente. Assim, na modelagem da função de controle do supervisor *S*, o conjunto de eventos Σ é particionado em $\Sigma = \Sigma_c \cup \Sigma_u$, onde Σ_c contém *eventos controláveis*, que podem ser desabilitados pela ação do supervisor, enquanto os elementos de Σ_u, denominados *eventos não controláveis*, são aqueles cuja ocorrência não pode ser evitada.

2.2.1.1 Modelagem dos Supervisores

A definição dos eventos que estão habilitados a ocorrer em um dado momento é realizada através da determinação de entradas de controle para a planta. Uma *entrada de controle* para o gerador *G* é um conjunto de eventos $\gamma \subseteq \Sigma$ que respeita a restrição $\Sigma_u \subseteq \gamma$. A condição $\Sigma_u \subseteq \gamma$ indica que os eventos não controláveis não podem ser desabilitados.

Seja Γ o conjunto de entrada de controles admissíveis, $2^{\Sigma_u} \subseteq \Gamma \subseteq 2^{\Sigma}$. Um *supervisor* é definido formalmente como uma função $h : L(G) \to \Gamma$ que associa a cada cadeia $s \in L(G)$, gerada pela planta, uma entrada de controle $\gamma = h(s)$ que restringe o comportamento da planta até a ocorrência de um novo evento.

O comportamento do sistema sob supervisão, denotado por $h|G$, é dado pela linguagem $L(h|G) \subseteq L(G)$, definida recursivamente por:

$\varepsilon \in L(h|G)$;

$se \in L(h|G)$ se e somente se $s \in L(h|G)$, $se \in L(G)$ e $e \in h(s)$.

Assim, em malha fechada, após a geração de uma cadeia *s*, o próximo evento será um elemento $e \in \Sigma$ tal que $e \in h(s)$ e $f(x_0, se)!$.

A linguagem $L_m(h|G)$, denominada *comportamento marcado de* $h|G$, é a parte de *G* que sobrevive à supervisão: $L_m(h|G) = L(h|G) \cap L_m(G)$. $L_m(h|G)$ representa o conjunto de tarefas realizadas sob supervisão.

Um supervisor *h* é dito *não bloqueante* para *G*, ou *próprio*, se $\overline{L_m(h|G)} = L(h|G)$. Assim, um supervisor é não bloqueante quando a linguagem gerada pelo sistema sob supervisão é coacessível, o que significa que, a partir de qualquer estado alcançável, o sistema sob supervisão sempre poderá evoluir até o completamento de uma tarefa.

2.2.1.2 Representação de Supervisores como Autômatos

Um supervisor *h* sobre uma planta *G* pode ser representado como uma estrutura de dados finita (T, Φ), denominada *representação por realização de estado*, onde $T = (\Sigma, Y, g, y_0, Y)$ é um autômato

definido sobre o mesmo alfabeto de G, e $\Phi : Y \rightarrow \Gamma$ é uma função que mapeia os estados de T em entradas de controle para G. O valor da função h para uma certa cadeia s gerada por G é obtido aplicando-se s ao autômato T, a partir de seu estado inicial, obtendo-se o estado $y' = g(y_0, s)$. Em seguida, a função Φ fornece a entrada de controle que deve ser aplicada após a cadeia s: $h(s) = \gamma = \Phi(y')$. Assim, dado um supervisor h, o par (T, Φ) realiza h se, para cada $s \in L(h|G)$, $\gamma = \Phi(y) = \Phi(g(y_0, s)) = h(s)$.

Na prática, um supervisor h pode ser representado por um gerador S, de tal forma que a ação de controle sobre G esteja implícita na estrutura de transição de S. Assim, transições não habilitadas por h não aparecem na estrutura de transição de S, e transições habilitadas por h e fisicamente possíveis, aparecem na estrutura de transição de S.

Assim, dado um supervisor h, Seja $S = (\Sigma, Y, g, y_0, Y)$ um gerador que represente h. Se $s \in L(h|G)$ e $e \in \Sigma$, as seguintes regras são válidas para a linguagem gerada por S:

$s \in L(S)$;

$e \in h(s)$ e $se \in L(G) \Rightarrow se \in L(S)$.

A ação de controle de S consiste em, uma vez estando em um estado $y \in Y$, desabilitar todos os eventos e_i não definidos em y, ou seja, tais que $g(y, e_i)!$ não é verdadeiro.

Embora seja possível representar ambos o supervisor e a planta por geradores S e G, existe uma diferença fundamental na semântica dos eventos para os dois geradores: em G, os eventos ocorrem espontaneamente, respeitadas as restrições impostas pela ação de controle, enquanto S tem seus eventos dirigidos pelos eventos de G.

Quando um supervisor h é representado por um gerador S, o comportamento $h|G$ pode ser calculado pela composição síncrona de S e G:

$h|G = S||G = (\Sigma, Y \times Q, \theta, (y_0, x_0), Y \times Q_m)$, onde

$$\theta((y,q),e) = \begin{cases} (g(y,e), f(q,e)), & \text{se } g(y,e)! \text{ e } f(q,e)! \\ \text{não definida}, & \text{caso contrário.} \end{cases}$$

O exemplo a seguir, adaptado a partir de Wonham (2001), ilustra a representação de supervisores por autômatos.

Exemplo 2.5 (Supervisor para alocação de dois recursos para dois usuários)
Seja o problema de controle de alocação de dois recursos para dois usuários. O usuário 1, para executar sua tarefa, aloca o recurso 1 (evento a_{11}), para em seguida alocar o recurso 2 (evento a_{12}). A devolução dos recursos ocorre em ordem inversa, representada pela seqüência r_{12} e r_{11}. Por sua vez, o usuário 2 aloca primeiramente o recurso 2 (evento a_{22}), em seguida o recurso 1 (evento a_{21}), devolvendo-os também em ordem inversa. Para ambos, apenas a alocação do recurso pode ser evitada

por um sistema de controle, ou seja, apenas os eventos a_{ji} são controláveis. O comportamento dos dois usuários é modelado pelos geradores G_1 e G_2, apresentados na figura 2.9.

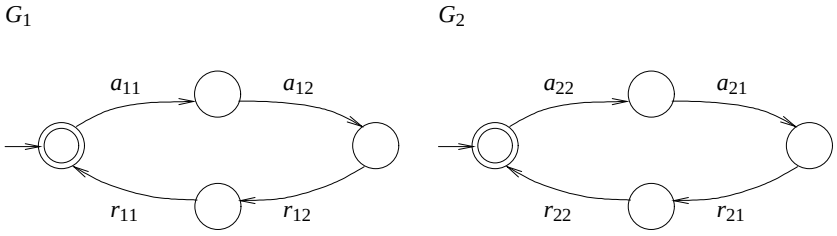

Figura 2.9: Usuários 1 e 2.

A planta sem controle é representada pelo comportamento conjunto dos dois usuários, modelado pelo gerador $G = G_1 \| G_2$ apresentado na figura 2.10. Este comportamento, no entanto, apresenta estados onde os recursos estariam alocados simultaneamente para os dois usuários (estados localizados dentro da região destacada). Um supervisor S para evitar que estes estados sejam atingidos pode ser obtido através da eliminação destes estados e das transições que levam a eles na estrutura de G, conforme apresentado na figura 2.11. Como, neste caso, o comportamento de S está incluso no comportamento de G, o sistema em malha fechada é dado por $S \| G = S$.

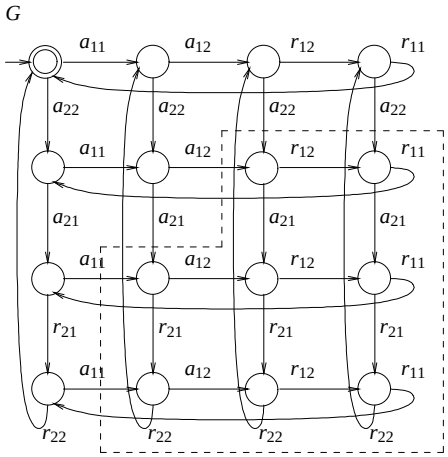

Figura 2.10: Comportamento conjunto dos usuários.

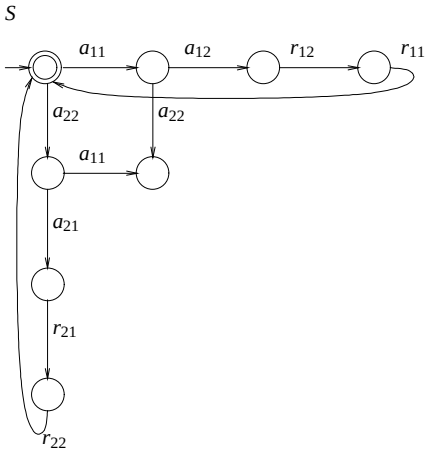

Figura 2.11: Supervisor bloqueante para G.

Neste caso, o supervisor foi obtido diretamente da eliminação de estados proibidos da estrutura da planta. No entanto, o comportamento do sistema sob supervisão $h|G = S\|G$ não é próprio, já que as cadeias $a_{11}a_{22}$ e $a_{22}a_{11}$ levam a um estado não coacessível. Este comportamento, não desejado, pode ser evitado através da eliminação do estado que corresponde ao bloqueio no sistema em malha fechada, ou seja, bastando que se tome como supervisor a componente *trim* de $S\|G$, apresentada na figura 2.12. □

O exemplo 2.5 sugere que procedimentos adequados para o projeto de supervisores podem ser desenvolvidos. No entanto, ainda é necessário que sejam discutidas as condições sob as quais um

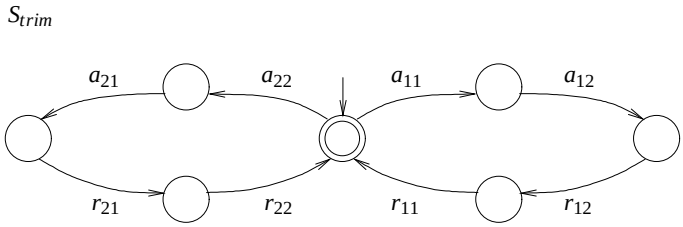

Figura 2.12: Supervisor não bloqueante para G.

dado comportamento em malha fechada pode ser obtido por supervisão. Além disso, é necessário que a especificação do comportamento desejado seja descrita formalmente. Estes tópicos são discutidos no item a seguir.

2.2.1.3 Existência de Supervisores

Dado um sistema a eventos discretos G com um comportamento descrito pelas linguagens $L(G)$, $L_m(G) \subseteq \Sigma^*$, deseja-se saber quais comportamentos podem ser obtidos por ação de algum supervisor. O comportamento desejado para G pode ser expresso de duas formas:

- Restringir o funcionamento através da especificação de uma linguagem-alvo $K \subseteq L(G)$ que representa o comportamento fisicamente possível desejado sob supervisão. Assim, deseja-se encontrar, se possível, um supervisor tal que o comportamento do sistema em malha fechada satisfaça $L(h|G) = K$.
- Modificar o funcionamento marcado preservando a propriedade de não-bloqueio. Para tanto, pode-se especificar uma linguagem-alvo $K \subseteq L_m(G)$ que representa as tarefas que se desejam ser completáveis sob supervisão. Deseja-se encontrar, se possível, um supervisor não bloqueante tal que o comportamento em malha fechada satisfaça $L_m(h|G) = K$.

Na prática, pode ser interessante expressar a especificação de comportamento do sistema em função de um subconjunto de eventos de Σ. Neste caso, dada uma *especificação genérica* $E \subseteq \Sigma_i^*$, com $\Sigma_i \subseteq \Sigma$, é possível expressá-la no alfabeto Σ através de sua projeção inversa: $P_i^{-1}(E)$, onde $P_i : \Sigma \to \Sigma_i$. Além disso, pode ser mais prático expressar o comportamento desejável como uma linguagem não contida em $L(G)$ (ou $L_m(G)$, no caso da especificação de comportamento marcado), o que é usualmente o caso de $P_i^{-1}(E)$. Neste caso, a especificação genérica pode ser adaptada para uma *especificação global* $K \subseteq L(G)$ (ou $K \subseteq L_m(G)$) que representa o comportamento do sistema restrito à especificação: $K = P_i^{-1}(E) \cap L(G) = E \| L(G)$ (ou $E \| L_m(G)$).

Para que seja possível determinar se um comportamento descrito por uma especificação global pode ser obtido por supervisão, foram introduzidas as propriedades de controlabilidade e L_m-fechamento de linguagens (Ramadge e Wonham, 1987b).

Definição 2.1 (Controlabilidade) Uma linguagem $K \subseteq \Sigma^*$ é dita *controlável* em relação a uma linguagem $L \subseteq \Sigma^*$ se:

$$\overline{K}\Sigma_u \cap L \subseteq \overline{K} \qquad \square$$

A controlabilidade em relação a uma linguagem L é também denominada *L-controlabilidade.* Além disto, pode-se omitir a referência à linguagem, caso L esteja implícita no contexto.

Assim, para uma linguagem K controlável com relação à L, a concatenação de qualquer prefixo s de K com um evento não controlável e_u, se_u, deve ser um prefixo de K, se for uma cadeia de L. Como casos particulares, $\emptyset$, L e Σ^* são linguagens controláveis em relação a L.

Dada uma planta G com alfabeto Σ e uma linguagem $M \subseteq \Sigma^*$, a classe de linguagens contidas em M e controláveis com relação a $L(G)$ é denotada por $C(M,G) = \{K | K \subseteq M$ e K é controlável em relação a $L(G)\}$. Esta classe é fechada para união, contendo portanto um elemento supremo, denotado $supC(M,G)$. O algoritmo para cálculo de $supC(M,G)$ a partir de M e G pode ser obtido em Wonham e Ramadge (1987).

Definição 2.2 (L_m-Fechamento) Uma linguagem $K \subseteq \Sigma^*$ é dita $L_m(G)$*-fechada* se:

$$K = \overline{K} \cap L_m(G) \qquad \square$$

Uma linguagem é $L_m(G)$-Fechada se todos os seus prefixos que são palavras de $L_m(G)$ forem também palavras de K. É importante ressaltar que a definição de L_m-fechamento implica $K \subseteq L_m(G)$.

Os conceitos de controlabilidade e L_m-Fechamento são fundamentais para a existência de um supervisor não bloqueante, como apresentado no Teorema 2.1.

Teorema 2.1 (Existência de supervisores, linguagem marcada) *(Ramadge e Wonham, 1987b)* Dados um gerador G, com linguagem marcada $L_m(G)$, e uma linguagem $K \subseteq L_m(G)$ não vazia, existe um supervisor não bloqueante h tal que $L_m(h|G) = K$ se e somente se K é $L_m(G)$-fechada e $L(G)$-controlável.

O problema formulado em termos de linguagem gerada ($K \subseteq L(G)$) pode ser visto como um caso particular do problema formulado em termos de linguagem marcada, no qual todos os estados da planta são marcados, conforme apresentado no Corolário 2.1.

Corolário 2.1 (Existência de supervisores, linguagem gerada) *(Ramadge e Wonham, 1987b)* Dados um gerador G, com linguagem gerada $L(G)$, e uma linguagem $K \subseteq L(G)$ não vazia, existe um supervisor não bloqueante h tal que $L(h|G) = K$ se e somente se K é prefixo-fechada e controlável.

Ainda, é possível introduzir uma generalização do controle supervisório não bloqueante na qual a ação do supervisor inclui, além do controle, marcação de estados. Neste caso, seja uma linguagem $K \subseteq L_m(G)$. Um *supervisor marcador* para G é definido novamente como uma função $h : L(G) \to \Gamma$. No entanto, o comportamento marcado de $h|G$ é agora definido por $L_m(h|G) = L(h|G) \cap K$ (em vez de $L_m(h|G) = L(h|G) \cap L_m(G)$). Neste caso, para que uma tarefa seja reconhecida no comportamento em malha fechada, além de ser uma tarefa definida originalmente em $L_m(G)$ e sobreviver à supervisão, esta tarefa deve também estar definida como tarefa na linguagem K. Neste sentido, é mais coerente dizer que a ação do supervisor inclui, em vez de marcação, desmarcação de estados.

Na representação de um supervisor marcador por um gerador S, com $L_m(S) \subseteq \Sigma$, S é definido como um gerador com estados marcados, $S = (\Sigma, Y, g, y_0, Y_m)$. Novamente, o comportamento do sistema sob supervisão pode ser calculado pela composição síncrona $S||G$:

$h|G = S||G = (\Sigma, Y \times Q, \theta, (y_0, x_0), Y_m \times Q_m)$, onde

$$\theta((y,q),e) = \begin{cases} (g(y,e), f(q,e)), & \text{se} \quad g(y,e)! \quad \text{e} \quad f(q,e)! \\ \quad \text{não definida}, & \text{caso contrário.} \end{cases}$$

Assumindo que um supervisor possa ser marcador, o seguinte resultado é enunciado:

Teorema 2.2 (Existência de supervisores marcadores) *(Wonham, 2001)* Dados um gerador G, com linguagem marcada $L_m(G)$, e uma linguagem $K \subseteq L_m(G)$ não vazia, existe um supervisor marcador não bloqueante h tal que $L_m(h|G) = K$ se e somente se K é $L(G)$-controlável.

É possível observar que a diferença fundamental entre os teoremas 2.1 e 2.2 reside no fato de que, no segundo caso, não é exigido que a linguagem K seja L_m-fechada.

2.2.1.4 Supervisor Minimamente Restritivo e não Bloqueante

Quando a condição de controlabilidade da linguagem K não é satisfeita, pode-se ainda calcular um supervisor h que restrinja minimamente o comportamento em malha fechada do sistema, de forma que o comportamento sob supervisão esteja contido em K. Este problema é usualmente enunciado como *problema de controle supervisório* (PCS).

Definição 2.3 (Problema de Controle Supervisório) Dada uma planta livre G e um comportamento sob supervisão desejado $K \subseteq L_m(G)$, encontrar um supervisor não bloqueante h tal que $L_m(h|G) \subseteq K$ e o comportamento $L_m(h|G)$ seja máximo, ou seja, não exista outro supervisor não bloqueante h' tal que $L_m(h|G) \subset L_m(h'|G) \subseteq K$.

É possível mostrar que a solução do problema de controle supervisório é tal que $L_m(h|G) = supC(K,G)$. Embora esta solução seja o comportamento ótimo que pode ser obtido por um supervisor,

diz-se que o PCS tem solução apenas quando o comportamento $supC(K,G)$ é satisfatório - o que nem sempre ocorre, dado que esta linguagem pode ser inclusive vazia. O comportamento mínimo desejado para o sistema sob supervisão pode ser especificado por uma linguagem $M \subseteq L_m(G)$. Assim, pode-se dizer que o PCS tem solução se e somente se $M \subseteq supC(K,G)$.

No entanto, para que os resultados sobre existência de supervisores enunciados no teorema 2.1 e no corolário 2.1 sejam válidos, são ainda necessárias considerações sobre o fechamento das máximas linguagens controláveis. Em particular, o seguinte resultado é válido:

Proposição 2.1 (L_m-Fechamento de supC(K,G)) *(Wonham, 2001)* Seja $E \subseteq \Sigma^*$ uma linguagem $L_m(G)$-fechada, e $K = E \cap L_m(G)$. Então $supC(K,G)$ é $L_m(G)$-fechada.

Como caso particular da proposição 2.1, enuncia-se a seguinte proposição:

Proposição 2.2 (Fechamento de supC(K,G)) *(Wonham, 2001)* Seja $E \subseteq \Sigma^*$ uma linguagem fechada, e $K = E \cap L(G)$. Então $supC(K,G)$ é fechada.

Para uma especificação genérica $E \subseteq \Sigma^*$ que não seja fechada ou L_m-fechada, a especificação global $K = E \cap L_m(G)$ pode, enfim, ser utilizada para o cálculo de um supervisor marcador. Assim, ao longo desta dissertação, será considerado que os supervisores calculados apresentam a propriedade de marcação de estados.

Dadas uma planta G, uma especificação genérica E e a especificação global equivalente K, o supervisor S, obtido no cálculo de $supC(K,G)$, usualmente tem um número de estados da ordem do produto dos números de estados de G e do gerador que reconhece E. No entanto, S incorpora, em sua estrutura, informação sobre restrições da função de transição de estados já impostas pela própria estrutura de G. Assim, em geral é possível encontrar um supervisor S', com menor número de estados do que S, tal que $S|G$ e $S'|G$ sejam equivalentes, ou seja, de forma que as linguagens obtidas em malha fechada sejam iguais.

Vaz e Wonham (1986) propõem um algoritmo para minimização do número de estados de supervisores, porém mostrando que este problema de minimização é NP-completo, o que significa que a complexidade computacional associada à minimização de um supervisor é exponencial no número de estados do supervisor. Como alternativa, algoritmos de complexidade polinomial, propostos recentemente, têm alcançado bons resultados na redução, não necessariamente ótima, do número de estados de supervisores (Su e Wonham, 2004). No entanto, também o custo computacional destes algoritmos cresce fortemente com o número de estados dos supervisores.

2.2.2 Controle de Sistemas com Eventos Forçáveis

Uma entrada de controle $\gamma \in \Gamma$ pode ser interpretada, em um ponto de vista mais genérico, como a especificação de quais são os possíveis eventos subsequentes na evolução do sistema.

Seja agora uma nova partição de Σ, $\Sigma = \Sigma_u \cup \Sigma_c \cup \Sigma_f$, como introduzida por Golaszewski e Ramadge (1987). Enquanto os eventos de Σ_u e Σ_c têm a mesma interpretação fornecida anteriormente, Σ_f representa uma nova classe de eventos, denominados *eventos forçáveis*, que modelam ações que ocorrem na planta se e somente se são forçadas por uma entrada externa, ou seja, não são espontâneas. Assume-se, neste caso, que é possível forçar estes eventos antes da ocorrência de eventos de Σ_u e Σ_c. Esta propriedade é modelada pela redefinição das entradas de controle admissíveis:

$$\Gamma_f = \left\{ \gamma \mid \gamma \in 2^{\Sigma} \text{ e } \begin{array}{lll} \gamma = \{e_f\} & , \text{ para algum } & e_f \in \Sigma_f \text{ ou} \\ \gamma \in 2^{\Sigma_u \cup \Sigma_c} & , \text{ com } & \Sigma_u \subseteq \gamma \end{array} \right\}.$$

A redefinição da natureza dos eventos permite que seja reformulada a noção de controlabilidade de linguagens, conforme a definição 2.4.

Definição 2.4 ((L,Γ_f)-Controlabilidade) *(Golaszewski e Ramadge, 1987)* Dada $L \subseteq \Sigma^*$, uma linguagem $K \subseteq L$ é dita (L,Γ_f)-*controlável* se, para toda cadeia $s \in \overline{K}$, exatamente uma das duas condições for verdadeira:

$$\begin{array}{lll} s\Sigma_u \cap L \subseteq \overline{K} & , \text{ com } \Sigma_K(s) \subseteq \Sigma_u \cup \Sigma_c & \text{ou} \\ \Sigma_K(s) = \{e_f\} & , \text{ para algum } e_f \in \Sigma_f. & \end{array}$$

No modelo original de Ramadge e Wonham (1987b), com alfabeto particionado em Σ_u e Σ_c, o conjunto $\Gamma = \{\gamma \mid \gamma \in 2^{\Sigma} \text{ e } \Sigma_u \subseteq \gamma\}$ é fechado para intersecção e união de conjuntos. Como, para qualquer $\gamma \in \Gamma$, é verdade que $\Sigma_u \subseteq \gamma \subseteq \Sigma$, Σ_u e Σ são respectivamente os elementos mínimo e máximo de Γ.

Por sua vez, o novo conjunto de entradas de controle admissíveis Γ_f não é fechado para união - por exemplo, $\gamma_1 = \{e_f\}$ e $\gamma_2 = \Sigma_u$ são dois elementos de Γ_f, porém sua união não é uma entrada admissível. Isto implica que, dada uma planta G com alfabeto Σ e uma linguagem $M \subseteq \Sigma^*$, a classe de linguagens contidas em M e $(L(G),\Gamma_f)$-controláveis, denotada por $C_f(M,G)$, não é garantidamente fechada para união, ou seja, $C_f(M,G)$ contém possivelmente diversos elementos máximos.

No entanto, dado um conjunto de entradas admissíveis Γ qualquer, é possível considerar um *supervisor não determinístico*, redefinindo o mapa Φ como $\Phi : Y \to 2^{\Gamma}$. Neste caso, o supervisor pode selecionar qualquer controle do subconjunto $\Phi(y) \subseteq \Gamma$ de maneira não determinística. Pode-se mostrar que a utilização de um supervisor não determinístico é equivalente a redefinir as entradas de controle admissíveis como o fechamento para união de Γ, $cl(\Gamma)$, e considerar um supervisor determinístico (T,Φ) com $\Phi : Y \to cl(\Gamma)$.

Para o caso Γ_f, o fechamento para união $cl(\Gamma_f)$ indica que, a cada mudança de estado na planta, o supervisor deve escolher entre forçar um evento contido em $\Phi(y) \cap \Sigma_f$ ou habilitar um conjunto de eventos contido em $\Phi(y) \cap (\Sigma_u \cup \Sigma_c)$.

A controlabilidade de linguagens pode, enfim, ser redefinida em função de um conjunto de entradas de controle admissíveis $cl(\Gamma_f)$:

Definição 2.5 ($(L, cl(\Gamma_f))$-Controlabilidade) Dado $L \subseteq \Sigma^*$, uma linguagem $K \subseteq L$ é dita $(L, cl(\Gamma_f))$-*controlável* se e somente se, para toda cadeia $s \in \overline{K}$, ao menos uma das duas condições for verdadeira:

$$s\Sigma_u \cap L \subseteq \overline{K} \qquad \text{ou}$$
$$\emptyset \subset \Sigma_K(s) \subseteq \Sigma_f.$$

Neste caso, dada uma linguagem $M \subseteq L_m(G)$ não controlável, o conjunto das linguagens contidas em M e $(L_m(G), cl(\Gamma_f))$-controláveis é fechado para união, contendo assim um elemento supremo, denotado por $supC_f(M, G)$.

As condições para existência de um supervisor marcador não bloqueante apresentadas no Teorema 2.2 podem, assim, ser estendidas para a abordagem com eventos forçáveis:

Teorema 2.3 (Existência de supervisores, abordagem com eventos forçáveis) *(Golaszewski e Ramadge, 1987)* Dados um gerador G, com linguagem marcada $L_m(G)$, e uma linguagem $K \subseteq L_m(G)$ não vazia, existe um supervisor (marcador) não bloqueante h tal que $L_m(h|G) = K$ se e somente se K é $(L_m(G), cl(\Gamma_f))$-controlável.

Para uma linguagem K não $(L_m(G), cl(\Gamma_f))$-controlável, o máximo comportamento que pode ser obtido sob supervisão é dado por $supC_f(K, G)$. Um algoritmo para cálculo de $supC_f(K, G)$ a partir de G e K é fornecido por Torrico (1999).

É apresentado agora um exemplo de aplicação da abordagem de controle com eventos forçáveis, adaptado de (Wonham, 2001).

Exemplo 2.6 (Controle Supervisório com Eventos Forçáveis)
Seja um tanque T, que deve ser enchido por um fluido através de uma válvula V. Esta válvula deve ser fechada para evitar transbordamento quando o tanque estiver cheio. A válvula V pode ser modelada pelo gerador V, apresentado na figura 2.13. O evento α é interpretado como o depósito de uma unidade definida de fluido no tanque, enquanto β e γ correspondem, respectivamente, ao fechamento e à abertura da válvula.

Para um tanque com capacidade N, é imposta a restrição de não-transbordamento do tanque. A modelagem desta especificação resulta no gerador T da figura 2.14, para $N = 3$. Note que, neste

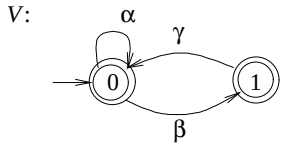

Figura 2.13: Modelo da válvula (planta).

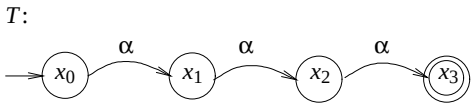

Figura 2.14: Especificação para o tanque.

caso, o modelo T corresponde a uma especificação genérica com alfabeto $\{\alpha\}$. Assim, o gerador que representa a especificação global é calculado por $K = V||T$.

Suponha que se deseje adotar a abordagem clássica para o projeto de um supervisor que satisfaça K. Neste caso, se $\Sigma_u = \{\alpha\}$ e $\Sigma_c = \{\beta, \gamma\}$, claramente $supC(K,V) = \emptyset$.

Suponha agora que $\Sigma_u = \{\alpha\}$, $\Sigma_c = \{\gamma\}$ e $\Sigma_f = \{\beta\}$. A modelagem de β como evento forçável significa que, após o depósito de uma unidade de fluido no tanque, é possível fechar a válvula antes do depósito de outra unidade. Neste caso, a ação de controle pode, no estado $(0, x_3)$ de $V \times T$, forçar a ocorrência de β. Assim, o supervisor S, representado pelo gerador S tal que $L_m(S) = supC_f(K,V)$, é apresentado na figura 2.15, para $N = 3$.

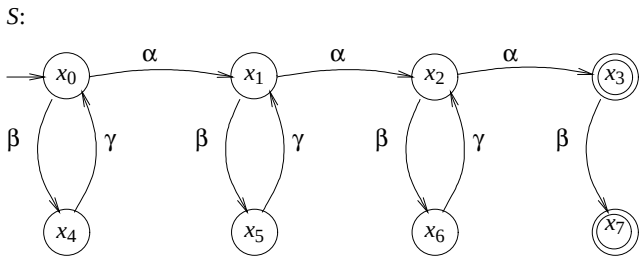

Figura 2.15: Supervisor para sistema de enchimento de tanque.

2.2.3 Modularidade em Sistemas a Eventos Discretos

A complexidade computacional do cálculo da máxima linguagem controlável, como proposta na abordagem clássica (Ramadge e Wonham, 1987b), é polinomial no número de estados dos modelos da planta G e da especificação K. No entanto, o número de estados da especificação e da planta cresce exponencialmente com o número de especificações e subplantas que as compõem. Como alternativa para contornar este problema, o controle modular visa a explorar a decomposição natural das especificações (Ramadge e Wonham, 1987a, 1988). Mais recentemente, foi prosposta também também a exploração da estrutura naturalmente descentralizada da planta (Queiroz, 2000).

Estas duas abordagens serão descritas a seguir.

2.2.3.1 Controle Modular

Quando é fornecido um conjunto de especificações globais $K_{Xi} \subseteq L_m(G)$, que representam restrições para uma planta G, é possível obter a especificação global $K = \cap_{i=1}^{n} K_{Xi}$ e projetar um supervisor *monolítico* S que satisfaça todas as especificações: $L_m(S|G) = supC(K,G)$.

Por outro lado, pode-se também construir um supervisor para cada especificação. Neste caso, a ação de controle sobre a planta deve ser calculada a partir da combinação das ações de controle individuais dos supervisores. Nesta abordagem, denominada controle modular, um evento controlável da planta será desabilitado sempre que este for desabilitado por algum dos supervisores, como ilustrado pela figura 2.16 para dois supervisores.

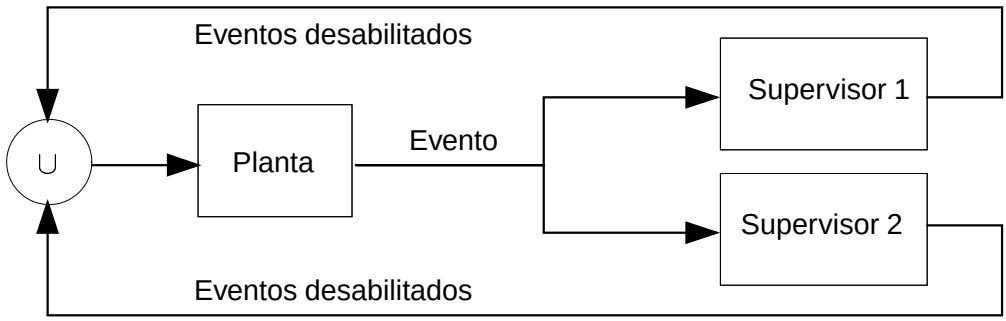

Figura 2.16: Ação de controle de supervisores modulares sobre uma planta.

