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

CONTRIBUIÇÕES AO CONTROLE HIERÁRQUICO DE SISTEMAS A EVENTOS DISCRETOS

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

ANTONIO EDUARDO CARRILHO DA CUNHA

Florianópolis, Abril de 2003

CONTRIBUIÇÕES AO CONTROLE HIERÁRQUICO DE SISTEMAS A EVENTOS DISCRETOS

Antonio Eduardo Carrilho da Cunha

'Esta Tese foi julgada adequada para a obtenção do título de Doutor em Engenharia Elétrica, Área de Concentração em *Automação e Sistemas*, 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.
Orientador

Geraldo Magela Pinheiro Gomes, Dr.
Co-Orientador

Edson Roberto De Pieri, Dr.
Coordenador do Programa de
Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

Fernando Antonio Campos Gomide, Dr.
UNICAMP

Rafael Santos Mendes, Dr.
UNICAMP

Guilherme Bittencourt, Dr.
UFSC

Eduardo Camponogara, Dr.
UFSC

Em nome de Deus,
o Mais Cheio de Graça,
o Mais Misericordioso.

"Se esse conhecimento pudesse ser obtido
simplesmente pelo que dizem os outros homens
não seria necessário entregar-se a tanto trabalho e esforço,
e ninguém se sacrificaria nessa busca.
Alguém vai à beira do mar e só vê água salgada, tubarões e peixes.
Ele diz: - Onde está essa pérola de que tanto falam?
Talvez não haja pérola alguma.
- Como seria possível obter a pérola simplesmente olhando o mar?
Mesmo que tivesse de esvaziar o mar cem mil vezes com uma taça,
a pérola jamais seria encontrada.
É preciso um mergulhador para encontrá-la."

Rumi, poeta persa do século XIII.

AGRADECIMENTOS

A **Deus**, nosso **Criador**, que ama e move toda sua criação, pela permissão de ter chegado até aqui.

Aos meus pais, **Antonio Carlos da Cunha** e **Sara Lucia Carrilho**, pela dedicação e amor na minha criação, que foram as bases fundamentais para o meu crescimento. Também agradeço aos meus pais o amor dedicado e o incentivo irrestrito que me foi dado durante a confecção deste trabalho.

Ao professor **José Eduardo Ribeiro Cury**, que me orientou nesta jornada que é conceber uma tese de doutorado, pela dedicação, tratamento atencioso e amizade. Foi pela orientação do professor Cury que eu aprendi *o que é* e *como fazer* um trabalho de pesquisa.

Ao professor **Geraldo Magela Pinheiro Gomes**, que é o tutor e mentor que permitiu e assegurou que esta tese acontecesse, pela total confiança, incentivo e amizade durante toda a confecção deste trabalho. Pela ação do professor Pinheiro que este trabalho pode acontecer enfim.

Ao professor **Bruce H. Krogh**, que orientou este trabalho durante o estágio sanduíche, pelo tratamento sempre atencioso, discussões profundas e amizade. Foi sob a orientação professor Krogh que eu aprendi as "regras do jogo"de se fazer pesquisa.

Ao senhores membros da banca examinadora, professores **Fernando Gomide**, **Rafael dos Santos Mendes**, **Guilherme Bittencourt** e **Eduardo Campanogara**, pelas observações sutis no momento oportuno, que contribuíram para o aperfeiçoamento deste trabalho.

Ao senhor **Omar Ali Shah**, grande amigo, pelo esclarecimento e orientação que foram-me dadas nas horas de necessidade. Agradeço muito a amizade do Sr. Omar Ali Shah.

Aos meus familiares, **Georgete Tauil**, **Bayron Nobre Filho**, **Gertrudes Tauil**, **Eduardo Tauil** (*in memoriam*), e **Georgia Tauil Nobre**, pelo amor e incentivo irrestritos a mim dedicados durante a confecção deste trabalho.

Ao Capitão **Carlos Eduardo da Mota Goes**, que foi o portador da notícia de que havia o oferecimento de uma vaga para um curso de Mestrado em Robótica na Universidade de São Carlos. Era um final de expediente em julho de 1997, ainda não sabia se ia fazer pós-graduação ou não, e eu estava então saindo de férias. A Robótica virou Automação, o Mestrado virou Doutorado, e São Carlos virou Santa Catarina, e por isso agradeço a *presença de espírito* do Capitão Goes por ter me chamado naquele momento para me passar a notícia.

Ao Capitão **Luis Paulo Gomes Ribeiro**, que me recebeu quando cheguei a Florianópolis para o início da pós-graduação, a quem devo muitas conversas inspiradoras e a quem dou ao crédito por despertar o meu interesse pelos sistemas a eventos discretos.

Aos professores da pós-graduação de quem fui aluno, professores **Jean-Marie Farines**, **Eugênio de Bona Castelan**, **Edson de Pieri**, **Alexandre Trofino Neto**, **Raul Guenter** e **Marcelo Stemmer**, por muito terem me ensinado e preparado para a jornada que estava por vir.

Aos colegas da pós-graduação, que são muitos, mas que vou nomear alguns, **Karina Acosta Barbosa**, **Jerusa Marchi Vaz**, **Alexandre Keller Albalustro**, **Julio Feller Golim**, **Carlos Cezar Bier** e **Daniel Martins**, por formarem essa "família"que existe no DAS, a se incentivar e apoiar.

Aos amigos do **Projeto Vida**, também colegas da pós-graduação, **Fernando Passold**, **Karen Farfan Campana**, **Sonia Palomino Bean**, **Dale Bean** e **Fábio Benevenutti**, pela amizade e companheirismo que temos até hoje.

Ao grande amigo **Marcos Banheti Rabelo Vallim**, também colega da pós-graduação, mas que acompanhou um pouco mais de perto a confecção deste trabalho. Muitas idéias e esclarecimentos surgiram para os nossos trabalhos nas nossas caminhadas ao entardecer na Avenida Beira Mar Norte. Ao Marcos agradeço uma amizade de irmão que surgiu durante o período em que vivi em Florianópolis.

Aos colegas do grupo de controle de sistemas a eventos discretos e sistemas híbridos sob a orientação do professor Cury, nomeando alguns, **César Claure Torrico**, **Max Hering de Queiróz**, **Tatiana Renata Garcia** e **André Bittencourt Leal**, pelo incentivo, conversas inspiradoras e apoio mútuo no desvendar deste universo que são os sistemas a eventos discretos.

Aos colegas de pós-graduação sob orientação do professor Krogh, a nomear alguns, **Ansgar Fehnker**, **Jim Kapinski**, **Paul Hubbard**, **Olaf Stursberg**, **Zhi Han** e **Collin Spencer**, pelo clima incentivo mútuo e discussões inspiradoras e instigantes que tive a oportunidade de ter no período em que convivemos.

Aos amigos da Tradição Sufi em Santa Catarina, a nomear alguns, **Oswaldo Micheluzzi**, **Virgo Micheluzzi**, **Marisa**, **Cleusa**, **Margarida Baird**, **José Faleiro**, **Giselle Guilhon de Camargo**, **Graça Fontes**, **Janete Correia**, **Mauricio Muller**, **Maria Luiza**, **Inês** e **João Guilherme**, por terem sido para mim uma segunda família em Florianópolis, com quem eu podia conversar, me aconselhar e ser recebido com carinho e amizade.

Aos amigos da Tradição Sufi no Rio de Janeiro, em particular, **Vitória**, **Heloisa**, **Carlos** e **Salomão Bernstein**, pelo incentivo, mesmo que à distância, neste período em que fiz a pós-graduação.

À senhora **Leni Saboya** e família, que me acolheram em sua residência no período final em que vivi em Florianópolis e de quem ouvi muitas palavras de incentivo.

Ao **Instituto Militar de Engenharia (IME)**, minha escola de formação como engenheiro, a qual me sinto endividado pela formação que recebi e que me permitiu galgar degraus mais altos como este do doutorado.

À **SD/1** do IME, em particular aos chefes que conheci, o coronel **Ronzani** e a professora **Wilma**, ao major **Marcílio** e à dona **Sônia**, pelo apoio prestado em todas as fases deste trabalho.

À **Universidade Federal de Santa Catarina (UFSC)**, escola que me acolheu para realizar a pós-graduação, e local onde muito aprendi e que guardo no meu coração.

Ao **Programa de Pós Graduação em Engenharia Elétrica** da UFSC, pelo acolhimento e cuidado nos assuntos administrativos da pós graduação. Particularmente agradeço ao apoio dos senhores **Wilson** e **Marcos**, da secretaria, pelo tratamento sempre atencioso ao cuidar dos assuntos dos alunos.

À **Universidade Carnegie Mellon (CMU)**, escola singular que me acolheu durante o estágio sanduíche, onde tive oportunidade de muito aprender no convívio com *mentes brilhantes*.

Ao **CNPq** pelo apoio financeiro à pesquisa.

Ao **Comando da 14^{a} Brigada de Infantaria Motorizada**, pelo tratamento atencioso durante todo período que estive adido a esta unidade situada no coração de Florianópolis.

À **Aditância do Exército Brasileiro em Washington**, pelo tratamento atencioso durante todo o período em que estive adido à esta singular unidade.

Ao **Exército Brasileiro** por pelo apoio e a confiança em mim depositados.

A todos aqueles que de alguma forma contribuíram para a confecção e conclusão desta tese e que, por esquecimento meu, não estão nomeados aqui.

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

CONTRIBUIÇÕES AO CONTROLE HIERÁRQUICO DE SISTEMAS A EVENTOS DISCRETOS

Antonio Eduardo Carrilho da Cunha

Abril / 2003

Orientador: José Eduardo Ribeiro Cury, Dr.
Co-Orientador: Geraldo Magela Pinheiro Gomes, Dr.
Área de Concentração: Controle, Automação e Informática Industrial
Palavras-chave: **Sistemas a Eventos Discretos, Controle Supervisório, Automação e Robótica**.
Número de Páginas: 272

Este trabalho traz uma série de contribuições à teoria de controle supervisório de Sistemas a Eventos Discretos (SEDs), particularmente ao controle hierárquico. No controle hierárquico de SEDs, a arquitetura de controle é decomposta em níveis hierarquicamente relacionados, cada um tratando de um aspecto distinto do problema de controle. Consistência hierárquica entre todos os pares de níveis consecutivos é o requisito fundamental para o funcionamento correto de uma hierarquia de controle. Apresenta-se uma nova classe de SEDs dotados de controle, chamados SEDs com marcação flexível, com o objetivo de se representar adequadamente os aspectos que surgem quando da abstração de um SED para o controle hierárquico. Propõe-se uma nova abordagem para o controle hierárquico de SEDs, baseada nos SEDs com marcação flexível, onde garante-se consistência hierárquica por construção e sem levar em conta nenhuma das condições impostas pelas outras abordagens da literatura. Na direção do tratamento de sistemas de maior porte e realistas, propõe-se um método fundamentado no raciocínio supor-garantir para construção da hierarquia de controle para SEDs compostos, que são sistemas formados por diversos subsistemas componentes. Uma ferramenta foi desenvolvida para implementação computacional dos modelos e algoritmos propostos. Aplicações baseadas em sistemas de controle reais são apresentadas: uma célula flexível de manufatura e um sistema de piloto automático para um automóvel.

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

CONTRIBUTIONS TO THE HIERARCHICAL CONTROL OF DISCRETE EVENT SYSTEMS

Antonio Eduardo Carrilho da Cunha

April / 2003

Advisor: José Eduardo Ribeiro Cury, Dr.
Co-advisor: Geraldo Magela Pinheiro Gomes, Dr.
Area of Concentration: Control, Automation and Industrial Computing
Keywords: **Discrete-Event Systems, Supervisory Control, Automation and Robotics**.
Number of Pages: 272

This work consists of a series of contributions for the supervisory control theory for Discrete Event Systems (DESs), particularly to the hierarchical control. In the hierarchical control of DESs, the control architecture is decomposed into hierarchically related levels, each one treating a distinct aspect of the control problem. Hierarchical consistency between every pair of consecutive levels is the fundamental requisite for the correct functionality of a control hierarchy. A new class of DES endowed with control, called DES with flexible marking, is presented with the objective to represent adequately the issues that appear in the abstraction of a DES for hierarchical control. A new approach for hierarchical control of DES is proposed, based on the DES with flexible marking, where hierarchical consistency is achieved by construction and considering none of the conditions prescribed by the other approaches of the literature. Towards the treatment of large-scale and realistic systems, a method based on the assume-guarantee reasoning is proposed for the construction of a control hierarchy of composed DES, that are systems formed by various component subsystems. A tool was developed for the computational implementation of the models and algorithms proposed. Applications to real-world control systems are presented: a flexible manufacturing system and an automatic cruise controller for an automobile.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviaturas

ASCII	*American Standard Code for Interchange of Information* (Código Padrão Americano para Intercâmbio de Informações)
CLP	Controlador Lógico-Programável
e.r.a.	em relação a
FL	*Finite Language* (Linguagem Finita)
FM	*Finite Machine* (Máquina de Estados Finitos)
OBDD	*Ordered Binary Decision Diagram* (Diagrama de Decisão Binário Ordenado)
PAAC	Piloto Automático Adaptativo Cooperativo
PCS	Problema de Controle Supervisório
RAM	*Random Access Memory* (Memória de Acesso Aleatório)
RE	*Regular Expression* (Expressão Regular)
RW	Ramadge-Wonham
SED	Sistema a Eventos Discretos
TCS	Teoria de Controle Supervisório

Lista de Símbolos

Conjuntos e funções

A, B, X, Y	conjuntos
$\mathbb{N}$	conjunto dos números naturais
x, y	elementos
$x \in A$	x pertence a A
$x \notin A$	x não pertence a A
$\|A\|$	número de elementos de A
$\emptyset$ ou $\{\}$	conjunto vazio
$A \subseteq B$	A está contido em B
$A \subset B$	A está contido propriamente em B
$A \supseteq B$	A contém B
$A \supset B$	A contém propriamente B
$A = B$	A é igual a B
$A \cap B$	intersecção de A com B
$A \cup B$	união de A com B
$A - B$	diferença entre A e B
A^c	complemento de A
2^A	conjunto das partes de A
$A \times B$	produto cartesiano de A com B
$f : A \to B$	função f de A em B
$f(x)$	valor que a função f toma em $x \in A$
$f^{-1} : B \to A$	função inversa de f, para o caso de f bijetiva

$f(X)$	imagem do conjunto $X \subseteq A$ pela função f
$f^{-1}(Y)$	imagem inversa do conjunto $Y \subseteq B$ pela função f
$f^{-1}(y)$	no caso de f possuir inversa, o valor de f^{-1} em $y \in B$, caso contrário, imagem inversa do conjunto $\{y\} \subseteq B$

Linguagens formais

Σ	alfabeto
σ	símbolo
$\sigma \in \Sigma$	símbolo em Σ
s, t	palavras
$\|s\|$	comprimento da palavra s
ϵ	palavra vazia
$s \in \Sigma^*$	palavra sobre Σ
Σ^+	conjunto de todas as palavras de comprimento finito e não nulo sobre Σ
Σ^*	conjunto de todas as palavras de comprimento finito sobre Σ
st	concatenação das palavras s e t
$s \leq t$	s é prefixo de t
$s < t$	s é prefixo estrito de t
L, K	linguagens
$L \subseteq \Sigma^*$	linguagem sobre Σ
$\overline{L}$	prefixo-fechamento da linguagem L
KL	concatenação das linguagens K e L
L^*	fechamento Kleene da linguagem L
$\Sigma_L(s)$	conjunto ativo dos símbolos em L após $s \in \overline{L}$
L/s	linguagem em L após $s \in \Sigma^*$
$p_i : \Sigma^* \to \Sigma_i^*$	projeção de palavras sobre o alfabeto Σ em palavras sobre $\Sigma_i \subseteq \Sigma$
$p_i(s)$	projeção da palavra $s \in \Sigma^*$ em Σ_i^*
$p_i(L)$	imagem da linguagem $L \subseteq \Sigma^*$ pela projeção p_i
$p_i^{-1}(L_i)$	imagem inversa da linguagem $L_i \subseteq \Sigma_i^*$ pela projeção p_i

$p_i^{-1}(t)$	imagem inversa de $\{t\} \subset \Sigma_i^*$ pela projeção p_i
	(também se denomina imagem inversa de t)
$K \| L$	produto síncrono das linguagens K e L
r, u	expressões regulares
$r + u$	expressão regular representando a união de duas linguagens
ru	expressão regular representando a concatenação de duas linguagens
r^*	expressão regular representando o fechamento Kleene de uma linguagem

Autômatos

G, G'	autômatos (deterministas de estados finitos)
$(\Sigma, Q, \delta, q_0, Q_m)$	representação de um autômato G
Σ	alfabeto de G
$\delta : Q \times \Sigma \to Q$	função de transição de G
Q	conjunto de estados de G
q_0	estado inicial de G
Q_m	conjunto de estados marcados de G
(q, σ, q')	transição com estado de origem q, estado de destino q' e
	etiqueta de evento σ
$\Sigma_G(q)$	conjunto ativo de símbolos em G no estado $q \in Q$
$\hat{\delta} : Q \times \Sigma^* \to Q$	extensão da função δ para palavras em Σ^*
L_G	linguagem gerada pelo autômato G
$L_{G,m}$	linguagem marcada pelo autômato G
$Ac(G)$	componente acessível do autômato G
$CoAc(G)$	componente co-acessível do autômato G
$trim(G)$	componente *trim* do autômato G
$proj(G, \Sigma_r)$	autômato projeção do autômato G no alfabeto Σ_r
G^D	equivalente determinista do autômato G
G_{min}	equivalente mínimo do autômato G
$G \| G'$	composição síncrona dos autômatos G e G'