Proposição 2.3 (Comportamento da supervisão modular) *(Ramadge e Wonham, 1988) Sejam S_1 e S_2 supervisores para G. Então, o supervisor $S = S_1 \cap S_2$ tem comportamento $L(S|G) = L(S_1|G) \cap L(S_2|G)$ e $L_m(S|G) = L_m(S_1|G) \cap L_m(S_2|G)$.*

Embora o cálculo dos supervisores garanta que cada ação de controle seja individualmente não bloqueante, é possível que a ação conjunta dos supervisores seja bloqueante, pois o cálculo de cada supervisor leva em consideração apenas uma visão parcial do sistema. No entanto, após o cálculo dos supervisores, é possível avaliar se a solução conjunta obtida é bloqueante, através do teste da condição de modularidade apresentada a seguir.

Duas linguagens L_1, $L_2 \subseteq \Sigma^*$ são ditas *modulares* se $\overline{L_1} \cap \overline{L_2} = \overline{L_1 \cap L_2}$. Isto significa que, para qualquer prefixo t comum a L_1 e L_2, existe uma cadeia s tal que ts seja um elemento de $L_1 \cap L_2$, ou seja, L_1 e L_2 compartilham também uma cadeia que contém este prefixo. É interessante destacar que, para quaisquer L_1, $L_2 \subseteq \Sigma^*$, é sempre verdade que $\overline{L_1 \cap L_2} \subseteq \overline{L_1} \cap \overline{L_2}$ - logo, duas linguagens L_1 e L_2 são modulares se e somente se $\overline{L_1} \cap \overline{L_2} \subseteq \overline{L_1 \cap L_2}$.

A modularidade de duas linguagens controláveis é condição suficiente para que sua intersecção seja também controlável, como enunciado na proposição 2.4:

Proposição 2.4 (Modularidade e controlabilidade) *(Wonham, 2001) Sejam linguagens K_1 e K_2, e uma planta G, definidas em Σ, tais que K_1 e K_2 são controláveis em relação a G. Se K_1 e K_2 são modulares, então $K_1 \cap K_2$ é controlável em relação a G.*

Sejam S_1 e S_2 supervisores próprios para G. Por definição, $\overline{L_m(S_i|G)} = L(S_i|G)$, $i = 1,2$. No entanto, para que o supervisor $S = S_1 \cap S_2$ seja próprio, é necessário que $\overline{L_m(S|G)} = L(S|G)$, ou seja, que $\overline{L_m(S_1|G) \cap L_m(S_1|G)} = L(S_1|G) \cap L(S_1|G) = L_m(S_1|G) \cap L_m(S_1|G)$. Assim, a modularidade é, também, condição necessária e suficiente para que o comportamento conjunto dos supervisores seja não bloqueante, como enunciado na proposicão 2.5.

Proposição 2.5 (Supervisão modular não bloqueante) *(Ramadge e Wonham, 1988) Sejam S_1 e S_2 supervisores para G, tais que $L_m(S_i|G) = K_i$, $i = 1,2$. Então, $S = S_1 \cap S_2$ é próprio para G se e somente se K_1 e K_2 são modulares.*

Para o caso de duas especificações não necessariamente controláveis K_1 e K_2, é necessária a verificação da modularidade das máximas linguagens controláveis contidas em K_1 e K_2, como enunciado na proposição 2.6.

Proposição 2.6 (Supervisão modular ótima não bloqueante) *(Ramadge e Wonham, 1988) Sejam $K_1, K_2 \subseteq \Sigma^*$ especificações para G. Então, se as linguagens $supC(K_1,G)$ e $supC(K_2,G)$ são modulares, $supC(K_1,G) \cap supC(K_2,G) = supC(K_1 \cap K_2, G)$.*

Assim, quando a modularidade dos supervisores é verificada, é possível não apenas afirmar que o comportamento conjunto dos supervisores modulares é próprio, como também garantir que a implementação modular é ótima, pois fornece a mesma solução obtida com a síntese e uso de um supervisor pela abordagem monolítica.

Embora enunciadas para dois supervisores, as proposições 2.3, 2.4, 2.5 e 2.6 podem ser generalizadas para um conjunto de especificações $K_{Xi} \subseteq \Sigma_i^*$, $i = 1,\ldots,n$. Neste caso, a condição de modularidade é dada por $\cap_{i=1}^n \overline{K_{Xi}} = \overline{\cap_{i=1}^n K_{Xi}}$. Ainda, se as especificações de comportamento do sistema são dadas sob a forma de especificações genéricas $E_{xi} \subseteq \Sigma_i^*$, $i = 1,\ldots,n$, pode-se obter o conjunto de especificações globais para aplicação da abordagem modular com $K_{Xi} = P_i^{-1}(E_{xi}) \cap L_m(G)$, $i = 1,\ldots,n$.

A abordagem modular é mais flexível, quando comparada com a abordagem monolítica, pois alterações de especificações refletem apenas em alterações no projeto e implementação dos supervisores correspondentes. No entanto, é necessário refazer o teste de modularidade a cada alteração, e caso a modularidade não seja verificada, pode ser necessária a alteração de outros supervisores.

2.2.3.2 Controle Modular Local

Como pôde ser observado na seção anterior, abordagem modular clássica não elimina a necessidade de representação da planta como um único gerador, mesmo quando o sistema é originalmente modelado por um conjunto de susbsistemas com certo grau de paralelismo. Este paralelismo, representado pela modelagem de susbsistemas com alfabetos disjuntos, não é utilizado para simplificação do projeto dos controladores - pelo contrário, acaba sendo uma causa direta do crescimento exponencial, em número de estados, do gerador que representa a planta.

A abordagem de controle modular local proposta por Queiroz (2000) permite explorar, além da modularidade das especificações, também a estrutura naturalmente descentralizada dos sistemas de manufatura automatizados, através da restrição da ação de controle de cada supervisor apenas aos subsistemas afetados pela especificação correspondente ao supervisor. Esta abordagem pressupõe que os sistemas de manufatura sejam modelados por sua Representação por Sistemas Produto (RSP) mais refinada, como definido na seção 2.1.3.1.

A estrutura de controle implementada pela abordagem modular local é ilustrada na figura 2.17, para o caso particular de duas especificações que restringem subsistemas em comum. Neste caso, a ação de controle sobre cada subsistema comum às duas subplantas é calculada como a união das desabilitações impostas pelos supervisores, de forma análoga à utilizada na abordagem de controle modular.

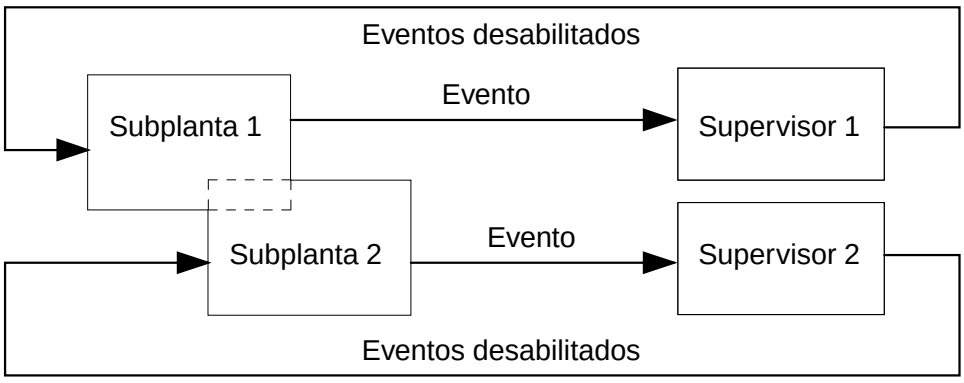

Figura 2.17: Estrutura de controle com supervisores localmente modulares.

Seja a RSP $\{G_j\}$, $j \in I_G = \{1,\ldots,m\}$ com $L(G_j) \subseteq \Sigma_j^*$ e $\Sigma_j \subseteq \Sigma$, $j \in I_G$. Dadas especificações genéricas E_{xi}, $i = 1,\ldots,n$, definidas sobre alfabetos $\Sigma_{xi} \subseteq \Sigma$, define-se a *planta local* G_{Xi} como a combinação dos susbsistemas da RSP que contém eventos em comum com Σ_{xi}: $G_{Xi} = \|_{i \in I_{Xi}} G_i$, com cada conjunto de índices I_{Xi} definido por $I_{Xi} = \{k \in I_G | \Sigma_k \cap \Sigma_{xi} \neq \emptyset\}$. O alfabeto Σ_{Xi} é definido por $\Sigma_{Xi} = \cup_{i \in I_{Xi}} \Sigma_i$.

As especificações genéricas podem agora ser expressas, em termos da planta local G_{Xi}, como *especificações locais* $E_{Xi} = E_{xi} \| L_m(G_{Xi})$. Para efeito de simplificação, sem perda de generalidade,

assume-se que a RSP contém apenas subsistemas restritos por alguma das especificações dadas, o que significa que $G = \|_{j=1}^{m} G_j = \|_{i=1}^{n} G_{Xi}$. Ainda, relembrando a definição de especificação global apresentada na seção 2.2.1.3, a cada especificação genérica E_{xi} corresponde também uma especificação global $K_{Xi} = E_{xi} \| L_m(G)$, que modela o comportamento da planta restrito por E_{xi}.

A expressão de uma especificação genérica E_{xi} como especificação local E_{Xi} permite que um problema de controle supervisório seja resolvido para a planta local G_{Xi}. Assim, para $i = 1, \ldots, n$, pode-se calcular $supC(E_{Xi}, G_{Xi})$, a máxima linguagem contida em E_{Xi} controlável em relação a G_{Xi}. Esta linguagem corresponde ao supervisor local que restringe apenas o comportamento das plantas que compõem G_{Xi}, de forma a satisfazer a especificação E_{xi}. No entanto, ainda que a representação por sistemas produto subjacente seja assíncrona, é importante destacar que as plantas locais podem conter plantas G_i em comum. Neste caso, é necessário garantir que as restrições impostas pelas especificações locais não sejam conflitantes.

Para a aplicação da abordagem modular local, apresentada neste item, é necessário que o conceito de modularidade apresentado na seção 2.2.3.1 seja estendido para linguagens definidas sobre alfabetos distintos. Assim, sejam linguagens $L_i \subseteq \Sigma_i^*$, $i = 1, \ldots, n$. O conjunto de linguagens $\{L_i\ , i = 1, \ldots, n\}$ é dito *localmente modular* se $\cap_{i=1}^{n} \overline{P_i^{-1}(L_i)} = \overline{\cap_{i=1}^{n} P_i^{-1}(L_i)}$, ou seja, se $\|_{i=1}^{n} \overline{L_i} = \overline{\|_{i=1}^{n} L_i}$. Se, para $i = 1, \ldots, n$, as linguagens forem marcadas por geradores *trim* G_i, é fácil provar que a modularidade local é verificada se e somente se $G = \|_{i=1}^{n} G_i$ for um gerador *trim*, o que implica a ausência de estados ou ciclos de bloqueio. Pode-se mostrar também que, para quaisquer $L_i \subseteq \Sigma_i^*$, $i = 1, \ldots, n$, é sempre verdade que $\overline{\|_{i=1}^{n} L_i} \subseteq \|_{i=1}^{n} \overline{L_i}$, o que significa que um conjunto de linguagens $\{L_i, i = 1, \ldots, n\}$ é localmente modular se e somente se $\|_{i=1}^{n} \overline{L_i} \subseteq \overline{\|_{i=1}^{n} L_i}$.

O seguinte resultado, utilizado na fundamentação da abordagem modular local, indica que um problema de controle supervisório com duas especificações pode ser naturalmente decomposto, quando a estas especificações estão associadas plantas locais assíncronas:

Proposição 2.7 *(Queiroz, 2000) Sejam* $K_1, L_1 \subseteq \Sigma_1^*$ *e* $K_2, L_2 \subseteq \Sigma_2^*$, *com* $\Sigma_1 \cap \Sigma_2 = \emptyset$. *Então* $supC(K_1, L_1) \|_{as} supC(K_2, L_2) = supC(K_1 \|_{as} K_2, L_1 \|_{as} L_2)$.

O principal resultado da abordagem de controle modular local é enunciado pela seguinte proposição:

Proposição 2.8 *(Queiroz, 2000) Dados um sistema com RSP formada por* G_j, $j = 1, \ldots, m$, *e as especificações genéricas* E_{xi}, $i = 1, \ldots, n$. *Sejam* E_{Xi}, K_{Xi}, G_{Xi} *e* G *definidos como nesta seção. Se* $\{supC(E_{Xi}, G_{Xi}),\ i = 1, \ldots, n\}$ *for localmente modular, então* $supC(\cap_{i=1}^{n} K_{Xi}, G) = \|_{i=1}^{n} supC(E_{Xi}, G_{Xi})$.

Assim, se o conjunto de supervisores locais calculado for localmente modular, o comportamento obtido pela supervisão das plantas locais é não bloqueante e equivalente ao comportamento obtido por supervisão monolítica.

Além da vantagem associada à redução da complexidade computacional na resolução do problema de controle supervisório (Queiroz, 2000), a abordagem modular local permite que a ação de controle seja realizada de forma descentralizada, pois os supervisores calculados só precisam observar eventos e atuar sobre um subconjunto das plantas assíncronas.

O exemplo de aplicação da abordagem de controle modular local, apresentado a seguir, foi retirado de Queiroz e Cury (2002).

Exemplo 2.7 (Supervisores localmente modulares para mesa giratória com quatro posições) Seja a planta a ser controlada uma mesa giratória para furação e teste de peças, conforme ilustrado na figura 2.18. Deseja-se projetar um sistema de controle que coordene as operações de giro da mesa, carga de peça, furação, teste e descarga. Suponha que a mesa não seja equipada com sensores para detecção da presença de peças nos estágios da mesa, ou que a informação referente a estes sensores não seja acessível para o sistema de controle.

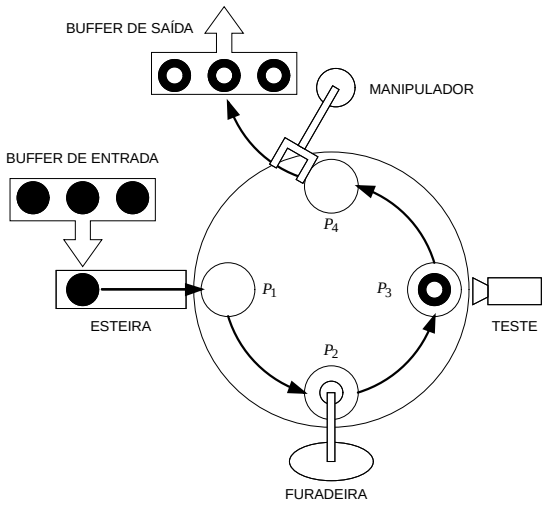

Figura 2.18: Esquemático da mesa giratória.

Para a modelagem do comportamento livre da planta, os elementos do sistema são denominados da seguinte forma: mesa M_0; esteira M_1; furadeira M_2; teste M_3; manipulador M_4. Sejam α_i e β_i eventos que identificam, respectivamente, o início e o fim do processamento no elemento M_i, para $i = 0,\ldots,4$. Por exemplo, evento α_2 indica que a furadeira começou a operar, o evento β_0 indica que a mesa concluiu um giro de 90^o, e assim por diante.

O comportamento livre de cada um dos componentes pode ser representado pelos geradores G_i, $i = 0,1,2,3,4$ (figura 2.19).

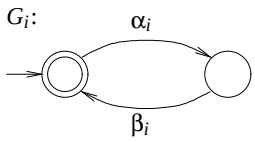

Figura 2.19: Comportamento livre dos componentes da mesa giratória.

Note que os geradores G_i são assíncronos. Assim, a representação por sistemas produto mais refinada é o próprio conjunto $\{G_0, G_1, G_2, G_3, G_4\}$. A planta livre $G = \|_{i=0}^{4} G_i$ é composta por um gerador com 32 estados e 160 transições.

O comportamento esperado para a planta pode ser descrito por um conjunto de especificações genéricas que visam, em conjunto, a restringir minimamente o funcionamento da mesa, de modo que ela possa operar com até quatro peças simultaneamente.

Assim, com a restrição *Ra*, indicada pelo gerador da figura 2.20, deseja-se evitar que a mesa gire sem operação realizada em ao menos um dos estágios P_1, P_2 ou P_3. Note que não é necessário que a mesa gire após uma operação de remoção de peça em P_4.

É também exigido, como restrição de segurança, que a mesa não gire durante a realização de uma operação. Além disso, uma operação não deve ser inicializada durante o giro da mesa. Para cada operação M_i da mesa, $i = 1, \ldots, 4$, estas duas restrições são descritas sob a forma de um gerador Rb_i, como ilustrado na figura 2.21.

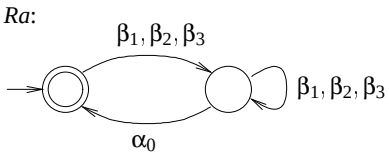

Figura 2.20: Restrição *Ra*.

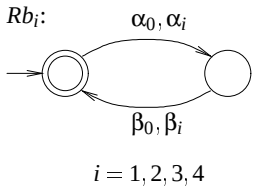

Figura 2.21: Restrições Rb_i.

Ainda, são propostas especificações para garantir o seqüenciamento das operações. Em função da ausência de sensores de presença nas posições da mesa, deve-se inferir sobre a presença de peça em um dado estágio da mesa apenas com base na seqüência de sinais de início e fim de operação dos estágios anteriores e de giros da mesa. Para tanto, é adotada a premissa de que existe um sensor de presença de peça no estágio de carga, que condiciona a ocorrência de α_1 à existência de peça disponível para carga. Destaca-se, novamente, que a informação deste sensor não pode ser acessada diretamente pelo sistema de controle.

Assim, para $i = 1, 2, 3$, são descritas especificações que visam a garantir que a operação do estágio correspondente à posição P_i não seja repetida antes de um giro da mesa, bem como impedir que uma operação na posição subseqüente seja iniciada sem presença de peça que deva ser operada (figura 2.22). Mais especificamente, deseja-se impedir dupla carga de peça em P_1 e furação sem peça bruta

em P_2 ($i = 1$); impedir dupla furação em P_2 e teste sem peça a ser testada em P_3 ($i = 2$); e impedir duplo teste em P_3 e descarga sem peça em P_4 ($i = 3$).

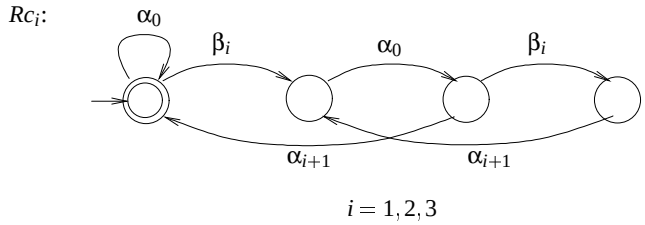

Figura 2.22: Restrições Rc_i.

Para as especificações R_x, $x = a, b1, b2, b3, b4, c1, c2, c3$, as plantas locais são dadas por: $G_A = G_0 \| G_1 \| G_2 \| G_3$, $G_{Bi} = G_0 \| G_i$, $i = 1, \ldots, 4$ e $G_{Ci} = G_0 \| G_i \| G_{i+1}$, $i = 1, 2, 3$. A figura 2.23 apresenta o gerador que representa as plantas locais G_{Bi}, para $i = 1, \ldots, 4$.

Para $x = a, b1, b2, b3, b4, c1, c2, c3$, cada especificação local E_X é calculada por $E_X = R_x \| G_x$. Para $x = b1, b2, b3, b4$, o gerador E_X é apresentado na figura 2.24.

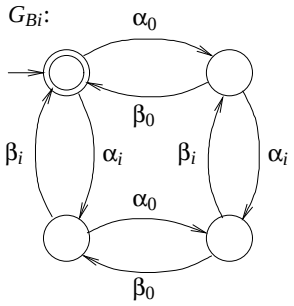

Figura 2.23: Plantas locais G_{Bi}.

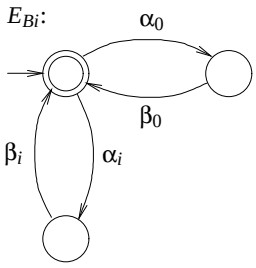

Figura 2.24: Especificações Locais E_{Bi}.

Assim, com base nas especificações locais e plantas locais, para o conjunto de índices $I_X = A, B1, B2, B3, C1, C2, C3, C4$ são calculados os supevisores minimamente restritivos S_X, $X \in I_X$, tais que $L(S_X) = supC(E_X, G_X)$.

Para verificação da propriedade de modularidade local do conjunto de supervisores locais, calcula-se o produto síncrono $\|_{\{X \in I_X\}} S_X$. Como o autômato obtido é *trim*, este conjunto é localmente modular. Os supervisores calculados tem um tamanho máximo de 32 estados. Para o mesmo sistema, a composição síncrona do conjunto de especificações genéricas fornece um gerador R com um total de 199 estados e 478 transições, mesmos números apresentados pela especificação global $K = R \| G$. O supervisor monolítico S, com linguagem $supC(K, G)$, é representado por um autômato *trim* com 151 estados e 350 transições.

Ainda, para cada supervisor local S_X é calculado sua representação mínima RS_X pelo algoritmo de Vaz e Wonham (1986). É importante destacar também que os algoritmos de minimimização e

redução de supervisores (Vaz e Wonham, 1986; Su e Wonham, 2004), em função de sua complexidade computacional, possuem aplicabilidade limitada para supervisores de grande porte, como é, em geral, um supervisor monolítico. Por esta razão, a redução de S não é apresentada, embora pudesse também ser também calculada, no caso em questão, com os recursos computacionais utilizados. Assim, considera-se o conjunto de supervisores $\{RS_X, X \in I_X\}$ e o supervisor S como solução final dos problemas de síntese modular local e de síntese monolítica de supervisores, respectivamente.

A tabela 2.7 apresenta o número de estados dos geradores envolvidos no processo de síntese. Em uma análise de pior caso, o esforço computacional $O(\cdot)$ para realização da síntese de cada supervisor S_X é polinomial no produto do número de estados da especificação local E_X com o número de estados da planta local G_X. A complexidade $O(\cdot)$ para a resolução completa do PCS pela abordagem modular local depende, também, da verificação da modularidade local do conjunto de supervisores locais (Queiroz, 2000). A complexidade computacional $O(\cdot)$ indicada na tabela 2.7 se refere à resolução do PCS pelas abordagens modular local e monolítica, sem considerar o procedimento de redução de supervisores.

Resolução Modular Local	E_x	G_X	E_X	S_X	RS_X	$O(\cdot)$
a	2	16	32	32	(2, 7)	$16^2 * 32^2$
$b_i, i = 1,2,3,4$	2	4	3	3	(2, 4)	$4^2 * 3^2$
$c_i, i = 1,2,3$	4	8	32	24	(4, 8)	$8^2 * 32^2$
Total (somatório)	22	56	140	116	(22,47)	
Resolução Monolítica	R	G	K	S		
	199	32	199	(151,350)		$32^2 * 199^2$

Tabela 2.1: Número de estados dos geradores e complexidade para o problema da mesa giratória (exemplo 2.7). Para as soluções finais, é apresentado o par (número de estados, número de transições).

O número total de estados e transições para a solução modular local é uma referência adequada para o esforço de programação necessário para a implementação física da solução, podendo, neste sentido, ser comparado com o número de estados e eventos da solução monolítica. □

O exemplo 2.7 ilustra também que, para sistemas com certo grau de simetria, é possível realizar o cálculo de supervisores sem especificação dos índices i dos eventos. Assim, é possível calcular apenas S_A, S_{Bi} e S_{Ci} - a especificação do índice i só se faz necessária para a verificação da modularidade local.

2.2.4 Controle Modular com Eventos Forçáveis

As abordagens para controle modular e modular local, como apresentadas respectivamente nos itens 2.2.3.1 e 2.2.3.2, são fundamentadas no fato de que o conjunto de entradas de controle válidas Γ é fechado para intersecção. É possível mostrar que esta propriedade garante que a intersecção de linguagens modulares e controláveis é controlável. Embora os resultados referentes a bloqueio

e modularidade enunciados independam da natureza dos eventos, a redefinição das entradas de controle válidas exige que novas condições sejam impostas sobre supervisores modulares, para que o comportamento conjunto dos supervisores seja controlável.

No caso particular da abordagem com eventos forçáveis, dadas linguagens K_1, $K_2 \subseteq L$, $(L, cl(\Gamma_f))$-controláveis, mesmo que K_1 e K_2 sejam modulares, é possível que $K_1 \cap K_2$ não seja $(L, cl(\Gamma_f))$-controlável, como ilustrado no exemplo 2.8.

Exemplo 2.8 (Não-controlabilidade de supervisores modulares com eventos forçáveis)
Sejam alfabetos $\Sigma_u = \{e_u\}$, $\Sigma_c = \emptyset$ e $\Sigma_f = \{e_{f1}, e_{f2}\}$. Sejam linguagens $L = \{\varepsilon, e_u, e_{f1}, e_{f2}\}$, $K_1 = \{\varepsilon, e_{f1}\}$ e $K_2 = \{\varepsilon, e_{f2}\}$. K_1 e K_2 são modulares, pois são prefixo-fechadas. Além disso, são linguagens $(L, cl(\Gamma_f))$-controláveis, pois, para $s = \varepsilon$, $\emptyset \subset \Sigma_{Ki}(s) = \{e_{fi}\} \subseteq \Sigma_f$, $i = 1, 2$.

No entanto, $\Sigma_{K_1 \cap K_2}(s) = \emptyset$. Além disso, $se_u \cap L = e_u \notin \overline{K_1 \cap K_2}$. Portanto, $K_1 \cap K_2$ não é $(L, cl(\Gamma_f))$-controlável. Esta situação é ilustrada na figura 2.25. □

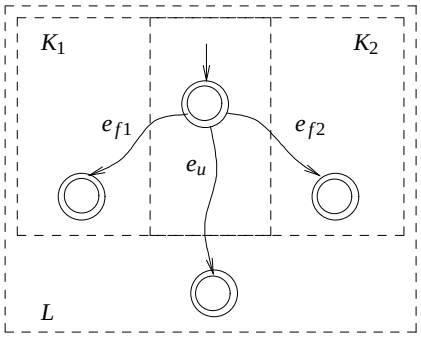

Figura 2.25: Não-controlabilidade causada por ação conjunta de supervisores com eventos forçáveis.

Portanto, para que a abordagem modular possa ser utilizada com eventos forçáveis, é necessário impor novas condições sobre as linguagens dos supervisores modulares. Este documento apresenta uma condição suficiente, baseada na suposição de que cada evento forçável pode ser atribuído a um único supervisor modular, responsável por forçar a ocorrência do evento, ação esta que não pode ser desabilitada pelos outros supervisores modulares.

Sejam linguagens $K_{Xi} \subseteq L$, $i = 1, \ldots, n$, $(L, cl(\Gamma_f))$-controláveis. Suponha que, para cada $e_{fk} \in \Sigma_f$, $k = 1, \ldots, |\Sigma_f|$, apenas o supervisor modular calculado para K_{Xi} seja responsável por forçar ocorrência de e_{fk}. Para simplificar a notação adotada nesta seção, diz-se neste caso que K_{Xi} força e_{fk}.

Para cada linguagem K_{Xi}, redefine-se a interpretação dos eventos de Σ da seguinte forma: sejam os alfabetos Σ_{Yi}, $i = 1, \ldots, n$, tais que $\Sigma_{c,Yi} = \Sigma_c \cup \Sigma_u \cup \{e_f \in \Sigma_f | e_f \text{ é forçado por } K_{Xi}\}$ e $\Sigma_{u,Yi} = \Sigma_f - \Sigma_{c,Yi}$. Note-se que, para esta interpretação dos eventos, a controlabilidade deve ser avaliada como enunciada na definição 2.1, ou seja, de acordo com a abordagem de eventos controláveis. Neste caso, diz-se que a linguagem K_{Xi} é (L, Γ_{Yi})-controlável se $\overline{K_{Xi}}\Sigma_{u,Yi} \cap L \subseteq \overline{K_{Xi}}$.

Esta nova condição de controlabilidade determina restrições para as entradas de controle válidas de forma diferente para cada supervisor, de forma que o comportamento conjunto dos supervisores sempre possa conter, ao menos, um evento forçável. Esta propriedade é enunciada na proposição 2.9:

Proposição 2.9 ($cl(\Gamma_f)$-Controlabilidade, linguagens modulares (1)) *Sejam linguagens $K_{Xi} \subseteq L$, $i = 1,\ldots,n$, definidas sobre $\Sigma = \Sigma_u \cup \Sigma_c \cup \Sigma_f$. Se o conjunto $\{K_{Xi} | i = 1,\ldots,n\}$ é modular, e se estas linguagens são $(L, cl(\Gamma_f))$-controláveis e $(\cup_{i=1}^{n} K_{Xi}, \Gamma_{Yi})$-controláveis, então $K = \cap_{i=1}^{n} K_{Xi}$ é $(L, cl(\Gamma_f))$-controlável.*

Prova:

Suponha que K_{Xi}, $i = 1,\ldots,n$ sejam $(L, cl(\Gamma_f))$-controláveis, mas que K não seja $(L, cl(\Gamma_f))$-controlável. Então, para algum $s \in \overline{K}$, é verdade que $\Sigma_u \cap \Sigma_K(s) \subset \Sigma_u \cap \Sigma_L(s)$ e que $\Sigma_f \cap \Sigma_K(s) = \emptyset$. Assim, existe ao menos um K_j tal que $\Sigma_u \cap \Sigma_{K_j}(s) \subset \Sigma_u \cap \Sigma_L(s)$. Como K_j é $(L, cl(\Gamma_f))$-controlável, existe um e_{fk} forçado por $\overline{K_j}$ tal que $\{e_{fk}\} \subseteq \Sigma_{K_j}(s) \subseteq \Sigma_f \cap \Sigma_L(s)$. No entanto, de $\Sigma_f \cap \Sigma_K(s) = \emptyset$, sabe-se que $e_{fk} \notin \Sigma_K(s)$, ou seja, $se_{fk} \notin \overline{K}$. Como K_{Xi} são linguagens modulares, $\overline{K} = \overline{\cap_{i=1}^{n} K_{Xi}} = \cap_{i=1}^{n} \overline{K_{Xi}}$. Portanto, $se_{fk} \notin \cap_{i=1}^{n} \overline{K_{Xi}}$. Assim, existe um $K_l \neq K_j$ tal que $se_{fk} \notin \overline{K_l}$, ou seja, $e_{fk} \notin \Sigma_{K_l}(s)$. No entanto, como $se_{fk} \in \overline{K_j}$, sabe-se que $se_{fk} \in \overline{\cup_{i=1}^{n} K_{Xi}}$. Neste caso, K_l não é $(\cup_{i=1}^{n} K_{Xi}, \Gamma_{Yl})$-controlável, pois $e_{fk} \in \Sigma_{s,Yl}$ implica $se_{fk} \in \overline{K_l}\Sigma_{u,Yl} \cap (\cup_{i=1}^{n} K_{Xi})$, mas $se_{fk} \notin \overline{K_l}$. □

A figura 2.26 ilustra a demonstração da proposição 2.9.

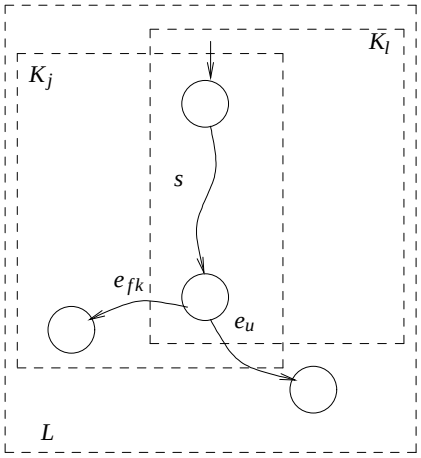

Figura 2.26: $K_j \cap K_l$ não é $(L, cl(\Gamma_f))$-controlável, pois K_l não é (K_j, Γ_{Yl})-controlável (ver proposição 2.9).

Com a redefinição da natureza dos eventos em cada linguagem K_i para Σ_{Yi}, se K_i é controlável em relação a $\cup_{i=1}^{n} K_{Xi}$, garante-se que a linguagem K_i não desabilita os eventos que são forçados pelas outras linguagens. Esta condição sendo satisfeita, as linguagens K_i podem ser utilizadas para implementação de supervisores modulares. Neste caso, o comportamento conjunto dos supervisores modulares pode ser calculado pela intersecção das linguagens K_i.

Uma consequência da proposição 2.9 é enunciada na seguinte proposição:

Proposição 2.10 ($cl(\Gamma_f)$-Controlabilidade, linguagens modulares (2)) *Sejam linguagens $K_{Xi} \subseteq L$, $i = 1,\ldots,n$, definidas sobre $\Sigma = \Sigma_u \cup \Sigma_c \cup \Sigma_f$. Se o conjunto $\{K_{Xi} | i = 1,\ldots,n\}$ é modular, e se estas linguagens são $(L, cl(\Gamma_f))$-controláveis e (L,Γ_{Yi})-controláveis, então $K = \cap_{i=1}^{n} K_{Xi}$ é $(L, cl(\Gamma_f))$-controlável.*

Prova:

Como $\cup_{i=1}^{n} K_{Xi} \subseteq L$, uma linguagem (L,Γ_{Yi})-controlável é também $(\cup_{i=1}^{n} K_{Xi}, \Gamma_{Yi})$-controlável. Assim, esta prova decorre diretamente da proposição 2.9. □

Seja agora o conjunto de entradas de controle $\Gamma_{uf} : \{\gamma | \Sigma_u \cup \Sigma_f \subseteq \gamma\}$. Para este conjunto, os eventos de Σ_f são considerados não controláveis, dentro da abordagem clássica. Assim, uma linguagem K_{Xi} é (L,Γ_{uf})-controlável quando $\overline{K_{Xi}}(\Sigma_u \cup \Sigma_f) \cap L \subseteq \overline{K_{Xi}}$. Seja ainda $supC_{uf}(K_{Xi}, L)$ a máxima linguagem (L,Γ_{uf})-controlável contida em K_{Xi}.

Proposição 2.11 (Γ_{uf}, Γ_f e Γ_{Yi}-Controlabilidade) *Seja K_{Xi} uma linguagem (L,Γ_{uf})-controlável. K_{Xi} é também $(L, cl(\Gamma_f))$-controlável e (L,Γ_{Yi})-controlável.*

Prova:

Primeiramente, $\overline{K_{Xi}}\Sigma_u \cap L \subseteq \overline{K_{Xi}}(\Sigma_u \cup \Sigma_f) \cap L \subseteq \overline{K_{Xi}}$. Logo, K_{Xi} é $(L, cl(\Gamma_f))$-controlável.

Por outro lado, $\Sigma_{u,Yi} \subseteq \Sigma_u \cup \Sigma_f$. Assim, $\overline{K_{Xi}}\Sigma_{u,Yi} \cap L \subseteq \overline{K_{Xi}}(\Sigma_u \cup \Sigma_f) \cap L \subseteq \overline{K_{Xi}}$. Portanto, K_{Xi} é também (L,Γ_{Yi})-controlável. □

Dadas especificações globais K_{Xi}, $i = 1,\ldots,n$, quando apenas o supervisor calculado para uma linguagem K_j corresponde o comportamento de forçar eventos de Σ_f, os resultados apresentados nas proposições 2.10 e 2.11 permitem que se enuncie a seguinte proposição:

Proposição 2.12 *Sejam linguagens $K_{Xi} \subseteq L$, $i = 1,\ldots,n$, definidas sobre $\Sigma = \Sigma_u \cup \Sigma_c \cup \Sigma_f$. Suponha que apenas o supervisor calculado a partir da especificação K_j, $j \in \{1,\ldots,n\}$, força os eventos de Σ_f. Seja o conjunto de linguagens $SUP = \{supC_{uf}(K_{Xi}, L) |\ i = 1,\ldots,n, i \neq j\} \cup \{supC_f(K_j, L)\}$. Se o conjunto SUP é modular, então a intersecção dos elementos de SUP é $(L, cl(\Gamma_f))$-controlável.*

Prova:

Para $i = 1,\ldots,n$, $i \neq j$, $supC_{uf}(K_{Xi}, L)$ é, pela proposição 2.11, $(L, cl(\Gamma_f))$-controlável e (L,Γ_{Yi})-controlável.

Por definição, $supC_f(K_j, L)$ é $(L, cl(\Gamma_f))$-controlável. Além disso, como apenas K_j força os eventos de Σ_f, $\Sigma_{c,Yj} = \Sigma$ e $\Sigma_{u,Yj} = \emptyset$. Assim, $supC_f(K_j, L)$ é também (L,Γ_{Yj}) controlável.

Logo, da proposição 2.10, $(\cap_{i=1,i\neq j}^{n} supC_{uf}(K_{Xi},L)) \cap supC_f(K_j,L)$ é $(L,cl(\Gamma_f))$-controlável. □

Assim, quando o conjunto de especificações globais K_{Xi}, $i = 1,\ldots,n$ contém apenas um elemento K_j responsável por forçar eventos de Σ_f, a proposição 2.12 sugere a adoção do seguinte procedimento para obtenção de supervisores modulares:

Proposição 2.13 (Procedimento de síntese modular, abordagem com eventos forçáveis) *Sejam linguagens $K_{Xi} \subseteq L$, $i = 1,\ldots,n$, definidas sobre $\Sigma = \Sigma_u \cup \Sigma_c \cup \Sigma_f$. Suponha que apenas o supervisor calculado a partir da especificação K_j, $j \in \{1,\ldots,n\}$, força os eventos de Σ_f. A seguinte seqüência de passos pode ser adotada para obtenção de supervisores com comportamento conjunto não bloqueante e $(L,cl(\Gamma_f))$-controlável:*

1. Utilizando a abordagem com eventos forçáveis, calcular a máxima linguagem $(L,cl(\Gamma_f))$-controlável contida em K_j.
2. Utilizando a abordagem de eventos controláveis, calcular, para cada uma das especificações globais restantes, a máxima linguagem (L,Γ_{uf})-controlável. Note que, aqui, os eventos forçáveis são considerados como não controláveis.
3. Verificar a modularidade do conjunto de supervisores calculados. □

Caso o teste de modularidade proposto no item 3 falhe, pode-se repetir o procedimento com um conjunto menos refinado de linguagens, resultante da combinação de algumas das linguagens.

Embora restritiva, a exigência de apenas um supervisor modular forçar todos os eventos é adequada para a demonstração de resultados apresentados no presente trabalho. No entanto, o comportamento obtido com esta abordagem não é garantidamente idêntico àquele obtido por pelo cálculo da máxima linguagem $(L,cl(\Gamma_f))$-controlável contida na intersecção das especificações globais, $supC_f(\cap_{i=1}^{n} K_{Xi},L)$.

Considere agora que um subconjunto $\{K_{Xi}, i = 1,\ldots,k\}$ das especificações globais corresponde a supervisores modulares que serão responsáveis por forçar eventos de Σ_f. Neste caso, é necessário também garantir que o comportamento conjunto dos supervisores calculados a partir destes supervisores, $\cap_{i=1}^{k} supC_f(K_{Xi},L)$, é controlável. Assim, a partir dos resultados das proposições 2.9 e 2.10, o seguinte procedimento é sugerido:

Proposição 2.14 (Procedimento de síntese modular, abordagem com eventos forçáveis) *Sejam linguagens $K_{Xi} \subseteq L$, $i = 1,\ldots,n$, definidas sobre $\Sigma = \Sigma_u \cup \Sigma_c \cup \Sigma_f$. Suponha que apenas os supervisores calculados a partir das especificações $K_{Xi}, i = 1,\ldots,k, k \leq n$, forçam eventos de Σ_f. A seguinte seqüência de passos pode ser adotada para obtenção de supervisores com comportamento conjunto não bloqueante e $(L,cl(\Gamma_f))$-controlável:*

1. Utilizando a abordagem com eventos forçáveis, calcular, para cada K_{Xi}, $i = 1,\ldots,k$, a máxima linguagem $(L, cl(\Gamma_f))$-controlável contida em K_{Xi}.

2. Verificar se as linguagens obtidas, $supC_f(K_{Xi}, L)$, $i = 1,\ldots,k$, são Γ_{Yi}-controláveis em relação à linguagem $\cup_{i=1}^{k} supC_f(K_{Xi}, L)$, ou em relação à linguagem L.

3. Utilizando a abordagem de eventos controláveis, calcular, para cada uma das especificações globais restantes, a máxima linguagem (L, Γ_{uf})-controlável. Aqui, novamente, os eventos forçáveis são considerados como não controláveis.

4. Verificar a modularidade do conjunto de supervisores calculados. □

Quando os testes de controlabilidade ou modularidade propostos falham, é possível repetir o procedimento com um conjunto menos refinado de linguagens. Em particular, quando o teste proposto no item 2 falha sucessivamente, a iteração do procedimento pode forçar a necessidade de que um único supervisor force eventos, como apresentado na proposição 2.13.

2.2.5 Controle Modular Local e Eventos Forçáveis

Embora tenha sido discutida, na seção anterior, a aplicabilidade da abordagem de eventos forçáveis para o controle modular, a extensão dos resultados mostrados para o controle modular local não é trivial. Mesmo em se tratando de supervisores que atuam sobre plantas locais assíncronas, é possível que o comportamento conjunto do sistema sob supervisão não seja controlável, como ilustrado no exemplo a seguir.

Exemplo 2.9 (Não-controlabilidade de supervisores assíncronos com eventos forçáveis)
Sejam alfabetos disjuntos $\Sigma_1 = \{c_1, c_2, u_1, u_2\}$ e $\Sigma_2 = \{f, u_3, u_4\}$, e as seguintes linguagens $L_1, K_1 \subseteq \Sigma_1^*$ e $L_2, K_2 \subseteq \Sigma_2^*$: $L_1 = \{\varepsilon, c_1, c_1u_1, c_2, c_2u_2\}$, $K_1 = \{\varepsilon, c_1, c_1u_1, c_2\}$, $L_2 = \{\varepsilon, f, u_3, u_4\}$, $K_2 = \{\varepsilon, f, u_3\}$. Sejam também $\Sigma_{c,1} \cup \Sigma_{c,2} = \{c_1, c_2\}$, $\Sigma_{u,1} \cup \Sigma_{u,2} = \{u_1, u_2, u_3, u_4\}$, $\Sigma_{f,1} \cup \Sigma_{f,2} = \{f\}$.

Neste caso, $supC_f(K_1, L_1) = \{\varepsilon, c_1, c_1u_1\}$ e $supC_f(K_2, L_2) = \{\varepsilon, f\}$. Assim, $supC_f(K_1, L_1) \|_{as}$ $supC_f(K_2, L_2) = \{\varepsilon, c_1, c_1u_1\} \|_{as} \{\varepsilon, f\}$. Por outro lado, pode-se verificar que $supC_f(K_1 \|_{as} K_2,$ $L_1 \|_{as} L_2) = \{\varepsilon, f, fc_1, fc_1u_1\}$.

Assim, $supC_f(K_1, L_1) \|_{as} supC_f(K_2, L_2) \not\subseteq supC_f(K_1 \|_{as} K_2, L_1 \|_{as} L_2)$. □

Uma discussão mais profunda sobre a combinação das abordagens de eventos forçáveis e controle modular local não é apresentada neste trabalho, podendo ser objeto de estudos posteriores.

Capítulo 3

Aproximações em Sistemas a Eventos Discretos

3.1 Introdução

Aproximação do comportamento é um artifício recorrente no projeto de controle de sistemas dinâmicos. No contexto de sistemas a eventos discretos, a aproximação do comportamento da planta foi introduzida na formulação do problema de controle supervisório robusto (Cury e Krogh, 1999). Neste problema, a partir de uma planta nominal e especificações de comportamento mínimo e máximo aceitáveis, deseja-se obter um supervisor que, além de fornecer a solução ótima para o PCS nominal, maximize a cardinalidade do conjunto de plantas para as quais o comportamento em malha fechada esteja contido nos limites pré-definidos. A robustez do supervisor obtido na resolução do PCS nominal é definida em termos de variações aceitáveis pela planta.

Em sistemas dinâmicos híbridos, nem sempre existe uma realização finita, em número de estados, para o comportamento da planta, ou, mesmo que exista, encontrá-la pode ser computacionalmente inviável. Por esta razão, o cálculo de aproximações conservadoras do comportamento do sistema é essencial para a resolução do problema de controle supervisório sobre sistemas híbridos (Cury *et al.*, 1998). Por aproximação conservadora, ou externa, entende-se um modelo para o sistema real que contemple todas as trajetórias possíveis do sistema, incluindo possivelmente trajetórias que não são possíveis.

Este capítulo visa a mostrar em que medida a aproximação externa do comportamento da planta pode também ser utilizada, no contexto do problema de controle supervisório, para redução tanto da complexidade computacional do cálculo do supervisor quanto do número de estados da representação obtida para o supervisor. As principais limitações do uso de aproximações externas serão descritas,

bem como as condições para que os problemas de controle supervisório aproximado e exato sejam equivalentes.

Além disso, este capítulo introduz um caso particular de aproximação externa, definida para um subconjunto dos alfabetos da planta. Para este caso particular, são deduzidos resultados que serão utilizados nos capítulos posteriores.

3.2 Aproximação Externa da Planta

Seja um sistema a eventos discretos representado por um gerador G, com comportamentos gerado e marcado definidos sobre um alfabeto Σ. Um gerador A é denominado *aproximação externa de G*, denotado por $A \geq G$, se A define linguagens gerada e marcada sobre o mesmo alfabeto Σ, tais que $L(G) \subseteq L(A)$ e $L_m(G) \subseteq L_m(A)$.

Dada uma especificação $K \subseteq \Sigma^*$ para G, não necessariamente contida em $L_m(G)$, o problema de controle supervisório pode agora ser resolvido para a planta aproximada A. O resultado da aplicação do supervisor h, obtido para A, sobre a planta G, pode ser calculado a partir da seguinte proposição:

Proposição 3.1 (Supervisores e aproximação externa) *(Cury* et al.*, 1998) Sejam geradores A e G, definidos sobre* Σ*, tais que* $A \geq G$*. Para qualquer supervisor* $h : \Sigma^* \to \Gamma$*,* $L(h|G) = L(h|A) \cap L(G)$ *e* $L_m(h|G) = L_m(h|A) \cap L_m(G)$. □

Corolário 3.1 *(Cury* et al.*, 1998) Sejam geradores A e G, com* $A \geq G$*. Para qualquer supervisor* $h : \Sigma^* \to \Gamma$*,* $h|A \geq h|G$. □

O corolário 3.1 indica que a aplicação de um supervisor é uma operação monotônica para aproximações. Como conseqüência desta propriedade, se um supervisor mantêm uma aproximação externa A sob limites determinados por uma especificação K, então também a planta exata G, sob ação do mesmo supervisor, não extrapola K. A relação entre limites inferiores e superiores de comportamento e as linguagens marcadas de A e G sob supervisão é mais precisamente estabelecida no corolário 3.2.

Corolário 3.2 *(Cury* et al.*, 1998) Sejam linguagens* $M, K \subseteq \Sigma^*$*, e geradores A e G, definidos sobre* Σ*, tais que* $A \geq G$*. Se* $M \subseteq L_m(h|A) \subseteq K$ *então* $M \cap L_m(G) \subseteq L_m(h|G) \subseteq K$. □

3.2.1 Bloqueio e Aproximação

A resolução do problema de controle supervisório para uma aproximação A da planta não garante que o supervisor encontrado h seja não bloqueante para G, ou seja, dado $\overline{L_m(h|A)} = L(h|A)$, nem

sempre é verdade que $\overline{L_m(h|G)} = L(h|G)$. Neste sentido, podem-se considerar duas alternativas para garantir que a propriedade de não-bloqueio seja mantida para a planta G:

- Estabelecer restrições sobre as aproximações externas válidas, de forma que se garanta que qualquer supervisor h obtido para uma aproximação externa válida seja também não bloqueante para G.
- Para um dado supervisor h obtido para uma aproximação externa A qualquer, verificar uma condição que garanta que h é não bloqueante para G.

A proposição a seguir apresenta uma condição necessária e suficiente para que um supervisor calculado para uma aproximação qualquer A de G não seja bloqueante para G.

Proposição 3.2 (Não-bloqueio e aproximação externa) *(Cury e Krogh, 1999) Sejam geradores A e G, com $A \geq G$. Dado um supervisor $h : \Sigma^* \to \Gamma$, não bloqueante para A, h será não bloqueante para G se e somente se $\overline{L_m(h|A)} \cap L(G) = \overline{L_m(h|A) \cap L_m(G)}$.* □

A prova da proposição 3.2 pode ser deduzida diretamente da definição de não-bloqueio e da proposição 3.1. Pode-se ainda mostrar que a condição de não-bloqueio apresentada é equivalente à condição de modularidade para as linguagens $L_m(h|A)$ e $L_m(G)$, quando se assume que G é trim. O exemplo 3.1 ilustra que, ainda que o problema de controle supervisório resolvido para a aproximação de uma planta forneça um supervisor não bloqueante para a planta aproximada, este supervisor pode ser bloqueante para a planta original.

Exemplo 3.1 (Dois usuários e dois recursos)
Seja o problema de controle de alocação de dois recursos para dois usuários, como definido no exemplo 2.5. As restrições de exclusão mútua no uso dos recursos podem ser expressas pelas especificações genéricas dadas pelos geradores E_i, $i = 1, 2$, ilustrados na figura 3.1.

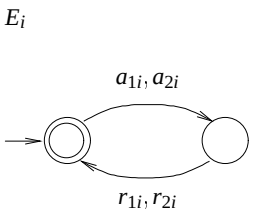

Figura 3.1: Especificações de exclusão mútua no uso dos recursos ($i = 1, 2$).

Pode-se verificar que a composição destas especificações genéricas com o modelo da planta global G, representado na figura 2.10, dá origem a uma especificação global idêntica ao supervisor bloqueante apresentado na figura 2.11. O cálculo do supervisor não bloqueante para G conduz à solução apresentada na figura 2.12.

Seja agora uma aproximação externa para o comportamento dos usuários 1 e 2, que permita que um usuário complete uma tarefa com a alocação de apenas um recurso, como representado pelos geradores $G_{1,ap}$ e $G_{2,ap}$ na figura 3.2.

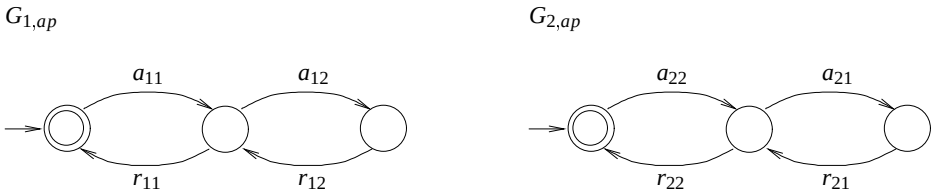

Figura 3.2: Modelos aproximados para os usuários 1 ($G_{1,ap}$) e 2 ($G_{2,ap}$).

De acordo com estes modelos, o usuário 1, para executar sua tarefa, aloca o recurso 1 (evento a_{11}), alocando possivelmente o recurso 2 (evento a_{12}) para a execução da tarefa, enquanto o usuário 2 aloca obrigatoriamente o recurso 2 (evento a_{22}) e eventualmente o recurso 1 (evento a_{21}) para a execução de sua tarefa.

Sejam G_{ap} e K_{ap}, respectivamente, a planta global para o modelo aproximado e a especificação global correspondente, $G_{ap} = G_{1,ap} \| G_{2,ap}$ e $K_{ap} = E_1 \| E_2 \| G_{ap}$. O gerador S_{ap} que reconhece a linguagem $supC(K_{ap}, G_{ap})$ é ilustrado na figura 3.3.

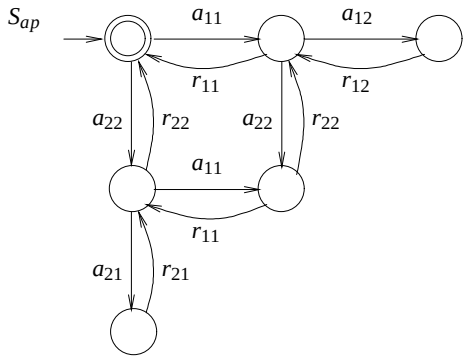

Figura 3.3: Supervisor para modelo aproximado dos usuários 1 e 2.

Para o modelo exato G, o comportamento em malha fechada obtido pela aplicação do supervisor S_{ap} pode ser calculado por $S = S_{ap} \| G$, conforme enunciado na proposição 3.1. O gerador S obtido é equivalente àquele representado na figura 2.11, que não é trim, como mencionado no exemplo 2.5. Logo, o supervisor S_{ap}, não bloqueante para G_{ap}, é bloqueante para G. □

3.2.2 Controlabilidade e Aproximação

Dada uma especificação qualquer K, contida na linguagem marcada de uma planta G, a existência de um supervisor h tal que $L_m(h|G) = K$ depende da controlabilidade de K com relação a G, como apresentado no capítulo 2.

Para uma aproximação A da planta G, a proposição a seguir mostra que, diferentemente da propriedade de não-bloqueio, a controlabilidade de K com relação à aproximação A é suficiente para que K seja também controlável com relação a G:

Proposição 3.3 (Controlabilidade e aproximação externa) *Sejam geradores A e G, definidos em Σ, tais que $A \geq G$. Seja também uma linguagem $K \subseteq \Sigma^*$. Se K é controlável em relação a A, então K é controlável em relação a G.*

Prova:

Como $A \geq G$, $\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}\Sigma_u \cap L(A) \subseteq \overline{K}$. □

Entretanto, uma linguagem K controlável em relação a G não é, necessariamente, controlável em relação a A. Em particular, para uma linguagem K qualquer, é possível que $supC(K,G)$ não seja controlável com relação a A, embora seja, por definição, controlável com relação a G. O exemplo a seguir exemplifica uma situação onde a volta da proposição 3.3 não é válida.

Exemplo 3.2 (Não-controlabilidade da aproximação)
Seja um SED G definido sobre $\Sigma = \{\alpha, \beta\}$, com linguagens $L_m(G) = \{\alpha, \alpha\beta\}$ e $L(G) = \overline{L_m(G)}$. Seja ainda $\Sigma_u = \{\beta\}$. Para uma especificação $K = L_m(G)$, claramente $supC(K,G) = L_m(G)$, ou seja, K é controlável com relação a G.

Suponha agora uma aproximação externa A de G, tal que $L_m(A) = L(A) = \Sigma^*$. Agora, a mesma especificação $K \subset L_m(A)$ não é controlável com relação a A, pois para $s = \varepsilon \in \overline{K}$, a cadeia $s\beta$ pertence a $L(A)$, mas não a $\overline{K}$. Neste caso, pode-se mostrar que $supC(K,A) = \emptyset$, o que significa que, embora o problema de controle supervisório possua solução quando formulado para K e G, não há solução satisfatória para o PCS formulado para K e A. □

Suponha agora que um problema de controle supervisório, com especificação genérica $E \subseteq \Sigma^*$, seja resolvido separadamente para a planta aproximada A e para a planta G. Para as linguagens $K_{ap} = E \cap L_m(A)$ e $K = E \cap L_m(G)$, os supervisores obtidos podem ser representados por $h_{ap} : L_m(h_{ap}|A) = supC(K_{ap}, A)$ e $h : L_m(h|G) = supC(K,G)$.

Pela proposição 3.3, $supC(K_{ap}, A)$ é controlável em relação a G. No entanto, a utilização de h_{ap} no sistema real pode ser muito restritiva, pois a aproximação A pode conter novas cadeias não controláveis que transgridam a especificação E, necessitando de uma ação de controle de desabilitação de eventos controláveis que, para a planta real G, não seria necessária.

Neste sentido, é desejável poder verificar se a aplicação na planta G de um supervisor calculado para A recupera o comportamento ótimo obtido com o supervisor calculado a partir de G. Em particular, quando $supC(K,G) \subseteq supC(K_{ap}, A)$, a seguinte proposição pode ser enunciada:

Proposição 3.4 *Sejam plantas G e A definidas sobre Σ, tais que A é uma aproximação externa de G, e seja $E \subseteq \Sigma^*$ uma linguagem possivelmente não contida em $L_m(G)$. Sejam ainda as linguagens $K = E \cap L_m(G)$ e $K_{ap} = E \cap L_m(A)$. Se as linguagens $supC(K_{ap}, A)$ e $L_m(G)$ são modulares, e além disso $supC(K,G) \subseteq supC(K_{ap}, A)$, então $supC(K_{ap}, A) \cap L_m(G) = supC(K,G)$.*

Prova:

$supC(K_{ap}, A) \cap L_m(G) \subseteq supC(K,G)$:

A linguagem $supC(K_{ap}, A)$ é, pela proposição 3.3, controlável em relação a G. Como a linguagem $L_m(G)$ é também controlável em relação a G, e as linguagens $supC(K_{ap}, A)$ e $L_m(G)$ são modulares, então $supC(K_{ap}, A) \cap L_m(G)$ é controlável em relação a G. Além disso, $supC(K_{ap}, A) \cap L_m(G) \subseteq E \cap L_m(A) \cap L_m(G) \subseteq K$. Assim, $supC(K_{ap}, A) \cap L_m(G)$ está contida na máxima linguagem contida em K controlável em relação a G.

$supC(K,G) \subseteq supC(K_{ap}, A) \cap L_m(G)$:

Deduz-se diretamente da hipótese $supC(K,G) \subseteq supC(K_{ap}, A)$ e de $supC(K,G) \subseteq L_m(G)$. □

Para que as condições da proposição 3.4 sejam verificadas, é necessário que a linguagem $supC(K,G)$ seja calculada. No entanto, como o objetivo da resolução do PCS aproximado é justamente evitar este cálculo, é desejável estabelecer restrições diretamente sobre as aproximações e especificações válidas, de forma que a equivalência da solução obtida com $supC(K,G)$ seja automática.

3.3 Aproximações para subalfabetos

Nas seções anteriores, foi observado que a utilização de aproximações externas da planta para a resolução de um problema de controle supervisório pode introduzir problemas associados tanto à condição de não-bloqueio quanto à controlabilidade.

Esta seção apresenta um caso particular para aproximação de plantas, em que apenas um subconjunto do alfabeto da planta tem o comportamento aproximado. Para este caso, mostra-se que os problemas associados ao bloqueio e a controlabilidade apresentados anteriormente não ocorrem, desde que a especificação associada ao PCS restrinja apenas eventos que não são aproximados.

Os resultados teóricos apresentados nesta seção são fundamentais no desenvolvimento dos capítulos posteriores.

Sejam alfabetos Σ_δ e Σ, tais que $\Sigma_\delta \subseteq \Sigma$, e a projeção natural $P_{-\delta} : \Sigma \to (\Sigma - \Sigma_\delta)$. Para uma planta G definida sobre Σ, uma *aproximação em* Σ_δ é uma aproximação $A \geq G$ tal que $P_{-\delta}(L(A)) = P_{-\delta}(L(G))$ e $P_{-\delta}(L_m(A)) = P_{-\delta}(L_m(G))$.

Uma aproximação em Σ_δ de uma planta G, denotada por A, é dita *máxima* se, para qualquer aproximação A' em Σ_δ de G, $A \geq A'$. É fácil verificar que, se A é a aproximação máxima em Σ_δ de G, então $A = P_{-\delta}^{-1}(P_{-\delta}(L(G))) = P_{-\delta}(L(G))\|_{as}\Sigma_\delta^*$.

Para as proposições apresentadas nesta seção, sejam também o alfabeto Σ_i, tal que $\Sigma_\delta \subseteq \Sigma_i \subseteq \Sigma$, e as projeções $P_i : \Sigma \to \Sigma_i$ e $P_{i-\delta} : \Sigma \to (\Sigma_i - \Sigma_\delta)$. Seja ainda a planta G_δ, definida em Σ_δ, tal que $L_m(G_\delta) = L(G_\delta) = \Sigma_\delta^*$.

Proposição 3.5 *Sejam G_i e $G_{i,ap}$ plantas definidas em Σ_i tais que $G_{i,ap}$ é uma aproximação externa em Σ_δ de G_i. Seja ainda uma planta G_q definida em $\Sigma - \Sigma_\delta$. Então $G_q\|G_{i,ap}$ é uma aproximação em Σ_δ da planta $G_q\|G_i$.*

Prova:

Primeiramente, $G_q\|G_{i,ap} \geq G_q\|G_i$ decorre de $G_{i,ap} \geq G_i$ e da monotonicidade do produto síncrono.

Além disso, $L_m(G_q) \subseteq (\Sigma - \Sigma_\delta)^*$ e $L_m(G_{i,ap}) \subseteq \Sigma_i^*$. Como $(\Sigma - \Sigma_\delta) \cap \Sigma_i = \Sigma_i - \Sigma_\delta \subseteq (\Sigma - \Sigma_\delta)$, pela propriedade 2.10, $P_{-\delta}(L_m(G_q)\|L_m(G_{i,ap})) = P_{-\delta}(L_m(G_q))\|P_{-\delta}(L_m(G_{i,ap}))$. Assim, $P_{-\delta}(L_m(G_q)\|L_m(G_{i,ap})) = P_{-\delta}(L_m(G_q))\|P_{-\delta}(L_m(G_{i,ap})) = L_m(G_q)\|\ P_{i-\delta}(L_m(G_{i,ap}))$.

Também é possível mostrar que $P_{-\delta}(L_m(G_q)\|L_m(G_i)) = L_m(G_q)\|\ P_{i-\delta}(L_m(G_i))$. Portanto, como $G_{i,ap}$ é uma aproximação de G_i, $P_{-\delta}(L_m(G_q)\|L_m(G_{i,ap})) = P_{-\delta}(L_m(G_q)\|L_m(G_i))$.

De maneira análoga, é possível mostrar que $P_{-\delta}(L(G_q)\|L(G_{i,ap})) = P_{-\delta}(L(G_q)\|L(G_i))$.

Logo, $G_q\|G_{i,ap}$ é uma aproximação em Σ_δ da planta $G_q\|G_i$. □

Proposição 3.6 *Sejam G_i e $G_{i,ap}$ plantas definidas em Σ_i tais que $G_{i,ap}$ é uma aproximação externa em Σ_δ de G_i. Seja ainda uma planta G_q definida em $\Sigma - \Sigma_\delta$, tal que $P_{i-\delta}(L_m(G_q)) \subseteq P_{i-\delta}(L_m(G_i))$. Então $P_{-\delta}(L_m(G_q\|G_{i,ap})) = P_{-\delta}(L_m(G_q\|G_i)) = L_m(G_q)$.*

Prova:

Como $P_{i-\delta}(L_m(G_q)) \subseteq P_{i-\delta}(L_m(G_i))$, $L_m(G_q) \subseteq P_{i-\delta}(L_m(G_q))\|_{as}(\Sigma - \Sigma_\delta)^* \subseteq P_{i-\delta}(L_m(G_i))\|_{as}(\Sigma - \Sigma_\delta)^*$.