M	autômato de Moore
$(\Sigma, Q, \delta, q_0, Q_m, T, w)$	representação de um autômato de Moore M
Σ, Q, δ, q_0 e Q_m	possuem mesmo significado que no autômato normal
T	alfabeto de saída do autômato de Moore M
$w : Q \to T$	função de saída para estados do autômato de Moore M

Controle Supervisório – Abordagem RW

S	sistemas a eventos discretos dotado de controle (planta)
Σ	conjunto de eventos ou alfabeto de S
$(L_S, L_{S,m}, \Gamma_S)$	representação do SED S em termos de linguagens
L_S	linguagem gerada de S
$L_{S,m}$	linguagem marcada de S
Γ_S	estrutura de controle de S (conjunto de entradas de controle)
$\gamma \in \Gamma_S$	entrada de controle para o SED S
Σ_c	conjunto de eventos controláveis de S
Σ_{nc}	conjunto de eventos não controláveis de S
(G, Γ_S)	representação em estados do SED S
G	autômato tal que $L_G = L_S$ e $L_{G,m} = L_{S,m}$
Γ_S	estrutura de controle de S (como antes)
(f, L_m)	supervisor (marcador) para S
$f : L_S \to \Gamma_S$	função de habilitação de eventos do supervisor
L_m	linguagem com as palavras que o supervisor deseja marcar em malha fechada
f/S	sistema em malha fechada da planta S com supervisor cuja função de habilitação é f
$L_{f/S}$	linguagem gerada em malha fechada
$L_{f/S,m}$	linguagem marcada em malha fechada
$K \subseteq L_S$	linguagem-alvo
$L_{f/S,m} = K$	diz-se que a linguagem K é realizável por supervisor para S

$C_{(L_S,\Sigma_{nc})}(K)$	conjunto das linguagens controláveis em relação a (e.r.a.) L_S e Σ_{nc} contidas em K
$\sup C_{(L_S,\Sigma_{nc})}(K)$	máxima linguagem controlável e.r.a. L_S e Σ_{nc} contida em K

Controle hierárquico de SEDs, definições básicas

S_{op}	SED do operador
S_{ge}	SED do gerente
Inf_{og}	canal de informação do operador para o gerente
Inf_{ge}	canal de informação do gerente
Con_{ge}	canal de controle do gerente
Com_{go}	canal de comando do gerente para o operador
Inf_{op}	canal de informação do operador
Con_{op}	canal de controle do operador
f_{ge}	supervisor para o gerente
f_{op}	supervisor para o operador
f_{ge}/S_{ge}	malha fechada para o gerente
f_{op}/S_{op}	malha fechada para o operador
$Inf_{og}(S_{op})$	imagem do comportamento do operador pelo canal de informação (igual a S_{ge})
$Inf_{og}(f_{op}/S_{op})$	imagem do comportamento em malha fechada do operador pelo canal de informação
Σ	alfabeto do operador
T	alfabeto do gerente
$\theta : L_{S_{op}} \rightarrow T^*$	mapa repórter
$\theta(t)$	θ-imagem de $t \in L_{S_{op}}$
$\theta(E_{op})$	θ-imagem da linguagem $E_{op} \subseteq L_{S_{op}}$
$\theta^{-1}(E_{ge})$	θ-imagem inversa da linguagem $E_{ge} \subseteq L_{S_{ge}}$
$\theta^{-1}(t)$	θ-imagem inversa da palavra $t \in L_{S_{ge}}$ (denota $\theta^{-1}(\{t\})$)
τ_0	evento silencioso

G_{op}	autômato de Moore representando o comportamento do sistema do operador mais o canal de informação
$w : L_{S_{op}} \rightarrow T \cup \{\tau_0\}$	mapa vocal
$L_{S_{op},voc}$	linguagem vocal de S_{op}
$s \in L_{S_{op},voc}$	palavra vocal de S_{op}
$s \in L_{S_{op}} - L_{S_{op},voc}$	palavra silenciosa de S_{op}
$\mathcal{X}_\tau$	conjunto de trechos silenciosos correspondentes ao evento τ
$\Sigma_{S_{op},voc}(s)$	conjunto ativo de eventos gerenciais em S_{op} após s (conjunto ativo de eventos relevantes em S_{op} após s)
$\theta_{voc}^{-1}(t)$	conjunto das palavras vocais cuja θ-imagem é t.
$L_{S_{op},voc}(s)$	conjunto de palavras não vazias que, concatenadas com $s \in L_{S_{op}}$, formam palavras vocais de S_{op} e cujos prefixos estritos, quando concatenados com s, formam palavras silenciosas de S_{op}

SED controlado de Pu (2000)

S	SED controlado
Σ	alfabeto de S
$(L_S, L_{S,m}, \mathcal{C}_S)$	representação do SED controlado S
L_S	linguagem gerada pelo SED controlado S
$L_{S,m}$	linguagem marcada pelo SED controlado S
$\mathcal{C}_S : L_S \rightarrow 2^\Sigma$	função de controle para o SED controlado S
$\gamma \in \mathcal{C}_S(s)$	opção de controle para a palavra $s \in L_S$
$K \subseteq L_S$	linguagem-alvo
$C_{(L_S,L_{S,m},\mathcal{C}_S)}(K)$	conjunto das linguagens $L_{S,m}$-fechadas e controláveis em relação a (e.r.a.) L_S e $\mathcal{C}_S$ contidas na linguagem K
$\sup C_{(L_S,L_{S,m},\mathcal{C}_S)}(K)$	máxima linguagem $L_{S,m}$-fechada e controlável e.r.a. L_S e $\mathcal{C}_S$ contida na linguagem K

SEDs com marcação flexível

S	SED com marcação flexível
Σ	alfabeto do SED com marcação flexível S
(L_S, Γ_S)	representação do SED com marcação flexível S por linguagens
L_S	linguagem gerada de S
Γ_S	estrutura de controle de S
$\Gamma_S(s)$	conjunto de controles da palavra $s \in L_S$
$(\gamma, \#) \in \Gamma_S(s)$	controle para a palavra $s \in L_S$
γ	conjunto de eventos habilitados do controle $(\gamma, \#)$
$\#$	indicador de marcação do controle $(\gamma, \#)$
$\# = M$	considera-se a palavra marcada
$\# = N$	considera-se a palavra não marcada
$\leq$	ordem parcial para o conjunto $\{M, N\}$
$\wedge$	*e* lógico para o conjunto $\{M, N\}$
$\vee$	*ou* lógico para o conjunto $\{M, N\}$
(G, Γ)	representação em estados do SED com marcação flexível S
G	autômato tal que $L_G = L_S$ e $L_{G,m} = \emptyset$
Γ	estrutura de controle dependente do estado
f	supervisor para o SED com marcação flexível S
f/S	sistema em malha fechada do SED com marcação flexível S com o supervisor f
$L_{f/S}$	linguagem gerada em malha fechada
$L_{f/S,m}$	linguagem marcada em malha fechada
$K \subseteq L_S$	linguagem-alvo
$K = L_{f/S,m}$	diz-se que K é realizável por supervisor para S
$C_{(L_S,\Gamma_S)}(K)$	conjunto das linguagens (L_S, Γ_S)-compatíveis contidas em K
$\sup C_{(L_S,\Gamma_S)}(K)$	máxima linguagem (L_S, Γ_S)-compatível contida em K
$\Omega : 2^{L_S} \rightarrow 2^{L_S}$	operador Ω para cálculo da máxima linguagem (L_S, Γ_S)-compatível contida numa dada linguagem

Nova abordagem para o controle hierárquico de SEDs

S	sistema do operador
P	sistema do gerente
Σ_r	conjunto de eventos relevantes do operador (alfabeto do gerente)
$pre_{voc}(s)$	prefixo vocal da palavra s
$S(s)$	subsistema para a palavra s em S
$\Gamma_{S,voc}(s)$	conjunto de controles vocais de s em S
$(\gamma_r, \#_r) \in \Gamma_P(t)$	controle para o gerente na palavra $t \in L_P$
$(\gamma_r, \#_r)^{\uparrow} \in \Gamma_{S,voc}(s)$	correspondente controle vocal para o operador (oculta-se a palavra correspondente do operador $s \in \theta_{voc}^{-1}(t)$)
$E_s(\gamma_r, \#_r)$	linguagem de especificação para implementação do controle $(\gamma_r, \#_r)$ no subsistema de s
$E_s(\gamma_r, \#_r)^{\uparrow}$	linguagem que implementa o controle $(\gamma_r, \#_r)$ no subsistema de s

Raciocínio supor-garantir

S, S'	SEDs com marcação flexível
$S \| S'$	produto síncrono dos SEDs S e S'
$S \leq S'$	S está contido em S'
$S \cup S'$	união dos SEDs S e S'
$\langle S, \Sigma_r \rangle$	máxima abstração consistente de S em relação a (e.r.a.) Σ_r
$[S, \Sigma_r]$	máxima abstração confiável de S em relação a (e.r.a.) Σ_r

Algoritmos

$\mathcal{O}(\cdot)$	complexidade assintótica de um algoritmo em tempo e espaço
$\|f(n)\|$	tempo ou espaço necessário para o cálculo da função (do algoritmo) $f(n)$ em função do parâmetro n

Capítulo 1

Introdução

Este trabalho traz uma série de contribuições à teoria de controle supervisório de Sistemas a Eventos Discretos, particularmente ao controle hierárquico.

Divide-se a exposição deste capítulo nas seguintes seções. Na seção 1.1 faz-se uma descrição do objeto de estudo. Na seção 1.2 descreve-se o desenvolvimento da tese. Na seção 1.3 resumem-se as principais contribuições. Por fim, na seção 1.4 apresenta-se a forma em que foi organizado o texto.

1.1 Objeto de estudo

As últimas décadas têm testemunhado uma presença cada vez maior de sistemas automatizados e informatizados em todos os ramos da atividade humana. Enumeram-se alguns exemplos de tais sistemas. Na fabricação, encontram-se os sistemas de manufatura flexível. Na automobilística, os sistemas de piloto automático e de prevenção de colisão. Nos serviços bancários, todo o sistema que cuida das transações *on line*, terminando nos caixas automáticos e os serviços disponíveis via *internet*. Na telefonia, os diversos serviços adicionais oferecidos pelas centrais digitais, tais como o *siga-me* ou o *chamada em espera*, entre outros. Na aviação, o controle de tráfego aéreo no pouso e decolagem de aviões. E, nas aplicações militares, os diversos tipos de veículos autônomos não tripulados e os robôs caça-minas, por exemplo. O projetista de tais sis-

temas deve garantir que estes atendam a requisitos do tipo confiabilidade, desempenho, segurança e resposta em tempo real.

Muitas das aplicações citadas acima possuem em comum a forma com que interagem com o ambiente a sua volta para tomar decisões de comportamento, que se dá pela recepção ou envio de estímulos, chamados *eventos*. Como exemplo, considere os eventos que marcam o início e o fim de uma tarefa por uma máquina num sistema de manufatura, ou o evento que marca a quebra da máquina durante a operação. Os *Sistemas a Eventos Discretos* (SEDs) são sistemas dinâmicos com espaço de estados discreto que evoluem com a ocorrência abrupta e instantânea de eventos (Cury 2001). No capítulo 2 de (Cury 2001) encontra-se uma descrição extensa de exemplos de aplicações em que SEDs são encontrados.

A natureza discreta dos SEDs faz com que os modelos matemáticos empregados sejam diferentes dos modelos matemáticos para os sistemas dinâmicos contínuos em tempo contínuo e em tempo discreto, estes fortemente fundamentados nas equações diferenciais e a diferenças, respectivamente. Existem diversos modelos para os SEDs, como, por exemplo, as redes de Petri, as cadeias de Markov, a álgebra de processos, a álgebra *max-plus*, e a teoria de linguagens e autômatos, entre outros (Cury 2001). Na verdade, não há um consenso de modelo descritivo para SEDs como são as equações diferenciais para os sistemas contínuos.

Dentre as abordagens para SEDs, a teoria de controle supervisório (TCS), iniciada por Ramadge e Wonham (1989), também conhecida por *abordagem Ramadge-Wonham (RW)*, trata da síntese de controladores para SEDs. Os modelos na abordagem RW fundamentam-se na teoria de linguagens e autômatos. Na abordagem RW, faz-se uma distinção clara entre o sistema a ser controlado, denominado *planta*, e o agente de controle, denominado *supervisor*. A planta reflete o comportamento fisicamente possível do sistema a ser controlado, isto é, todas as possibilidades sem aplicação de controle, prevendo, inclusive, a ocorrência de seqüências de eventos que levem a situações indesejadas. O papel do supervisor é observar a seqüência de eventos gerada pela planta e, com base nesta observação, exercer controle sobre a planta na forma de proibição da

ocorrência de alguns eventos. O que a abordagem RW fornece é um método de síntese de tais supervisores, dados os modelos da planta e a lista de especificações de comportamentos desejados. Algumas referências básicas para a abordagem RW são, além de Cury (2001), Ramadge e Wonham (1987b), Wonham e Ramadge (1987), Ramadge e Wonham (1989), Kumar e Garg (1995), Cassandras e Lafortune (1999) e Wonham (2002b).

Muito embora a complexidade dos algoritmos de síntese na abordagem RW seja polinomial em relação ao número de estados dos sistemas de transição que representam a planta e as especificações, o número de estados da planta e das especificações varia exponencialmente com o número de seus componentes (Ramadge e Wonham 1989). Assim, um problema fundamental da abordagem RW é a explosão combinatória dos estados dos modelos em função do número de componentes. Esta explosão combinatória de estados torna a complexidade dos algoritmos implicitamente exponencial em relação ao número de componentes da planta e das especificações.

A explosão combinatória dos estados também acarreta o crescimento exponencial, em relação ao número de componentes da planta e das especificações, do número de estados do supervisor sintetizado. Isso é problemático quando se considera que a lógica de controle resultante do supervisor destina-se a ser implementada em algum sistema de controle discreto, seja um computador de processo, o processador de um sistema embutido[1], ou um controlador lógico-programável (CLP). O grande número de estados dos supervisores gerados pelos algoritmos de síntese da abordagem RW para sistemas reais torna tais supervisores, na prática, *ilegíveis*, no sentido de impossibilitar o entendimento da lógica de controle por inspeção visual da listagem (ou gráfico) do supervisor. Somado a isso os algoritmos conhecidos para minimizar o número de estados de um supervisor são de complexidade exponencial em relação ao número de estados do mesmo (Vaz e Wonham 1986, Ramadge e Wonham 1989). A ilegibilidade dos supervisores é uma das causas da dificuldade de aceitação da abordagem RW na engenharia de projeto de sistemas de controle discreto.

[1] *Embedded system*, em inglês.

A decomposição estrutural da planta e das especificações tem sido um ponto explorado na literatura para o problema da explosão combinatória de estados na abordagem RW. No sentido da decomposição horizontal do sistema estão as abordagens de controle modular e descentralizado, tais como os trabalhos de Ramadge e Wonham (1987a), Rudie e Wonham (1992), e no recente desenvolvimento em (de Queiroz e Cury 2002a). Também como uma forma de decomposição estrutural encontra-se a exploração da simetria da arquitetura para redução da complexidade, objeto do trabalho de Eyzell e Cury (2001).

O *controle supervisório hierárquico de SEDs* explora a decomposição vertical da arquitetura do sistema. Por controle hierárquico entende-se a subdivisão de um problema de controle complexo em problemas associados a níveis de abstração diferentes. O conceito de hierarquia de controle faz-se presente em diversas organizações humanas, como por exemplo as militares. Mesarovic et al. (1970) apontam que a principal propriedade de uma estrutura hierárquica de controle é que o modelo de controle disponível a qualquer nível da hierarquia possa ser utilizado com a garantia de que o nível imediatamente inferior vai responder como desejado ou esperado, propriedade esta denominada *consistência hierárquica*. Numa hierarquia militar, a consistência hierárquica concretiza-se no fato de que um comandante deve conhecer a capacidade dos seus subordinados para que possa dar as ordens.

Zhong e Wonham (1990) introduzem o controle hierárquico na abordagem RW. Trata-se de um esquema de supervisão hierárquica de dois níveis numa metáfora de uma fábrica: o nível superior associado a um *gerente*, e o inferior, a um *operador*. O comportamento dos sistemas é descrito por linguagens prefixo-fechadas, e ambos os sistemas possuem estrutura de controle segundo a abordagem RW, definida pelo particionamento do alfabeto do sistema em subconjuntos controláveis e não controláveis. Formaliza-se o conceito de consistência hierárquica dentro da abordagem RW e a apresenta-se condição chamada *consistência de controle estrita* sobre o SED do operador para assegurar a consistência hierárquica.

Wong e Wonham (1996a) dão seqüência ao trabalho de Zhong e Wonham (1990),

agora tratando sistemas com comportamento descrito por linguagens marcadas. Neste contexto, surge um problema de bloqueio para o supervisor do operador no esquema de supervisão hierárquica de dois níveis (Wong e Wonham 1996a). Para tratamento desse problema, Wong e Wonham (1996a) introduzem as condições de *mapa repórter observador* e *consistência de marcação*. No trabalho de Wong e Wonham (1996a) surgiem de estruturas de controle não convencionais para modelar o gerente, sendo apresentado um formalismo algébrico para tais estruturas de controle. O mapa repórter observador, surgido inicialmente no contexto do controle hierárquico tem importância evidenciada em trabalhos mais recentes, como em (Wong e Wonham 1998), (Wong 1998) e (Wong et al. 2000). A construção de um mapa repórter observador é objeto dos trabalhos de Wong (1994), Guan (1997) e Wong e Wonham (2000).

A proposta de estruturas de controle não convencionais de Wong e Wonham (1996a) é puramente algébrica, faltando uma contrapartida concreta. Pu (2000) propõe um modelo generalizado para SEDs dotados de controle, que vem a ser suporte para uma proposta de controle hierárquico baseada em *abstrações consistentes* de SEDs.

Paralelamente, Hubbard e Caines (2002) apresentam uma abordagem para o controle hierárquico por agregação de estados. O ponto de partida é o trabalho de Caines e Wei (1995), que trata de agregação de estados e alcançabilidade para SEDs. O modelo dos SEDs usado por Hubbard e Caines (2002) corresponde ao da abordagem RW padrão e resultados semelhantes aos de Wong e Wonham (1996a) são obtidos para assegurar a consistência hierárquica num esquema de supervisão hierárquica de dois níveis.

Todas as abordagens anteriormente citadas para o controle hierárquico de SEDs correspondem a modelagens do tipo *de baixo para cima* (*bottom-up*), pois modelam os sistemas a partir de um nível mais detalhado para chegar em modelos simplificados e abstratos. Em contrapartida, existem também abordagens do tipo *de cima para baixo* (*top-down*), isto é, de modelos simples e abstratos a modelos detalhados. Dentro das abordagens *de cima para baixo*, Brave e Heymann (1993) tratam do controle de máquinas de estados hierárquicas, baseadas nas *Statecharts* de Harel (1987). O objetivo

de Brave e Heymann (1993) é a redução da complexidade da síntese de supervisores por estratégias de busca hierarquizada nos estados. Gohari-Moghadam e Wonham (1998) apresentam uma abordagem baseada em autômatos e linguagens para as máquinas de estado hierárquicas de Brave e Heymann (1993). A modelagem *de cima para baixo* para o controle supervisório de SEDs dentro da abordagem RW é também tratada por Wang (1995) e Ma (1999). Leduc et al. (2001) aplicam à abordagem *de cima para baixo* para a verificação de controladores para sistemas de grande porte. As abordagens *de cima para baixo* para o controle hierárquico de SEDs não fazem parte dos objetos de estudo deste trabalho.

Encontram-se algumas aplicações do controle hierárquico de SEDs na literatura. O trabalho de da Cunha e Cury (2000) utiliza o controle hierárquico para redução de modelos de SEDs. Gohari e Wonham (2000) apresentam uma estratégia de redução de modelos por controle hierárquico para SEDs temporizados. Wong e Wonham (1996b) tratam o problema de controle hierárquico de SEDs temporizados usando o formalismo estabelecido por Wong e Wonham (1996a). Dois trabalhos relacionam o controle hierárquico à solução do problema de não modularidade no controle modular (Ramadge e Wonham 1989): Wong et al. (1995) propõem um coordenador hierárquico para mediar os supervisores não modulares, e Wong e Wonham (1998) propõem interfaces hierárquicas para resolverem os conflitos entre os supervisores modulares. Park e Lim (2001) apresentam uma hierarquia de dois níveis para resolver um problema de controle robusto de uma célula de manufatura. Zad et al. (1998) e Zad (1999) tratam o problema do diagnóstico de falhas de SEDs por meio de um diagnosticador, um sistema construído por abstração do sistema observado, com o objetivo de identificar seqüências geradas que definem falhas.

1.2 Desenvolvimento

O desenvolvimento desta tese se fez em quatro partes. A primeira parte foi uma pesquisa bibliográfica sobre controle hierárquico de SEDs dentro da teoria de controle

supervisório. A segunda, foi o desenvolvimento de uma nova classe de SEDs dotados de controle, que foi aplicada, na terceira parte, para se fazer uma proposta de hierarquia de dois níveis com consistência hierárquica. Por último, foi proposto um método para o controle hierárquico de SEDs compostos, isto é, SEDs formados por diversos componentes.

Na revisão bibliográfica, focalizaram-se os trabalhos de Zhong e Wonham (1990), Wong e Wonham (1996a) e Pu (2000). Os três trabalhos partem de uma abordagem que usa linguagens para definir o problema de controle hierárquico. Já o trabalho de Hubbard e Caines (2002) parte de uma perspectiva de agregação de estados para definir o problema de controle hierárquico, o que o diferencia das outras abordagens. Chega-se, em (Hubbard e Caines 2002), a resultados equivalentes aos de Zhong e Wonham (1990) e Wong e Wonham (1996a), sendo que expressos em outro formalismo. Nesta tese utiliza-se muito da nomenclatura criada por Zhong e Wonham (1990) para o controle supervisório hierárquico de SEDs como, por exemplo, a analogia de uma hierarquia de controle a uma fábrica, onde se denomina o sistema do nível superior como *sistema do gerente* e o sistema do nível inferior como *sistema do operador*.

Considera-se que, se fossem seguidos os mesmos caminhos que os trabalhos anteriores, chegar-se-ia a resultados semelhantes. Por outro lado, julga-se interessante a existência de um método para construir hierarquias de SEDs com consistência hierárquica, e além disso, que seja disponível uma ferramenta computacional. Acredita-se que isso abre caminho para que aplicações surjam e novas problemáticas sejam tratadas. Com isto em mente, começou-se a trabalhar com uma hierarquia de dois níveis.

De forma semelhante aos outros trabalhos, o principal problema encontrado foi o bloqueio no controle hierárquico. A solução que se apontava como a mais viável era a definição do sistema do gerente diferente do modelo padrão da abordagem RW. Extensões do modelo padrão da abordagem RW foram considerados por Wong e Wonham (1996a) e Pu (2000) na tentativa de resolver o problema de controle hierárquico. Entretanto mesmo aplicando os respectivos modelos de Wong e Wonham (1996a) ou Pu (2000) ainda são necessárias condições adicionais para garantir-se consistência

hierárquica. Desenvolveu-se então uma nova classe de SED dotada de estrutura de controle onde permite-se a flexibilização tanto do atributo de controle de um evento quanto da definição de uma palavra como tarefa em malha fechada. Denominou-se esta nova classe por *SEDs com marcação flexível*. Os SEDs com marcação flexível foram desenvolvidos em conjunto com outro trabalho de tese sobre controle hierárquico de SEDs sob a mesma orientação (Torrico 2003).

Definiu-se então uma hierarquia de dois níveis, onde o sistema do operador é um SED na abordagem padrão RW e o sistema do gerente é um SED com marcação flexível, ambos relacionados de uma forma a ser especificada. Mostrou-se então que a hierarquia proposta possui naturalmente consistência hierárquica. Adicionalmente, propôs-se um método de construção de tal hierarquia de SEDs. Posteriormente, os resultados foram estendidos para uma hierarquia de dois níveis onde ambos os níveis são SEDs com marcação flexível. Essa extensão faz com que os resultados sejam válidos para hierarquias com mais de dois níveis.

Tendo como aplicação o projeto de um controlador discreto para um sistema de piloto automático de um automóvel (Girard et al. 2001), explorou-se na quarta etapa da tese o controle hierárquico para SEDs compostos. O objetivo era, tendo um sistema composto para o operador e um conjunto de eventos para a coordenação, construir um SED para o gerente sem ter de se construir o SED composto do operador antes. A construção dessa forma é interessante pelo fato de que o número de estados de um SED composto cresce exponencialmente com o número de componentes (Cury 2001), e este número de estados influi na construção do SED do gerente. Tal problemática foi estudada anteriormente por Wonham e Zhong (1990) e Pu (2000), mas apenas soluções *ad hoc* foram encontradas, sem proposta de métodos construtivos. Desenvolveu-se um método de construção do SED do gerente sem necessidade da construção do SED composto do operador, baseado no raciocínio supor-garantir (Stark 1985, Alur e Henzinger 1999, Maier 2001).

Paralelamente, para tratamento dos exemplos de aplicação considerados, foi desenvolvida uma ferramenta computacional para controle hierárquico de SEDs. A

ferramenta foi construída em C++ usando a biblioteca de funções para autômatos e linguagens chamada **Grail** (Raymond e Wood 1996a). Primeiramente, foi necessário implementar na ferramenta as funções de manipulação e de controle supervisório para o SED com marcação flexível e, em seguida, implementaram-se as funções de controle hierárquico. A ferramenta foi testada para diversos exemplos da literatura e todos os resultados conferidos. A ferramenta está disponível em `http://www.das.ufsc.br/~aecc`.

1.3 Resumo das Contribuições

As principais contribuições desta tese são listadas a seguir:

- Uma revisão dos principais conceitos do controle hierárquico de SEDs que permite comparar as abordagens correlatas da literatura;
- Uma nova classe de SEDs dotados de controle, os SEDs com marcação flexível, que são adequados para modelar a abstração dos sistemas que surge no controle hierárquico;
- Um esquema de controle hierárquico baseado nos SEDs com marcação flexível, onde consistência hierárquica é garantida de forma direta sem consideração das condições impostas pelas outras abordagens da literatura;
- Um método baseado no raciocínio supor-garantir para construção do sistema do gerente quando o sistema do operador é um sistema composto, e
- A implementação de todos os métodos e técnicas propostas numa ferramenta computacional.

O SED com marcação flexível foi desenvolvido em conjunto com outro trabalho de tese de doutorado (Torrico 2003), sendo então considerado como uma contribuição conjunta surgida nas duas teses.

1.4 Organização do texto

No capítulo 2, apresentam-se conceitos introdutórios referentes à teoria de controle supervisório segundo a abordagem RW. No capítulo 3, faz-se uma revisão bibliográfica do controle hierárquico de SEDs. No capítulo 4, o material fundamental para os SEDs com marcação flexível é introduzido. No capítulo 5, descreve-se o método de controle hierárquico de SEDs baseado nos SEDs com marcação flexível. No capítulo 6, apresenta-se a extensão do método de controle hierárquico do capítulo 5 para SEDs compostos. No capítulo 7 fazem-se considerações finais sobre o trabalho. Um descritivo da ferramenta para controle hierárquico de SEDs, apresentado na forma de um manual do usuário, encontra-se no apêndice A.

Capítulo 2

O controle supervisório de SEDs

Neste capítulo apresenta-se, de forma resumida, a conceituação preliminar necessária para o trabalho.[1]

A organização deste capítulo é como segue. Na seção 2.1 faz-se uma caracterização dos sistemas a eventos discretos, ressaltando os aspectos que são foco de atenção deste trabalho. Nas três seções seguintes apresenta-se o ferramental matemático necessário ao trabalho, a saber, conceitos de conjuntos e funções na seção 2.2, conceitos de linguagens formais na seção 2.3, e conceitos de autômatos na seção 2.4. Por fim, na seção 2.5 apresentam-se os fundamentos da abordagem Ramadge-Wonham para o controle de sistemas a eventos discretos.

2.1 Caracterização dos SEDs

Esta seção apresenta uma descrição preliminar sobre a caracterização dos SEDs e seus diversos modelos, trazendo o foco de atenção para o modelo de interesse deste trabalho.

[1] A apresentação deste capítulo, em alguns pontos, é bastante concisa, visto que grande parte do material já foi detalhado em outros trabalhos relacionados, tais como Ziller (1993), Martins (1999), Torrico (1999), de Queiroz (2000), González (2000) e Cury (2001). O leitor familiarizado com o material introdutório da teoria de controle supervisório de SEDs pode optar em passar direto à leitura do próximo capítulo, a partir do qual é apresentado apenas material particular a esta tese. Para tal leitor sugere-se que use este capítulo como forma de referência aos conceitos básicos da teoria de controle supervisório à medida que estes forem surgindo nos capítulos seguintes.

Os sistemas a eventos discretos (SEDs) englobam uma grande variedade de sistemas físicos que surgiram com o desenvolvimento tecnológico, os ditos *sistemas feitos pelo homem* (Wonham 2002b). Incluem-se em tais sistemas, os sistemas flexíveis de manufatura, os sistemas de controle de tráfego, os sistemas de gerenciamento de bancos de dados, os sistemas logísticos (para armazenamento de bens ou entrega de serviços), os protocolos de comunicação, e as redes de comunicação de dados (Wonham 2002b).

Os SEDs são sistemas dinâmicos discretos, isto é, com espaço de estados discreto, que evoluem com a ocorrência abrupta de eventos físicos, em intervalos de tempo irregulares e desconhecidos (Cury 2001).

Considere a seguinte discussão sobre o comportamento de um SED (Cassandras e Lafortune 1999). Entre a ocorrência de dois eventos consecutivos, diz-se que o sistema permanece num determinado estado, e a ocorrência de um evento pode causar a transição ou mudança de estado do sistema. A evolução do sistema no tempo pode ser representada pela trajetória de estados percorrida, conforme ilustrado na figura 2.1. Na figura 2.1, além dos tempos onde ocorrem as transições, assinalam-se os eventos correspondentes. Nesta trajetória ocorrem os eventos α, β e λ. A figura 2.1 ilustra o fato de que a ocorrência de um mesmo evento pode ter efeitos diferentes no sistema, dependendo do estado onde ocorre (Cury 2001). Por exemplo, a ocorrência do evento α no estado x_1 leva o sistema ao estado x_4, já no estado x_3 leva o sistema ao estado x_1.

A trajetória de estados na figura 2.1 pode continuar indefinidamente, inclusive com a ocorrência de outros eventos, não representados na figura. Para os sistemas de interesse deste trabalho, assume-se que o conjunto de eventos possíveis é finito. Já o conjunto de estados é, em geral, representado por um conjunto discreto, não necessariamente com um número finito de elementos. Entretanto, os sistemas tratados nos procedimentos computacionais apresentados neste trabalho possuem número finito de estados. Alguns estados são de interesse particular. O estado inicial corresponde ao estado em que o sistema se encontra antes de ocorrer o primeiro evento. Há outro conjunto de estados que é de interesse particular para a teoria de controle supervisório de SEDs. Esses estados correspondem à finalização de uma tarefa pelo sistema, chamados

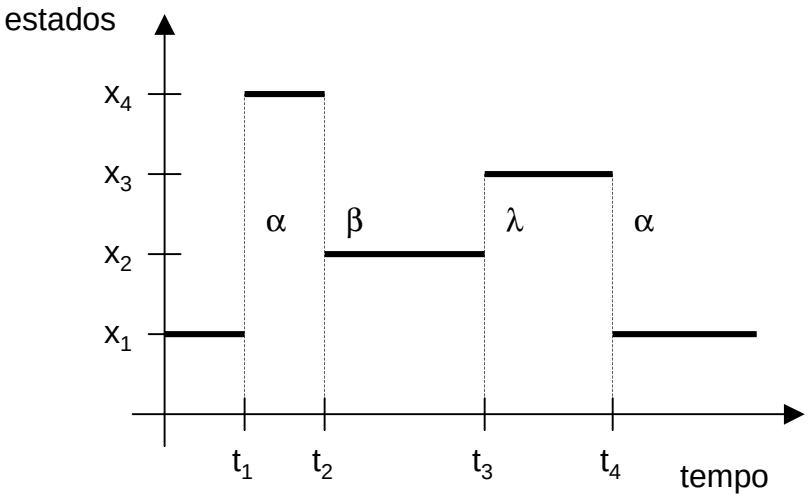


Figura 2.1: Trajetória de estados de um SED.

estados marcados. Por exemplo, num sistema de montagem de uma peça, cada estado do sistema pode corresponder a um estágio da montagem, enquanto que o estado marcado corresponde à finalização da montagem.

Os SEDs contrastam com os sistemas dinâmicos contínuos, cujo espaço de estados é não enumerável, como o conjunto dos reais. No sistema contínuo a evolução do estado do sistema é uma função do tempo, uma variável independente. Dependendo do tipo de evolução temporal de um sistema contínuo, a transição de estados é caracterizada por uma equação diferencial, no caso do tempo contínuo, ou uma equação a diferenças, no caso do tempo discreto. A figura 2.2 representa uma trajetória típica de estados de um sistema contínuo regido por uma equação diferencial (Cury 2001).

A trajetória de estados de um SED pode ser caracterizada pela seqüência de eventos correspondentes percorrida, por exemplo, $\alpha_1\alpha_2 \ldots \alpha_n$, eventualmente incluindo o tempo em que o evento ocorre, isto é, $(\alpha_1, t_1)(\alpha_2, t_2) \ldots (\alpha_n, t_n)$. A qualidade da informação depende dos objetivos de aplicação (Cury 2001).