Assim, é verdade que $L_m(G_q)\|P_{i-\delta}(L_m(G_i)) = L_m(G_q) \cap (P_{i-\delta}(L_m(G_i))\|_{as}(\Sigma - \Sigma_\delta)^*) = L_m(G_q)$.

Portanto, utilizando a propriedade 2.10, $P_{-\delta}(L_m(G_q\|G_i)) = P_{-\delta}(L_m(G_q))\|P_{-\delta}(L_m(G_i)) = L_m(G_q)\|\ P_{i-\delta}(L_m(G_i)) = L_m(G_q)$. Analogamente, como $P_{i-\delta}(L_m(G_i)) = P_{i-\delta}(L_m(G_{i,ap}))$, é possível mostrar que $P_{-\delta}(L_m(G_q\|G_{i,ap})) = L_m(G_q)$. □

Proposição 3.7 *Seja uma linguagem K_q, definida em $\Sigma - \Sigma_\delta$, e plantas A e G, definidas em Σ, tais que A é a aproximação máxima em Σ_δ de G. Então $P_{-\delta}^{-1}(K_q)$ e $L_m(A)$ são modulares se e somente se K_q e $P_{-\delta}(L_m(G))$ são também modulares.*

Prova:

Primeiramente, $\overline{P_{-\delta}^{-1}(K_q)} \cap \overline{L_m(A)} = \overline{P_{-\delta}^{-1}(K_q)} \cap \overline{P_{-\delta}^{-1}(P_{-\delta}(L_m(G)))} = P_{-\delta}^{-1}(\overline{K_q}) \cap P_{-\delta}^{-1}(\overline{P_{-\delta}(L_m(G))}) = P_{-\delta}^{-1}(\overline{K_q} \cap \overline{P_{-\delta}(L_m(G))})$.

Além disso, $\overline{P_{-\delta}^{-1}(K_q) \cap L_m(A)} = \overline{P_{-\delta}^{-1}(K_q) \cap P_{-\delta}^{-1}(P_{-\delta}(L_m(G)))} = \overline{P_{-\delta}^{-1}(K_q \cap P_{-\delta}(L_m(G)))} = P_{-\delta}^{-1}(\overline{K_q \cap P_{-\delta}(L_m(G))})$.

i) (Somente se) suponha $P_{-\delta}^{-1}(K_q)$ e $L_m(A)$ modulares. Então, $P_{-\delta}^{-1}(\overline{K_q} \cap \overline{P_{-\delta}(L_m(G))}) = P_{-\delta}^{-1}(\overline{K_q \cap P_{-\delta}(L_m(G))})$.

Fazendo a projeção desta equação em $P_{-\delta}$, $\overline{K_q} \cap \overline{P_{-\delta}(L_m(G))} = \overline{K_q \cap P_{-\delta}(L_m(G))}$. Portanto, as linguagens K_q e $P_{-\delta}(L_m(G))$ são, também, modulares.

ii) (Se) se as linguagens K_q e $P_{-\delta}(L_m(G))$ são modulares, $\overline{K_q} \cap \overline{P_{-\delta}(L_m(G))} = \overline{K_q \cap P_{-\delta}(L_m(G))}$.

A modularidade de $P_{-\delta}^{-1}(K_q)$ e $L_m(A)$ decorre da projeção inversa desta igualdade em $P_{-\delta}$. □

Proposição 3.8 *Seja uma linguagem K_q, definida em $\Sigma - \Sigma_\delta$, e plantas A e G, definidas em Σ, tais que A é a aproximação máxima em Σ_δ de G. Se $P_{-\delta}^{-1}(K_q)$ e $L_m(A)$ são modulares, então $P_{-\delta}^{-1}(K_q)$ e $L_m(G)$ são também modulares.*

Prova:

i) $P_{-\delta}(\overline{P_{-\delta}^{-1}(K_q) \cap L_m(G)}) = \overline{P_{-\delta}(K_q \| L_m(G))} = \overline{P_{-\delta}(K_q) \| P_{-\delta}(L_m(G))} = \overline{K_q \cap P_{-\delta}(L_m(G))}$.

ii) $P_{-\delta}(\overline{P_{-\delta}^{-1}(K_q)} \cap \overline{L_m(G)}) = P_{-\delta}(\overline{K_q} \| \overline{L_m(G)}) = P_{-\delta}(\overline{K_q}) \| P_{-\delta}(\overline{L_m(G)}) = \overline{P_{-\delta}(K_q)} \| \overline{P_{-\delta}(L_m(G))} = \overline{K_q} \cap \overline{P_{-\delta}(L_m(G))}$.

Suponha que as linguagens $P_{-\delta}^{-1}(K_q)$ e $L_m(G)$ não são modulares. Então existe uma cadeia $s \in \Sigma^*$ tal que $s \in \overline{P_{-\delta}^{-1}(K_q)} \cap \overline{L_m(G)}$, mas $s \notin \overline{P_{-\delta}^{-1}(K_q) \cap L_m(G)}$. Logo, existe ao menos uma cadeia $st \in L_m(G)$ tal que, para qualquer t' tal que $\varepsilon \prec t' \preceq t$, $st' \notin \overline{P_{-\delta}^{-1}(K_q)}$. Além disso, se t' é tal que $st' \in \overline{L_m(G)} - \overline{P_{-\delta}^{-1}(K_q)}$, pode-se afirmar que $P_{-\delta}(t') \neq \varepsilon$, pois caso contrário, $P_{-\delta}(st') = P_{-\delta}(s) \in \overline{K_q}$, e portanto $st' \in \overline{P_{-\delta}^{-1}(K_q)}$.

Em particular, seja s tal que, para qualquer $st' \succ s$ tal que $st' \in \overline{L_m(G)}$, $st' \notin \overline{P_{-\delta}^{-1}(K_q)}$.

Assim, para qualquer t' tal que $st' \in L_m(G) \subseteq \overline{L_m(G)}$, $P_{-\delta}(st') \in P_{-\delta}(L_m(G))$, mas $P_{-\delta}(st') = P_{-\delta}(s)P_{-\delta}(t') \notin \overline{K_q} \supseteq K_q$. Logo, $P_{-\delta}(s) \notin \overline{K_q \cap P_{-\delta}(L_m(G))}$.

No entanto, por (ii), é fácil verificar que $P_{-\delta}(s) \in \overline{K_q} \cap \overline{P_{-\delta}(L_m(G))}$.

Logo, K_q e $P_{-\delta}(L_m(G))$ não são modulares, o que contradiz a proposição 3.7. □

Como conseqüência da proposição 3.8, o conjunto de linguagens $\{L_m(G) \mid A$ é uma aproximação máxima em Σ_δ de $G\}$ constitui uma classe de linguagens para as quais a linguagem K_q é localmente modular, se K_q e A são localmente modulares. Assim, dada uma especificação K_q definida em uma linguagem $\Sigma_q \subseteq \Sigma$, se K_q é controlável em relação a A e K_q e A são localmente modulares, um supervisor S não bloqueante para A tal que $L_m(S|A) = K_q \| L_m(A)$ é também não bloqueante para G, quando a aproximação é restrita aos eventos de $\Sigma - \Sigma_q$.

As proposições a seguir objetivam mostrar que, restringindo a aproximação externa da planta a um subconjunto de eventos, também o problema associado à controlabilidade, apresentado na seção 3.2.2, pode ser contornado. Para tanto, é apresentado o seguinte resultado preliminar:

Proposição 3.9 *Sejam uma linguagem K_q e uma planta G_q, definidas em $\Sigma - \Sigma_\delta$, tais que $K_q \subseteq L_m(G_q)$. Seja também a planta G_i, definida em Σ_i, tal que $P_{i-\delta}(L_m(G_q)) \subseteq P_{i-\delta}(L_m(G_i))$. Então $P_{-\delta}(K_q \| L_m(G_i)) = K_q$. Além disso, $K_q \| L_m(G_i) = P_{-\delta}^{-1} P_{-\delta}(K_q \| L_m(G_i)) \| L_m(G_i)$.*

Prova:

Primeiramente, da proposição 3.6, $P_{-\delta}(L_m(G_q \| G_i)) = L_m(G_q)$. Além disso, como $K_q \subseteq L_m(G_q)$, $P_{i-\delta}(K_q) \subseteq P_{i-\delta}(L_m(G_q)) \subseteq P_{i-\delta}(L_m(G_i))$. Logo, pela mesma proposição, também é verdade que $P_{-\delta}(K_q \| L_m(G_i)) = K_q$.

Ainda, $P_{-\delta}^{-1} P_{-\delta}(K_q \| L_m(G_i)) = P_{-\delta}^{-1}(K_q)$, o que implica $P_{-\delta}^{-1} P_{-\delta}(K_q \| L_m(G_i)) \| L_m(G_i) = P_{-\delta}^{-1}(K_q) \| L_m(G_i) = K_q \| L_m(G_i)$. □

Desta forma, a linguagem $K_q \| L_m(G_i) \subseteq L_m(G_q \| G_i)$ pode ser recuperada quando projetada em $\Sigma - \Sigma_\delta$, bastando para isso compor o resultado da projeção com a linguagem gerada pela planta $L_m(G_q \| G_i)$. Diz-se, neste caso, que a linguagem $K_q \| L_m(G_i)$ é *normal* em relação à projeção $P_{-\delta}$ e à planta $L_m(G_q \| G_i)$ (Lin e Wonham, 1988). A seguintes proposições fornecem resultados importantes sobre as propriedades de normalidade e controlabilidade:

Proposição 3.10 *(Wonham, 2001) Para alfabetos $\Sigma_1 \subseteq \Sigma$, seja uma planta G, $L(G) \subseteq \Sigma^*$, e seja também a projeção $P : \Sigma \to \Sigma_1$. Seja uma linguagem $K \subseteq L_m(G)$, normal em relação a P e $L_m(G)$. Se G é trim, e as linguagens $P^{-1}(P(K))$ e $L_m(G)$ são modulares, então $\overline{K}$ é normal em relação a P e $L(G)$.*

Proposição 3.11 *(Lin e Wonham, 1990) Para alfabetos $\Sigma_1 \subseteq \Sigma$, seja uma planta G, $L(G) \subseteq \Sigma^*$, e seja uma linguagem $K \subseteq L_m(G)$, controlável com relação a G. Seja também a projeção $P : \Sigma \to \Sigma_1$. Se $\overline{K}$ é normal em relação a P e $L(G)$, então $P(K)$ é controlável com relação a $P(L(G))$.*

As proposições 3.9, 3.10 e 3.11 permitem que a seguinte relação entre a controlabilidade das linguagens $K_q||L_m(G_i)$ e K_q seja enunciada:

Proposição 3.12 *Sejam uma linguagem K_q e uma planta G_q, definidas em $\Sigma - \Sigma_\delta$, tais que $K_q \subseteq L_m(G_q)$. Seja também a planta G_i, definida em Σ_i, tal que $P_{i-\delta}(L_m(G_q)) \subseteq P_{i-\delta}(L_m(G_i))$. Se $K_q||L_m(G_i)$ é controlável com relação a $G_q||G_i$ e as plantas G_q, G_i e $G_q||G_i$ são trim, então K_q é controlável com relação a G_q.*

Prova:

Pela proposição 3.11, deve-se mostrar que $\overline{K_q||L_m(G_i)}$ é normal com relação a $P_{-\delta}$ e a $L(G_q||G_i)$.

Como, pela proposição 3.9, $K_q||L_m(G_i)$ é normal com relação a $P_{-\delta}$ e $L_m(G_q||G_i)$, pela proposição 3.10, basta mostrar que $P_{-\delta}^{-1}(P_{-\delta}(K_q||L_m(G_i)))$ e $L_m(G_q||G_i)$ são modulares, pois $G_q||G_i$ foi assumido trim.

Note que, pela proposição 3.9, $P_{-\delta}^{-1}(P_{-\delta}(K_q||L_m(G_i))) = P_{-\delta}^{-1}(K_q)$.

Além disso, como G_q é trim e $K_q \subseteq L_m(G_q)$, então K_q e $L_m(G_q)$ são modulares. Logo $P_{-\delta}^{-1}(K_q)$ e $P_{-\delta}^{-1}(L_m(G_q))$ são modulares. Portanto, utilizando a proposição 3.8, $L_m(G_q||G_i)$ e $P_{-\delta}^{-1}(K_q)$ são modulares. □

Para uma linguagem $K_q||L_m(G_i)$ não controlável com relação a $G_q||G_i$, uma relação equivalente à enunciada na proposição 3.12 pode ser apresentada, quando a linguagem $\overline{supC(K_q||L_m(G_i), G_q||G_i)}$ é normal em relação a $P_{-\delta}$ e $L(G_q||G_i)$. A seguinte proposição indica que esta condição é satisfeita por $supC(K_q||L_m(G_i), G_q||G_i)$:

Proposição 3.13 *Sejam uma linguagem K_q e uma planta G_q, definidas em $\Sigma - \Sigma_\delta$, tais que $K_q \subseteq L_m(G_q)$. Seja também a planta G_i, definida em Σ_i, tal que $P_{i-\delta}(L_m(G_q)) \subseteq P_{i-\delta}(L_m(G_i))$. Suponha, adicionalmente, que Σ_δ contém apenas eventos não controláveis. Então $\overline{supC(K_q||L_m(G_i), G_q||G_i)}$ é normal em relação a $P_{-\delta}$ e $L(G_q||G_i)$.*

Prova:

Suponha o caso particular $\Sigma_\delta = \{\delta\}$.

Basta provar que, para qualquer $t \in L(G_q||G_i)$, tal que $P_{-\delta}(s) = P_{-\delta}(t)$ para algum $s \in \overline{supC(K_q||L_m(G_i), G_q||G_i)}$, é verdade que $t \in \overline{supC(K_q||L_m(G_i), G_q||G_i)}$.

Suponha que existe t que não satisfaça esta propriedade. Neste caso, é possível afirmar que existe um prefixo v, comum a t e s, tal que $v\delta \preceq s$ e $v\delta \not\preceq t$, ou $v\delta \preceq t$ e $v\delta \not\preceq s$. A segunda condição certamente não é verdade, pois, como $v \in \overline{supC(K_q||L_m(G_i), G_q||G_i)}$ e $v\delta \in L(G_q||G_i)$, a continuação de v por δ deveria ser habilitada por $supC(K_q||L_m(G_i), G_q||G_i)$.

Neste caso, é possível mostrar que uma continuação de v sem δ é também controlável. Portanto t é controlável, o que implica a contradição $t \in supC(K_q \| L_m(G_i), G_q \| G_i)$. □

Como conseqüência das proposições 3.11 e 3.13, o seguinte resultado pode ser enunciado:

Proposição 3.14 *Sejam uma linguagem K_q e uma planta G_q, definidas em $\Sigma - \Sigma_\delta$, tais que $K_q \subseteq L_m(G_q)$. Seja também a planta G_i, definida em Σ_i, tal que $P_{i-\delta}(L_m(G_q)) \subseteq P_{i-\delta}(L_m(G_i))$. Suponha, adicionalmente, que Σ_δ contém apenas eventos não controláveis. Então $P_{-\delta}(supC(K_q \| L_m(G_i), G_q \| G_i))$ é controlável em relação a G_q.*

O dois resultados que seguem apresentam relações adicionais entre projeção inversa e controlabilidade:

Proposição 3.15 *(Queiroz, 2000) Sejam uma linguagem K_q e uma planta G_q, definidas em $\Sigma - \Sigma_\delta$, tais que K_q é controlável em relação a G_q. Então $P_{-\delta}^{-1}(K_q)$ é controlável em relação a $A = G_q \| G_\delta$.*

Proposição 3.16 *Sejam uma linguagem K_q e uma planta G_q, definidas em $\Sigma - \Sigma_\delta$, tais que K_q é controlável em relação a G_q, e uma planta $A = G_q \| G_\delta$, definida em Σ. Para qualquer $G \leq A$, $P_{-\delta}^{-1}(K_q)$ é controlável em relação a G.*

Prova:

Decorre das proposições 3.3 e 3.15. □

Finalmente, com base nos resultados apresentados nesta seção, o principal resultado deste capítulo pode ser enunciado:

Proposição 3.17 *Sejam uma linguagem K_q e uma planta G_q, definidas em $\Sigma - \Sigma_\delta$. Sejam ainda as plantas G_i, definida em Σ_i, e $G = G_q \| G_i$, definida em Σ, tais que $P_{i-\delta}(L_m(G_q)) \subseteq P_{i-\delta}(L_m(G_i))$. Suponha, adicionalmente, que Σ_δ contém apenas eventos não controláveis. Então $supC(K_q \cap L_m(G_q), G_q) \| L_m(G_i) = supC(P_{-\delta}^{-1}(K_q) \cap L_m(G), G)$.*

Prova:

Sejam as linguagens $K = P_{-\delta}^{-1}(K_q)$ e $K_{sup} = P_{-\delta}^{-1}(supC(K_q \cap L_m(G_q), G_q))$, e a planta $A = G_q \|_{as} G_\delta$. Sejam ainda as linguagens $K_G = K \cap L_m(G)$ (especificação K restrita à planta G) e $K_A = K \cap L_m(A)$.

i) $supC(K_q \cap L_m(G_q), G_q) \| L_m(G_i) \subseteq supC(K \cap L_m(G), G)$:

Primeiramente, $supC(K_q \cap L_m(G_q), G_q) \| L_m(G_i) = supC(K_q \cap L_m(G_q), G_q) \| L_m(G_q) \| L_m(G_i) = P_{-\delta}^{-1}(supC(K_q \cap L_m(G_q), G_q)) \cap (L_m(G_q) \| L_m(G_i)) = K_{sup} \cap L_m(G)$.

Como $supC(K_q \cap L_m(G_q), G_q)$ é controlável em relação a G_q, pelo corolário 3.16, K_{sup} é controlável em relação a G. Além disso, $L_m(G)$ é também controlável em relação a G. Ainda, como as linguagens $supC(K_q \cap L_m(G_q), G_q)$ e $L_m(G_q)$ são modulares, também K_{sup} e $L_m(A)$ são modulares. Então, pela proposição 3.8, K_{sup} e $L_m(G)$ são modulares.

Portanto, da modularidade de K_{sup} e $L_m(G)$ e da controlabilidade destas linguagens com relação a G, pode-se afirmar que $K_{sup} \cap L_m(G)$ é controlável em relação a G.

Além disso, como $supC(K_q \cap L_m(G_q), G_q) \subseteq K_q$, $K_{sup} \subseteq K$, e portanto $K_{sup} \cap L_m(G) \subseteq K \cap L_m(G)$.

Logo, $supC(K_q \cap L_m(G_q), G_q) \| L_m(G_i) = K_{sup} \cap L_m(G) \subseteq supC(K \cap L_m(G), G)$.

ii) $supC(K \cap L_m(G), G) \subseteq supC(K_q \cap L_m(G_q), G_q) \| L_m(G_i)$:

Como $supC(K_G, G) \subseteq L_m(G) = L_m(G_q) \| L_m(G_i) \subseteq P_i^{-1}(L_m(G_i))$, basta mostrar que $supC(K_G, G) \subseteq K_{sup}$.

Primeiramente, de $supC(K_G, G) \subseteq K_G$, pode-se deduzir que $P_{-\delta}^{-1}(P_{-\delta}(supC(K_G, G))) \subseteq P_{-\delta}^{-1}(P_{-\delta}(K_G)) \subseteq P_{-\delta}^{-1}(P_{-\delta}(K) \cap P_{-\delta}(L_m(G))) = P_{-\delta}^{-1}(P_{-\delta}(K)) \cap P_{-\delta}^{-1}(P_{-\delta}(L_m(G))) = P_{-\delta}^{-1}(P_{-\delta}(P_{-\delta}^{-1}(K_q))) \cap L_m(A) = P_{-\delta}^{-1}(K_q) \cap L_m(A) = K_A$.

Como $supC(K_G, G)$ é controlável em relação a G, então, pelas proposições 3.6 e 3.14, $P_{-\delta}(supC(K_G, G))$ é controlável em relação a G_q. Novamente, pela proposição 3.15, $P_{-\delta}^{-1}(P_{-\delta}(supC(K_G, G)))$ é controlável com relação a A. Então, como $P_{-\delta}^{-1}(P_{-\delta}(supC(K_G, G))) \subseteq K_A$, $P_{-\delta}^{-1}(P_{-\delta}(supC(K_G, G))) \subseteq supC(K_A, A)$.

Além disso, $supC(K_G, G) \subseteq P_{-\delta}^{-1}(P_{-\delta}(supC(K_G, G)))$. Assim, $supC(K_G, G) \subseteq supC(K_A, A)$.

Logo, $supC(K_G, G) \subseteq supC(K_A, A) = supC(P_{-\delta}^{-1}(K_q) \cap P_{-\delta}^{-1}(L_m(G_q)), A) = supC(P_{-\delta}^{-1}(K_q \cap L_m(G_q)), A) = supC((K_q \cap L_m(G_q)) \|_{as} \Sigma_\delta^*, G_q \|_{as} G_\delta)$.

Ainda, pela proposição 2.7, $supC((K_q \cap L_m(G_q)) \|_{as} \Sigma_\delta^*, G_q \|_{as} G_\delta) = supC(K_q \cap L_m(G_q),\ G_q) \|_{as} supC(\Sigma_\delta^*, G_\delta) = K_{sup}$.

Assim, $supC(K_G, G) \subseteq K_{sup}$. □

A proposição 3.17 indica que, sob certas condições, a introdução de uma planta adicional G_i em um problema de controle supervisório não influencia no resultado final do problema. O capítulo a seguir visa a mostrar que é possível entender a introdução de um sensor em um SED como a introdução de uma planta adicional que satisfaz estas condições.

Capítulo 4

Sensores em Controle Supervisório de SEDs

4.1 Introdução

Recentemente, diversos problemas envolvendo alocação e minimização de sensores em sistemas a eventos discretos foram propostos na literatura. No entanto, estes trabalhos estão baseados principalmente no controle de SEDs com observação parcial de eventos (Lin e Wonham, 1988), considerando que, dado um alfabeto Σ fixo, a alocação de um sensor no sistema real determina a observabilidade do evento correspondente em Σ. Enquanto os trabalhos de Young e Garg (1993) e Haji-Valizadeh e Loparo (1996) propõem algoritmos para escolhas com número mínimo de sensores de forma que o comportamento em malha fechada seja igual a um comportamento especificado, Yoo e Lafortune (2002) mostram que a determinação de conjuntos mínimos para estes problemas é NP-completa. É interessante notar que a observação parcial de eventos pode ser vista como um problema de redução no número de eventos do gerador que representa o supervisor (Haji-Valizadeh e Loparo, 1996).

Neste trabalho, a discussão sobre sensores para sistemas a eventos discretos é revisitada, porém agora dentro do contexto do controle supervisório com observação total de eventos. Entretanto, a observabilidade total não representa, como nos trabalhos anteriores, a existência de sensores para todos os eventos do alfabeto, já que outra conotação é associada aos sensores.

Para o presente trabalho, um sensor é, essencialmente, um elemento que sinaliza a ocorrência de uma seqüência de eventos no sistema original (sem sensor). A alocação do sensor significa adicionar o evento do sensor ao alfabeto da planta original, e adaptar a linguagem da planta para indicar, explicitamente, quando o sensor é ativado, ou *sensibilizado*. A adaptação do modelo da planta para incluir o sensor pode ser realizada pela combinação do modelo original da planta com um modelo que indique todas as situações onde o sensor pode ser ativado, denominado modelo exato do sensor.

Na abordagem considerada, para um problema previamente definido, deseja-se que a introdução de sensores permita que os geradores associados ao cálculo do supervisor possam ser representados por geradores menores, de forma a reduzir tanto o custo computacional da resolução do problema quanto o número de estados da solução final.

Pretende-se avaliar também como as abordagens de controle modular e modular local podem ser influenciadas pela alocação de sensores. Nos exemplos apresentados, mostra-se que a introdução de sensores permite que se obtenha redução na representação das especificações genéricas. No entanto, a representação por sistemas produto da planta após a introdução do modelo do sensor é, de forma geral, menos refinada do que a original. Ainda assim, o custo computacional do cálculo dos supervisores locais pode ser reduzido em relação ao problema original, desde que sejam utilizadas aproximações para o comportamento do sensor.

O exemplo a seguir é apresentado como motivação para a discussão sobre inserção de sensores em SEDs realizada neste capítulo.

Exemplo 4.1 (Mesa Giratória com Posição Intermediária)
Seja o problema da mesa giratória apresentado no exemplo 2.7. Considere agora uma nova configuração da mesa, com a agregação de uma posição intermediária P', de forma que sejam necessários dois passos da mesa para transladar uma peça da posição P_1 para a posição P_2, conforme ilustrado na figura 4.1. Note que, agora, cada passo da mesa corresponde a um giro de 72 graus.

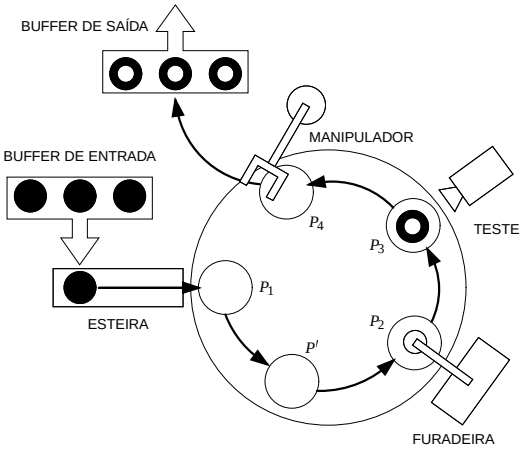

Figura 4.1: Mesa aumentada de uma posição P'.

Embora uma alteração da planta física tenha sido efetuada, a modelagem da planta livre pode ser realizada pela mesma RSP descrita no exemplo 2.7. Por outro lado, esta nova configuração física leva à necessidade de que algumas restrições utilizadas no projeto dos supervisores localmente modulares sejam adaptadas. Primeiramente, pode-se notar que, como não há realização de operação em P', o estado de P_2 não pode ser deduzido a partir de eventos gerados na posição precedente. Por esta razão, torna-se necessário agora armazenar, em uma única especificação Rc'_1, informação sobre as

duas posições anteriores a P_2, de forma que seja possível inferir sobre a presença ou ausência de peça nesta posição após um passo da mesa. Este comportamento pode ser representado pelo gerador Rc'_1, apresentado na figura 4.2. Note que este gerador tem 8 estados, contra 4 da especificação anterior Rc_1.

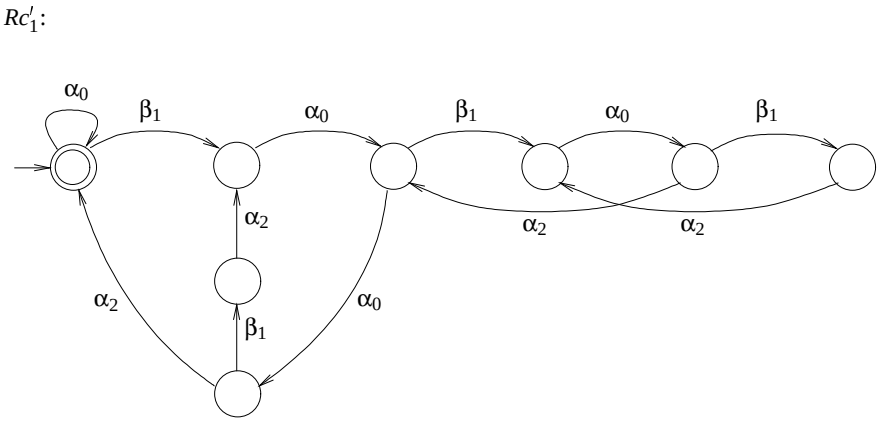

Figura 4.2: A especificação Rc_1 é substituída por Rc'_1, para refletir a alteração na planta.

Também a especificação Ra, de 2 estados, deve agora ser substituída por uma nova especificação Ra', com 3 estados, que permita dois giros subsequentes da mesa após a operação de carga em P_1 (figura 4.3).

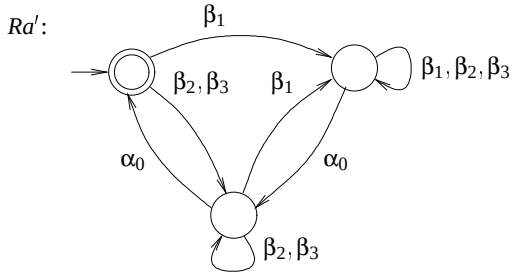

Figura 4.3: A especificação Ra é substituída por Ra', para refletir a alteração na planta.

As plantas locais para as especificações Ra' e Rc'_1 são idênticas àquelas para as especificações Ra e Rc_1, respectivamente. Assim, para o novo conjunto de especificações $\{Ra', Rb_1, Rb_2, Rb_3, Rb_4, Rc'_1, Rc_2, Rc_3\}$, são calculados os supervisores locais correspondentes. Também para este caso, a modularidade local do conjunto $\{Sa', Sb_1, Sb_2, Sb_3, Sb_4, Sc'_1, Sc_2, Sc_3\}$ é verificada. A tabela 4.1 apresenta o número de estados dos geradores envolvidos nos processos de síntese modular local e monolítica, bem como o número de estados e transições para os supervisores reduzidos.

Com o acrescimo da posição P', o supervisor modular local para Ra passa de 32 para 48 estados, enquanto o supervisor para Rc_1 passa de 24 para 48 estados. Este aumento pode ser observado também na comparação dos supervisores reduzidos. Pode-se observar, portanto, que a necessidade de armazenar informações sobre um maior número de posições da mesa em uma única especifica-

Resolução Modular Local	E_x	G_X	E_X	S_X	RS_X
a'	3	16	48	48	$(3,11)$
$b_i, i = 1,2,3,4$	2	4	3	3	$(2,4)$
c'_1	8	8	48	48	$(8,16)$
$c_i, i = 2,3$	4	8	32	24	$(4,8)$
Total (somatório)	27	56	172	156	$(27,59)$
Resolução Monolítica	R	G	K	S	
	399	32	399	303	

Tabela 4.1: Número de estados dos geradores para o problema da mesa com 5 posições (exemplo 4.1). Para as soluções finais, é apresentado o par (número de estados, número de transições).

ção gera um crescimento exponencial do número de estados da especificação, e por conseqüência no cálculo do supervisor correspondente. Assim, a alteração proposta provoca um aumento indesejável do custo computacional do cálculo da solução modular local, ainda que a planta alterada seja funcionalmente semelhante à original. Note, também, que a informação que deve ser armazenada para utilização por estas especificações é de presença de peça na posição P', que pode ser deduzida a partir de um depósito de peça em P_1, seguido da execução de um passo da mesa. □

Como será apresentado na seção seguinte, as especificações que foram alteradas em função da inserção de uma posição intermediária da mesa, no exemplo 4.1, podem ser expressas como especificações menores, desde que para tanto seja alocado um sensor que indique a presença de peça em P'. Para isso, é necessário que se discuta como modelar o comportamento associado ao sensor.

4.2 Modelagem Exata de Sensores em SEDs

Seja uma planta representada por um gerador G_q, definido sobre um alfabeto Σ_q. Suponha que se pretenda adicionar à planta um sensor T, cuja sensibilização é modelada por um evento representado pelo símbolo δ, que não pertence ao alfabeto original de G_q. Suponha ainda que, dadas as configurações reais de G_q e da alocação de T, a sensibilização de T possa ser deduzida a partir de cadeias definidas em $L(G_q)$. Pretende-se projetar um modelo para a planta que contenha explicitamente o evento δ, a partir da especificação da linguagem que identifique todas as cadeias que sensibilizam T. Por sua vez, a dessensibilização de T não será modelada como um evento específico, mas sim deduzida, após cada sensibilização, diretamente a partir de cadeias definidas em $L(G_q)$.