A teoria de Sistemas a Eventos Discretos advém de diversas outras áreas, como a Teoria de Sistemas, a Ciência da Computação, os Sistemas em Tempo Real, os Sistemas de Manufatura e a Pesquisa Operacional (Boel et al. 2002). Existem diversos modelos

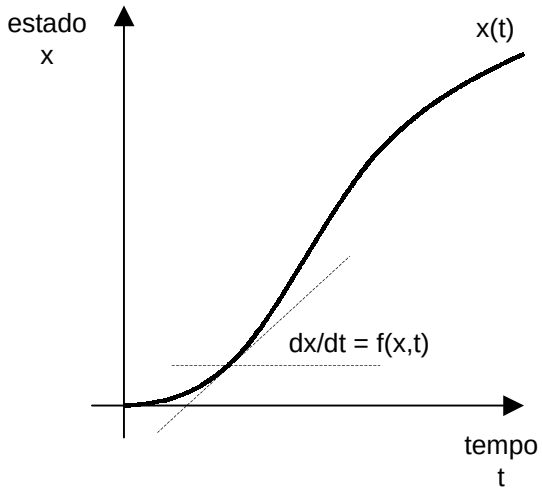


Figura 2.2: Trajetória de estados de um sistema contínuo.

para SEDs, sem que nenhum tenha se firmado como paradigma, como são as equações diferenciais e a diferenças para os sistemas contínuos. Os modelos refletem os diferentes tipos de SEDs e os diferentes objetos de estudo (Cury 2001). Alguns dos modelos utilizados são os autômatos, as linguagens formais, as redes de Petri, as cadeias de Markov, a álgebra Max-Plus, entre outros (Cury 2001). Existem diversas abordagens para SEDs, seja para o controle, a análise de desempenho, a verificação formal de propriedades entre outras. Em (Boel et al. 2002) encontra-se um panorama recente do estado da arte da pesquisa sobre SEDs. Das abordagens para os SEDs, o interesse deste trabalho é na teoria de controle supervisório (TCS), iniciada por Ramadge e Wonham (1989). A TCS é uma abordagem para controle de SEDs, sob a ótica da Teoria de Controle, que modela os sistemas por linguagens formais e autômatos.

Nas próximas três seções são apresentados os fundamentos teóricos usados na teoria de controle supervisório para SEDs, sendo, em seguida, introduzida a abordagem Ramadge-Wonham (RW) para o controle de SEDs.

2.2 Conjuntos e funções

Introduz-se nesta seção a conceituação preliminar de conjuntos e funções que é utilizada sistematicamente neste trabalho. Baseia-se esta exposição no capítulo 1 de Lima (1989), remetendo-se o leitor interessado às obras lá citadas.

Um conjunto é formado de objetos, chamados de elementos. A relação básica entre um elemento e um conjunto é a relação de pertinência. Se um elemento x pertence a um conjunto A, denota-se $x \in A$, e, caso contrário, $x \notin A$. Denota-se o número de elementos do conjunto A por $|A|$. O conjunto sem elementos é denominado conjunto vazio e denotado $\emptyset$ ou $\{\}$. Se todos os elementos de um conjunto A também são elementos de um conjunto B, então afirma-se que A está contido em B, ou que A é subconjunto de B, ou que A é parte de B, e denota-se $A \subseteq B$ (ou ainda que B contém A e $B \supseteq A$). Adicionalmente a $A \subseteq B$, se existe $y \in B$ tal que $y \notin A$, então afirma-se que A está contido propriamente em B, ou que A é subconjunto próprio de B, e denota-se $A \subset B$ (ou ainda que B contém propriamente A e $B \supset A$). Dois conjuntos são ditos iguais se e somente se possuem os mesmos elementos, ou seja, $A = B$ se e somente se $A \subseteq B$ e $B \subseteq A$. Por definição, o conjunto vazio é subconjunto de qualquer conjunto.

Sejam A e B dois conjuntos, então definem-se as operações de união, $A \cup B = \{x : x \in A \text{ ou } x \in B\}$, interseção, $A \cap B = \{x : x \in A \text{ e } x \in B\}$, e diferença, $A - B = \{x : x \in A \text{ e } x \notin B\}$. Freqüentemente, tem-se um conjunto $\mathbb{U}$, denominado conjunto universo, que contém todos os objetos que ocorrem numa certa discussão. Neste caso, a diferença $\mathbb{U} - A$ chama-se simplesmente o complemento do conjunto A, denotado A^c. O conjunto das partes de um conjunto A, ou conjunto de subconjuntos de A, denotado 2^A, é definido por $2^A = \{X : X \subseteq A\}$. O produto cartesiano de dois conjuntos A e B, denotado $A \times B$, é o conjunto de todos os pares ordenados (a, b) para os quais $a \in A$ e $b \in B$.

Uma função $f : A \to B$ consta de três partes, um conjunto A, chamado domínio da função, ou o conjunto onde a função é definida, um conjunto B, chamado contradomínio

da função, ou o conjunto onde a função toma valores, e uma regra que permite associar, de um modo bem determinado, a cada elemento $x \in A$ um único elemento $f(x) \in B$, chamado valor que a função assume em x. A natureza da regra que mostra como obter o valor $f(x) \in B$ quando é dado $x \in A$ é inteiramente arbitrária, sendo sujeita apenas a duas condições: (i) não deve haver exceções, isto é, a fim de que f tenha o conjunto A como domínio, a regra deve fornecer $f(x)$ para todo $x \in A$, e (ii) não deve haver ambigüidades, a cada $x \in A$ a regra deve fazer corresponder um único $f(x)$ em B. Uma função parcial $f : A \to B$ é uma função onde a condição (i) para a regra de atribuição de um valor $f(x) \in B$ dado $x \in A$ é relaxada, isto é, a função não é definida para todo $x \in A$.

Uma função $f : A \to B$ chama-se injetiva quando dados x, y quaisquer em A, $f(x) = f(y)$ implica $x = y$. Em outras palavras, quando $x \neq y$ em A implica $f(x) \neq f(y)$ em B. Uma função $f : A \to B$ chama-se sobrejetiva quando, para todo $y \in B$ existe pelo menos um $x \in A$ tal que $f(x) = y$. Uma função $f : A \to B$ chama-se bijetiva quando é injetiva e sobrejetiva ao mesmo tempo. Dada uma função bijetiva $f : A \to B$, define-se a função inversa $f^{-1} : B \to A$ para todo $y \in B$ por $f^{-1}(y) = x$, onde $x = f(y)$.

Dada uma função $f : A \to B$ e uma parte $X \subseteq A$, chama-se imagem de X pela função f ao conjunto $f(X)$ formado pelos valores $f(x)$ que f assume nos pontos $x \in X$. Assim, $f(X) = \{f(x) : x \in X\}$. O conjunto $f(X)$ é um subconjunto de B. Para que $f : A \to B$ seja sobrejetiva, é necessário e suficiente que $f(A) = B$. Em geral, tem-se $f(A) \subseteq B$. O conjunto $f(A)$ é chamado a imagem da função f.

Dada uma função $f : A \to B$, considere um conjunto $Y \subseteq B$. A imagem inversa de Y pela função f é o conjunto $f^{-1}(Y)$, formado por todos os $x \in A$ tais que $f(x) \in Y$. Assim, $f^{-1}(Y) = \{x \in A : f(x) \in Y\}$. Note-se que pode ocorrer $f^{-1}(Y) = \emptyset$, mesmo que $Y \subseteq B$ seja não vazio. Isso se dá quando $Y \cap f(A) = \emptyset$, isto é, quando Y não tem pontos em comum com a imagem de f. Dado $\{y\} \subseteq B$, escreve-se $f^{-1}(y)$ em vez de $f^{-1}(\{y\})$. Pode acontecer que $f^{-1}(y)$ possua mais que um elemento, pois f pode não ser injetiva. Neste trabalho, usa-se o símbolo f^{-1} para denotar a inversa de uma

função f e a imagem inversa de um conjunto por uma função f.

2.3 Linguagens formais

Esta seção apresenta o primeiro conjunto de conceitos de base para o trabalho, correspondente a alguns fundamentos da teoria de linguagens formais. Citam-se como principais referências para o aprofundamento os trabalhos de Hopcroft e Ullmann (1979), Cassandras e Lafortune (1999), Wonham (2002b), Kumar e Garg (1995), Menezes (2001) e Cury (2001).

2.3.1 Alfabetos e palavras

Um alfabeto é um conjunto finito e não vazio de símbolos. Uma palavra[2] finita sobre um alfabeto Σ é qualquer justaposição de um número finito de símbolos em Σ, na forma $\sigma_1\sigma_2\ldots\sigma_n$, com $\sigma_i \in \Sigma$, para $i \in \{1, 2, \ldots, n\}$ e $n \in \mathbb{N}$. O comprimento de uma palavra s, denotado por $|s|$, é o número de símbolos que a compõe. A palavra vazia, denotada pela letra grega ϵ, é a palavra de comprimento nulo. Define-se Σ^+ o conjunto de todas as palavras de comprimento finito e não nulo sobre o alfabeto Σ. Também define-se $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$, o conjunto Σ^+ acrescido da palavra vazia.

A concatenação de duas palavras s e s' sobre o alfabeto Σ, denotada ss', onde $s = \sigma_1\ldots\sigma_k$ e $s' = \sigma_{k+1}\ldots\sigma_{k+n}$ é definida por $ss' = \sigma_1\ldots\sigma_k\sigma_{k+1}\ldots\sigma_{k+n}$. A palavra vazia é o elemento identidade da concatenação de palavras, isto é, $s\epsilon = \epsilon s = s$ para qualquer palavra s. Um prefixo de uma palavra $s \in \Sigma^*$ é qualquer palavra $t \in \Sigma^*$ tal que, concatenada com outra palavra $u \in \Sigma^*$, forma a palavra s, isto é, $s = tu$. Denota-se que t é prefixo de s por $t \leq s$. Dada a palavra s, a palavra vazia e a própria palavra s são prefixos de s, uma vez que a condição $s\epsilon = \epsilon s = s$ é sempre satisfeita. Um prefixo estrito da palavra s é qualquer prefixo de s, diga-se $t \leq s$, tal que $t \neq s$. Denota-se que t é prefixo estrito de s por $t < s$. Dada a palavra s, denota-se por $\overline{s}$ o conjunto de todos os prefixos de s.

[2]Também chamada de cadeia ou seqüência.

2.3.2 Linguagens: definição e operações

Dado um alfabeto Σ, qualquer subconjunto de Σ^* é uma linguagem sobre Σ. Os conjuntos $\emptyset$, dito linguagem vazia, e Σ^* são linguagens sobre Σ. O prefixo-fechamento de uma linguagem L sobre o alfabeto Σ, denotado $\overline{L}$, é o conjunto de todos os prefixos de palavras em L, isto é,

$$\overline{L} \;=\; \{t \in \Sigma^* : (\exists s \in L)\; t \leq s\}. \tag{2.1}$$

Uma linguagem L é dita prefixo-fechada se $L = \overline{L}$. As linguagens também atendem às operações usuais sobre conjuntos, tais como a união $\cup$ e a interseção $\cap$, por exemplo. A concatenação de duas linguagens K e L sobre um alfabeto Σ, denotada KL, é definida por $KL = \{kl \in \Sigma^* : (k \in K) \text{ e } (l \in L)\}$. O fechamento Kleene de uma linguagem K sobre um alfabeto Σ, denotado por K^*, é definido por $K^* = \bigcup_{i=0}^{\infty} K^i$, onde K^i denota a i-ésima concatenação de K consigo mesma e $K^0 = \{\epsilon\}$. Para uma linguagem L sobre um alfabeto Σ e a palavra $s \in \overline{L}$, o conjunto ativo de símbolos de L após s, denotado por $\Sigma_L(s)$, é o conjunto de todos os símbolos $\sigma \in \Sigma$ tais que $s\sigma \in \overline{L}$, isto é,

$$\Sigma_L(s) \;=\; \{\sigma \in \Sigma : s\sigma \in \overline{L}\} \tag{2.2}$$

Exemplo 2.1 (Operações sobre linguagens (Cury 2001)) *Seja o alfabeto* $\Sigma = \{a, b, c\}$ *e as linguagens* $L_1 = \{\epsilon, a, abb\}$ *e* $L_2 = \{c\}$. *Verifica-se que* L_1 *e* L_2 *não são prefixo-fechadas. Verifica-se também que* $L_1L_2 = \{c, ac, abbc\}$, $\overline{L_1} = \{\epsilon, a, ab, abb\}$, $L_2^* = \{\epsilon, c, cc, ccc, ...\}$, $L_1 \cup L_2 = \{\epsilon, a, abb, c\}$ *e que* $\Sigma_{L_1}(ab) = \{b\}$.

2.3.3 Produto síncrono de linguagens

Para um alfabeto Σ e um alfabeto $\Sigma_i \subseteq \Sigma$, a projeção de palavras sobre Σ em palavras sobre Σ_i é uma função $p_i : \Sigma^* \to \Sigma_i^*$ definida recursivamente por $p_i(\epsilon) = \epsilon$,

$$p_i(\sigma) \quad = \quad \begin{cases} \epsilon & \text{se } \sigma \notin \Sigma_i \\ \sigma & \text{se } \sigma \in \Sigma_i \end{cases} \tag{2.3}$$

para $\sigma \in \Sigma$, e

$$p_i(s\sigma) \quad = \quad p_i(s)p_i(\sigma) \tag{2.4}$$

para $\sigma \in \Sigma$ e $s \in \Sigma^*$. A ação de p_i sobre a palavra s é apenas a de apagar todas as ocorrências de símbolos σ que não estejam em Σ_i, resultando uma palavra $p_i(s) \in \Sigma_i^*$. A imagem da linguagem $L \subseteq \Sigma^*$ pela projeção p_i é definida por:

$$p_i(L) \quad = \quad \{p_i(s) \in \Sigma_i^* : s \in L\}. \tag{2.5}$$

Por fim, para a linguagem $L_i \subseteq \Sigma_i^*$, a imagem inversa pela projeção p_i é definida por:

$$p_i^{-1}(L_i) \quad = \quad \{s \in \Sigma^* : (\exists t \in L_i)\ p_i(s) = t\}. \tag{2.6}$$

Em particular, para $L_i = \{t\} \subseteq \Sigma_i^*$, escreve-se $p_i^{-1}(t)$ no lugar de $p_i^{-1}(\{t\})$ e diz-se imagem inversa de t.

Sejam os alfabetos Σ_1 e Σ_2, sendo possível que $\Sigma_1 \cap \Sigma_2 \neq \emptyset$, e as linguagens $L_1 \subseteq \Sigma_1^*$ e $L_2 \subseteq \Sigma_2^*$. Define-se o produto síncrono de L_1 e L_2, denotado $L_1 \| L_2$, como sendo uma linguagem sobre $\Sigma = \Sigma_1 \cup \Sigma_2$ definida por

$$L_1 \| L_2 = p_1^{-1}(L_1) \cap p_2^{-1}(L_2), \tag{2.7}$$

onde $p_i^{-1}(L_i)$ é a imagem inversa da linguagem $L_i \subseteq \Sigma_i^*$, conforme a equação (2.6). Note que se $u \in L_1 \| L_2$, então $p_1(u) \in L_1$ e $p_2(u) \in L_2$. Quando $\Sigma_1 = \Sigma_2$ o produto síncrono corresponde à interseção de linguagens, e quando $\Sigma_1 \cap \Sigma_2 = \emptyset$, chama-se o produto de

assíncrono.[3] O entendimento do produto síncrono de linguagens será complementado quando da definição da composição síncrona de autômatos, apresentada na seção 2.4.5.

2.3.4 Expressões regulares

Para um alfabeto Σ, uma expressão regular é definida recursivamente de acordo com as seguintes regras:

1. (a) $\emptyset$ é uma expressão regular, a representar a linguagem vazia.

   (b) ϵ é uma expressão regular, a representar a linguagem $\{\epsilon\}$.

   (c) σ é uma expressão regular, a representar a linguagem $\{\sigma\}$, com $\sigma \in \Sigma$.

2. Se r e s são expressões regulares, a corresponder às linguagens L_r e L_s, respectivamente, então rs, $r + s$, r^* e s^* são expressões regulares, a representar as linguagens $L_r L_s$, $L_r \cup L_s$, L_r^* e L_s^*, respectivamente.

3. Toda expressão regular é obtida pela aplicação das regras 1 e 2 um número finito de vezes.

Exemplo 2.2 (Expressões regulares (Cury 2001)) *Seja o alfabeto* $\Sigma = \{a, b, c\}$, *então as expressões regulares* $(a + b)c^*$ *e* $(ab)^* + c$ *representam as linguagens* $\{a, b, ac, bc, acc, bcc, \ldots\}$ *e* $\{c, \epsilon, ab, abab, ababab, \ldots\}$, *respectivamente.*

Qualquer linguagem que possa ser descrita por uma expressão regular é dita uma linguagem regular. As linguagens regulares relacionam-se aos autômatos determinstas de estados finitos, apresentados na próxima seção.

2.4 Autômatos

Esta seção apresenta o segundo conjunto de conceitos de base para o trabalho, correspondente a alguns fundamentos da teoria de autômatos. Novamente citam-se como

[3] No original em inglês *shuffle product*.

referências os trabalhos de Hopcroft e Ullmann (1979), Cassandras e Lafortune (1999), Wonham (2002b), Kumar e Garg (1995), Menezes (2001) e Cury (2001).

2.4.1 Autômatos deterministas de estados finitos

Um autômato determinista de estados finitos é uma quíntupla $G = (\Sigma, Q, \delta, q_0, Q_m)$, onde:

- Q é um conjunto finito de estados,
- Σ representa um alfabeto finito e não vazio,
- $\delta : Q \times \Sigma \rightarrow Q$ é a função de transição, parcial, a significar que não há necessidade da função ser definida para todo elemento de Σ para todo estado de Q,
- q_0 é o estado inicial do autômato, com $q_0 \in Q$, e
- Q_m é o conjunto de estados marcados[4], com $Q_m \subseteq Q$.

Define-se uma transição do autômato $G = (\Sigma, Q, \delta, q_0, Q_m)$ por uma tripla $(q, \sigma, q') \in Q \times \Sigma \times Q$ onde $q' = \delta(q, \sigma)$. Numa transição (q, σ, q'), q é o estado de origem, q' é o estado de destino e σ é a etiqueta da transição (Wonham 2002b).

Um autômato $G = (\Sigma, Q, \delta, q_0, Q_m)$ pode ser visto como um dispositivo que, inicialmente no estado q_0, lá permanece até a ocorrência de um símbolo que dispara uma transição definida para q_0 até um novo estado. O processo continua baseado nas transições definidas em δ.

Referenciar-se-á um autômato determinista de estados finitos simplesmente por autômato quando o contexto não exigir a distinção.

Representa-se um autômato de forma gráfica por um grafo dirigido, onde os nós representam os estados e os arcos etiquetados representam as transições entre os estados. O estado inicial é identificado por uma seta apontando para si e os estados marcados são representados por círculos duplos. A figura 2.3 representa um autômato determinista de estados finitos (Cury 2001).

[4]Também referenciados por estados finais.

$G = (\Sigma, Q, \delta, q_0, Q_m)$

$\Sigma = \{a, b, c\}$
$Q = \{0, 1, 2\}$
$q_0 = 0$
$Q_m = \{0, 2\}$

	símbolos			
estados	δ	a	b	c
	0	0	-	2
	1	0	1	-
	2	1	2	1

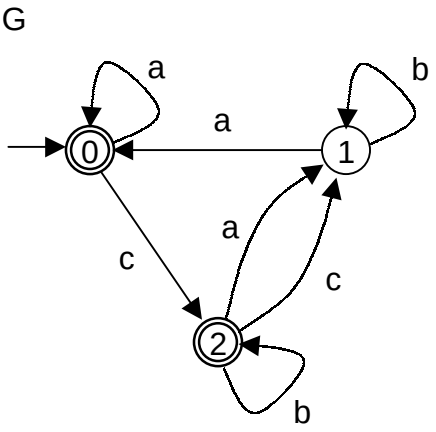


Figura 2.3: Autômato determinista de estados finitos.

A definição de um autômato determinista de estados finitos aqui apresentada difere da definição clássica dos livros de teoria de linguagens formais e autômatos, como (Hopcroft e Ullmann 1979), por permitir que a função de transição seja parcial. Tradicionalmente, na teoria de controle supervisório, o autômato como definido acima denomina-se *gerador*, como pode ser visto em (Wonham 2002b). Entretanto, este trabalho segue a nomenclatura de literatura recente, tais como Cassandras e Lafortune (1999) e Cury (2001), onde se usa o termo autômato para denominar um gerador.

Para um autômato $G = (\Sigma, Q, \delta, q_0, Q_m)$, define-se o conjunto ativo de símbolos de G no estado $q \in Q$, denotado $\Sigma_G(q)$, pelo conjunto:

$$\Sigma_G(q) \quad = \quad \{\sigma \in \Sigma : \delta(q, \sigma) \text{ definido}\}. \tag{2.8}$$

No exemplo da figura 2.3, $\Sigma_G(0) = \{a, c\}$ e $\Sigma_G(1) = \{a, b\}$.

2.4.2 Linguagens de um autômato

Estende-se a função de transição δ para palavras sobre Σ, denotada $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$, pela forma recursiva $\hat{\delta}(q, \epsilon) = q$ e

$$\hat{\delta}(q, s\sigma) \quad = \quad \delta(\hat{\delta}(q, s), \sigma) \tag{2.9}$$

para $q \in Q$, $\sigma \in \Sigma$, $s \in \Sigma^*$ e sempre que $q' = \hat{\delta}(q_0, s)$ e $\delta(q', \sigma)$ estejam definidos. No exemplo da figura 2.3, $\hat{\delta}(0, cba) = \delta(\hat{\delta}(0, cb), a) = \delta(\delta(\delta(0, c), b)a) = \delta(\delta(2, b), a) = \delta(2, a) = 1$ (Cury 2001).

Na literatura tradicional do controle supervisório de SEDs associam-se duas linguagens a um autômato $G = (\Sigma, Q, \delta, q_0, Q_m)$, a saber, a linguagem gerada $L_G \subseteq \Sigma^*$, definida por

$$L_G \quad = \quad \{s \in \Sigma^* : \hat{\delta}(q_0, s) \text{ definida}\}, \tag{2.10}$$

e a linguagem marcada $L_{G,m} \subseteq L_G$, definida por

$$L_{G,m} \quad = \quad \{s \in L_G : \hat{\delta}(q_0, s) \in Q_m\}. \tag{2.11}$$

A linguagem gerada representa todas as palavras que podem ser seguidas pelo autômato, partindo do estado inicial, enquanto que, a linguagem marcada representa todas as palavras que, partindo do estado inicial, levam o autômato a um estado marcado. As linguagens gerada e marcada do autômato G da figura 2.3 estão representadas na figura 2.4.

Na literatura da teoria de linguagens e autômatos referencia-se a linguagem marcada do autômato por linguagem aceita ou reconhecida pelo autômato e, como a função de transição do autômato não é parcial, a linguagem gerada pelo autômato é Σ^*, onde Σ é o alfabeto do autômato.

Seja um autômato G com alfabeto Σ e as linguagens L e L_m sobre Σ. Quando $L = L_G$ e $L_m = L_{G,m}$ diz-se que L é gerada por G e L_m é marcada por G, ou alternativamente, que G gera L e marca L_m.

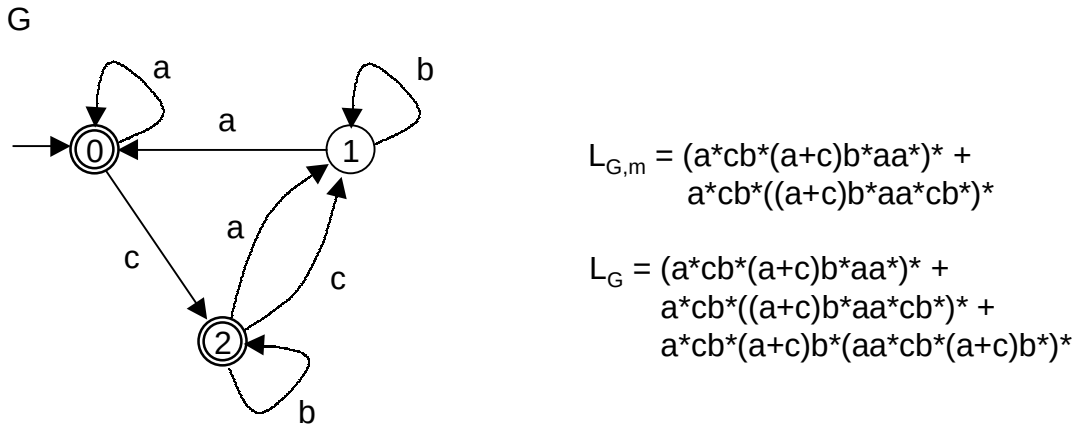


Figura 2.4: Linguagens para o autômato da figura 2.3.

Um autômato G é usado para representar o comportamento de um SED S, onde os símbolos do alfabeto de G representam os eventos gerados por S. A linguagem L_G representa todas as seqüências de eventos que o sistema pode gerar e a linguagem $L_{G,m}$ representa todas as seqüências de eventos que correspondem a tarefas completas do sistema.

Um resultado da literatura relaciona os autômatos deterministas de estados finitos às linguagens regulares. Pelo teorema de Kleene, se a linguagem L é regular então existe um autômato determinista de estados finitos G tal que $L_{G,m} = L$, e que, se o autômato G é determinista de estados finitos então $L_{G,m}$ é regular (Hopcroft e Ullmann 1979). Assim, pode-se afirmar que os autômatos deterministas de estados finitos e as expressões regulares possuem o mesmo poder de expressão, a significar que ambos conseguem representar as mesma classe de linguagens, as linguagens regulares.

Dada uma linguagem regular L existe mais de um autômato G tal que $L_{G,m} = L$ (Hopcroft e Ullmann 1979). Dois autômatos G_1 e G_2 tais que $L_{G_1} = L_{G_2}$ e $L_{G_1,m} = L_{G_2,m}$ são ditos equivalentes. Dois autômatos G_1 e G_2 são ditos isomorfos quando $G_1 = G_2$, a menos de uma renomeação de estados.

2.4.3 Acessibilidade e co-acessibilidade de um autômato

Dado um autômato $G = (\Sigma, Q, \delta, q_0, Q_m)$, um estado $q \in Q$ é dito acessível se $q = \hat{\delta}(q_0, u)$ para algum $u \in \Sigma^*$, isto é, se q pode ser alcançado a partir do estado inicial q_0 seguindo transições de G. O autômato G é dito ser acessível se todos os seus estados forem acessíveis. A componente acessível de um autômato G, denotada $Ac(G)$ é obtida a partir de G por eliminação dos estados não acessíveis e transições associadas (Cury 2001). Por outro lado, o estado $q \in Q$ é dito ser co-acessível, se houver uma seqüência de transições em G que parte de q e chega num estado marcado. Um autômato G é dito ser co-acessível se todos os seus estados forem co-acessíveis. Assim como existe a componente acessível, define-se a componente co-acessível do autômato G, denotada $CoAc(G)$, por eliminação dos estados não co-acessíveis de G e transições associadas (Cury 2001). A condição de co-acessibilidade de um autômato G é descrita em forma de linguagens pela condição $L_G = \overline{L_{G,m}}$ (Cury 2001). Por fim, diz-se que um autômato é *trim* quando este é acessível e co-acessível. Também define-se a componente *trim* de um autômato G, denotada $trim(G)$, obtida a partir de G por eliminação dos estados não acessíveis e não co-acessíveis. A componente *trim* de um autômato G é obtida fazendo-se $trim(G) = CoAc(Ac(G)) = Ac(CoAc(G))$ (Cury et al. 2001). Na figura 2.5 ilustra-se um autômato G, a componente acessível $Ac(G)$, a componente co-acessível $CoAc(G)$ e a componente $trim(G)$ (Cassandras e Lafortune 1999).

A definição de acessibilidade e co-acessibilidade de um autômato são usadas para caracterizar situações de bloqueio em um SED (Cury 2001). Seja um SED S cujo comportamento é representado por um autômato G. Diz-se que o SED S é não bloqueante se e somente se $L_G = \overline{L_{G,m}}$, isto é, se o autômato correspondente for *trim*. A condição de bloqueio de um SED caracteriza algumas situações indesejadas no funcionamento do sistema. A situação de *deadlock* é caracterizada pela existência de um estado de onde o sistema não pode mais evoluir, não correspondendo a uma tarefa completa do sistema. A situação de *livelock* é caracterizada pela existência de um conjunto de estados onde a evolução do sistema fica confinada, sem que nenhuma tarefa do sistema seja completa.

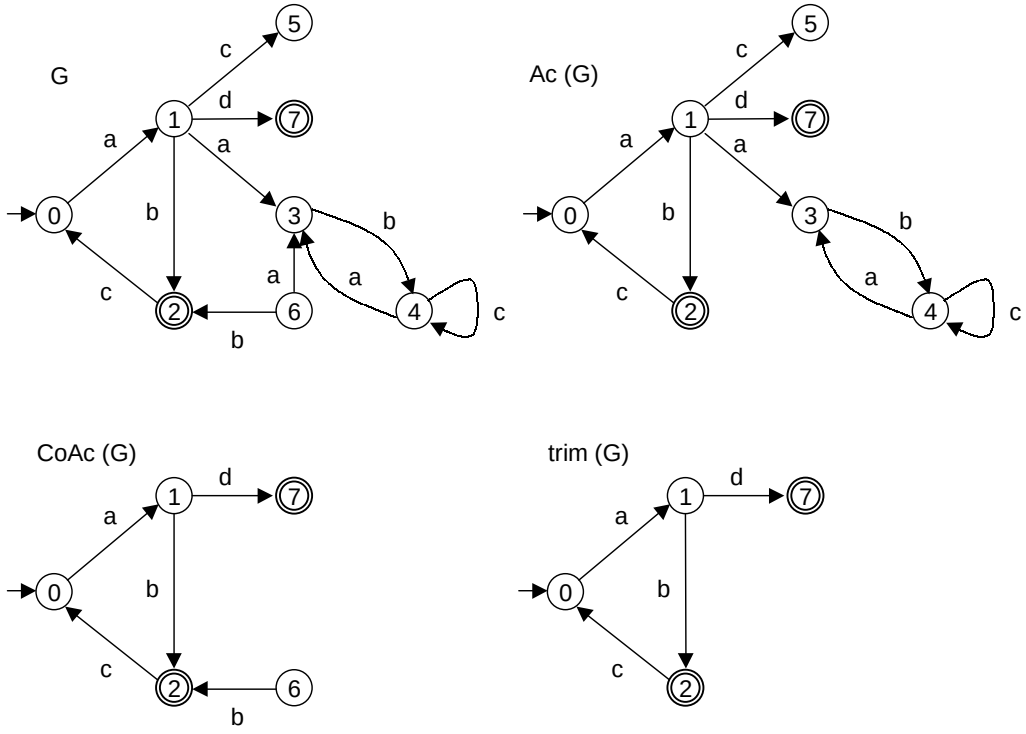


Figura 2.5: Componente acessível, co-acessível e *trim* de um autômato.

Se o autômato G da figura 2.5 representa o comportamento de um SED, verifica-se que o SED é bloqueante. O estado 5 de G na figura 2.5 corresponde a um *deadlock*, ao passo que os estados 3 e 4 correspondem a um *livelock*. Já o estado 7 da figura 2.5 não corresponde a um *deadlock* pois este corresponde a uma tarefa completa do sistema, isto é, 7 é um estado marcado.

2.4.4 Equivalentes Determinista e Mínimo de um autômato

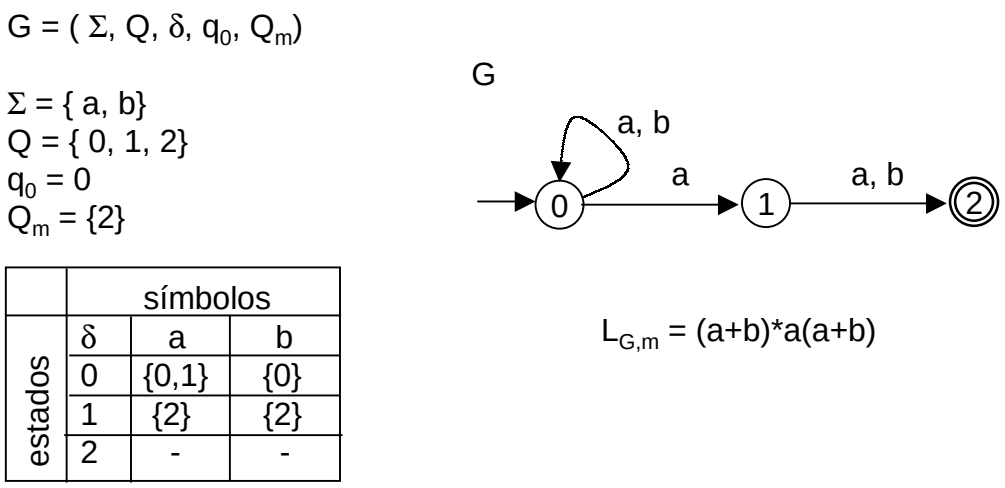
Um autômato não determinista de estados finitos é uma quíntupla $G = (\Sigma, Q, \delta, q_0, Q_m)$, onde Q, Σ, q_0 e Q_m são definidos como para o autômato determinista, e δ é uma função de transição definida por $\delta : Q \times \Sigma \rightarrow 2^Q$, onde 2^Q é o conjunto das partes de Q. As linguagens gerada e marcada por um autômato não determinista de estados finitos são definidas conforme as linguagens de um autômatos determinista de estados finitos (Cury 2001). Os autômatos não deterministas podem ser uma alternativa aos autômatos deterministas para modelar sistemas físicos, seja por simplicidade de obtenção dos mesmos, seja como meio de exprimir a falta de informações sobre o sistema que se está modelando (Cury 2001). Como será mostrado nos capítulos 3 e 5, autômatos não deterministas surgem em uma das etapas da construção dos sistemas do gerente no controle hierárquico de SEDs.