Para um sistema em sua representação por sistemas produto, G_q pode corresponder a uma planta local, definida pela combinação das plantas assíncronas que contêm eventos em comum com as cadeias que definem a sensibilização do sensor.

Para a modelagem da planta acrescida do sensor T, define-se o alfabeto $\Sigma_{q\delta} = \Sigma_q \cup \{\delta\}$. Sejam ainda alfabetos Σ_d, Σ_{ss}, Σ_s que satisfaçam as seguintes relações: $\Sigma_d \subseteq \Sigma_{ss} \subseteq \Sigma_q$ e $\Sigma_s = \Sigma_{ss} \dot{\cup} \{\delta\}$.

O alfabeto Σ_{ss} corresponde ao subconjunto de eventos da planta G_q que são relevantes para a sensibilização do sensor T. As cadeias de eventos definidas em Σ_{ss} que sensibilizam T são especificadas atráves de uma linguagem denominada *linguagem sensibilizadora*, representada por $L_{ss} \subseteq \Sigma_{ss}^*$. Por sua vez, o alfabeto Σ_d define os *eventos dessensibilizadores* de T. Neste caso, após uma cadeia reconhecida por L_{ss}, a ocorrência de um evento de Σ_d indica que T deixou de estar sensibilizado.

Sejam também as projeções $P_d : \Sigma_{q\delta} \to \Sigma_d$, $P_{ss} : \Sigma_{q\delta} \to \Sigma_{ss}$, $P_s : \Sigma_{q\delta} \to \Sigma_s$ e $P_\delta : \Sigma_{q\delta} \to \{\delta\}$.

Uma linguagem $L_{ss} \subseteq \Sigma_{ss}^*$ pode ser utilizada como linguagem sensibilizadora para a planta G_q se satisfizer as seguintes condições:

- O comportamento gerado pela linguagem sensibilizadora não deve restringir o comportamento original gerado pela planta livre.
- A linguagem sensibilizadora deve indicar que o sensor deixa de estar sensibilizado antes que uma nova sensibilização ocorra ou simultaneamente à nova sensibilização.

As condições apresentadas podem ser expressas formalmente por meio da definição a seguir:

Definição 4.1 (Linguagem sensibilizadora) Sejam uma planta G_q, os alfabetos Σ_d, Σ_{ss}, e as projeções P_d e P_{ss}, como definidos nesta seção. Uma linguagem $L_{ss} \subseteq \Sigma_{ss}^*$ é dita linguagem sensibilizadora do sensor T para G_q se satisfaz as seguintes condições:

1. $P_{ss}(L(G_q)) \subseteq \overline{L_{ss}}$;
2. para quaisquer cadeias $u \in L_{ss}$ e $uv \in L_{ss}$, se $v \neq \varepsilon$, então v contém ao menos um elemento de Σ_d, ou seja, $P_d(v) \neq \varepsilon$. □

Note que, se $\overline{L_{ss}} = \Sigma_{ss}^*$, a condição 1 da definição 4.1 é automaticamente é satisfeita. Além disso, para o caso particular $\Sigma_d = \Sigma_{ss}$, a segunda condição é sempre satisfeita, pois $P_d(v) = v \neq \varepsilon$. Ainda de acordo com a segunda condição, para o caso particular em que o último evento de v pertence a Σ_d, é possível que as indicações de dessensibilização e de uma nova sensibilização sejam simultâneas.

A partir da especificação da linguagem L_{ss}, que identifica como tarefa completa cada sensibilização do sensor T, e do alfabeto Σ_d, que identifica os eventos responsáveis por sua dessensibilização, é possível construir um gerador trim G_s que represente o *modelo exato do sensor*. A linguagem gerada por G_s representa um refinamento de L_{ss} para introduzir explicitamente transições etiquetadas por um evento δ, que indica a sensibilização de T, conforme apresentado na definição a seguir:

Definição 4.2 (Modelo Exato de um Sensor) Sejam os alfabetos Σ_d, Σ_{ss}, Σ_s e as projeções P_d, P_{ss} e P_δ, como definidos nesta seção. Seja ainda $L_{ss} \subseteq \Sigma_{ss}^*$ a linguagem sensibilizadora do sensor T. O modelo exato de T, G_s, é um gerador definido em Σ_s que satisfaz as seguintes propriedades:

1. $L_m(G_s) = L(G_s) \subseteq \Sigma_s^*$.
2. $P_{ss}(L(G_s)) = \overline{L_{ss}}$.
3. Para qualquer cadeia $uv \in L(G_s)$, tal que $P_{ss}(u) \in L_{ss}$ e $P_{ss}(uv) \in L_{ss}$, se nenhum prefixo próprio t de v satisfizer $P_{ss}(ut) \in L_{ss}$, então v contém exatamente um evento δ, ou seja, $P_\delta(v) = \delta$.
4. Se, para $u \in \Sigma_s^*$, $P_{ss}(u) \in L_{ss}$ mas não existe $t \prec u$ tal que $P_{ss}(t) \in L_{ss}$, então $P_\delta(u) = \varepsilon$.
5. Para qualquer cadeia $uve_d \in \Sigma_s^*$, tal que $P_{ss}(u) \in L_{ss}$, $v \in (\Sigma_s - \Sigma_d)^*$, $e_d \in \Sigma_d$ e $P_{ss}(uve_d) \in \overline{L_{ss}}$, pode-se afirmar que $uve_d \in L(G_s)$ se e somente se $P_\delta(v) = \delta$ (note que, pela definição de L_{ss}, $P_d(v) = \emptyset$ implica que, para qualquer cadeia $t \preceq v$, $ut \notin L_{ss}$). □

De acordo com a definição 4.2, a linguagem gerada por G_s apresenta um evento δ após cada cadeia marcada por L_{ss}, como indicado no item 3. O item 4 da definição garante que δ não é introduzido antes de que uma primeira sensibilização ocorra, ou seja, de que um estado marcado de L_{ss} seja atingido. Além disso, o item 5 indica que o evento δ é introduzido antes da ocorrência de qualquer evento de Σ_d posterior à cadeia marcada por L_{ss}, mas que os eventos de $\Sigma_{ss} - \Sigma_d$ posteriores à cadeia marcada são assíncronos em relação a δ, podendo assim ocorrer antes ou depois de δ.

A partir das definições 4.1 e 4.2, pode-se propor um algoritmo para cálculo de G_s, dadas a linguagem sensibilizadora e o conjunto dos eventos dessensibizadores:

Proposição 4.1 (Algoritmo para cálculo do modelo exato para um sensor) Sejam um gerador trim $G_{ss} = \{X_{ss}, \Sigma_{ss}, f_{ss}, x_{0,ss}, X_{m,ss}\}$ que reconheça a linguagem sensibilizadora L_{ss}, e o alfabeto $\Sigma_d \subseteq \Sigma_{ss}$. Para a construção do modelo exato do sensor T, a seguinte seqüência de passos pode ser seguida:

1. Escolher um estado marcado, $x \in X_{m,ss}$.
2. Identificar, a partir deste estado, o conjunto X_a de estados x_j, $j = 1, \ldots, n$, alcançáveis por eventos de $\Sigma_{ss} - \Sigma_d$, e acrescentar a este conjunto o próprio estado x: $X_a = \{x_j \in X_{ss} | \exists u \in (\Sigma_{ss} - \Sigma_d)^*, f_{ss}(x,u)!\}$.
3. Para cada estado $x_j \in X_a$, criar um novo estado x'_j, e definir $f(x'_j, \delta) = x_j$.
4. Deslocar todas as transições que chegam no estado x para o estado x', ou seja, para todos (x_k, e_i) tais que $f_{ss}(x_k, e_i) = x$, fazer $f_{ss}(x_k, e_i) = x'$.

5. Quaisquer eventos $e_i \in \Sigma_{ss} - \Sigma_d$ definidos nos estados de X_a terão suas transições replicadas, ou seja, se $f(x_j, e_i) = x_k$ então $f(x'_j, e_i) = x'_k$, onde x'_j e x'_k são, respectivamente, os novos estados criados a partir de x_j e x_k no passo 3.

6. Desmarcar o estado x.

7. Se ainda houver estados marcados, retornar ao passo 1.

8. Marcar todos os estados.

9. Minimizar o gerador obtido. □

Como caso particular do algoritmo, se $\Sigma_{ss} = \Sigma_d$, apenas os estados marcados de G_{ss} serão duplicados no passo 3. Além disso, nenhuma transição será replicada no passo 5. Portanto, neste caso o modelo G_s corresponde a um refinamento dos estados marcados de G_{ss} para incluir explicitamente o evento δ.

Exemplo 4.2 (Modelagem de Sensores em SEDs)
Seja o problema da mesa giratória com posição extra, como descrito no exemplo 4.1. Suponha agora que um sensor de presença de peça T seja acrescentado à posição P', como ilustrado na figura 4.4.

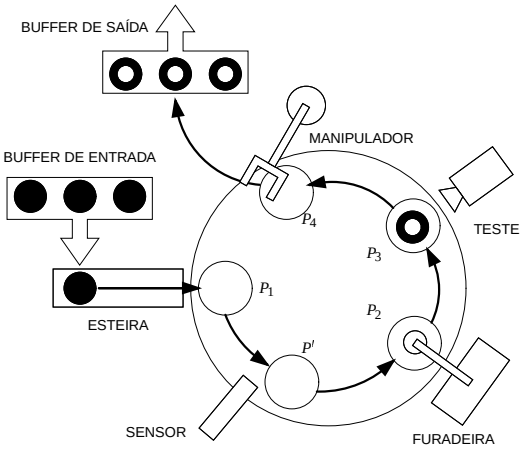

Figura 4.4: Mesa giratória com cinco posições

Para que uma peça esteja presente na posição P', é necessário que seja realizado um depósito de peça na posição P_1, e que em seguida a peça seja deslocada em um passo da mesa. No entanto, suponha que o sensor possa detectar a presença de peça sendo transferida para P' em algum momento após a inicialização do giro da mesa - possivelmente antes da conclusão do giro - em função, por exemplo, de um alcance variável do sensor.

Para a construção da linguagem sensibilizadora a partir dos eventos já existentes na modelagem da planta, considere os eventos β_1 e α_0, que indicam, respectivamente, o depósito de uma peça em

P_1 e o início de um passo da mesa. Para $\Sigma_{ss} = \{\alpha_0, \beta_1\}$, a linguagem sensibilizadora L_{ss} para T é reconhecida pelo gerador G_{ss}, apresentado na figura 4.5.

Note que, na modelagem dos elementos da mesa giratória no exemplo 2.7, os eventos β_1 e α_0 estão contidos, respectivamente, nas plantas G_1 e G_0. Seja então $G_q = G_0 \| G_1$. A linguagem gerada por G_{ss} marca cadeias $\beta_1\alpha_0$, sem no entanto desabilitar a ocorrência dos eventos - de fato, G_{ss} é um autômato, ou seja, $\overline{L_{ss}} = \Sigma_{ss}^*$. Assim, claramente $P_{ss}(L(G_q)) \subseteq \overline{L_{ss}}$.

Assume-se que o sensor T deixa de estar sensibilizado quando um novo passo da mesa é iniciado, o que significa que o término de operação da esteira não influi na dessensibilização de T. Portanto, $\Sigma_d = \{\alpha_0\}$. Neste caso particular, a ocorrência de α_0 após uma sensibilização indica, simultaneamente, a dessensibilização e uma nova sensibilização, o que no entanto não transgride a definição 4.1.

Para $\Sigma_{ss} = \{\alpha_0, \beta_1\}$ e $\Sigma_d = \{\alpha_0\}$, o algoritmo para obtenção do modelo exato do sensor fornece o gerador da figura 4.6.

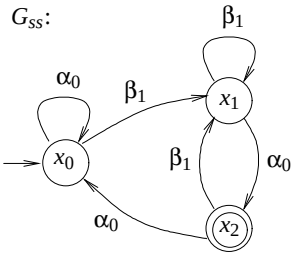

Figura 4.5: Linguagem sensibilizadora de um sensor em P', para $\Sigma_{ss} = \{\alpha_0, \beta_1\}$.

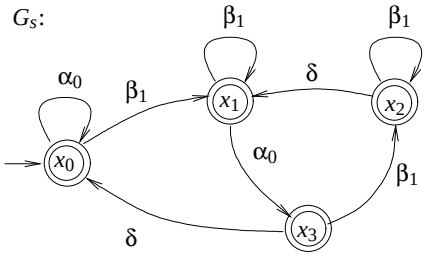

Figura 4.6: Modelo exato do sensor para um único evento dessensibilizador ($\Sigma_d = \{\alpha_0\}$).

Considere agora que o sensor T é apenas ativado quando a mesa completa um passo, transferindo uma peça da posição P_1 para P'. Neste caso, pode-se incluir no alfabeto Σ_{ss} explicitamente o evento β_0, que indica o término de um passo da mesa. Assim, é possível também modelar o comportamento do mesmo sensor com $\Sigma_{ss} = \{\alpha_0, \beta_0, \beta_1\}$. A linguagem sensibilizadora, neste caso, é aquela gerada por $G_{ss,2}$, apresentado na figura 4.7. Considerando novamente $\Sigma_d = \{\alpha_0\}$, o modelo exato do sensor corresponde ao gerador $G_{s,2}$, apresentado na figura 4.8. □

4.3 Modelagem da Linguagem Sensibilizadora por Evento Auxiliar

Em vez de inferir sobre a sensibilização de T a partir da marcação associada ao gerador da linguagem sensibilizadora L_{ss}, pode-se usar um evento auxiliar $\tau \notin \Sigma_{q\delta}$ para indicar o momento a partir do qual T pode ser sensibilizado. Para tanto, sejam os alfabetos $\Sigma_{ss\tau} = \Sigma_{ss} \dot{\cup} \{\tau\}$, $\Sigma_{s\tau} = \Sigma_s \dot{\cup} \{\tau\}$, $\Sigma_{q\tau} = \Sigma_q \dot{\cup} \{\tau\}$, $\Sigma_{q\tau\delta} = \Sigma_q \dot{\cup} \{\tau\} \dot{\cup} \{\delta\}$, $\Sigma_{d\tau\delta} = \Sigma_d \dot{\cup} \{\tau\} \dot{\cup} \{\delta\}$, e as projeções $P_{ss} : \Sigma_{q\tau\delta} \to \Sigma_{ss}$, $P_d : \Sigma_{q\tau\delta} \to \Sigma_d$ e $P_s = \Sigma_{q\tau\delta} \to \Sigma_s$.

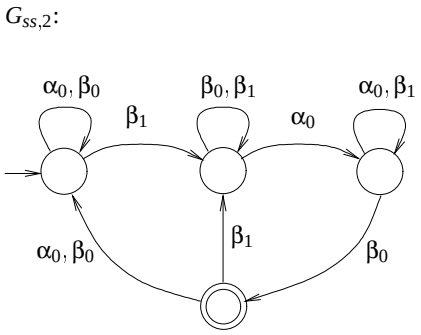

Figura 4.7: Linguagem sensibilizadora do sensor em P', para $\Sigma_{ss} = \{\alpha_0, \beta_0, \beta_1\}$.

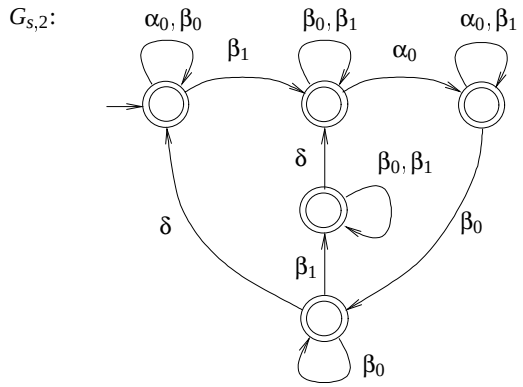

Figura 4.8: Modelo exato do sensor para a linguagem sensibilizadora $L(G_{ss,2})$.

Definição 4.3 (Linguagem sensibilizadora genérica) Sejam uma planta G_q e os alfabetos Σ_d e Σ_{ss}, como definidos na seção anterior, bem como o alfabeto $\Sigma_{ss\tau}$ e as projeções P_{ss} e P_d, como definidos nesta seção. Uma linguagem $L_{ss\tau} \subseteq \Sigma_{ss\tau}^*$ é dita *linguagem sensibilizadora genérica* do sensor T para G_q se satisfaz as seguintes condições:

1. $L_{ss\tau}$ é prefixo-fechada.
2. $P_{ss}(L(G_q)) \subseteq P_{ss}(L_{ss\tau})$.
3. Para qualquer cadeia $u\tau v\tau \in L_{ss\tau}$, v contém ao menos um elemento de Σ_d, ou seja, $P_d(v) \neq \varepsilon$. □

A especificação da linguagem sensibilizadora na forma genérica permite o cálculo do modelo exato do sensor de uma forma mais direta, apenas com o uso das operações de produto síncrono e projeção de linguagens. Seja, para tanto, a *linguagem auxiliar* $L_{aux} \subseteq \Sigma_{d\tau\delta}$, reconhecida pelo gerador da figura 4.9. Agora, o modelo exato do sensor pode ser calculado por $L(G_s) = P_s(L_{ss\tau} \| L_{aux})$.

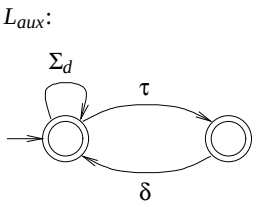

Figura 4.9: Linguagem que introduz evento δ na linguagem sensibilizadora genérica.

A composição de $L_{ss\tau}$ com L_{aux} expressa o fato de que os eventos de $\Sigma_{ss} - \Sigma_d$ posteriores a τ são assíncronos em relação a δ. Além disso, se $\Sigma_{ss} = \Sigma_d$, $L(G_s)$ corresponde à substituição de τ por δ em $L_{ss\tau}$.

Pode-se mostrar que qualquer linguagem sensibilizadora definida em Σ_{ss} pode ser convertida para a forma genérica, bastando para tanto que cada estado marcado seja refinado pelo acréscimo de uma

transição etiquetada por τ. A utilização da linguagem sensibilizadora genérica é vantajosa para a demonstração de alguns resultados apresentados no próximo capítulo.

Exemplo 4.3 (Modelagem de Sensores em SEDs com Linguagem Sensibilizadora Genérica) Para o posicionamento do sensor descrito no exemplo 4.2, considerando novamente $\Sigma_{ss} = \{\alpha_0, \beta_1\}$ e $\Sigma_d = \{\alpha_0\}$, a linguagem sensibilizadora genérica é reconhecida pelo gerador $G_{ss\tau}$ (figura 4.10), enquanto a linguagem L_{aux} é a linguagem reconhecida pelo gerador da figura 4.11.

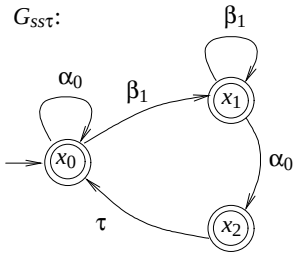

Figura 4.10: Linguagem sensibilizadora genérica do sensor em P', para $\Sigma_{ss} = \{\alpha_0, \beta_1\}$.

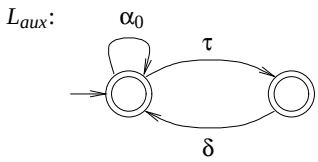

Figura 4.11: Linguagem auxiliar para $\Sigma_d = \{\alpha_0\}$.

Para $G_q = G_0 \| G_1$, pode-se verificar que $P_{ss}(L(G_q)) \subseteq P_{ss}(L(G_{ss\tau})) = \Sigma_{ss}^*$. Além disso, τ só está definido no estado x_2 de $G_{ss\tau}$, que só é acessível a partir de x_1, pela transição etiquetada por $\alpha_0 \in \Sigma_d$. Assim, $L(G_{ss\tau})$ satisfaz as condições da definição 4.3.

É possível verificar que a linguagem $P_s(L(G_{ss\tau}) \| L_{aux})$, obtida neste caso, é equivalente àquela gerada por G_s obtido no exemplo 4.2. Assim, o modelo para o sensor, obtido aqui, é equivalente ao modelo apresentado na figura 4.6. □

4.4 O Problema de Controle Supervisório com Introdução de Sensores

Seja um problema de controle supervisório, enunciado para uma planta G_q definida sobre Σ_q e uma especificação genérica E definida sobre $\Sigma_e \subseteq \Sigma_q$. Como apresentado no capítulo 2, para a especificação global $K_q = E \| L_m(G_q)$, a solução do PCS é dada pela linguagem $supC(K_q, G_q)$.

Pode-se alterar este problema pela inclusão de um sensor T, conforme descrito na seção 4.2. Assume-se que o evento δ, que representa a sensibilização de T, é de natureza não controlável. Assim, dado o particionamento dos eventos de Σ_q em eventos controláveis $\Sigma_{c,q}$ e não controláveis $\Sigma_{u,q}$, o alfabeto $\Sigma_{q\delta}$ pode ser particionado em $\Sigma_{u,q\delta} = \Sigma_{u,q} \cup \{\delta\}$ e $\Sigma_{c,q\delta} = \Sigma_{c,q}$.

Com a modelagem do comportamento do sensor pelo gerador G_s, o PCS pode agora ser reformulado em termos da planta livre $G_{qs} = G_q \| G_s$ e da especificação global $K_{qs} = E \| L_m(G_{qs})$, definidas em $\Sigma_{q\delta}$. Este problema é denominado *Problema de Controle Supervisório com Introdução de Sensor.* A relação entre a solução para este problema, dada por $supC(K_{qs}, G_{qs})$, e a solução do PCS original é determinada pelas proposições que seguem.

Proposição 4.2 *Sejam G_q e G_s plantas definidas, respectivamente, em Σ_q e Σ_s, tais que G_s é o modelo exato de um sensor T para G_q. Para o evento δ do sensor, sejam também o alfabeto $\Sigma_\delta = \{\delta\}$ e a planta G_δ, tais que $L_m(G_\delta) = \Sigma_\delta^*$. Então $G_q \|_{as} G_\delta$ é uma aproximação externa em Σ_δ para $G_q \| G_s$.*

Prova: Seja a planta $G_{s,ap}$, definida sobre Σ_s, tal que $L_m(G_{s,ap}) = \Sigma_s^*$. Então, $G_q \|_{as} G_\delta = G_q \| G_{s,ap}$.

Seja novamente o alfabeto $\Sigma_{qs} = \Sigma_q \dot{\cup} \Sigma_\delta$. Note que G_q está definida em $\Sigma_{qs} - \Sigma_\delta$, e que $G_{s,ap}$ é uma aproximação em Σ_δ de G_s, definida sobre um alfabeto Σ_s tal que $\Sigma_\delta \subseteq \Sigma_s \subseteq \Sigma_{qs}$. Assim, pela proposição 3.5, $G_q \| G_{s,ap}$ é uma aproximação de $G_q \| G_s$. □

Como conseqüência das proposições 3.6, 3.17 e 4.2, o seguinte resultado pode ser enunciado:

Proposição 4.3 *Sejam K_q, G_q, G_s, K_{qs} e G_{qs}, como definidos neste capítulo. Então $supC(K_{qs}, G_{qs}) = supC(K_q, G_q) \| L_m(G_s)$. Seja ainda $P_{q-\delta} : \Sigma_{q\delta} \to \Sigma_q$. Então $P_{q-\delta}(supC(K_{qs}, G_{qs})) = supC(K_q, G_q)$.*

A proposição 4.3 indica que o problema de controle supervisório original não é alterado pela introdução de um sensor na planta original. Assim, os problemas de controle supervisório original e com introdução de sensores são ditos equivalentes.

O objetivo da introdução de um sensor é permitir que a especificação genérica E possa ser re-expressa por um conjunto de especificações genéricas $E_{si} \subseteq \Sigma_{si}^*$, com $\Sigma_{si} \subseteq \Sigma_e \cup \{\delta\}$, $i = 1, \ldots, k$, de modo que $K_{qs} = E \| L_m(G_s) \| L_m(G_q) = (\|_{i=1}^k E_{si}) \| L_m(G_s) \| L_m(G_q)$. Embora esta decomposição da especificação não seja feita, neste trabalho, de forma sistemática, em muitos casos a introdução do sensor permite que as especificações E_{si} sejam projetadas a partir da compreensão das restrições impostas por E. Em particular, pode-se desejar que $(\|_{i=1}^k E_{si}) \| L_m(G_s) = E \| L_m(G_s)$.

Com a decomposição da especificação, a abordagem modular pode ser explorada para cada especificação $K_{Si} = E_{si} \| L_m(G_s) \| L_m(G_q)$, caso a planta $L_m(G_q)$ corresponda à planta global do PCS original.

4.4.1 Introdução de Sensores e a Representação por Sistemas Produto

Suponha agora uma planta em sua representação por sistemas produto RSP_0, para a qual são dadas especificações E_a, E_b e E_c, como ilustrado na figura 4.12.

Acrescenta-se à planta um sensor, cujo modelo exato é representado pelo gerador G_s, que contém eventos em comum com as subplantas G_j a G_n. Neste caso, o sensor para a planta $G_q = \|_{i=j}^n G_i$ influi no refinamento da RSP original, que deve ser alterada para uma representação onde seja considerado o sincronismo entre as subplantas e G_s. A nova representação, denominada RSP_1, é ilustrada na figura 4.13.

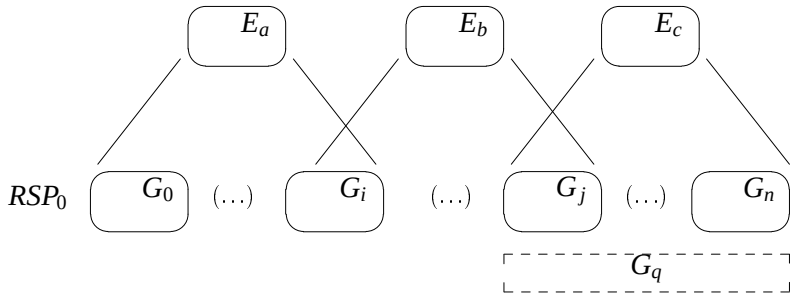

Figura 4.12: RSP e especificações genéricas para um problema de controle supervisório sem sensor.

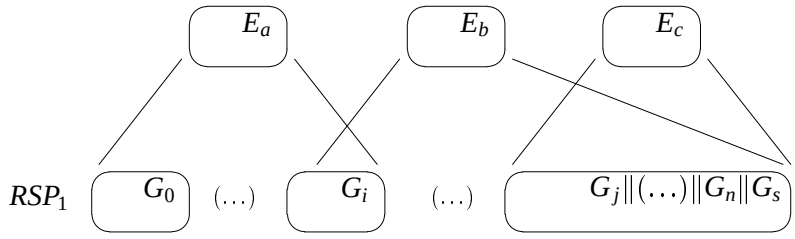

Figura 4.13: RSP e especificações genéricas para o mesmo problema, com inclusão de sensor.

Note que, quando o modelo do sensor é acrescentado à planta, para algumas especificações a planta local passa a ser significativamente maior e abranger um maior número de eventos, como é o caso da planta local para a especificação E_b na figura 4.13.

A introdução do modelo do sensor tem por objetivo permitir que especificações que compartilham eventos com G_s sejam reescritas como novas especificações com menor número de estados, como ilustrado na figura 4.14.

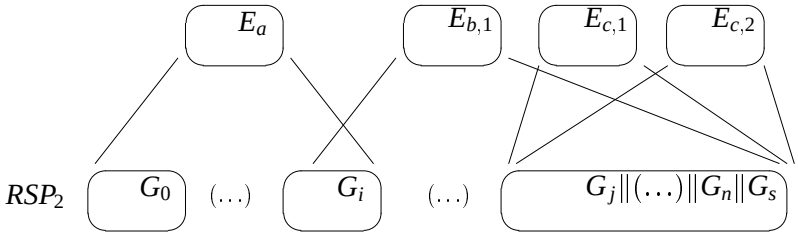

Figura 4.14: RSP para o problema com inclusão de sensor, com decomposição das especificações.

No entanto, como o ganho em complexidade computacional introduzido pela abordagem modular local depende do refinamento da planta local, a introdução do modelo exato do sensor torna a solução mais custosa, como é ilustrado no exemplo 4.4.

Exemplo 4.4 (Mesa Giratória com Posição Intermediária e Sensor)
Para o problema da mesa giratória com posição extra e sensor, descrito inicialmente no exemplo 4.2, a introdução do modelo do sensor G_s apresentado na figura 4.6 exige que a representação por sistemas produto da planta seja reformulada, pois G_s possui eventos em comum com as plantas G_0 e G_1. Para

que seja possível a utilização da abordagem modular local, a planta livre é representada pelo seguinte conjunto de sistemas assíncronos: $G_{SP1} = G_0 \| G_1 \| G_s$; $G_{SP2} = G_2$; $G_{SP3} = G_3$; $G_{SP4} = G_4$.

Com a introdução de T, as especificações de comportamento das figuras Ra' e Rc'_1 (figuras 4.3 e 4.2) podem ser reexpressas pela utilização do evento δ.

A especificação Ra_s, equivalente a Ra', habilita a execução de giro da mesa apenas quando uma operação foi concluída nos estágios P_1, P_2 ou P_3 ou quando há peça presente na posição P' (figura 4.15). Por sua vez, a restrição imposta pela especificação Rc'_1 pode ser expressa por duas especificações $Rc_{0,s}$ e $Rc_{1,s}$ (figuras 4.16 e 4.17).

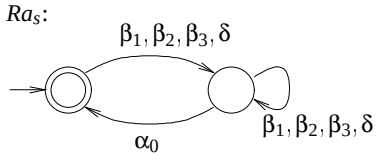

Figura 4.15: Restrição Ra_s.

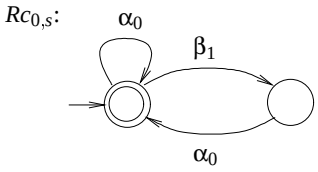

Figura 4.16: Restrição $Rc_{0,s}$.

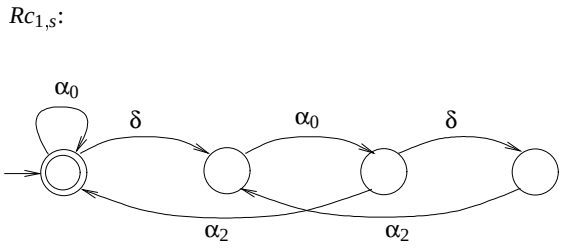

Figura 4.17: Restrição $Rc_{1,s}$.

As demais especificações genéricas apresentadas são mantidas como no exemplo 2.7. Assim, $Rb_{i,s} = Rb_i$, $i = 1,2,3,4$ (figura 4.18) e $Rc_{i,s} = Rc_i$, $i = 2,3$ (figura 4.19).

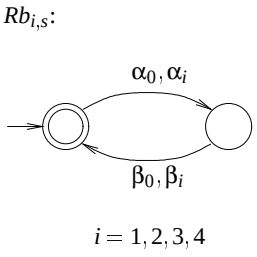

Figura 4.18: Restrições $Rb_{i,s}$.

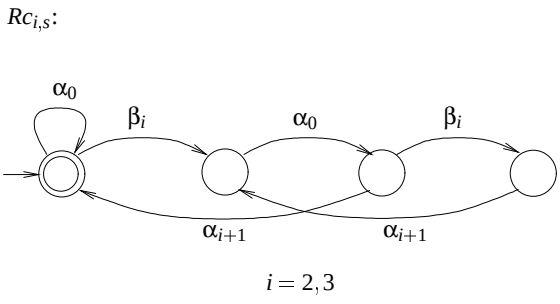

Figura 4.19: Restrições $Rc_{i,s}$, i=2,3.