Um resultado da teoria de autômatos e linguagens diz que toda linguagem marcada por um autômato não determinista é também marcada por um autômato determinista (Cury 2001). Dado um autômato não determinista de estados finitos $G = (\Sigma, Q, \delta, q_0, Q_m)$, constrói-se um autômato determinista de estados finitos G^D equivalente a G, isto é, tal que $L_{G^D} = L_G$ e $L_{G^D,m} = L_{G,m}$, da seguinte forma (Cury 2001). Define-se $G^D = Ac(\Sigma, Q^D, \delta^D, q_0^D, Q_m^D)$, onde:

- $Q^D = 2^Q$,
- $q_0^D = \{q_0\}$,
- $\delta^D(q^D, \sigma) = \bigcup_{q \in q^D} \delta(q, \sigma)$, com $q^D \in Q^D$ e $\sigma \in \Sigma$, e

- $Q_m^D = \{q^D \in Q^D : q^D \cap Q_m \neq \emptyset\}$.

A construção indicada acima, conhecida por construção do equivalente determinista[5], baseia-se na construção recursiva da função de transição δ^D, a partir do estado inicial q_0^D. Considere o autômato não determinista G da figura 2.6, o autômato determinista G^D equivalente obtido pela construção acima, é representado na figura 2.7.



	símbolos		
estados	δ	a	b
	0	{0,1}	{0}
	1	{2}	{2}
	2	-	-

Figura 2.6: Autômato não determinista.

O problema da construção do equivalente determinista é o número de estados do autômato determinista resultante. Para um autômato não determinista com n estados, o tamanho do autômato determinista equivalente é limitado a 2^n estados. Este potencial problema de crescimento exponencial do equivalente determinista está presente na construção dos sistemas para o gerente no controle hierárquico de SEDs, comentado na seção 5.3.

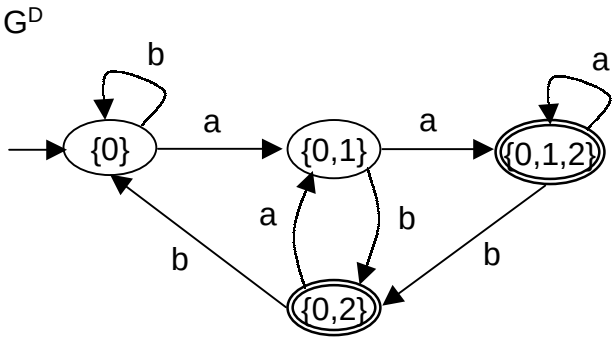
O autômato resultante do procedimento de obtenção do equivalente determinista possivelmente não possui número mínimo de estados para representar as mesmas linguagens do autômato não determinista original. Entretanto, este autômato pode ter seu número de estados minimizado, tendo as linguagens marcada e gerada preservada,

[5] *Subset construction* em inglês.

$G^D = (\Sigma, Q^D, \delta^D, q_0{}^D, Q_m{}^D)$

$\Sigma = \{ a, b\}$
$Q^D = \{ \{0\}, \{0,1\}, \{0,2\}, \{0,1,2\}\}$
$q_0{}^D = \{0\}$
$Q_m{}^D = \{ \{0,2\}, \{0,1,2\}\}$

	símbolos		
estados	δ^D	a	b
	{0}	{0,1}	{0}
	{0,1}	{0,1,2}	{0,2}
	{0,2}	{0,1}	{0}
	{0,1,2}	{0,1,2}	{0,2}



$L_{GDm} = (a+b)^*a(a+b)$

Figura 2.7: Equivalente determinista.

ao se aplicar o procedimento descrito a seguir. Dado um autômato G é possível encontrar um autômato G_{min} equivalente a G com número mínimo de estados, dito autômato mínimo equivalente (Hopcroft e Ullmann 1979). O procedimento de obtenção de um autômato mínimo equivalente não é detalhado neste trabalho, podendo este ser encontrado em (Hopcroft e Ullmann 1979), por exemplo. O interesse em se obter um autômato mínimo equivalente a um dado autômato é, por exemplo, obter uma descrição compacta, em termos de número de estados, de um dado SED.

2.4.5 Composição síncrona de autômatos

Na teoria de controle supervisório, um SED pode ser visto como formado por subsistemas componentes atuando em paralelo. A ação do controlador impõe restrições de coordenação ao funcionamento conjunto dos componentes.

Numa abordagem global para a modelagem de um SED, procura-se obter diretamente o autômato determinista de estados finitos que represente todas as seqüências de eventos que o sistema pode gerar e tarefas que pode completar. Cury (2001) comenta

que, para sistemas de grande porte esta pode ser uma tarefa de grande complexidade, além do que, qualquer alteração no sistema, seja por inclusão ou retirada de componentes ou modificação na lógica de controle, requer a reconstrução do modelo como um todo. Uma abordagem local para a modelagem parte do princípio de que se pode construir modelos locais para cada subsistema e restrição de coordenação, e que se pode compor os mesmos para obter um modelo de sistema global. Cury (2001) também comenta que uma modelagem local acarreta maior facilidade na obtenção dos modelos de grande porte, e permite pressupor que alterações numa parte do sistema somente acarretem modificações no modelo específico da parte afetada. O que se introduz a seguir é uma forma de composição de autômatos que é usada na modelagem local de SEDs.

Dados dois autômatos $G_1 = (Q_1, \Sigma_1, \delta_1, q_{0,1}, Q_{m,1})$ e $G_2 = (Q_2, \Sigma_2, \delta_2, q_{0,2}, Q_{m,2})$, define-se a composição síncrona de G_1 e G_2, denotada $G_1 \| G_2$, pelo autômato

$$G_1 \| G_2 \;=\; Ac(Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta_{12}, (q_{0,1}, q_{0,2}), Q_{m,1} \times Q_{m,2}), \tag{2.12}$$

onde $\delta_{12} : (Q_1 \times Q_2) \times (\Sigma_1 \cup \Sigma_2) \to (Q_1 \times Q_2)$ é definida por:

$$\delta_{12}((q_1, q_2), \sigma) \;=\; \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)) & \text{se } \sigma \in \Sigma_{G_1}(q_1) \cap \Sigma_{G_2}(q_2) \\ (\delta_1(q_1, \sigma), q_2) & \text{se } \sigma \in \Sigma_{G_1}(q_1) - \Sigma_2 \\ (q_1, \delta_2(q_2, \sigma)) & \text{se } \sigma \in \Sigma_{G_2}(q_2) - \Sigma_1 \\ \text{não definida} & \text{caso contrário,} \end{cases} \tag{2.13}$$

para $(q_1, q_2) \in Q_1 \times Q_2$ e $\sigma \in \Sigma_1 \cup \Sigma_2$.

Na composição síncrona, um símbolo comum a G_1 e G_2 só ocorre se puder ser executado sincronamente nos dois autômatos. Os demais símbolos ocorrem assincronamente, isto é, de modo independente em cada um dos autômatos. Considere os autômatos G_1 e G_2 na figura 2.8, com alfabetos $\Sigma_1 = \{a, b\}$ e $\Sigma_2 = \{b, c\}$, respectivamente. A composição síncrona de G_1 com G_2, o autômato $G_1 \| G_2$, também é representada na figura 2.8 (Wonham 2002b).

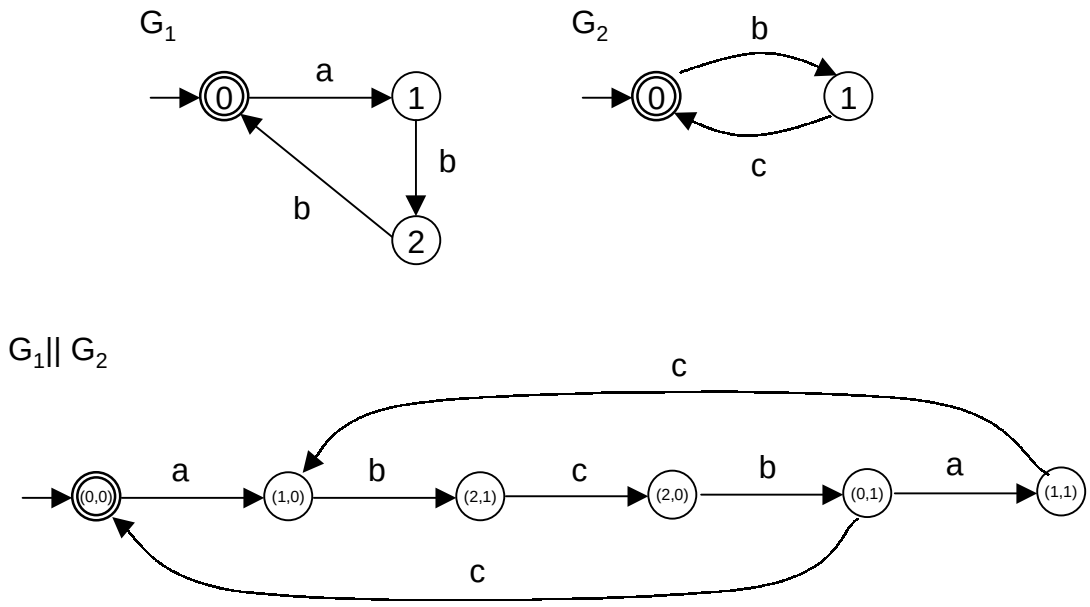


Figura 2.8: Composição síncrona.

Verifica-se que se $G = G_1 \| G_2$, então $L_G = L_{G_1} \| L_{G_2}$ e $L_{G,m} = L_{G_1,m} \| L_{G_2,m}$, isto é, as linguagens da composição síncrona são os produtos síncronos das linguagens respectivas dos autômatos componentes (Wonham 2002b). Para o caso em que $\Sigma_1 = \Sigma_2$, todos os eventos devem ser sincronizados entre G_1 e G_2 e tem-se $L_{G_1 \| G_2} = L_{G_1} \cap L_{G_2}$ e $L_{G_1 \| G_2,m} = L_{G_1,m} \cap L_{G_2,m}$, isto é, as linguagens da composição síncrona são as interseções das linguagens respectivas dos componentes. Pode-se provar que a composição síncrona é comutativa, isto é, $G_1 \| G_2 = G_2 \| G_1$, e associativa, isto é, $G_1 \| (G_2 \| G_3) = (G_1 \| G_2) \| G_3$, assim, pode-se estender a composição síncrona para mais de dois autômatos (Wonham 2002b).

2.4.6 Autômato projeção

Dados um autômato $G = (\Sigma, Q, \delta, q_0, Q_m)$ e um alfabeto $\Sigma_r \subseteq \Sigma$, define-se o autômato projeção do autômato G no alfabeto Σ_r, denotado $proj(G, \Sigma_r)$, pelo autômato $G' = (\Sigma_r, Q', \delta', q_0, Q'_m)$ obtido pelo algoritmo 2.1. No algoritmo 2.1, a notação $\leftarrow$ representa a atribuição e os textos entre chaves correspondem a comentários.

entradas
Autômato: $G = (\Sigma, Q, \delta, q_0, Q_m)$
Alfabeto: $\Sigma_r \subseteq \Sigma$
início
$Q' \leftarrow \{q_0\}$ {Inicializa Q'}
$Q'_m \leftarrow \emptyset$ {Inicializa Q'_m}
para todo $q \in Q$ para o qual existe $q' \in Q$ e $\sigma \in \Sigma_r$ tais que $\delta(q', \sigma) = q$ **faça**
  $Q' \leftarrow Q' \cup \{q\}$ {Calcula o conjunto de estados Q'}
fim para
para todo $q \in Q'$ **faça**
  $Q(q) \leftarrow \emptyset$ {Inicializa $Q(q)$}
  $Q'(q) \leftarrow \emptyset$ {Inicializa $Q'(q)$}
  para todo $\sigma \in \Sigma_r$ **faça**
    $\delta'(q, \sigma) \leftarrow \emptyset$ {Inicializa $\delta'(q, \sigma)$}
  fim para
  {Calcula $Q(q)$ e $\delta'(q, \sigma)$ por uma recursão}
  enquanto $Q'(q) \neq \emptyset$ **faça**
    Selecione e retire q' de $Q'(q)$
    para todo $q'' \in Q$ e $\sigma \in \Sigma$ tais que $\delta(q', \sigma) = q''$ **faça**
      se $\sigma \notin \Sigma_r$ **então**
        se $q'' \notin Q(q)$ **então**
          $Q(q) \leftarrow Q(q) \cup \{q''\}$
          $Q'(q) \leftarrow Q'(q) \cup \{q''\}$
        fim se
      senão
        $\delta'(q, \sigma) \leftarrow \delta'(q, \sigma) \cup \{q''\}$
      fim se
    fim para
  fim enquanto
  {Testa se deve considerar q como marcado}
  se $Q(q) \cap Q_m \neq \emptyset$ **então**
    $Q'_m \leftarrow Q'_m \cup \{q\}$
  fim se
fim para
fim
saída
Autômato: $G' = (\Sigma_r, Q', \delta', q_0, Q'_m)$

Algoritmo 2.1: Calcula o autômato projeção.

No algoritmo 2.1, o conjunto de estados Q' é um subconjunto do conjunto de estados Q contendo os estados de entrada de transições com eventos em Σ_r, mais o estado inicial. Dado o estado q de Q', o conjunto de estados $Q(q) \subseteq Q$ corresponde aos estados em Q alcançados a partir de q por transições envolvendo símbolos em $\Sigma - \Sigma_r$. Define-se uma transição entre dois estados q e q' de Q' envolvendo o símbolo σ em Σ_r quando algum estado de $Q(q)$ alcança q' por uma transição em G envolvendo o símbolo σ. Por fim, o estado $q \in Q$ é marcado se o conjunto de estados $Q(q)$ contém um estado marcado de G.

O autômato projeção $G' = proj(G, \Sigma_r)$ obtido pelo algoritmo 2.1 é possivelmente não determinista, e pode-se provar que $p_r(L_G) = L_{G'}$ e $p_r(L_{G,m}) = L_{G',m}$, onde $p_r : \Sigma^* \rightarrow \Sigma_r^*$ é a projeção de palavras sobre Σ em palavras sobre Σ_r.

Seja o autômato G na figura 2.9 e considere $\Sigma_r = \{a, b, c\}$, então representa-se $G' = proj(G, \Sigma_r)$ na figura 2.9.

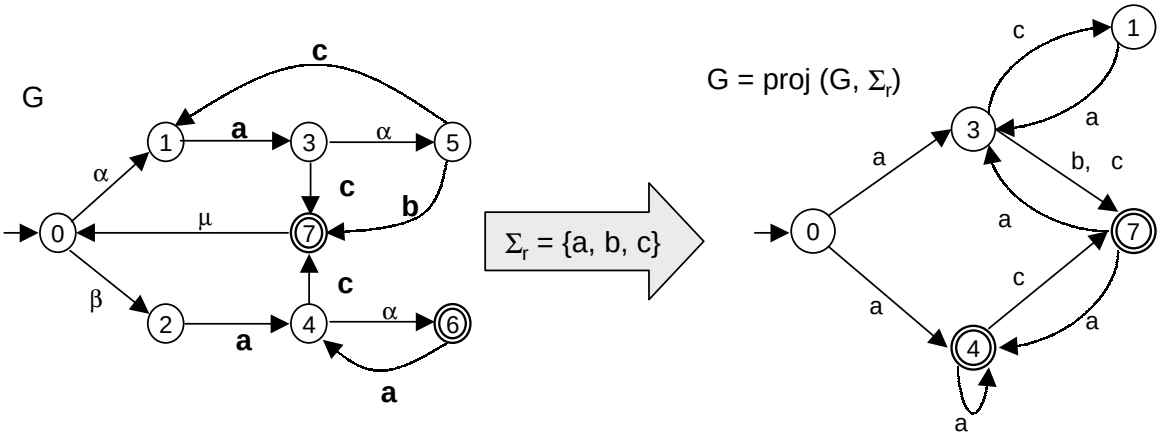


Figura 2.9: Autômatos-projeção.

Em (Kumar e Garg 1995) define-se uma forma alternativa de autômato projeção, com construção baseada em autômatos não deterministas com ϵ-transições e a operação de ϵ-fechamento (Hopcroft e Ullmann 1979). Não se adota esta definição neste trabalho pois tal construção não garante a propriedade de preservar como estados do autômato projeção os estados de entrada de transições com eventos no alfabeto de projeção.

No controle hierárquico de SEDs, um autômato projeção com a propriedade citada acima é utilizado na construção do sistema para o gerente num esquema de supervisão hierárquica de dois níveis, vide capítulo 5.

2.4.7 Autômato de Moore

Define-se um autômato de Moore[6] pela sêxtupla $G = (\Sigma, Q, \delta, q_0, Q_m, T, w)$ (Wonham 2002b), onde

- Σ é o alfabeto de entrada,
- Q é o conjunto de estados,
- $\delta : Q \times \Sigma \rightarrow Q$ é a função de transição de estados, possivelmente parcial,
- q_0 é o estado inicial, com $q_0 \in Q$, e
- Q_m é o conjunto de estados marcados, com $Q_m \subseteq Q$,

conforme a definição de um autômato. Adicionalmente, o autômato de Moore possui mais dois elementos, a saber:

- T, o alfabeto de saída, e
- $w : Q \rightarrow T$, uma função de saída definida sobre os estados.

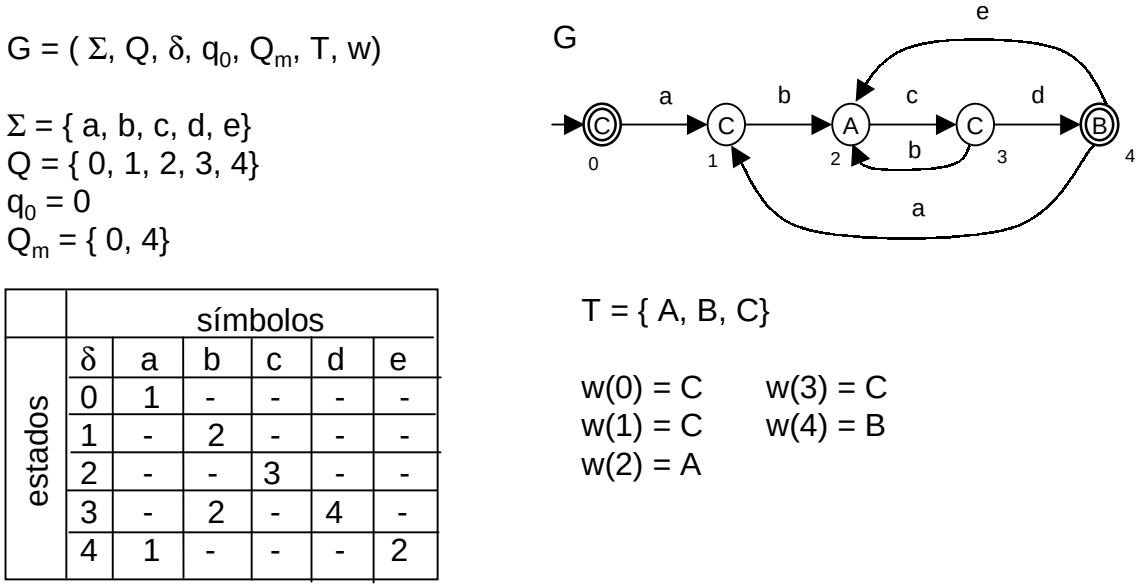
De forma geral, um autômato de Moore é um autômato com uma função que atribui um símbolo de saída a cada estado. A definição de autômato de Moore acima difere da definição padrão, em (Hopcroft e Ullmann 1979), por incluir um conjunto de estados marcados.

Basicamente, o funcionamento de um autômato de Moore corresponde ao reconhecimento de seqüências de símbolos de entrada $\sigma_0\sigma_1 \ldots \sigma_n \in \Sigma^*$, conforme um autômato normal, ao passo que a trajetória de estados percorrida gera uma seqüência de símbolos

[6] Também denominado Máquina de Moore na literatura da teoria de autômatos e linguagens.

de saída $\tau_0\tau_1 \ldots \tau_n \in T^*$ (Hopcroft e Ullmann 1979). Na teoria de controle supervisório, atribuem-se duas linguagens ao autômato de Moore, a saber, a linguagem gerada $L_G \subseteq \Sigma^*$ e a linguagem marcada $L_{G,m} \subseteq L_G$, conforme as linguagens correspondentes para os autômatos normais, vide equações (2.10) e (2.11). Não se associa uma linguagem à seqüência de símbolos de saída de estados do autômato de Moore (Wonham 2002b).

Neste trabalho, representa-se um autômato de Moore graficamente por um grafo dirigido, conforme um autômato normal, com as seguintes particularidades: a identificação dos estados fica sob os círculos que os representam, a identificação das transições fica sobre as setas, e a identificação da saída do estado fica dentro do círculo que o representa. Na figura 2.10 representa-se um autômato de Moore.



	símbolos					
estados	δ	a	b	c	d	e
	0	1	-	-	-	-
	1	-	2	-	-	-
	2	-	-	3	-	-
	3	-	2	-	4	-
	4	1	-	-	-	2

Figura 2.10: Autômato de Moore.

No controle hierárquico de SEDs, um autômato de Moore é usado para representar dois elementos em um esquema de supervisão hierárquica. O primeiro elemento, representado pelas linguagens gerada e marcada do autômato de Moore, é o comportamento do sistema do operador. O segundo, representado pelas etiquetas de saída para os

estados do autômato de Moore, é um canal de informação que notifica a ocorrência de novos eventos ao sistema do gerente a partir das seqüências de eventos geradas pelo sistema do operador.

2.5 Abordagem Ramadge-Wonham (RW)

Nesta seção apresenta-se uma introdução à abordagem RW para controle supervisório de SEDs. Algumas referências para o leitor interessado em mais detalhes são Ramadge e Wonham (1989), Kumar e Garg (1995), Cassandras e Lafortune (1999), Wonham (2002b), e Cury (2001).

2.5.1 Esquema de supervisão

A abordagem Ramadge-Wonham (RW) para o controle supervisório de SEDs faz distinção entre o sistema a controlar, denominado planta, e o agente controlador, denominado supervisor (Ramadge e Wonham 1989).

A planta corresponde, em geral, a um conjunto de sistemas componentes atuando em paralelo para realizar uma dada função (Cury 2001). Cada componente possui um comportamento individual e a composição dos comportamentos dos componentes determina o comportamento da planta. A planta modela todos os comportamentos possíveis em malha aberta, inclusive os indesejáveis para o funcionamento conjunto dos componentes.

À planta deseja-se impor um conjunto de restrições de coordenação, a fim de que o funcionamento conjunto dos componentes atenda a certas especificações. Tais especificações podem expressar requisitos de segurança, como para evitar uma seqüência de eventos indesejada, justiça, como para garantir o acesso justo a um recurso comum, ou a imposição de uma seqüência desejada de eventos, como para exprimir um programa de produção (Cury 2001).

De modo a fazer com que os componentes da planta atuem de forma coordenada, Ramadge e Wonham (1989) introduzem um agente de controle denominado supervisor.

O supervisor interage com a planta numa estrutura em malha fechada como indicado na figura 2.11. A planta é suposta gerar eventos de forma espontânea, e o supervisor é suposto ter a habilidade de proibir a ocorrência de certos eventos na planta. Um evento cuja ocorrência é permitida pelo supervisor é dito habilitado, caso contrário, o evento é dito desabilitado. A entrada de controle aplicada pelo supervisor à planta corresponde a um conjunto informando que eventos estão habilitados. Em malha fechada, o supervisor observa os eventos que são gerados pela planta, e à geração de um novo evento, aplica uma nova entrada de controle à planta informando que eventos estão habilitados após o evento gerado. A ação de controle do supervisor é dita passiva, pois esta apenas indica quais os eventos habilitados e não força a geração de um evento.

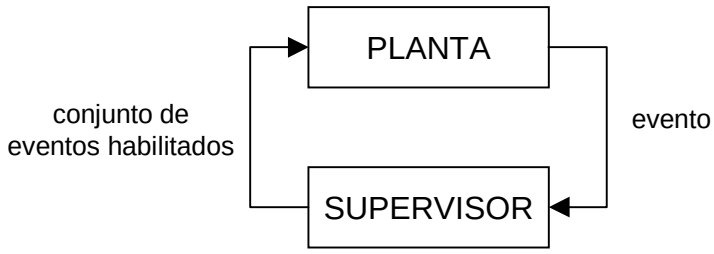


Figura 2.11: Esquema de controle supervisório na abordagem RW.

Nesta exposição, adota-se a perspectiva do controle supervisório com marcação, uma extensão da abordagem RW original (Wonham 2002b), onde, além de poder desabilitar eventos, o supervisor pode selecionar quais tarefas em malha aberta da planta são consideradas como tarefas em malha fechada. Isso permite, por exemplo, que numa célula de manufatura onde a tarefa em malha aberta seja a produção de uma peça, possa-se definir como uma tarefa em malha fechada a produção de um lote com, por exemplo, dez peças.

2.5.2 Modelo da planta

Na abordagem RW, representa-se um SED S com conjunto de eventos Σ por uma tripla $(L_S, L_{S,m}, \Gamma_S)$, onde L_S e $L_{S,m}$ são linguagens sobre Σ, com L_S prefixo-fechada e

$L_{S,m}$ contida em L_S, e Γ_S é um subconjunto de 2^{Σ}, denominado estrutura de controle de S (Ramadge e Wonham 1989). Diz-se que Σ é o alfabeto do SED S ou que S é um SED sobre o alfabeto Σ. A linguagem L_S representa todas as seqüências de eventos que o sistema S pode gerar, ao passo que a linguagem $L_{S,m}$ representa todas as seqüências geradas que definem tarefas completas do sistema. Γ_S é um conjunto de entradas de controle $\gamma \subseteq \Sigma$, onde cada entrada γ contém um possível conjunto de eventos habilitados que o supervisor pode aplicar à planta, isto é, a ocorrência dos eventos pertencentes ao controle γ está permitida. Os outros eventos da planta que não estão no controle γ estão desabilitados, a significar que sua ocorrência está proibida. Alternativamente, pode-se representar o comportamento da planta S por um autômato G tal que $L_G = L_S$ e $L_{G,m} = L_{S,m}$. Assim, é comum também se representar a planta por um par (G, Γ_S) (Ramadge e Wonham 1989).

O mecanismo de controle de eventos proposto por Ramadge e Wonham (1989) corresponde ao particionamento do conjunto de eventos Σ em um subconjunto de eventos controláveis Σ_c e um subconjunto de eventos não controláveis Σ_{nc}. Os eventos controláveis são eventos que podem ser desabilitados, isto é, sua ocorrência pode ser proibida por ação de controle. Por exemplo, num sistema de manufatura, considera-se o evento que marca o início de operação de uma máquina como controlável. Os eventos não controláveis são eventos que, por sua natureza, não podem ser desabilitados, isto é, não se pode proibir sua ocorrência por ação de controle. Por exemplo, novamente num sistema de manufatura, consideram-se os eventos que marcam o fim de operação de uma máquina ou a sua quebra como não controláveis.

O particionamento do conjunto de eventos Σ em eventos controláveis e não controláveis implica algumas propriedades estruturais para o conjunto de entradas de controle Γ_S (Ramadge e Wonham 1989). É natural que uma entrada de controle sempre deva conter os eventos não controláveis de uma planta, pois estes não podem ser desabilitados, assim o conjunto Γ_S possui a seguinte propriedade:

$$\Gamma_S \quad = \quad \{\gamma \in 2^{\Sigma} : \Sigma_{nc} \subseteq \gamma\}. \tag{2.14}$$

Outra propriedade de Γ_S é que este é fechado em relação à união (Ramadge e Wonham 1989).

Exemplo 2.3 (Máquina de três estados) *Considere o seguinte modelo de uma máquina num sistema de manufatura (Ramadge e Wonham 1989). A máquina pode estar em três estados, a saber: parada (P), trabalhando (T) ou quebrada (Q). Inicialmente parada, a máquina pode iniciar a trabalhar, gerando o evento* início de operação *(α). Trabalhando, a máquina pode terminar a operação com sucesso e voltar a ficar parada, gerando o evento* fim de operação *(β), ou pode quebrar, gerando o evento* quebra *(λ). Estando a máquina quebrada, pode-se reparar a máquina, gerando-se o evento* reparo *(μ), e voltando a máquina a estar parada. Consideram-se os eventos de início de operação e reparo como controláveis, enquanto os eventos de fim de operação e quebra como não controláveis. Consideram-se também que tarefas completas do sistema correspondem a ciclos do tipo* parada-trabalhando-parada *ou* parada-trabalhando-quebrada-parada*, ou a intercalação de tais ciclos. A figura 2.12 representa o SED S que modela a máquina de três estados segundo a abordagem RW. O autômato G representa as linguagens gerada e marcada pelo sistema. A representação usual da abordagem RW para os eventos controláveis num autômato corresponde a uma linha que corta a seta que representa as transições correspondentes (Cury 2001). O conjunto de entradas de controle Γ_S também é representado na figura 2.12.*

2.5.3 Modelo do supervisor

Na abordagem RW, um supervisor para o SED S é definido por um par (f, L_m), onde $f : L_S \to \Gamma_S$ é uma função e $L_m \subseteq L_{S,m}$ é uma linguagem (Wonham 2002b). A função f associa a cada $s \in L_S$ uma entrada de controle $f(s) = \gamma \in \Gamma_S$, isto é, um conjunto de eventos habilitados. A linguagem L_m especifica quais das tarefas de S em malha aberta devem ser consideradas tarefas em malha fechada.

O supervisor aqui apresentado corresponde ao controle supervisório com marcação[7]

[7]No original em inglês, *marking supervisory control*.

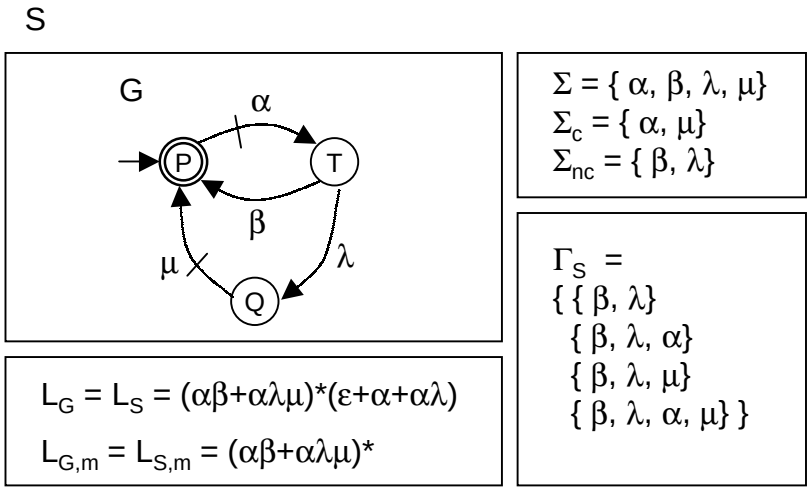


Figura 2.12: Máquina de três estados.

(Wonham 2002b), onde o supervisor possui a habilidade adicional de *desmarcar* palavras em malha fechada, isto é, não reconhecer como tarefa do sistema em malha fechada uma palavra que corresponde a uma tarefa do sistema em malha aberta (Wonham 2002b, Cury 2001). Diz-se que tal supervisor é um supervisor marcador[8]. A hipótese de um supervisor marcador acarreta uma simplificação das condições e do formalismo para determinar a existência de um supervisor que realize uma dada linguagem no sistema, como será visto a seguir.

Define-se o sistema em malha fechada f/S pelo sistema resultante da ação conjunta da planta S sob controle do supervisor (f, L_m). O comportamento em malha fechada é caracterizado por duas linguagens sobre Σ, a saber, a linguagem gerada em malha fechada $L_{f/S}$, definida recursivamente por $\epsilon \in L_{f/S}$, e

$$s\sigma \in L_{f/S} \iff ((s \in L_{f/S}) \text{ e } (s\sigma \in L_S) \text{ e } (\sigma \in f(s))), \tag{2.15}$$

[8]No original em inglês, *marking supervisor*.

onde $\sigma \in \Sigma$ e $s \in \Sigma^*$, e a linguagem marcada em malha fechada, $L_{f/S,m}$, por

$$L_{f/S,m} \quad = \quad L_{f/S} \cap L_m. \tag{2.16}$$

A definição da linguagem $L_{f/S}$ significa que uma seqüência de eventos $s\sigma$ pode ocorrer sob supervisão se e somente se s pode ocorrer sob supervisão, $s\sigma$ é uma seqüência fisicamente possível na planta, e σ é habilitado por f após s. Já a definição da linguagem $L_{f/S,m}$ significa que as palavras marcadas em malha fechada são as palavras marcadas em L_m que sobrevivem à supervisão. Observa-se que $L_{f/S}$ é prefixo-fechada, que $L_{f/S,m} \subseteq L_{f/S}$, e que, em geral, $\overline{L_{f/S,m}} \subseteq L_{f/S}$ (Ramadge e Wonham 1989). A última expressão acima significa que é possível que no sistema em malha fechada haja palavras geradas que não são prefixos de tarefas completas do sistema. Assim, é possível que o sistema em malha fechada seja bloqueante, como visto na subseção 2.4.3. Diz-se que um supervisor f para um SED S é não bloqueante quando $\overline{L_{f/S,m}} = L_{f/S}$, caso contrário f é dito bloqueante. Como visto anteriormente, o bloqueio num SED corresponde a situações indesejadas no comportamento lógico do sistema. Assim, é de interesse do projetista que o supervisor que implementa um comportamento desejado em malha fechada seja não bloqueante.

Uma forma de representar um supervisor (f, L_m) para um SED S é por meio de um autômato F que marque L_m e cuja ação de controle da função f sobre S esteja implícita na estrutura de transição de F. Seja G o autômato que representa o comportamento em malha aberta de S. Prova-se que o comportamento do sistema em malha fechada f/S é representado pela composição síncrona $F\|G$, ou seja, $L_{F\|G} = L_{f/S}$ e $L_{F\|G,m} = L_{f/S,m}$ (Cury 2001).