Neste caso, as plantas locais para o conjunto de especificações $\{Ra_s, Rb_{1,s}, Rb_{2,s}, Rb_{3,s}, Rb_{4,s}, Rc_{0,s}, Rc_{1,s}, Rc_{2,s}, Rc_{3,s}\}$ são: $G_{A,s} = G_{SP1} \| G_{SP2} \| G_{SP3}$; $G_{B1,s} = G_{SP1}$; $G_{Bi,s} = G_{SP1} \| G_{SPi}$, para $i = 2,3,4$; $G_{C0,s} = G_{SP1}$; $G_{C1,s} = G_{SP1} \| G_{SP2}$; e $G_{Ci,s} = G_{SP1} \| G_{SPi} \| G_{SPi+1}$, para $i = 2,3$.

A resolução do problema de controle modular local com as especificações dadas também gera uma

solução modular local. Ainda, é possível mostrar que a solução obtida é equivalente àquela obtida no exemplo 4.1 - para tanto, basta verificar que o comportamento global obtido, quanto projetado no alfabeto sem δ, é equivalente ao comportamento obtido naquele exemplo. A tabela 4.4 apresenta as propriedades dos geradores associados à resolução do problema de controle supervisório com sensor, apresentado neste exemplo.

Resolução Modular Local	E_x	G_X	E_X	S_X	RS_X
a_s	2	64	96	$(96,400)$	$(2,9)$
$b_{1,s}$	2	16	10	$(10,17)$	$(2,4)$
$b_{i,s}, i = 2,3,4$	2	32	24	$(24,64)$	$(2,4)$
$c_{0,s}$	2	16	16	$(12,23)$	$(2,4)$
$c_{1,s}$	4	32	96	$(96,264)$	$(4,6)$
$c_{i,s}, i = 2,3$	4	64	256	$(192,680)$	$(4,8)$
Total (somatório)				$(670,2256)$	$(24,51)$
Resolução Monolítica	R	G	K	S	
	400	128	599	$(455,1198)$	

Tabela 4.2: Número de estados dos geradores para o problema da mesa com sensor (exemplo 4.4). Para as soluções finais, é apresentado o par (número de estados, número de transições).

Pode-se observar que, embora os supervisores locais obtidos apresentem um número expressivamente maior de estados do que a solução original apresentada no exemplo 4.1, a solução obtida com minimização dos supervisores possui menor número de estados do que a solução minimizada obtida na resolução do PCS original, em função de supervisores minimizados com menor número de estados obtidos para as restrições Ra_s, $Rc_{0,s}$ e $Rc_{1,s}$. No entanto, o processo de minimização, neste caso, é mais custoso, em função do maior número de estados dos supervisores locais. □

4.5 Alocação de Sensores

Baseado na modelagem de sensores pela definição de sua linguagem sensibilizadora, como introduzido neste capítulo, é possível investigar a dependência entre a alocação de sensores e a complexidade computacional da resolução do problema de controle supervisório com sensores. A posição de alocação é representada diretamente na linguagem sensibilizadora e, portanto, no modelo no sensor. Embora a alocação ótima de sensores não seja estudada no presente trabalho, no exemplo a seguir fica clara a relevância da alocação.

Exemplo 4.5 (*Buffer* com capacidade oito)
Seja uma linha de transferência composta por duas máquinas M_1 e M_2, similares, e um *buffer* intermediário de capacidade $N = 8$ (figura 4.20).

Para cada uma das máquinas, é possível identificar os sinais de início e fim de operação por

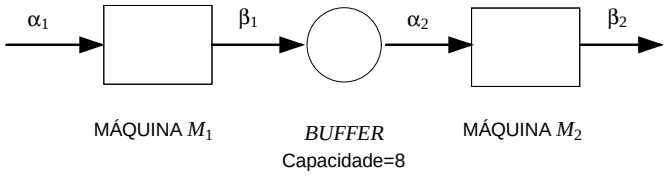

Figura 4.20: Planta com duas máquinas e *buffer* de capacidade oito.

eventos α_i e β_i, $i = 1,2$, sendo que apenas os eventos de início de operação α_i são controláveis. Assim, os comportamentos livres das máquinas M_1 e M_2 podem ser modelados, respectivamente, pelos geradores G_i, definidos sobre $\Sigma_i = \{\alpha_i, \beta_i\}$, $i = 1,2$ (figura 4.21).

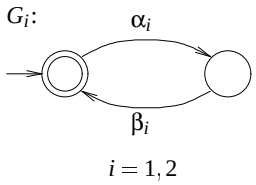

Figura 4.21: Comportamento livre do sistema.

Deseja-se projetar um controlador que evite tanto *underflow* quanto *overflow* do *buffer*. A especificação genérica deste comportamento pode ser expressa por uma linguagem E, definida sobre $\Sigma_e = \{\alpha_2, \beta_1\}$, e reconhecida pelo gerador apresentado na figura 4.22. Note que o número de estados deste gerador cresce linearmente com a capacidade do *buffer*.

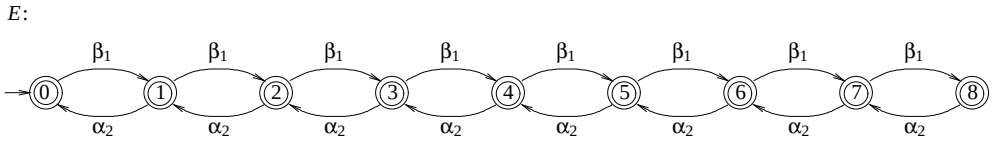

Figura 4.22: Restrição de funcionamento do sistema *buffer*, para evitar *underflow* quanto *overflow* do *buffer*.

Ainda, é importante destacar que a especificação E poderia ser escrita como duas especificações que evitassem separadamente *underflow* e *overflow* do *buffer*, que seriam, no entanto, especificações com mesmo número de estados de E.

Suponha agora que dois sensores de presença de peça possam ser alocados no *buffer*. Assim, para identificar que um determinado número de peças no *buffer* é ultrapassado em virtude do depósito de uma peça por M_1, é alocado o sensor T_1, cuja sensibilização é identificada pelo sinal δ_1. De maneira análoga, o sinal δ_2, proveniente do sensor T_2, identifica quando a retirada de peça no *buffer* por M_2 provoca o atingimento de um dado número de peças.

Neste caso, ao comportamento livre do sistema deve ser agregado o comportamento dos sensores. Para tanto, devem ser especificadas as linguagens que reconhecem a sensibilização de T_1 e T_2. Sejam

os alfabetos $\Sigma_{ss,1} = \Sigma_{ss,2} = \Sigma_e$, que definem linguagens sensibilizadoras $L_{ss,i} \subseteq \Sigma^*_{ss,i}$, $i = 1,2$, para os sensores T_1 e T_2.

As figuras 4.23, 4.24 e 4.25 apresentam geradores que reconhecem linguagens sensibilizadoras para três possíveis escolhas de posicionamento dos sensores:

- Caso A: T_1 indica a transição de 3 para 4 peças no *buffer*, enquanto T_2 indica a transição de 5 para 4 peças.
- Caso B: sensores T_1 e T_2 posicionados de forma a indicar, respectivamente, as transições de 6 para 7 peças e de 2 peças para 1 peça.
- Caso C: sensores T_1 e T_2 posicionados de forma a indicar, respectivamente, as transições de 7 para 8 peças e de 1 peça para nenhuma peça.

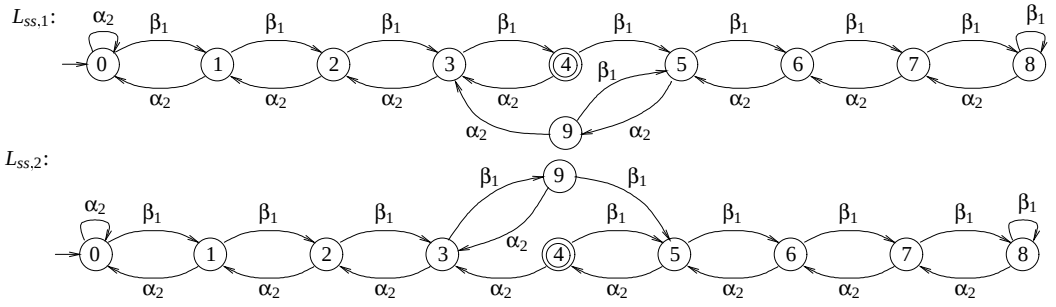

Figura 4.23: Linguagens sensibilizadoras para os sensores T_1 e T_2, conforme o caso A.

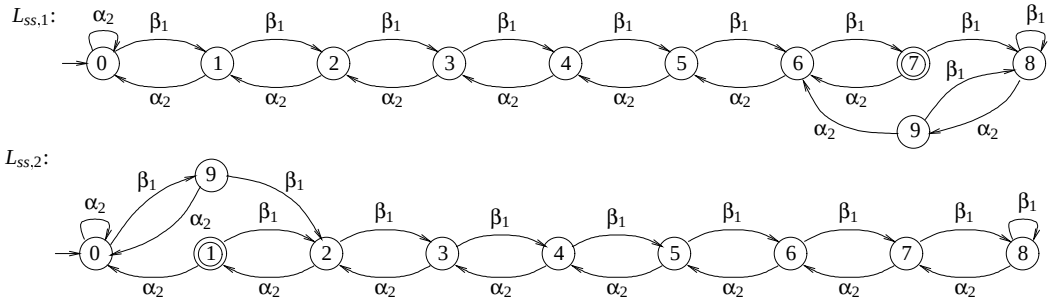

Figura 4.24: Linguagens sensibilizadoras para os sensores T_1 e T_2, conforme o caso B.

É importante destacar que as linguagens sensibilizadoras não devem restringir o comportamento original do sistema, o que é expresso pelas condições $P_{ss,i}(L(G_q)) \subseteq \overline{L_{ss,i}}$, $i = 1,2$, para os casos A, B e C. No exemplo em questão, isto significa que as linguagens sensibilizadoras não devem introduzir automaticamente as restrições de *underflow* e *overflow*, já que estas não são garantidas pela configuração física do sistema, devendo ser impostas pela supervisão. Neste sentido, a ocorrência dos eventos α_2 e β_1 é habilitada (como *self-loop*), respectivamente, nos estados 0 e 8 dos geradores apresentados, nos casos A e B, bem como nos estados 0 e 9 para $L_{ss,1}$ e 9 e 8 para $L_{ss,2}$, no caso C.

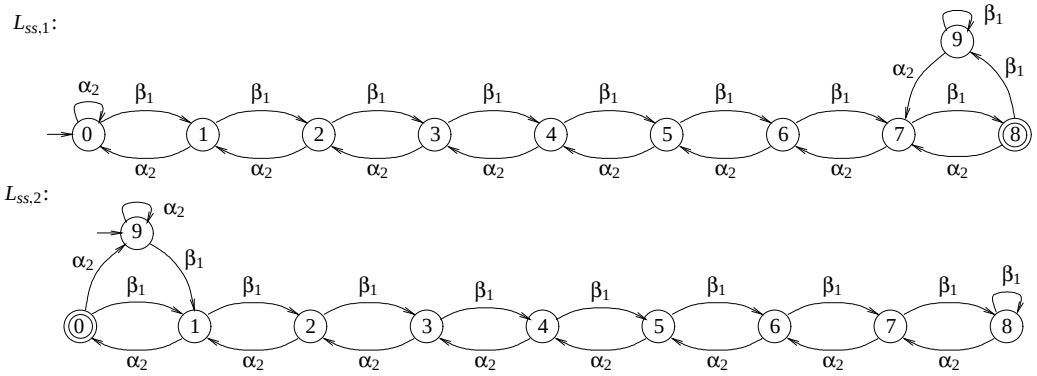

Figura 4.25: Linguagens sensibilizadoras para os sensores T_1 e T_2, conforme o caso C.

Considerando que qualquer mudança no *buffer* posterior à sensibilização de um sensor o dessensibiliza, $\Sigma_d = \{\alpha_2, \beta_1\}$. Note que esta hipótese não corresponde, necessariamente, ao comportamento de um sensor real, mas é adequada para indicar que o evento do sensor ocorre imediatamente após o atingimento do estado marcardo da linguagem sensibilizadora.

Assim, são calculados os modelos exatos dos sensores G_{si}, definidos sobre $\Sigma_{si} = \{\alpha_2, \beta_1, \delta_i\}$, $i = 1, 2$, para cada um dos casos enunciados. As figuras 4.26, 4.27 e 4.28 apresentam os modelos exatos dos sensores, G_{s1} e G_{s2}, para as três escolhas de posicionamento.

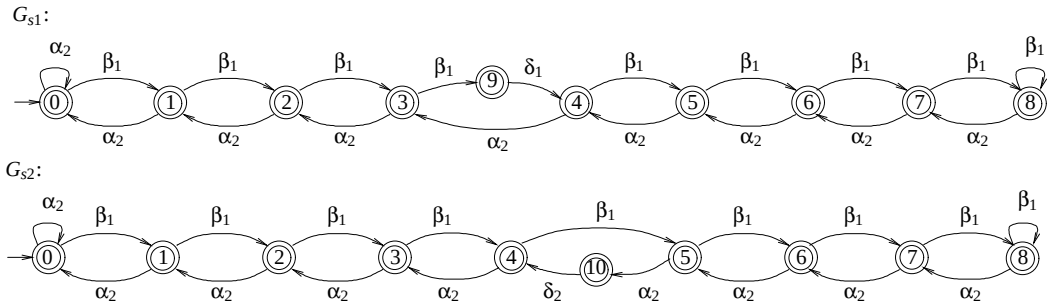

Figura 4.26: Modelos exatos dos sensores T_1 e T_2 em posições intermediárias do *buffer* (caso A).

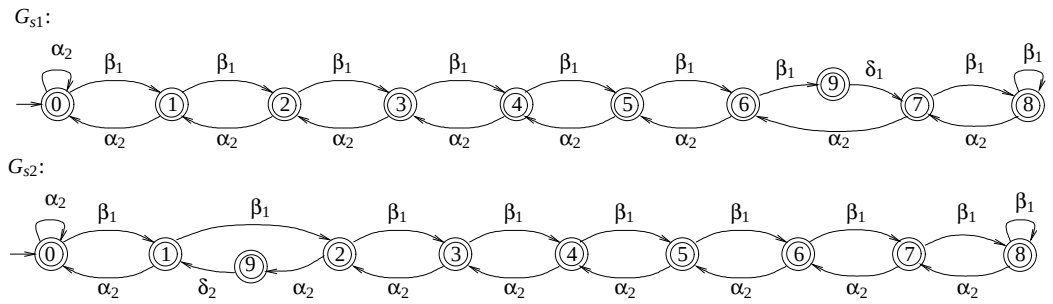

Figura 4.27: Modelos exatos dos sensores T_1 e T_2 próximos às extremidades do *buffer* (caso B).

Como ambos os alfabetos dos modelos exatos para T_1 e T_2 possuem eventos em comum com as plantas G_1 e G_2, a representação por sistemas produto mínima é a composição $G = G_{s1} \| G_{s2} \| G_1 \| G_2$,

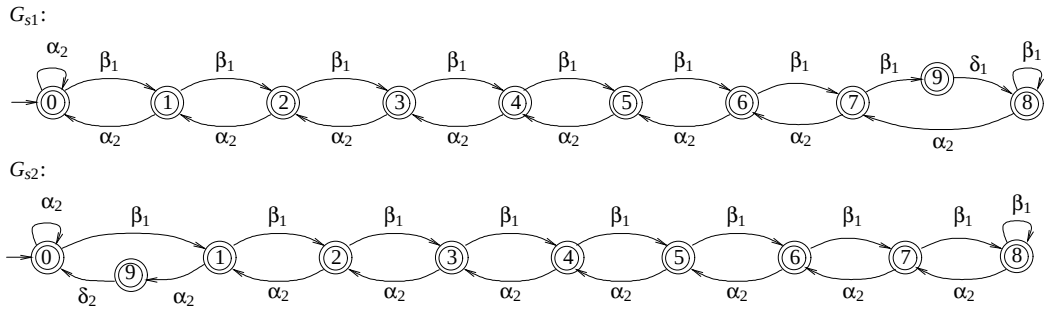

Figura 4.28: Modelos exatos dos sensores T_1 e T_2 nas extremidades do *buffer* (caso C).

o que significa que, neste problema, as abordagens modular e modular local são equivalentes.

Com a introdução de sensores na planta, a especificação E pode ser reescrita como duas especificações modulares genéricas $E_{s1} \subseteq \Sigma_{s1}^*$ e $E_{s2} \subseteq \Sigma_{s2}^*$, para evitar, respectivamente, a ocorrência de *overflbw* e *underflbw* no *buffer*. As figuras 4.29, 4.30 e 4.31 apresentam as especificações E_{s1} e E_{s2} para os casos A, B e C, respectivamente.

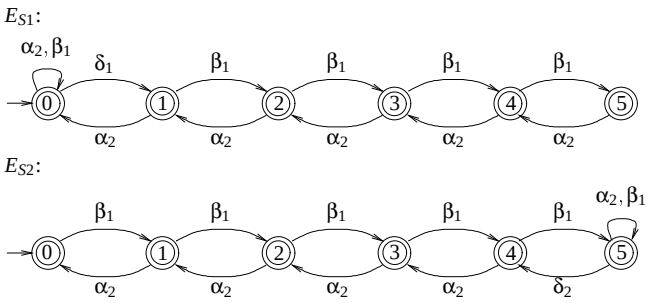

Figura 4.29: Especificações de comportamento para o caso A.

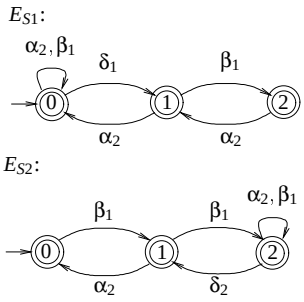

Figura 4.30: Especificações de comportamento para o caso B.

Figura 4.31: Especificações de comportamento para o caso C.

O cálculo dos supervisores modulares para as especificações E_{s1} e E_{s2} fornece os supervisores S_1 e S_2, com as propriedades apresentadas na tabela 4.3 para os três casos apresentados. Em todos os casos, os supervisores S_1 e S_2 são modulares. Além disso, sua aplicação é equivalente à aplicação do supervisor monolítico S calculado a partir da especificação original E, ou seja, $S_1 \| S_2 = S \| L_m(G_s)$.

Pode-se observar que, para este problema, a posição dos sensores no *buffer* influencia diretamente o número de estados das especificações e, portanto, a complexidade computacional do cálculo dos supervisores correspondentes, como observado na tabela 4.3. No entanto, novamente, a utilização de

sensores associados à modelagem exata de seu comportamento torna mais dispendioso o cálculo da máxima linguagem controlável, além de não fornecer supervisores com menor número de estados. □

Problema	$O(\cdot)$	Supervisor	Estados	Transições
Planta sem sensores	$4^2 * 9^2$	S	34	64
Planta com sensores, caso A	$88^2 * 6^2$	S_1	42	82
		S_2	44	86
Planta com sensores, caso B	$88^2 * 3^2$	S_1	42	82
		S_2	44	86
Planta com sensores, caso C	$88^2 * 2^2$	S_1	40	77
		S_2	44	86

Tabela 4.3: Complexidade computacional (para o cálculo de cada supervisor modular) e propriedades dos supervisores para os casos A, B e C.

4.6 Aproximação para Sensores

Na seção anterior, um problema de controle supervisório foi resolvido para diferentes posições de alocação de sensores. No entanto, a utilização dos modelos exatos de comportamento dos sensores não introduziu ganho, em termos de redução do custo computacional da resolução do problema, ainda que tenha possibilitado a utilização de especificações genéricas menores para síntese dos supervisores pela abordagem modular. Nesta seção, mostra-se que este ganho pode ser obtido quando se utiliza uma aproximação externa do comportamento do sensor.

Para $K_{qs} = (\|_{i=1}^{k} E_{si}) \| L_m(G_q) \| L_m(G_s)$, como definido na seção 4.4, seja agora a aproximação $K_{qs,ap} = (\|_{i=1}^{k} E_{si}) \| L_m(G_q) \| L_m(G_\delta)$. É possível mostrar que $K_{qs,ap}$ é uma aproximação em $\Sigma_\delta = \{\delta\}$ para K_{qs}. Pretende-se agora avaliar a relação existente entre a solução do problema de controle enunciado para K_{qs} e $G_q \| G_s$, e a solução do *Problema de Controle Supervisório com Aproximação de Sensor,* representado pela especificação $K_{qs,ap}$ e pela planta $G_q \| G_\delta$.

Considere novamente a representação por sistemas produto RSP_0 de uma planta sem sensores, ilustrada na figura 4.12. Agora, em vez de acrescentar o modelo exato do sensor à RSP_0, é acrescentada apenas a planta local G_δ, como definida anteriormente, onde δ é o evento que representa a ativação do sensor. Na RSP obtida, apresentada na figura 4.32, a planta é a aproximação máxima para δ da planta representada na RSP_1. Como visto na seção 4.4, estas duas representações são equivalentes, quando as especificações não utilizam o sensor, ou seja, não são decompostas a partir da introdução do sensor.

Entretanto, com as especificações apresentadas na figura 4.14, não há garantia de que se possa utilizar a aproximação do sensor. Em diversos casos, porém, a arquitetura apresentada na figura 4.33 é equivalente às anteriores, como é possível observar nos dois exemplos apresentados a seguir.

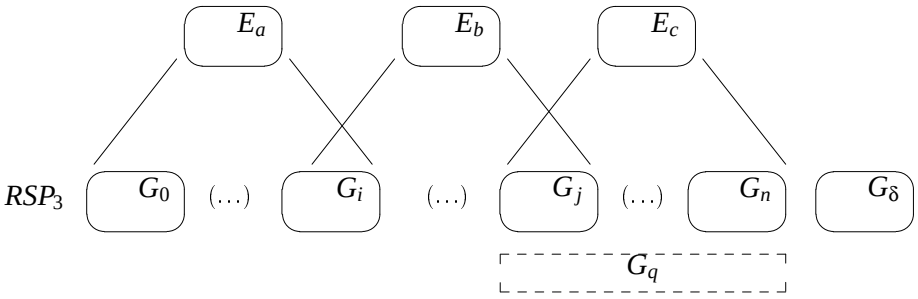

Figura 4.32: RSP para o problema com inclusão de sensor, com aproximação máxima da planta.

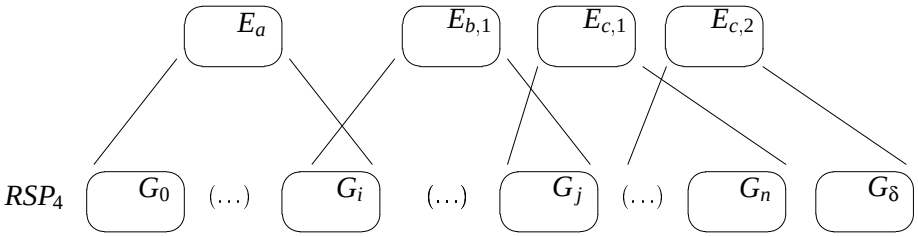

Figura 4.33: RSP para o problema com inclusão de sensor aproximado e decomposição das especificações.

Exemplo 4.6 (*Buffer* com capacidade oito, aproximação de sensores)
Considere novamente o problema do *buffer* de capacidade 8 apresentado no exemplo 4.5, com os sensores alocados conforme descrito nos casos A, B e C. Agora, em vez de modelar o comportamento exato dos sensores, cada um dos sensores pode ser modelado por uma aproximação externa dada por geradores $G_{si,ap}$, $i = 1,2$, com um único estado, onde os eventos β_1, α_2 e δ_i são definidos em *self-loop*, como ilustrado na figura 4.34.

O conjunto de plantas $\{G_1, G_2, G_{s1,ap}, G_{s2,ap}\}$ possui uma representação por sistema produto implícita, mais refinada do que aquela obtida com o comportamento exato dos sensores. Esta RSP é dada pelo conjunto de geradores $\{G_1, G_2, G_{\delta 1}, G_{\delta 2}\}$, ilustrados na figura 4.35. Note que o alfabeto de $G_{\delta i}$ é diferente daquele de $G_{si,ap}$, $i = 1,2$, apenas porque foram omitidos os eventos β_1 e α_2. Pode-se verificar ainda que o modelo composto $G_{ap} = G_1 \| G_2 \| G_{s1,ap} \| G_{s2,ap}$ (figura 4.36) é uma aproximação externa de $G = G_1 \| G_2 \| G_{s1} \| G_{s2}$. Além disso, G_{ap} é idêntico para os casos A, B e C.

Esta aproximação do comportamento dos sensores exige que as especificações sejam reescritas, para que as funções de transição das especificações sejam totais com relação aos eventos não controláveis δ_1 e δ_2. Assim, para o caso A, ao gerador E_{s1} são incluídas transições $f(x_i, \delta_1) = x_1$, $\forall x_i \in X_{Es1}$, enquanto ao gerador E_{s2} são incluídas transições $f(x_i, \delta_2) = x_4$, $\forall x_i \in X_{Es2}$. Para o caso B, ao gerador E_{s1} são incluídas transições $f(x_i, \delta_1) = x_1$, $\forall x_i \in X_{Es1}$, enquanto ao gerador E_{s2} são incluídas transições $f(x_i, \delta_2) = x_1$, $\forall x_i \in X_{Es2}$. Estas modificações dão origem às especificações $E_{s1,ap}$ e $E_{s2,ap}$, apresentadas na figura 4.37 para o caso B. No caso C, o evento δ_1 é definido em *self-loop* no estado x_1 de E_{s1}, enquanto δ_2 é definido em *self-loop* no estado x_0 de E_{s2}, dando origem às especificações

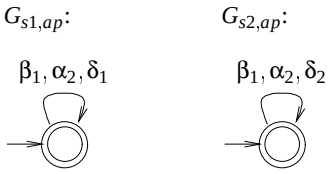

Figura 4.34: Modelos aproximados para os sensores.

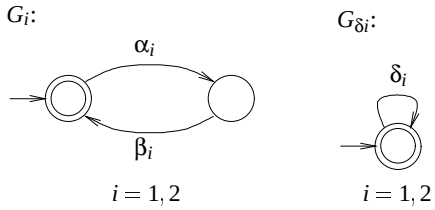

Figura 4.35: Representação por sistema produto com modelo aproximado para os sensores.

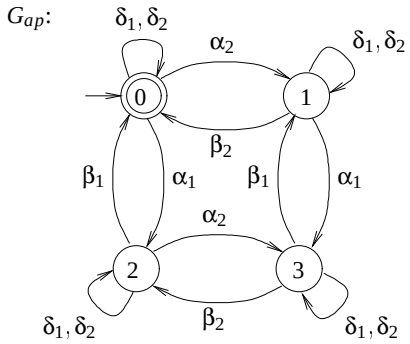

Figura 4.36: Modelo global da planta com aproximação dos sensores.

apresentadas na figura 4.38. Note que, se usadas como especificações para o problema de síntese com comportamento exato dos sensores, estas novas especificações são equivalentes às anteriores, pois no modelo exato do sensor as novas transições nunca irão ocorrer. No entanto, as modificações propostas para completar as especificações são empíricas, baseadas na compreensão de que o sensor T_1 é utilizado com o objetivo de detectar quantas peças podem ser colocadas no *buffer* até que se atinja a capacidade máxima, enquanto o sensor T_2 é utilizado para detectar a quantidade de peças que ainda pode ser retirada do *buffer* até seu esvaziamento.

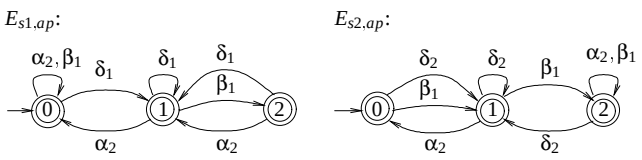

Figura 4.37: Especificações para o caso B (modelo aproximado).

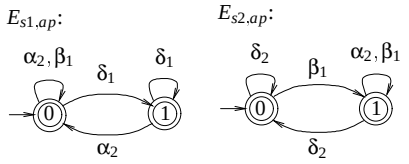

Figura 4.38: Especificações para o caso C (modelo aproximado).

Em cada um dos casos enunciados, para cada especificação $E_{si,ap}$ é calculada a planta local $G_{Si,ap} = G_1 \| G_2 \| G_{si,ap}$, $i = 1, 2$. Os supervisores modulares $S_{1,ap}$ e $S_{2,ap}$, calculados a partir da aproximação do comportamento dos sensores, são definidos sobre os alfabetos $\Sigma_{q\delta_1}$ e $\Sigma_{q\delta_2}$, respectivamente, e têm as propriedades apresentadas na tabela 4.4.

Para os três casos, pode-se verificar que os supervisores $S_{1,ap}$ e $S_{2,ap}$ são localmente modulares. Além disso, para os casos A e B, é também verdade que o comportamento do sistema real sob supervisão aproximada é equivalente ao comportamento sob supervisão exata: $S_{1,ap} \| S_{2,ap} \| G = S_1 \| S_2$.

No entanto, para o caso C, o supervisor obtido para a planta aproximada G_{ap} é muito restritivo. Para este caso, os supervisores são apresentados nas figuras 4.39 e 4.40. Pode-se verificar que

Problema	$O(\cdot)$	Supervisor	Estados	Transições
Planta aproximada, caso A	$4^2 * 6^2$	$S_{1,ap}$	22	64
		$S_{2,ap}$	24	70
Planta aproximada, caso B	$4^2 * 3^2$	$S_{1,ap}$	10	28
		$S_{2,ap}$	12	34
Planta aproximada, caso C	$4^2 * 2^2$	$S_{1,ap}$	4	8
		$S_{2,ap}$	8	22

Tabela 4.4: Complexidade computacional (para o cálculo de cada supervisor modular local) e propriedades dos supervisores para a aproximação da planta, casos A, B e C.

$L(S_{1,ap}\|S_{2,ap}\|G) = \{\varepsilon\}$ - note que os eventos α_1 e β_1 pertencem ao alfabeto que define $S_{1,ap}$, embora não estejam definidos nos estados. Assim, a planta física não evolui, porque a máquina M_1 não sai do estado inicial.

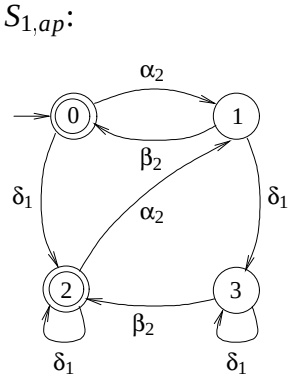

Figura 4.39: Supervisor S_1 para o caso C (modelo aproximado).

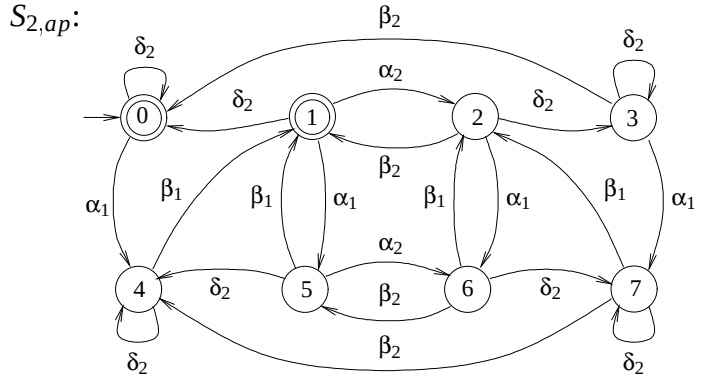

Figura 4.40: Supervisor S_2 para o caso C (modelo aproximado).

Para entender por que o sistema não evolui no caso C, pode-se verificar que a especificação $E_{s1,ap}$ apresenta uma não-controlabilidade com relação à planta aproximada G_{ap} que não pode ser contornada. Seja, para tanto, a projeção $P_s : \Sigma_{q\delta1\delta2} \to \Sigma_{s1} \cup \Sigma_{s2}$. Pode-se observar que $\alpha_1\delta_1\beta_1 \in L(G_{ap})$ e $\alpha_1\delta_1 \in P_s^{-1}(E_{s1,ap})$, mas $\alpha_1\delta_1\beta_1 \notin P_s^{-1}(E_{s1,ap})$. No entanto, a cadeia $\alpha_1\delta_1\beta_1$ não pertence também à linguagem gerada pela planta $L(G)$, o que significa que esta não-controlabilidade não existe para a planta G. No modelo exato da planta, no estado onde δ_1 está definido, é possível inibir β_1, inibindo α_1. □

Exemplo 4.7 (Mesa Giratória com Posição Intermediária e Sensor Aproximado)
Para o problema da mesa giratória com sensor, considere também a aproximação máxima em Σ_δ de G_s, onde $\Sigma_\delta = \{\delta\}$. Como a planta G_δ é assincrona com relação a G_i, $i = 0,1,2,3,4$, a representação por sistemas produto é composta pelas seguintes plantas: G_0, G_1, G_2, G_3, G_4 e G_δ. Neste caso, não há perda de refinamento da RSP original apresentada no exemplo 2.7. Note que, na especificação $Rc_{1,s}$, apresentada na figura 4.17, o evento δ não está definido em alguns estados. Nestes estados,

acrescenta-se este evento em *self-loop*, dando origem à especificação $Rc_{1s,ap}$ (figura 4.41).

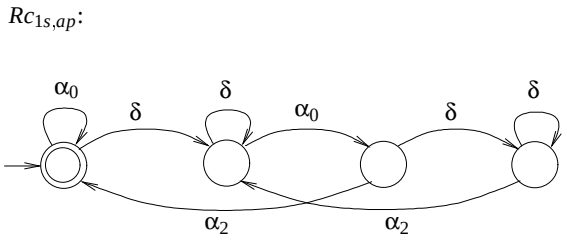

Figura 4.41: Restrição $Rc_{1s,ap}$.

Sejam portanto as especificações $\{Ra_s, Rb_{1,s}, Rb_{2,s}, Rb_{3,s}, Rb_{4,s}, Rc_{0,s}, Rc_{1s,ap}, Rc_{2,s}, Rc_{3,s}\}$, com as demais especificações definidas no exemplo 4.4. Neste caso, as plantas locais são definidas como: $G_{A,s} = G_0 \| G_1 \| G_2 \| G_3 \| G_\delta$; $G_{Bi,s} = G_0 \| G_i$, para $i = 1,2,3,4$; $G_{C0,s} = G_0 \| G_1$; $G_{C1,s} = G_0 \| G_2 \| G_\delta$; $G_{Ci,s} = G_0 \| G_i \| G_{i+1}$, para $i = 2,3$.

Novamente, o conjunto de supervisores obtidos é localmente modular. Os geradores relacionados à solução do problema têm suas propriedades relevantes apresentadas na tabela 4.7.

Resolução Modular Local	E_x	G_X	E_X	S_X	RS_X
a_s	2	16	32	$(32,152)$	$(2,9)$
$b_{i,s}, i = 1,2,3,4$	2	4	3	$(3,4)$	$(2,4)$
$c_{0,s}$	2	4	8	$(6,16)$	$(2,4)$
$c_{1s,ap}$	4	4	8	$(16,40)$	$(4,8)$
$c_{i,s}, i = 2,3$	4	8	32	$(24,52)$	$(4,8)$
Total (somatório)				$(114,328)$	$(24,53)$
Resolução Monolítica	R	G	K	S	
	912	32	400	$(304,1007)$	

Tabela 4.5: Número de estados dos geradores para o problema da mesa com sensor aproximado (exemplo 4.7). Para os supervisores, é representado o par (número de estados, número de transições).

Pode-se verificar que a solução, neste caso, é também equivalente à obtida no exemplo 4.1, quando projetada no alfabeto sem o evento do sensor. Além disso, os supervisores minimizados obtidos são similares aos obtidos pela resolução do mesmo problema sem aproximação dos sensores, apresentada no exemplo 4.4. A única diferença consiste em duas transições adicionais no supervisor $RSc_{1s,ap}$ em relação a $RSc_{1,s}$, em função dos *self-loops* acrescentados para permitir a resolução do PCS com aproximação do sensor.

Os supervisores não minimizados, obtidos aqui, possuem menor número de estados do que aqueles obtidos nos exemplos 4.1 e 4.4. Em particular, Sa_s possui 32 estados, contra 48 de Sa', o que significa que o acréscimo em número de estados decorrente da introdução da posição P' pôde ser contornado pela alocação de um sensor em P'. Também os supervisores $Sc_{0,s}$ e $Sc_{1s,ap}$ possuem 6 e 16 estados, contra 48 do supervisor Sc'_1. □

Finalmente, pelos resultados apresentados no capítulo 4, pode-se afirmar que o problema de controle supervisório com modelos exatos de sensores possui solução garantidamente equivalente à solução do problema original. Porém, a utilização dos modelos exatos não introduz ganho, em termos de redução da complexidade computacional envolvida no cálculo de cada supervisor modular local. Este ganho pode ser introduzido pela utilização de aproximações do comportamento dos sensores, porém neste caso não se garante que a solução obtida seja adequada.

Capítulo 5

Emulação de Sensores

5.1 Introdução

Em algumas situações, a utilização de aproximações do comportamento de sensores é suficiente para que o problema de controle supervisório com sensores aproximados tenha solução equivalente à solução do PCS original, porém com menor custo computacional. No entanto, em geral à introdução de um sensor real em uma planta física corresponde um maior custo econômico, motivando por exemplo trabalhos discutindo a minimização do número de sensores em sistemas a eventos discretos (Young e Garg, 1993; Haji-Valizadeh e Loparo, 1996; Yoo e Lafortune, 2002).

Por outro lado, com a introdução de um sensor à planta e a modelagem de seu comportamento exato, é possível observar que o modelo da planta com sensor indica, inequivocamente, todas as situações nas quais o sensor pode ser sensibilizado. Neste sentido, desde que o modelo da planta possa gerar espontaneamente o evento correspondente à sensibilização, não há necessidade de que se introduza um sensor na planta física. Pode-se falar, neste caso, em uma emulação do comportamento do sensor no nível do modelo da planta, ou, no caso da abordagem de controle modular local, em emulação no nível da representação por sistemas produto.

Com uma aproximação do comportamento do sensor, a emulação no nível do modelo da planta não mais faz sentido, já que cadeias do modelo que contenham o evento de sensibilização podem não ter contrapartida física - o caso mais evidente é a aproximação do sensor por *self-loops* em todos os estados do modelo da planta.

Este capítulo aborda a emulação do comportamento do sensor no nível dos supervisores, definida como a utilização do modelo exato do sensor como especificação adicional na resolução do problema de controle supervisório. Objetiva-se mostrar que, quando a planta original é combinada com o modelo aproximado do sensor, a utilização de emulação pode permitir que o comportamento obtido em malha fechada seja equivalente ao obtido quando a planta é combinada com o modelo exato do

sensor, com a vantagem adicional de que a emulação permite que se prescinda de sensores reais. Para tanto, é necessária uma mudança na interpretação da controlabilidade do evento de sensibilização, como será discutido a seguir.

5.2 Síntese de Supervisores com Emulação de Sensores

Seja o sistema apresentado na figura 4.33, onde pode-se notar que a planta, apresentada em sua representação por sistemas produto, contém a aproximação máxima do comportamento de um sensor, representada por G_δ. Além disso, as especificações genéricas são definidas sobre alfabetos que possivelmente contêm o evento do sensor. Como apresentado no capítulo 4, a solução deste problema não apenas pode ser mais restritiva do que a solução do PCS original, ilustrado na figura 4.12, como também depende da redução adotada para as especificações genéricas originais. Além disso, a equivalência da solução dos dois problemas, no capítulo anterior, foi testada a partir da resolução do problema original, que se deseja evitar, como ressaltado no capítulo 3.

Suponha agora que, ao sistema apresentado na figura 4.33, seja acrescentada uma especificação genérica $E_s = L(G_s)$, obtendo-se a estrutura apresentada na figura 5.1.

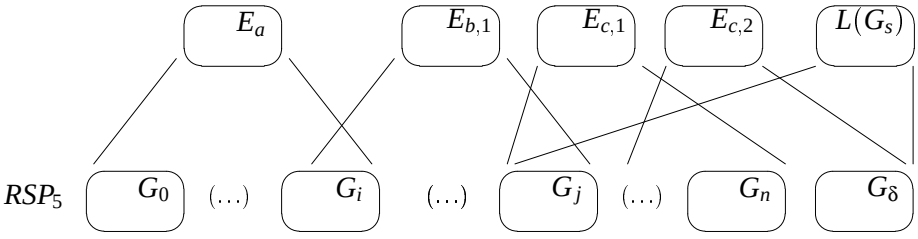

Figura 5.1: RSP do sistema exemplo com inclusão de G_s como especificação.

Pode-se pensar a especificação E_s como elemento de emulação do comportamento do sensor, compreendida como a substituição do sensor físico por um mecanismo interno ao dispositivo de controle. Neste sentido, o supervisor que emula o sensor exerce influência sobre a ocorrência do evento δ, o que significa que este evento, neste caso, deve ser controlável ou forçável.

O supervisor modular calculado a partir da especificação E_s restringe o comportamento das plantas de forma que a linguagem sob supervisão reproduza o comportamento da planta com sensor.

As seções que seguem discutem a controlabilidade da especificação E_s, bem como a equivalência da solução do problema ilustrado na figura 5.1 e do PCS original, sob a ótica das abordagens de controle supervisório com eventos controláveis e com eventos forçáveis.

5.3 Emulação por Evento Controlável

Para as proposições que seguem, considere os alfabetos $\Sigma_d \subseteq \Sigma_{ss} \subseteq \Sigma_q \subseteq \Sigma$, $\Sigma_{i\tau} = \Sigma_i \dot{\cup} \{\tau\}$, $\Sigma_{i\delta} = \Sigma_i \dot{\cup} \{\delta\}$ e $\Sigma_{i\tau\delta} = \Sigma_i \dot{\cup} \{\tau\} \dot{\cup} \{\delta\}$, $i \in \{d, ss, q\}$, e $\Sigma_s = \Sigma_{ss\delta}$, como definidos nas seções 4.2 e 4.3, com $\tau, \delta \notin \Sigma$.

Sejam também as projeções $P_{ss} : \Sigma_{q\tau\delta} \to \Sigma_{ss}$, $P_d : \Sigma_{q\tau\delta} \to \Sigma_d$ e $P_s = \Sigma_{q\tau\delta} \to \Sigma_s$, como definidas na seção 4.3, e a projeção $P_{-\delta} : \Sigma_{q\delta} \to \Sigma_q$.

Dada uma planta com alfabeto Σ, considere uma representação por sistemas produto $\{G_i, i = 1, \ldots, m\}$. Para um sensor T para esta planta, sejam a linguagem sensibilizadora genérica $L_{ss\tau} \subseteq \Sigma^*_{ss\tau}$ e a linguagem linguagem auxiliar $L_{aux} \subseteq \Sigma_{d\tau\delta}$. Seja ainda a planta local $G_q = \|_{i=j}^{m} G_i$, $L_m(G_q) \subseteq \Sigma^*_q$, definida como a composição das plantas da RSP cujos alfabetos contêm eventos em comum com o alfabeto Σ_{ss}.

Considere ainda o modelo exato do sensor G_s, que reconhece a linguagem $L_m(G_s) = L(G_s) = P_s(L_{ss\tau} \| L_{aux}) \subseteq \Sigma^*_s$, e o gerador G_δ tal que $L_m(G_\delta) = L(G_\delta) = \delta^*$.

A controlabilidade dos eventos de Σ_q é definida pelo particionamento deste alfabeto em eventos não controláveis $\Sigma_{u,q}$ e controláveis $\Sigma_{c,q}$. Definem-se também os alfabetos $\Sigma_{u,ss} = \Sigma_{u,q} \cap \Sigma_{ss}$ e $\Sigma_{c,ss} = \Sigma_{c,q} \cap \Sigma_{ss}$. Note que, em princípio, a controlabilidade dos eventos τ e δ não é determinada.

Finalmente, para $i \in \{ss, q\}$, sejam uma linguagem $K \subseteq \Sigma^*_{i\delta}$ e uma planta G com $L_m(G) \subseteq \Sigma^*_{i\delta}$. K é (δc)-*controlável* em relação a G se $\overline{K}\Sigma_{u,i} \cap L(G) \subseteq \overline{K}$. Para uma linguagem $M \subseteq \Sigma^*_{i\delta}$, não necessariamente (δc)-controlável, a classe de linguagens contidas em M e (δc)-controláveis em relação a G é denotada por $C_{\delta c}(M, G)$ e contém um elemento supremo, $supC_{\delta c}(M, G)$.

Analogamente, K é (δu)-*controlável* em relação a G se $\overline{K}(\Sigma_{u,i} \cup \{\delta\}) \cap L(G) \subseteq \overline{K}$. Define-se também, para uma linguagem $M \subseteq \Sigma^*_{i\delta}$, não necessariamente (δu)-controlável, a classe de linguagens contidas em M e (δu)-controláveis em relação a G, denotada por $C_{\delta u}(M, G)$ e contendo um elemento supremo, $supC_{\delta u}(M, G)$.

Proposição 5.1 *Sejam os alfabetos Σ_d e $\Sigma_{c,ss}$, a projeção P_{ss}, os geradores G_s e G_δ e a linguagem $L_{ss\tau}$, como definidos anteriormente. Assuma que o evento τ é não controlável. Se $\Sigma_d \subseteq \Sigma_{c,ss}$, então $L(G_s)$ é (δc)-controlável em relação a $P_{ss}(L_{ss\tau}) \|_{as} L(G_\delta)$.*

Prova: Assume-se que o termo controlável se refere à (δc)-controlabilidade. Seja também a projeção $P_\tau : \Sigma_{ss\tau\delta} \to \{\tau\}$.

Considere a linguagem $L_{aux} \|_{as} (\Sigma_{ss} - \Sigma_d)^*$, reconhecida pelo gerador da figura 5.2. A linguagem L_{aux}, definida em $\Sigma_{d\tau\delta}$, possui apenas eventos controláveis, com exceção de τ. Assim, no gerador apresentado para $L_{aux} \|_{as} (\Sigma_{ss} - \Sigma_d)^*$, eventos não controláveis de Σ_{ss} estão definidos em todos os estados.

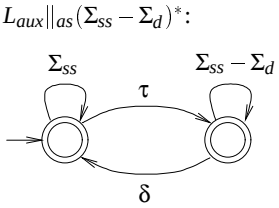

Figura 5.2: Gerador que reconhece a linguagem $L_{aux}||_{as}(\Sigma_{ss}-\Sigma_d)^*$.

Seja $s \in L_{aux}||_{as}(\Sigma_{ss}-\Sigma_d)^*$, tal que $s\tau \in L_{ss\tau}||_{as}L(G_\delta)$.

Se $P_\tau(s)=\varepsilon$, então $s \in \Sigma_{ss\delta}$ e, portanto, o gerador que reconhece $L_{aux}||_{as}(\Sigma_{ss}-\Sigma_d)^*$ se encontra no estado inicial, para o qual τ está definido. Assim, $s\tau \in L_{aux}||_{as}(\Sigma_{ss}-\Sigma_d)^*$.

Suponha agora $P_\tau(s)\neq\varepsilon$. Então, pela definição de linguagem sensibilizadora genérica, é possível escrever $s\tau = t\tau v\tau$, com $v \in \Sigma_{ss\delta}$ e $P_d(v)\neq\varepsilon$. Mas, se $P_d(v)\neq\varepsilon$, e $t\tau v \in L_{aux}||_{as}(\Sigma_{ss}-\Sigma_d)^*$, então o gerador $L_{aux}||_{as}(\Sigma_{ss}-\Sigma_d)^*$ retornou ao estado inicial, ou seja, $t\tau v\tau = s\tau \in L_{aux}||_{as}(\Sigma_{ss}-\Sigma_d)^*$. Logo, $L_{aux}||_{as}(\Sigma_{ss}-\Sigma_d)^*$ é controlável em relaçao a $L_{ss\tau}||_{as}L(G_\delta)$.

Além disso, $L_{ss\tau}||_{as}L(G_\delta)$ é também controlável em relação a $L_{ss\tau}||_{as}L(G_\delta)$. Como $L_{aux}||_{as}(\Sigma_{ss}-\Sigma_d)^*$ e $L_{ss\tau}||_{as}L(G_\delta)$ são linguagens prefixo-fechadas, então $(L_{aux}||(\Sigma_{ss}-\Sigma_d)^*)||(L_{ss\tau}||L(G_\delta)) = L_{aux}||L_{ss\tau}$ é também controlável em relação a $L_{ss\tau}||_{as}L(G_\delta)$.

Para quaisquer $s \in L_{aux}||_{as}(\Sigma_{ss}-\Sigma_d)^*$, $e_i \in \Sigma_{ss}-\Sigma_d$, $se_i \in L_{aux}||_{as}(\Sigma_{ss}-\Sigma_d)^*$. Logo, $se_i \in L_{aux}||L_{ss\tau}$ se e somente se $se_i \in L_{ss\tau}||L(G_\delta)$. Portanto $P_s(se_i)=P_s(s)e_i \in P_s(L_{aux}||L_{ss\tau})$ se $P_s(s)e_i \in P_s(L_{ss\tau}||L(G_\delta))$. Portanto, qualquer cadeia $P_s(s) \in P_s(L_{aux}||L_{ss\tau})$ contém como continuações válidas todos os eventos não controláveis e_i tais que $P_s(s)e_i \in P_s(L_{ss\tau}||L(G_\delta))$.

Portanto, $L(G_s) = P_s(L_{aux}||L_{ss\tau})$ é controlável em relação a $P_s(L_{ss\tau}||_{as}L(G_\delta)) = P_s(L_{ss\tau})||_{as}P_s(L(G_\delta)) = P_{ss}(L_{ss\tau})||_{as}L(G_\delta)$. □

Proposição 5.2 *Sejam os geradores G_q e G_s como definidos anteriormente. Então $L_m(G_q)$ e $L(G_s)$ são localmente modulares.*

Prova:

Sejam G tal que $L_m(G)=L(G_s)||\Sigma_q^*$ e A com $L_m(A)=(\Sigma_q\cup\Sigma_\delta)^*$. Seja também $K_q=L_m(G_q)$. $K_q||_{as}L(G_\delta)$ e $L_m(A)$ são modulares. Como A é a aproximação externa máxima em $\{\delta\}$ de G, pela proposição 3.8, $K_q||_{as}L(G_\delta)$ e $L_m(G)$ são modulares, ou seja, $L_m(G_q)$ e $L(G_s)$ são localmente modulares. □

Proposição 5.3 *Sejam os alfabetos Σ_d e $\Sigma_{c,q}$ e os geradores G_q, G_s e G_δ, como definidos anteriormente. Se $\Sigma_d \subseteq \Sigma_{c,q}$, então $L_m(G_q||G_s)$ é (δc)-controlável em relação a $L(G_q||G_\delta)$.*

Prova:

Assume-se que o termo "controlável" se refere à (δc)-controlabilidade. Seja também a linguagem $M_{ss} = P_{ss}(L_{ss\tau})\|\Sigma_q^*$.

i) $\Sigma_d \subseteq \Sigma_{c,q}$ implica $\Sigma_d \subseteq \Sigma_{c,q} \cap \Sigma_{ss} = \Sigma_{c,ss}$. Da proposição 5.1, $L(G_s)$ é controlável em relação a $P_{ss}(L_{ss\tau})\|_{as}L(G_\delta)$. Então, $L(G_s)\|\Sigma_q^*$ é controlável em relação a $P_{ss}(L_{ss\tau})\|L(G_\delta)\|\Sigma_q^* = M_{ss}\|L(G_\delta)$.

ii) Da definição de linguagem sensibilizadora, $P_{ss}(L(G_q)) \subseteq P_{ss}(L_{ss\tau})$.

Assim, $L(G_q) \subseteq P_{ss}^{-1}(P_{ss}(L(G_q))) = P_{ss}(L(G_q))\|\Sigma_q^* \subseteq M_{ss}$, e portanto $L(G_q\|G_\delta) \subseteq M_{ss}\|L(G_\delta)$. Pela proposição 3.3, $L(G_s)\|\Sigma_q^*$ é controlável em relação a $L(G_q\|G_\delta)$.

iii) $L_m(G_q\|G_\delta)$ é controlável em relação a $L(G_q\|G_\delta)$.

iv) De (ii) e (iii), $L(G_s)\|\Sigma_q^*$ e $L_m(G_q\|G_\delta)$ são linguagens controláveis em relação a $L(G_q\|G_\delta)$. Além disso, pela proposição 5.2, estas linguagens são modulares. Logo, sua intersecção é controlável em relação a $L(G_q\|G_\delta)$.

Assim, $L(G_s)\|\Sigma_q^*\|L_m(G_q\|G_\delta) = L_m(G_q\|G_s)$ é controlável em relação a $L(G_q\|G_\delta)$. □

A proposição 5.3 indica que, dada uma planta livre, acrescentar o modelo exato G_s de um sensor ao seu comportamento é equivalente a restringir o seu comportamento com uma especificação equivalente a G_s, já que $L_m(G_q\|G_s) = supC_{\delta c}(L_m(G_q\|G_s), G_q\|G_\delta)$.

Para a planta representada pelo gerador G_q, seja agora uma especificação genérica E definida em $\Sigma_e \subseteq \Sigma_q$, e seja a especificação global $K_q = E\|L_m(G_q) \subseteq \Sigma_q$. Para um conjunto de índices $i \in I_k$, seja também um conjunto de especificações genéricas $E_{si} \subseteq \Sigma_{si}^*$, com $\Sigma_{si} \subseteq \Sigma_{q\delta}$, tais que, para as especificações $K_{si} = E_{si}\|L_m(G_q) \subseteq \Sigma_{q\delta}^*$, seja verdade que $K_q\|L(G_s) = (\underset{i\in I_k}{\|} K_{si})\|L(G_s) \subseteq \Sigma_{q\delta}^*$.

O seguinte resultado preliminar indica que, para especificações que não contêm o evento δ em seu alfabeto, a controlabilidade deste evento não influi no cálculo do supervisor correspondente.

Proposição 5.4 *Sejam os geradores G_q e G_δ e a linguagem E, como definidos nesta seção. Então* $supC_{\delta c}(E\|L_m(G_q\|G_\delta), G_q\|G_\delta) = supC_{\delta u}(E\|L_m(G_q\|G_\delta), G_q\|G_\delta)$.

Prova:

Como Σ_q e $\{\delta\}$ são disjuntos, pela proposição 2.7, $supC_{\delta c}(E\|L_m(G_q\|G_\delta), G_q\|G_\delta) = supC(E\|L_m(G_q), G_q)\|_{as}supC_{\delta c}(L(G_\delta), G_\delta) = supC(E\|L_m(G_q), G_q)\|_{as}L(G_\delta) = supC(E\|L_m(G_q), G_q)\|_{as}supC_{\delta u}(L(G_\delta), G_\delta) = supC_{\delta u}(E\|L_m(G_q\|G_\delta), G_q\|G_\delta)$. □

A seguinte proposição apresenta o principal resultado deste capítulo:

Proposição 5.5 *Sejam os geradores G_q, G_s e G_δ como definidos anteriormente. Sejam também especificações E, K_q, E_{si} e K_{si}, $i \in I_k$, tais que $K_q \| L(G_s) = (\underset{i\in I_k}{\|} K_{si}) \| L(G_s)$. Se $\Sigma_d \subseteq \Sigma_{c,q}$, e se o conjunto de linguagens $\{supC_{\delta c}(K_{si}, G_q \| G_\delta),\ i \in I_k\} \cup \{L_m(G_q \| G_s)\}$ é modular, então $(\underset{i\in I_k}{\|} supC_{\delta c}(K_{si}, G_q \| G_\delta)) \| G_s = supC(K_q, G_q) \| G_s$. Além disso, para a projeção $P_{-\delta}$ definida anteriormente, $P_{-\delta}((\underset{i\in I_k}{\|} supC_{\delta c}(K_{si}, G_q \| G_\delta)) \| G_s) = supC(K_q, G_q)$.*

Prova:

Seja $K_{q\delta} = K_q \|_{as} L(G_\delta)$.

Por definição, $supC_{\delta c}(K_{q\delta}, G_q \| G_\delta)$ e $L_m(G_q \| G_\delta)$ são modulares. Como $supC_{\delta c}(K_{q\delta}, G_q \| G_\delta) = P_{-\delta}^{-1}(supC(K_q, G_q))$ e $G_q \| G_\delta$ é a aproximação externa máxima em $\{\delta\}$ de $G_q \| G_s$, pela proposição 3.8, $supC_{\delta c}(K_{q\delta}, G_q \| G_\delta)$ e $L_m(G_q \| G_s)$ são modulares. Como, pela proposição 5.3, $supC_{\delta c}(L_m(G_q \| G_s), G_q \| G_\delta) = L_m(G_q \| G_s)$, então $supC_{\delta c}(K_{q\delta}, G_q \| G_\delta)$ e $supC_{\delta c}(L_m(G_q \| G_s), G_q \| G_\delta)$ são modulares.

Logo, $supC(K_q, G_q) \| L(G_s) = supC_{\delta c}(K_{q\delta}, G_q \| G_\delta) \| L_m(G_q \| G_s) = supC_{\delta c}(K_{q\delta}, G_q \| G_\delta) \| supC_{\delta c}(L_m(G_q \| G_s), G_q \| G_\delta) = supC_{\delta c}(K_{q\delta} \| L_m(G_q \| G_s), G_q \| G_\delta) = supC_{\delta c}(K_q \| L(G_s),\ G_q \| G_\delta) = supC_{\delta c}((\underset{i\in I_k}{\|} K_{si}) \| L(G_s),\ G_q \| G_\delta) = supC_{\delta c}((\underset{i\in I_k}{\|} K_{si}) \| L_m(G_q \| G_s),\ G_q \| G_\delta) = (\underset{i\in I_k}{\|} supC_{\delta c}(K_{si}, G_q \| G_\delta)) \| supC_{\delta c}(L_m(G_q \| G_s), G_q \| G_\delta) = (\underset{i\in I_k}{\|} supC_{\delta c}(K_{si}, G_q \| G_\delta)) \| G_s$.

Além disso, pela proposição 3.6, $P_{-\delta}((\underset{i\in I_k}{\|} supC_{\delta c}(K_{si}, G_q \| G_\delta)) \| G_s) = P_{-\delta}(supC(K_q, G_q) \| L(G_s)) = supC(K_q, G_q)$. □

Quando os eventos dessensibilizadores são controláveis, a proposição 5.5 garante que o comportamento obtido com emulação de G_s é equivalente ao do problema de controle original, sem que seja necessária a resolução do problema original. Esta equivalência é válida para qualquer redução das especificações tal que $K_q \| L(G_s) = (\|_i K_{si}) \| L(G_s)$, desde que respeitada a condição de modularidade imposta na proposição.

A controlabilidade dos eventos dessensibilizadores garante que o evento δ possa ser introduzido antes da próxima ocorrência de eventos de Σ_d, já que a especificação E_s desabilita estes eventos antes da ocorrência de δ. Por outro lado, quando o modelo do sensor é introduzido à planta, esta ordenação de eventos é imposta diretamente pelo modelo da planta.

A partir da proposição 5.5, pode-se propor a seguinte metodologia para resolução do problema de controle com emulação de sensores, quando os eventos dessensibilizadores são controláveis:

Proposição 5.6 (Procedimento para resolução do PCS com emulação de sensores (controláveis)) *Seja um problema de controle supervisório original para uma planta em sua representação por sistemas produto, e seja um sensor T para esta planta.*

1. Modelar a planta G, sem o comportamento do sensor, como uma RSP.
2. Modelar a linguagem sensibilizadora L_{ss}, ou a linguagem sensibilizadora genérica $L_{ss\tau}$.
3. Calcular o comportamento exato do sensor $L(G_s)$, e acrescentá-lo como uma nova especificação genérica.
4. Incluir na RSP a aproximação máxima G_δ, tal que $L_m(G_\delta) = \delta^*$.
5. Reduzir as especificações originais a partir do uso do sensor.
6. Resolver o problema pela abordagem de controle modular local. Note que não é necessário calcular $supC_{\delta c}(L(G_s), G_q \| G_\delta)$, pois $L(G_s)$ é (δc)-controlável.
7. Realizar o teste de modularidade local, incluindo a linguagem $L(G_s)$ no teste.

Para ilustrar este procedimento, o problema da mesa com 5 posições é retomado no exemplo a seguir.

Exemplo 5.1 (Mesa Giratória com Posição Intermediária e Sensor Emulado)
Para o problema da mesa giratória com sensor aproximado, considere também a emulação do modelo G_s. Novamente, a representação por sistemas produto é composta pelas seguintes plantas: G_0, G_1, G_2, G_3, G_4 e G_δ, sem perda de refinamento da RSP original apresentada no exemplo 2.7.

No entanto, diferentemente do exemplo 4.7, a especificação $Rc_{1,s}$ da figura 4.17 não precisa ser alterada, pois o evento δ é, agora, controlável.

Ao conjunto de especificações do sistema, é incluída $R_s = L(G_s)$. Assim, para o conjunto de índices $I_k = \{a_s, b_{1,s}, b_{2,s}, b_{3,s}, b_{4,s}, c_{0,s}, c_{1,s}, c_{2,s}, c_{3,s}, s\}$ seja o conjunto de especificações $\{Ri, i \in I_k\}$, que contém R_s e as demais especificações como definidas no exemplo 4.4. Neste caso, a planta local para R_s é dada por $G_S = G_0 \| G_1 \| G_\delta$, enquanto as demais plantas locais são novamente definidas como no exemplo 4.7: $G_{A,s} = G_0 \| G_1 \| G_2 \| G_3 \| G_\delta$; $G_{Bi,s} = G_0 \| G_i$, para $i = 1,2,3,4$; $G_{C0,s} = G_0 \| G_1$; $G_{C1,s} = G_0 \| G_2 \| G_\delta$; $G_{Ci,s} = G_0 \| G_i \| G_{i+1}$, para $i = 2,3$.

A tabela 5.1 apresenta o número de estados dos geradores envolvidos na resolução do problema, bem como o número de transições das soluções finais para a resolução monolítica e modular local. Note que o procedimento para o cálculo de S_S, que reconhece a linguagem $supC_{\delta c}(R_s \| G_S, G_S)$, foi realizado, embora este cálculo seja desnecessário, em função da controlabilidade de R_s. Ainda, pode-se verificar que RS_s e R_s são equivalentes.

O conjunto de supervisores obtidos na resolução deste problema pela abordagem modular local é localmente modular. Logo, a solução obtida é equivalente à solução o obtida no exemplo 4.1, o que pode ser verificado pelo cálculo de $P_{-\delta}(L_m(\underset{i \in I_k}{\|} S_I))$. Note que esta verificação não se faz aqui

Resolução Modular Local	E_x	G_X	E_X	S_X	RS_X
a_s	2	16	32	$(32,152)$	$(2,9)$
$b_{i,s}, i = 1,2,3,4$	2	4	3	$(3,4)$	$(2,4)$
$c_{0,s}$	2	4	8	$(6,16)$	$(2,4)$
$c_{1,s}$	4	4	8	$(16,32)$	$(4,6)$
$c_{i,s}, i = 2,3$	4	8	32	$(24,52)$	$(4,8)$
s	4	4	16	$(16,36)$	$(4,8)$
Total (somatório)				$(130,356)$	$(28,59)$
Resolução Monolítica	R	G	K	S	
	1368	32	599	$(455,1198)$	

Tabela 5.1: Número de estados dos geradores para o problema da mesa com sensor emulado (exemplo 5.1). Para os supervisores, é representado o par (número de estados, número de transições).

necessária, como enunciado pela proposição 5.5.

É possível observar que os supervisores obtidos são calculados para a planta aproximada, o que significa que possuem o mesmo número de estados daqueles obtidos na resolução com aproximação do sensor. Exceção é feita para $Sc_{1,s}$, pois na emulação do sensor não houve necessidade de que a função de transição do gerador que reconhece $Rc_{1,s}$ fosse alterada para ser total com relação ao evento δ. Ainda, é possível observar que os supervisores locais minimizados obtidos aqui são equivalentes aos obtidos quando da utilização do modelo exato do sensor na planta.

Enfim, tanto os supervisores locais quanto os supervisores locais minimizados, obtidos com a emulação do sensor na posição P', constituem soluções com menor número de estados do que aqueles obtidos no exemplo 4.1. Para cada especificação local, a introdução do sensor possibilitou também a redução da complexidade computacional do cálculo do supervisor. □

No entanto, quando Σ_d contém eventos não controláveis, a abordagem de eventos forçáveis fornece o mecanismo desejado para antecipação de δ, como será mostrado a seguir.

5.4 Emulação por Evento Forçável

Considere novamente, para $i \in \{ss, q\}$, uma linguagem K e uma planta G, tais que $K \subseteq L_m(G) \subseteq \Sigma^*_{i\delta}$. K é (δf)-controlável em relação a G se, para toda cadeia $s \in \overline{K}$, $s\Sigma_{u,i} \cap L(G) \subseteq \overline{K}$ ou $\emptyset \subset \Sigma_K(s) \subseteq \{\delta\}$. A última condição pode, também, ser escrita como $\Sigma_K(s) = \delta$.

Para $M \subseteq \Sigma^*_{i\delta}$, não necessariamente (δf)-controlável em relação a G, a classe de linguagens contidas em M e (δf)-controláveis em relação a G é denotada por $C_{\delta f}(M,G)$ e contém um elemento supremo, $supC_{\delta f}(M,G)$. Note que é possível mostrar, utilizando as definições 2.1 e 2.4, que $supC_{\delta u}(M,G) \subseteq supC_{\delta c}(M,G) \subseteq supC_{\delta f}(M,G)$.

Para as proposições que seguem, suponha o caso particular $\Sigma_q = \Sigma_{ss} = \Sigma_d$. De fato, a principal restrição deste caso particular consiste em assumir que todos os eventos da linguagem sensibilizadora são também dessensibilizadores, pois, se uma linguagem sensibilizadora genérica $L_{ss\tau}$ é definida em um alfabeto $\Sigma_{ss\tau} \subset \Sigma_{q\tau}$, pode-se considerar como nova linguagem sensibilizadora $L'_{ss\tau} = L_{ss\tau}\|_{as}(\Sigma_q - \Sigma_{ss})^* \subseteq \Sigma^*_{q\tau}$.

Proposição 5.7 *Sejam os alfabetos $\Sigma_{ss} = \Sigma_d$, a projeção P_{ss}, os geradores G_s e G_δ e a linguagem $L_{ss\tau}$, como definidos na seção anterior. Então $L(G_s)$ é (δf)-controlável em relação a $P_{ss}(L_{ss\tau})\|_{as}L(G_\delta)$.*

Prova:

Assume-se que o termo controlável se refere à (δf)-controlabilidade. Sejam também a projeção $P_\tau : \Sigma_{ss\tau\delta} \to \{\tau\}$. Como $\Sigma_{ss} = \Sigma_d$, $L_{aux}\|_{as}(\Sigma_{ss} - \Sigma_d)^* = L_{aux}$.

Logo, apenas no estado inicial do gerador que reconhece L_{aux} os eventos de Σ_{ss} são definidos. No segundo estado, apenas o evento δ está definido. Logo, para qualquer $s \in L_{aux}$, se s leva ao estado inicial de L_{aux}, então $s\Sigma_{u,ss} \subseteq s\Sigma_{ss} \subseteq L_{aux}$, enquanto que, se s leva ao segundo estado, $\Sigma_{Laux}(s) = \{\delta\}$. Portanto L_{aux} é controlável em relação a qualquer linguagem definida em $\Sigma_{ss\tau\delta}$, em particular em relação a $L_{ss\tau}\|_{as}L(G_\delta)$.

Além disso, $L_{ss\tau}\|_{as}L(G_\delta)$ é também controlável em relação a $L_{ss\tau}\|_{as}L(G_\delta)$. Sabe-se que L_{aux} e $L_{ss\tau}\|_{as}L(G_\delta)$ são linguagens prefixo-fechadas. Considere que L_{aux} força o evento δ. Então, como a concatenação de qualquer cadeia de $L_{ss\tau}\|_{as}L(G_\delta)$ com o evento δ pertence a $L_{ss\tau}\|_{as}L(G_\delta)$, então, pela proposição 2.9, $L_{aux} \cap (L_{ss\tau}\|L(G_\delta)) = L_{aux}\|L_{ss\tau}$ é também controlável em relação a $L_{ss\tau}\|_{as}L(G_\delta)$.

Deseja-se agora mostrar que $L(G_s) = P_s(L_{aux}\|L_{ss\tau})$ é controlável com relação a $P_s(L_{ss\tau}\|_{as}L(G_\delta))$. Note que, como $\Sigma_{ss\tau\delta} = \Sigma_{q\tau\delta}$, a projeção P_s retira apenas o evento τ. A demonstração que segue se baseia no fato de que qualquer cadeia $s \in L_{aux}$ pode ser escrita em uma das três seguintes formas: $s = t\tau\delta v$, com $v \in \Sigma^*_{ss}$; $s \in \Sigma^*_{ss}$; ou $s = t\tau$, para t qualquer.

Suponha o caso $s \in L_{aux}\|L_{ss\tau}$ da forma $s = t\tau\delta v$, com $v \in \Sigma^*_{ss}$. Neste caso, se_i é definido em L_{aux} para qualquer $e_i \in \Sigma_{ss}$. Logo, quando $s \in L_{aux}\|L_{ss\tau}$, $se_i \in L_{aux}\|L_{ss\tau}$ se e somente se $se_i \in L_{ss\tau}\|L(G_\delta)$, e portanto $P_s(se_i) = P_s(s)e_i \in P_s(L_{aux}\|L_{ss\tau})$ se $P_s(se_i) = P_s(s)e_i \in P_s(L_{ss\tau}\|L(G_\delta))$. Em particular, todos os eventos não controláveis definidos após $P_s(s)$ em $P_s(L_{ss\tau}\|L(G_\delta))$ estão também definidos em $P_s(L_{aux}\|L_{ss\tau}) = L(G_s)$, o que garante que a cadeia $P_s(s)$ de $L(G_s)$ satisfaz a condição de controlabilidade em relação a $P_s(L_{ss\tau}\|L(G_\delta))$ associada a eventos não controláveis.

Da mesma forma, quando $s \in L_{aux}\|L_{ss\tau}$ é tal que $P_\tau(s) = \emptyset$, se_i é definido em L_{aux} para qualquer $e_i \in \Sigma_{ss}$. Neste caso, a prova da controlabilidade de s é análoga à utilizada para s da forma $s = t\tau\delta v$.

Suponha agora que uma cadeia $s \in L_{aux}\|L_{ss\tau}$ seja da forma $s = t\tau$. Da linguagem L_{aux} pode-se deduzir que $t\tau\delta \in L_{aux}$ e que, para todo $e_i \in \Sigma_{ss}$, $t\tau e_i \notin L_{aux}$. Portanto, para $P_s(t\tau) = P_s(t) \in P_s(L_{aux}\|L_{ss\tau})$, é verdade que $P_s(t\tau\delta) = P_s(t)\delta \in P_s(L_{aux}\|L_{ss\tau})$ e, para qualquer $e_i \in \Sigma_{ss}$, $P_s(t\tau e_i) = P_s(t)e_i$. Se

$t\tau e_i \notin L_{aux}$ implica $P_s(t)e_i \notin P_s(L_{aux}\|L_{ss\tau})$ para qualquer $e_i \in \Sigma_{ss}$, então $P_s(s) \in P_s(L_{aux}\|L_{ss\tau}) = L(G_s)$ satisfaz a condição de controlabilidade $\Sigma_{L(Gs)}(P_s(s)) = \delta$. Se tal implicação não é válida, ou seja, existe e_i com $P_s(t)e_i \in P_s(L_{aux}\|L_{ss\tau})$, então existe uma cadeia $s' \in L_{aux}\|L_{ss\tau}$ tal que $P_s(t\tau) = P_s(s')$ e $s'e_i \in L_{aux}\|L_{ss\tau}$, o que significa que $P_s(s)e_i = P_s(s')e_i \in L_{aux}\|L_{ss\tau}$. Neste caso, a cadeia s' certamente é da forma $s' = t\tau\delta v$, $v \in \Sigma_{ss}^*$, ou da forma $s' \in \Sigma_{ss}^*$, o que garante que $P_s(s') = P_s(s)$ satisfaz a condição de permitir todas as continuações não controláveis válidas em $P_s(L_{ss\tau}\|L(G_\delta))$.

Portanto, qualquer cadeia de $L(G_s)$ é controlável em relação a $P_s(L_{ss\tau}\|L(G_\delta))$, e assim $L(G_s)$ é controlável em relação a $P_s(L_{ss\tau}\|_{as}L(G_\delta))$. Utilizando a propriedade 2.10, que é válida independentemente da natureza dos eventos envolvidos, $P_s(L_{ss\tau}\|_{as}L(G_\delta)) = P_s(L_{ss\tau})\|_{as}P_s(L(G_\delta)) = P_{ss}(L_{ss\tau})\|_{as}L(G_\delta)$. □

Proposição 5.8 *Sejam os alfabetos $\Sigma_q = \Sigma_{ss} = \Sigma_d$, e os geradores G_q, G_s e G_δ, como definidos na seção anterior. Então $L_m(G_q\|G_s)$ é (δf)-controlável em relação a $L(G_q\|G_\delta)$.*

Prova:

Assume-se que o termo "controlável" se refere à (δf)-controlabilidade. Sejam também a linguagem $L_{ss\tau}$ e a projeção P_{ss}, como definidas anteriormente.

i) $\Sigma_{ss} = \Sigma_d$. Da proposição 5.7, $L(G_s)$ é controlável em relação a $P_{ss}(L_{ss\tau})\|_{as}L(G_\delta)$.

ii) Pela definição de controlabilidade, para qualquer cadeia $t \in L(G_s)$, ao menos uma das condições é verdadeira: $t\Sigma_{u,q} \cap (P_{ss}(L_{ss\tau})\|_{as}L(G_\delta)) \subseteq L(G_s)$ ou $\emptyset \subset \Sigma_{L(Gs)}(t) \subseteq \{\delta\}$.

Por outro lado, da definição de linguagem sensibilizadora, $P_{ss}(L(G_q)) \subseteq P_{ss}(L_{ss\tau})$. Como $\Sigma_q = \Sigma_{ss}$, $P_{ss}(L(G_q)) = L(G_q)$. Assim, $L(G_q\|G_\delta) \subseteq P_{ss}(L_{ss\tau})\|_{as}L(G_\delta)$.

Logo, para toda cadeia $t \in L(G_s)$, se $t\Sigma_{u,q} \cap (P_{ss}(L_{ss\tau})\|_{as}L(G_\delta)) \subseteq L(G_s)$, então $t\Sigma_{u,q} \cap L(G_q\|G_\delta) \subseteq L(G_s)$. Se a segunda condição de controlabilidade é satisfeita, ou seja, se $\emptyset \subset \Sigma_{L(Gs)}(t) \subseteq \{\delta\}$, então esta condição se mantém inalterada quando a controlabilidade é verificada com relação a $L(G_q\|G_\delta)$.

Portanto, $L(G_s)$ é, também, controlável em relação a $L(G_q\|G_\delta)$.

iii) $L_m(G_q\|G_\delta)$ é controlável em relação a $L(G_q\|G_\delta)$.

iv) De (ii) e (iii), $L(G_s)$ e $L_m(G_q\|G_\delta)$ são linguagens controláveis em relação a $L(G_q\|G_\delta)$. Também, pela proposição 5.2, estas linguagens são modulares. Além disso, pode-se considerar que $L(G_s)$ força o evento δ, pois para qualquer cadeia $t \in \overline{L_m(G_q\|G_\delta)}$, $t\delta \in \overline{L_m(G_q\|G_\delta)}$. Logo, pela proposição 2.9, $L(G_s)\|L_m(G_q\|G_\delta)$ é controlável em relação a $L(G_q\|G_\delta)$.

Assim, $L(G_s)\|L_m(G_q\|G_\delta) = L_m(G_q\|G_s)$ é (δf)-controlável em relação a $L(G_q\|G_\delta)$. □

Como $L_m(G_q\|G_s)$ é (δf)-controlável em relação a $L(G_q\|G_\delta)$, existe um supervisor próprio que restringe o comportamento da planta livre $G_q\|G_\delta$ de forma que a linguagem em malha fechada obtida seja igual a $L_m(G_q\|G_s)$. Assim, diz-se que este supervisor emula a introdução do sensor na planta. Quando da emulação do sensor, como se considera como planta livre $G_q\|G_\delta$, para que o comportamento de um conjunto de supervisores seja (δf)-controlável, os demais supervisores não devem desabilitar o evento δ, como enunciado na proposição 2.9. Assim, para as demais especificações, deve ser calculada a máxima linguagem contida na especificação e (δu)-controlável em relação a planta $G_q\|G_\delta$. Desta forma, o comportamento conjunto dos supervisores obtidos é (δf)-controlável, como enunciado na proposição a seguir.

Proposição 5.9 *Sejam os geradores G_q, G_s e G_δ, definidos sobre Σ_q, Σ_s e $\{\delta\}$, com $\Sigma_s = \Sigma_q \cup \{\delta\}$, tais que G_s é o modelo exato de um sensor com evento δ para G_q, calculado com $\Sigma_d = \Sigma_{ss}$. Sejam também especificações, K_q e K_{si}, $i \in I_k$, tais que $K_q\|L(G_s) = (\underset{i\in I_k}{\|} K_{si})\|L(G_s)$. Se o conjunto de linguagens $\{supC_{\delta u}(K_{si}, G_q\|G_\delta),\ i \in I_k\} \cup \{L_m(G_q\|G_s)\}$ é modular, então a intersecção destas linguagens é (δf)-controlável com relação a $G_q\|G_\delta$.*

Prova:

Para qualquer $i \in I_k$, $supC_{\delta u}(K_{si}, G_q\|G_\delta)$ é, por definição, (δu)-controlável com relação a $G_q\|G_\delta$, e portanto também (δf)-controlável com relação a esta planta. Além disso, pela proposição 5.7, $L_m(G_q\|G_s)$ é também (δf)-controlável em relação a $G_q\|G_\delta$. Como estas linguagens são modulares, então, pela proposição 2.9, a intersecção destas linguagens é (δf)-controlável em relação a $G_q\|G_\delta$. □

A proposição 5.9 enuncia que o resultado obtido com emulação de um sensor com evento forçável é igual ao resultado obtido com a introdução do modelo com aproximação máxima do mesmo sensor, pois $(\cap_{i\in I_k} supC_{\delta u}(K_{si}, G_q\|G_\delta))$ corresponde ao comportamento conjunto dos supervisores calculados no PCS com aproximação do sensor, enquanto a intersecção desta linguagem com $L_m(G_q\|G_s)$ corresponde à aplicação dos supervisores obtidos na planta acrescida do sensor.

Como conseqüência, a emulação de sensores por eventos forçáveis não garante que o comportamento obtido será equivalente ao do problema de controle supervisório original. O exemplo a seguir ilustra esta situação.

Exemplo 5.2 (*Buffer* com capacidade oito, emulação de sensores)
Considere novamente o problema do *buffer* de capacidade 8 apresentado no exemplo 4.6, com aproximação máxima do comportamento dos sensores, alocados conforme descrito nos casos A, B e C.

Novamente, a representação por sistema produto associada à planta é dada pelo conjunto de geradores $\{G_1, G_2, G_{\delta 1}, G_{\delta 2}\}$, ilustrados na figura 4.35, enquanto a planta sem sensores é representada por $G_q = G_1\|G_2$.

Considere como especificações de comportamento novamente as linguagens $E_{s1,ap}$ e $E_{s2,ap}$, apresentadas nas figuras 4.37 para o caso B e 4.38 para o caso C.

Cada linguagem sensibilizadora apresentada no exemplo 4.6 pode ser expressa também em sua forma genérica. Para o caso *B*, estas linguagens são ilustradas na figura 5.3.

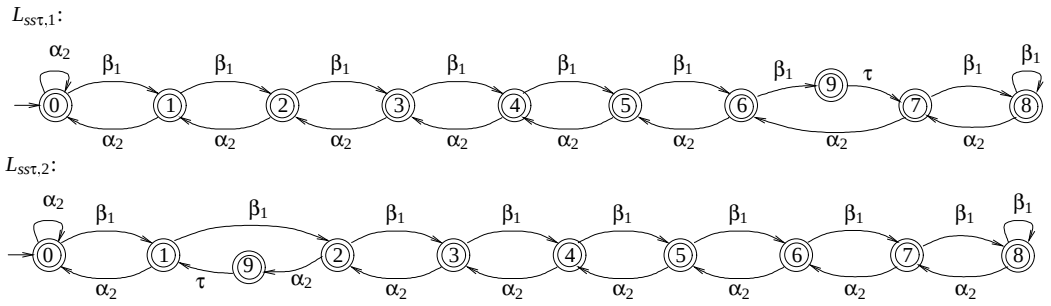

Figura 5.3: Linguagens sensibilizadoras (genéricas) para os sensores T_1 e T_2, conforme o caso B.

Para cada caso, com o objetivo de emular os sensores, considere novas linguagens sensibilizadoras $L'_{ss\tau,1} = L_{ss\tau,1}||\Sigma_1^*||\Sigma_2^*$ e $L'_{ss\tau,2} = L_{ss\tau,2}||\Sigma_1^*||\Sigma_2^*$. Note que, neste caso, todos os eventos do alfabeto $\Sigma_1 \cup \Sigma_2$ devem ser incluídos como eventos dessensibilizadores. Assim, são calculados os modelos exatos G_{s1} e G_{s2}, que introduzem os eventos δ_1 e δ_2. Estes modelos correspondem às novas especificações $E_{s3} = L(G_{s1})$ e $E_{s4} = L(G_{s2})$.

Os supervisores modulares $S_{1,ap}$ e $S_{2,ap}$ são, neste caso, equivalentes aos apresentados anteriormente, pois no seu cálculo os eventos δ_1 e δ_2 são considerados não controláveis. Ainda, $E_{s3}||L_m(G_q)$ é $(\delta_1 f)$-controlável em relação a $L(G_q||G_{\delta 1})$, enquanto $E_{s4}||L_m(G_q)$ é $(\delta_2 f)$-controlável em relação a $L(G_q||G_{\delta 2})$. Sejam ainda $K_{s3} = (E_{s3}||L_m(G_q))||_{as}L(G_{\delta 2})$ e $K_{s4} = (E_{s4}||L_m(G_q))||_{as}L(G_{\delta 1})$ Considerando que o supervisor correspondente a K_{s3} força δ_1 e que o supervisor correspondente a K_{s4} força δ_2, pode-se verificar que K_{s3} não desabilita δ_2 e que K_{s4} não desabilita δ_1, o que significa que o comportamento conjunto destes supervisores é $(\delta_i f)$-controlável, $i = 1, 2$, como enunciado na proposição 2.9.

Para os casos A, B e C, a solução global obtida com emulação dos sensores é equivalente àquela obtida com a aproximação do comportamento dos sensores, e portanto equivalente à solução do problema de controle supervisório sem sensores apenas para os casos A e B. □

Assim como para o problema de controle supervisório com sensor aproximado, é desejável neste caso o desenvolvimento de condições que, quando verificadas, garantam que a utilização de emulação por eventos forçáveis do sensor provê solução equivalente à solução do PCS original.

Capítulo 6

Conclusão

Esta dissertação de mestrado apresentou um estudo sobre sensores em sistemas a eventos discretos, sob uma ótica diferente daquelas apresentadas por Young e Garg (1993) e trabalhos relacionados. Na abordagem apresentada, o sinal do sensor não corresponde a um evento já presente no modelo da planta, como na perspectiva destes trabalhos, mas sim a uma cadeia de eventos que indica a ativação do sensor. A compreensão aqui adotada do que vem a ser um sensor parece mais adequada, na medida em que estes trabalhos associam a observabilidade dos eventos diretamente à existência de sensores. Em um sistema com supervisão monolítica, por exemplo, eventos que correspondam a comandos executados pelo sistema de controle são naturalmente observáveis, sem que haja associação natural com sensores reais.

Foram introduzidos formalmente os conceitos de modelos exato e aproximado de sensores, para os quais foi mostrado o resultado, intuitivamente trivial, de que a combinação destes modelos com o modelo original da planta não altera a solução final do problema de controle supervisório, quando as especificações originais são mantidas.

As principais contribuições apresentadas resultam da combinação da modelagem de sensores com a utilização da metodologia de controle modular local (Queiroz, 2000). Para os casos em que especificações podem ser reduzidas a partir da utilização do evento do sensor, foi mostrado que o uso de aproximações para os modelos dos sensores permite a redução da complexidade do cálculo dos supervisores locais correspondentes, e que esta redução depende da alocação dos sensores. No entanto, não há garantia de que a solução global obtida seja equivalente à solução do problema de controle original (sem sensores). Além disso, não foi indicado um procedimento sistemático para modificação das especificações reduzidas para o problema de controle supervisório com sensores aproximados.

Entretanto, para classe de sensores que são desativados por eventos controláveis, a utilização do modelo exato de um sensor como especificação adicional (emulação do sensor) permitiu que o problema de controle supervisório, resolvido pela abordagem modular local, também gerasse soluções

com menor complexidade associada ao cálculo de cada supervisor. Adicionalmente, a emulação é uma alternativa economicamente adequada, pois dispensa a alocação de um sensor real.

A hipótese de que apenas eventos controláveis desativam os sensores é razoável para alguns problemas, como por exemplo os que envolvem a movimentação de peças e sensores de fim de curso, já que nestes casos a movimentação costuma ser acionada pelo sistema de controle. Porém, de maneira geral, é possível que os eventos de dessensibilização não sejam controláveis. Para estes casos, a emulação dos sensores depende da utilização da abordagem com eventos forçáveis (Golaszewski e Ramadge, 1987).

No caso da emulação de sensores por eventos forçáveis, exige-se que a abordagem apresentada no presente trabalho seja aprofundada, no sentido do desenvolvimento de condições que assegurem a equivalência entre o problema de controle original e o problema de controle com sensores emulados. Alguns resultados preliminares, com o objetivo de combinar as abordagens de controle modular e de eventos forçáveis, foram apresentados.

Ressalta-se, aqui, que o modelo exato do sensor é redundante com relação às possibilidades de ativação do sensor real. Em sistemas computacionais, redundância de informação pode ser útil para a identificação de faltas, como é exemplo a ativação anormal do sensor real por interferência de um objeto estranho, não previsto na modelagem. Vê-se, desta forma, que a eliminação de um sensor real deve ser proposta com cautela.

Outro ponto pendente que exige estudo posterior é a sistematização do procedimento de redução das especificações. Neste caso, é possível que os resultados de minimização e redução de supervisores (Vaz e Wonham, 1986; Su e Wonham, 2004) possam servir como ponto de partida.

O presente trabalho carece de uma análise sobre a complexidade computacional associada aos diferentes problemas propostos. Para o caso da utilização de emulação de sensores, deseja-se também investigar se os modelos dos sensores realmente necessitam ser incluídos na verificação da modularidade local dos supervisores obtidos.

Como perspectivas de trabalhos futuros, algumas simplificações implícitas na construção do modelo dos sensores podem ser objeto de estudo mais aprofundado. Nesta direção, pode-se desejar que os eventos correspondentes à dessensibilização não pertençam ao alfabeto da linguagem que sensibiliza o sensor. Além disso, outra perspectiva de extensão da abordagem apresentada consiste em associar a desativação de um sensor a linguagens dessensibilizadoras, em vez de eventos.

É interessante notar, finalmente, que o sensor, como definido no presente trabalho, pode ser entendido como uma abstração de cadeias que possuem um significado particular. Por esta razão, o uso da abordagem hierárquica (Zhong e Wonham, 1990; Wong e Wonham, 1992; Cunha e Cury, 2002; Torrico e Cury, 2002) parece constituir uma alternativa natural para modelagem de sensores.

Referências Bibliográficas

BRANDIN, B.; WONHAM, W. M. Supervisory control of timed discrete-event systems. *IEEE Trans. Automatic Control*, 1994. volume 39, nº 2, págs. 329–342.

BRYANT, R. E. Graph based algorithms for boolean function manipulation. *IEEE Trans. Computers*, 1986. volume 35, nº 8, págs. 677–691.

CARDOSO, Janette; VALETTE, Robert. *Redes de Petri*. 1ª ed. Florianópolis, Brasil: Editora da UFSC, 1997.

CUNHA, A. E. C.; CURY, J. E. R. Hierarchically consistent controlled discrete event systems. In: *Proc. 15th IFAC World Congress on Automatic Control*. Barcelona, Spain, 2002. págs. 1–6.

CURY, J. E. R.; KROGH, B. H. Robustness of supervisors for discrete-event systems. *IEEE Trans. Automatic Control*, 1999. volume 44, nº 2, págs. 376–379.

CURY, J. E. R.; KROGH, B. H.; NIINOMI, T. Synthesis of supervisory controllers for hybrid systems based on approximating automata. *IEEE Trans. Automatic Control*, 1998. volume 43, nº 4, págs. 564–568.

EYZELL, J. M.; CURY, J. E. R. Symmetry in the supervisory control problem. In: *Proc. 4th Internat. Workshop on Discrete Event Systems*. Cagliari, Italy, 1998.

GOHARI, P.; WONHAM, W. M. Reduced supervisors for timed discrete-event systems. In: *Discrete Event Systems - Analysis and Control (Proc. 5th Internat. Workshop on Discrete Event Systems)*. Kluwer Academic Publishers, 2000. págs. 119–130.

GOLASZEWSKI, C. H.; RAMADGE, P. J. Control of discrete event processes with forced events. In: *Proc. 26th IEEE Conf. Decision and Control*. Los Angeles, CA, 1987. págs. 247–251.

HAJI-VALIZADEH, A.; LOPARO, K. A. Minimizing the cardinality of an events set for supervisors of discrete-event dynamical systems. *IEEE Trans. Automatic Control*, 1996. volume 41, nº 11, págs. 1579–1593.

HEYMANN, M. Concurrency and discrete event control. In: *Proc. 28th IEEE Conf. Decision and Control*. 1989. págs. 103–112.

HOPCROFT, John E.; ULLMAN, Jefferey D. *Introduction to Automata Theory, Languages, and Computation*. Reading, Massachusets, USA: Adison-Wesley Publishing Company, 1979.

ÇINLAIR, E. *Introduction to Stochastic Processes*. Englewood Cliffs, N.J., USA: Prentice-Hall, Inc., 1975.

KLEINROCK, L. *Queueing Systems*. Canada: John Wiley & Sons, 1975.

LIN, F.; WONHAM, W. M. On observability of discrete-event systems. *Information Sciences*, 1988. volume 44, nº 3, págs. 173–198.

—. Decentralized control and coordination of discrete event systems with partial observation. *IEEE Trans. Automatic Control*, 1990. volume 35, nº 12, págs. 1330–1337.

MANNA, Z.; PNUELI, A. *The Temporal Logic of Reactive and Concurrent Systems*. New York, NY, USA: Springer-Verlag, 1992.

QUEIROZ, M. H.; CURY, J. E. R. Synthesis and implementation of local modular supervisory control for a manufacturing cell. In: *Proc. 6th Internat. Workshop on Discrete Event Systems*. Zaragoza, Spain, 2002. págs. 377–382.

QUEIROZ, M. H.; CURY, J. E. R.; WONHAM, W. M. Multi-tasking supervisory control of discrete-event systems. In: *Proc. 7th Internat. Workshop on Discrete Event Systems*. Reims, France, 2004.

QUEIROZ, Max Hering de. *Controle Supervisório Modular de Sistemas de Grande Porte*. Dissertação (mestrado em engenharia elétrica), Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis, 2000.

RAMADGE, P. J. Some tractable supervisory control problems for discrete-event systems modeled by büchi automata. *IEEE Trans. Automatic Control*, 1989. volume 34, nº 1, págs. 10–19.

RAMADGE, P. J.; WONHAM, W. M. Modular feedback logic for discrete event systems. *SIAM J. Control and Optimization*, 1987a. volume 25, nº 5, págs. 1202–1218.

—. Supervisory control of a class of discrete-event processes. *SIAM J. Control and Optimization*, 1987b. volume 25, nº 1, págs. 206–230.

—. Modular supervisory control of discrete event systems. *Mathematics of control of discrete event systems*, 1988. volume 1, nº 1, págs. 13–30.

—. The control of discrete event systems. *Proc. IEEE, Special Issue on Dynamics of Discrete Event Systems*, 1989. volume 77, nº 1, págs. 81–98.

SANTOS, Eduardo Alves Portela. *Contribuições ao projeto conceitual de sistemas de manipulação e montagem automatizados*. Tese (doutorado em engenharia elétrica), Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis, 2003.

SU, R.; WONHAM, W. M. Supervisor reduction for discrete-event systems. *Discrete Event Dynamic Systems*, 2004. volume 14, nº 1, págs. 31–53.

TORRICO, C. R. C.; CURY, J. E. R. Hierarchical supervisory control of discrete event systems based on state aggregation. In: *Proc. 15th IFAC World Congress on Automatic Control*. Barcelona, Spain, 2002. págs. 1–6.

TORRICO, César Rafael Claure. *Implementação de Controle Supervisório de Sistemas a Eventos Discretos Aplicado a Processos de Manufatura*. Dissertação (mestrado em engenharia elétrica), Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis, 1999.

VAZ, A. F.; WONHAM, W. M. On supervisor reduction in discrete-event systems. *Internat. J. Control*, 1986. volume 44, nº 2, págs. 475–491.

WONG, K. C.; WONHAM, W. M. Hierarchical and modular control of discrete-event systems. In: *Proc. 30th Allerton Conference on Communication, Control and Computing*. Monticello, IL, 1992. págs. 614–623.

WONHAM, W. M. *Notes on Control of Discrete-Event Systems*. Notas de aula, Dept. of Electrical and Computer Engineering, University of Toronto, Toronto, 2001.

WONHAM, W. M.; RAMADGE, P. J. On the supremal controllable sublanguage of a given language. *SIAM J. Control and Optimization*, 1987. volume 25, nº 3, págs. 637–659.

YOO, T.; LAFORTUNE, S. NP-completeness of sensor selection problems arising in partially-observed discrete-event systems. *IEEE Trans. Automatic Control*, 2002. volume 47, nº 9, págs. 1495–1499.

YOUNG, S.; GARG, V. Optimal sensor and actuator choices for discrete event systems. In: *Proc. 31st Allerton Conference on Communication, Control and Computing*. Allerton, IL, 1993.

ZHONG, H.; WONHAM, W. M. On the consistency of hierarchical supervision in discrete-event systems. *IEEE Trans. Automatic Control*, 1990. volume 35, nº 10, págs. 1125–1134.