2.5.4 O problema de controle supervisório

De forma geral, deseja-se que, em malha fechada o sistema atenda a certas restrições de funcionamento (Cury 2001). Por exemplo, pode ser que S admita seqüências de eventos que sejam indesejáveis por violarem uma condição que deseja se impor ao sistema. Isso

corresponde à existência de estados proibidos no autômato que representa S. Pode ser ainda o caso de palavras que violem o ordenamento desejado para os eventos, como por exemplo, a justiça no acesso a recursos. Na abordagem RW, expressam-se as restrições de coordenação por intermédio de uma linguagem-alvo $K \subseteq L_{S,m}$, que exprime o comportamento desejado para o sistema em malha fechada. Neste ponto, introduz-se o problema de controle supervisório dentro da abordagem RW.

Definição 2.1 (Problema de controle supervisório (PCS)) *Dados um SED S e uma linguagem-alvo $K \subseteq L_{S,m}$, encontre um supervisor não bloqueante (f, L_m) para S tal que $\emptyset \subset L_{f/S,m} \subseteq K$.*

A formulação apresentada na definição 2.1 corresponde a uma forma simplificada do PCS encontrada na literatura (Ramadge e Wonham 1989). Na definição 2.1, requer-se apenas encontrar um supervisor não bloqueante tal que o comportamento resultante na supervisão seja não vazio e atenda a K, a representar um comportamento desejado para o sistema. Em formulações mais detalhadas do PCS, define-se uma outra linguagem não vazia que representa um comportamento mínimo aceitável em malha fechada (Cury 2001). Introduz-se a seguir um conceito apresentado por Ramadge e Wonham (1989) que é utilizado na solução do PCS.

Dados um SED S sobre um alfabeto Σ, com Σ particionado em Σ_c (eventos controláveis) e Σ_{nc} (eventos não controláveis), e a linguagem $K \subseteq L_S$, diz-se que K é controlável em relação a (e.r.a.) L_S e Σ_{nc} se e somente se, para todo $s \in \overline{K}$ e $\sigma \in \Sigma_{nc}$, se $s\sigma \in L_S$ então $s\sigma \in \overline{K}$ (Ramadge e Wonham 1989). Em palavras, K é controlável e.r.a. L_S e Σ_{nc} se e somente se, para qualquer prefixo s de uma palavra de K e evento não controlável σ, se $s\sigma$ é fisicamente possível na planta, então $s\sigma$ é obrigatoriamente um prefixo de alguma palavra de K. De forma compacta, diz-se que K é controlável e.r.a. L_S e Σ_{nc} se e somente se

$$\overline{K}\Sigma_{nc} \cap L_S \subseteq \overline{K}, \tag{2.17}$$

onde $\overline{K}\Sigma_{nc} = \{s\sigma : s \in \overline{K} \text{ e } \sigma \in \Sigma_{nc}\}$. Quando está subentendido qual é o sistema em questão, diz-se simplesmente que a linguagem é controlável. Observe que $\emptyset$ e L_S são controláveis (Ramadge e Wonham 1989).

Exemplo 2.4 (Linguagem controlável (Cury 2001)) *Seja a máquina de três estados no exemplo 2.3, com* $L_S = (\alpha\beta + \alpha\lambda\mu)^*(\epsilon + \alpha + \alpha\lambda)$. *Considere a linguagem* $K_1 = (\alpha\lambda\mu)^*$. *Escrevem-se:*

$$\overline{K_1} \;=\; (\alpha\lambda\mu)^*(\epsilon + \alpha + \alpha\lambda) \text{ e}$$

$$\overline{K_1}\Sigma_{nc} \cap L_S \;=\; (\alpha\lambda\mu)^*(\alpha\beta + \alpha\mu).$$

Observe que a palavra $\alpha\beta$ *não pertence ao prefixo de* K_1, *então,* K_1 *é não controlável e.r.a.* L_S *e* Σ_{nc}. *Considere agora a linguagem* $K_2 = \{\alpha\beta, \alpha\mu\}$. *Escrevem-se:*

$$\overline{K_2} \;=\; \{\epsilon, \alpha, \alpha\beta, \alpha\mu\} \text{ e}$$

$$\overline{K_2}\Sigma_{nc} \cap L_S \;=\; \{\alpha\beta, \alpha\mu\}.$$

Então, K_2 *é controlável e.r.a.* L_S *e* Σ_{nc}.

Dados um SED S e uma linguagem $K \subseteq L_{S,m}$, diz-se que K é realizável por supervisor não bloqueante para S se existe um supervisor não bloqueante f para S tal que $L_{f/S,m} = K$. A proposição 2.1 apresenta um resultado fundamental para a existência de supervisores não bloqueantes que realizem uma dada linguagem.

Proposição 2.1 (Existência de supervisores (Wonham 2002b)) *Dados um SED* S *sobre o alfabeto* Σ *e com conjunto de eventos não controláveis* $\Sigma_{nc} \subseteq \Sigma$, *e uma linguagem-alvo* $K \subseteq L_{S,m}$, *existe um supervisor não bloqueante* (f, L_m) *para* S *tal que* $L_{f/S,m} = K$ *se e somente se* K *for controlável e.r.a.* L_S *e* Σ_{nc}.

Para uma linguagem $K \subseteq L_{S,m}$, seja

$$C_{(L_S,\Sigma_{nc})}(K) \;=\; \{J \subseteq K : \overline{J}\Sigma_{nc} \cap L_S \subseteq \overline{J}\}, \tag{2.18}$$

o conjunto das linguagens controláveis e.r.a. L_S e Σ_{nc} contidas em K. Em (Ramadge e Wonham 1989) prova-se que o conjunto $C_{(L_S,\Sigma_{nc})}(K)$ é não vazio e fechado em relação à união, possuindo assim um único elemento supremo, a máxima linguagem controlável e.r.a. L_S e Σ_{nc} contida em K, denotada $\sup C_{(L_S,\Sigma_{nc})}(K)$. A proposição 2.2 formaliza a condição de existência de solução para o PCS da definição 2.1.

Proposição 2.2 (Solução do PCS (Wonham 2002b)) *O PCS da definição 2.1 possui solução se e somente se* $\sup C_{(L_S,\Sigma_{nc})}(K) \supset \emptyset$.

A linguagem $\sup C_{(L_S,\Sigma_{nc})}(K)$ representa o comportamento menos restritivo possível que se pode realizar no sistema S para atender à linguagem-alvo K. Qualquer que seja a linguagem K' controlável e.r.a. L_S e Σ_{nc} tal que $\emptyset \subset K' \subset \sup C_{(L_S,\Sigma_{nc})}(K)$, tem-se que K' é mais restritiva ao sistema que $\sup C_{(L_S,\Sigma_{nc})}(K)$. Por outro lado, para qualquer linguagem K'' tal que $\sup C_{(L_S,\Sigma_{nc})}(K) \subset K'' \subseteq K$, tem-se que embora K'' atenda a K, K'' não pode ser realizada por supervisor não bloqueante para S. Assim, um supervisor (f, L_m) tal que $L_{f/S,m} = \sup C_{(L_S,\Sigma_{nc})}(K)$ é ótimo, no sentido de restringir o sistema de forma mínima possível para seguir o especificado pela linguagem-alvo K. Pode-se fazer $L_m = \sup C_{(L_S,\Sigma_{nc})}(K)$ para o supervisor, por exemplo. Assim, uma representação do supervisor ótimo é um autômato que marque $\sup C_{(L_S,\Sigma_{nc})}(K)$.

Na abordagem RW original, o supervisor não possui a habilidade de desmarcar palavras, como o supervisor marcador (Ramadge e Wonham 1989). O supervisor é então representado apenas pela função f definida anteriormente. A linguagem marcada em malha fechada, antes definida conforme a equação (2.16), passa a ser definida por $L_{f/S,m} = L_{f/S} \cap L_{S,m}$, isto é, esta contém todas as palavras marcadas em malha aberta que *sobrevivem* à supervisão. Assim outro requisito surge no resultado equivalente à proposição 2.1 para que uma dada linguagem seja realizável por supervisor não bloqueante (Wonham 2002b). O requisito adicional diz que a linguagem K deve ser $L_{S,m}$-fechada, condição definida por $K = \overline{K} \cap L_{S,m}$, o que significa que qualquer palavra do prefixo de K que corresponda a uma palavra marcada da planta, deve ser também uma palavra pertencente a K (Wonham 2002b). Isso mostra que se perde a

habilidade de especificar uma palavra marcada em malha aberta como não marcada em malha fechada, como no caso da definição de uma tarefa como sendo a fabricação de um lote de peças, não apenas uma peça, comentado anteriormente. Assim, torna-se mais complexo escrever uma especificação que faça referência ao corte ou modificações de tarefas em malha aberta. Essa forma de definir o controle supervisório também acaba por gerar mais complexidade no formalismo. Por exemplo, a solução ótima para o equivalente ao PCS da definição 2.1 passa corresponder ao cálculo da máxima linguagem controlável e $L_{S,m}$-fechada contida numa dada linguagem (Ramadge e Wonham 1989).

2.5.5 O método de síntese de supervisores

Enuncia-se o método básico para síntese de supervisores proposto por Ramadge e Wonham (1989) como solução do PCS na definição 2.1. O método consiste em três passos, a saber (Cury 2001):

1. Obtenção do modelo da planta a ser controlada. Isso consiste em se obter o autômato G que representa o comportamento da planta, mais o particionamento do alfabeto do sistema em subconjuntos de eventos controláveis e não controláveis. Para um sistema composto por diversos componentes, esta etapa corresponde à descrição local dos autômatos e particionamento dos eventos para cada componente, seguida da obtenção do autômato do sistema completo por composição destes autômatos.

2. Obtenção de um modelo das especificações a serem respeitadas em malha fechada. Isto consiste em se obter um autômato R que marca a linguagem-alvo. De forma geral, primeiro modelam-se especificações de coordenação locais aos componentes do sistema. Em seguida, faz-se a composição destas especificações com o modelo da planta. O autômato resultante ainda pode ser modificado para incluir especificações que levem em conta o comportamento global da planta, como é o caso de especificações de estado proibido.

3. Síntese da lógica de controle não bloqueante e ótima. Esta etapa consiste no cálculo da máxima linguagem controlável contida na linguagem-alvo.

No apêndice A ilustra-se o método de síntese pela solução de um problema de controle supervisório para uma célula de manufatura.

Um quarto passo adicional ao método acima seria a implementação da lógica de controle obtida num programa de controle real. O autômato que marca a máxima linguagem controlável pode ser usado para gerar um programa de controle que implementa a supervisão desejada, usando-se, por exemplo, um jogador de autômatos (Cury 2001). Para sistemas flexíveis de manufatura comandados por controladores lógico-programáveis (CLP) existem algumas experiências de implementação de supervisores em diferentes arquiteturas de controle, como por exemplo, em (Brandin 1996), (Lauzon et al. 1996), (Leduc 1996), (Fabian e Hellgren 1998), (Martins e Cury 1997), (Torrico 1999) e (de Queiroz e Cury 2002b). A área de implementação de supervisores para sistemas reais é ainda um ponto que está sendo explorado na literatura.

2.5.6 Comentários sobre a abordagem RW

A abordagem RW fornece então um método de síntese de controladores lógicos para SEDs. Entretanto, esta abordagem possui alguns problemas intrínsecos que dificultam sua aplicação para sistemas reais (Cury et al. 2001).

O primeiro problema diz respeito à complexidade computacional da síntese de supervisores. Para comentar este problema introduz-se a notação da complexidade assintótica, que é uma possível forma de medir o custo computacional de um algoritmo (Rudie 1988). Diz-se que uma função $f(n)$ é de complexidade $\mathcal{O}(g(n))$ quando, para algum N e algum M, inteiros não nulos, $|f(n)| < M \cdot |g(n)|$ para todo $n \geq N$, onde $|f(n)|$ representa o tempo (ou espaço) requerido para completar o algoritmo com entrada de tamanho n (Rudie 1988). Diz-se que $f(n)$ é de complexidade polinomial se $g(n)$ for um polinômio em n, e de complexidade exponencial se $g(n)$ for uma função exponencial em n. Por exemplo, para o cálculo da máxima linguagem controlável, para a planta

representada por um autômato de n estados e as especificações representadas por um autômato de m estados, a complexidade é $\mathcal{O}(n^2m^2)$ (Ramadge e Wonham 1989).

Em síntese, pode-se afirmar que todos os algoritmos envolvidos na síntese de supervisores na abordagem RW são de complexidade polinomial em relação ao número de estados do autômato que representa a composição síncrona da planta com as especificações (Ramadge e Wonham 1989). Entretanto, o número de estados deste autômato depende exponencialmente do número de componentes da planta e especificações, pois o autômato corresponde à composição síncrona dos autômatos que representam esses componentes. Assim, embora a complexidade da síntese seja polinomial em relação ao número de estados do autômato que representa a composição síncrona da planta com as especificações, esta é exponencial em relação ao número de componentes da planta e especificações. Isso também influi no tamanho, em número de estados, do supervisor resultante da síntese, que também fica limitado pelo tamanho do autômato que representa a composição síncrona da planta com as especificações. Por exemplo, considere uma planta com 10 componentes, representados por autômatos de 2 estados, e especificações de coordenação somando 10, também representadas por autômatos de 2 estados. Assim, o valor que limita o número de estados do supervisor resultante da síntese é $2^{20} = 1.048.576$.

Existem na literatura diversas extensões para a abordagem RW que tratam de técnicas de decomposição estrutural da planta e especificações para lidar com a complexidade computacional inerente à síntese de supervisores. Citam-se por exemplo, a abordagem de controle modular (Ramadge e Wonham 1989), o controle modular local (de Queiroz e Cury 2002a), a exploração da simetria (González et al. 2001) e o controle hierárquico.

Outro problema que limita a aplicabilidade do método de síntese de supervisores para SEDs da abordagem RW é a legibilidade do programa de controle gerado (Cury 2001). Observa-se que até mesmo para aplicações industriais de pequeno porte, o supervisor calculado pela abordagem RW possui potencialmente de centenas a milhares de estados, vide, por exemplo, (Wonham 2002b). Assim, torna-se complexo para o

engenheiro projetista de tais sistemas analisar heuristicamente a política de controle resultante. Assim, encontra-se uma certa resistência por parte do engenheiro projetista a aplicar este método (Cury 2001). A modularização do controlador é uma possível solução para este problema, gerando pequenos controladores locais no lugar de um controlador único, cuja política de controle individual pode ficar facilmente legível para o projetista (Cury 2001). Entretanto, existem casos para os quais essa modularização não se aplica, por não cumprimento de certas condições (de Queiroz e Cury 2002a). Nesses casos, outras técnicas, como o controle hierárquico, podem ser aplicadas como será visto nos capítulos seguintes.

Existem diversas ferramentas para a síntese de supervisores para SEDs dentro da abordagem RW e suas extensões. Citam-se, por exemplo, a ferramenta **TCT** desenvolvida pela Universidade de Toronto (Wonham 2002a), a **UMDES Software Lybrary** desenvolvida pela Universidade de Michigan (UMDES Group 1998), a **DESCO** desenvolvida pela Universidade de Chelmers (Boel et al. 2002, CDES Group 2002), a ferramenta **VALID** desenvolvida pela Siemens (Boel et al. 2002), a ferramenta **CONDES** desenvolvida pela Universidade Federal de Santa Catarina (Torrico 1999), e a ferramenta de controle supervisório baseada na ferramenta **GRAIL** desenvolvida pela Universidade Federal de Santa Catarina (vide apêndice A). Na sua maioria, são ferramentas de pesquisa para o meio acadêmico, sem fins comerciais.

Capítulo 3

Revisão bibliográfica do controle hierárquico de SEDs

Neste capítulo faz-se uma revisão dos principais formalismos do controle hierárquico de SEDs.

A organização do capítulo é como segue. Na seção 3.1 discute-se o problema de controle hierárquico para SEDs. Na seção 3.2 apresentam-se os resultados da abordagem de Zhong e Wonham (1990) para o controle hierárquico formulado para linguagens prefixo-fechadas. Na seção 3.3 descrevem-se os resultados de Wong e Wonham (1996a) e Wong e Wonham (1998) para a formulação do problema de controle hierárquico para linguagens marcadas. Na seção 3.4 mostram-se os resultados de Pu (2000) para o controle hierárquico baseado abstrações consistentes. Por fim, na seção 3.5 discutem-se os principais pontos das abordagens apresentadas.

3.1 O problema de controle hierárquico

Nesta seção, apresenta-se a formulação do problema de controle hierárquico para SEDs.

Considera-se o problema de controle hierárquico reduzido a dois níveis, sem perda de generalidade. Isso se dá porque as propriedades são avaliadas entre dois níveis consecutivos, o que as torna extensíveis a uma hierarquia de vários níveis. Adota-se,

conforme em (Zhong e Wonham 1990), a metáfora de uma fábrica, onde se define um nível de abstração para o operador e um para o gerente.

No esquema de supervisão hierárquica de dois níveis representado na figura 3.1, S_{op} é um SED dotado de controle que corresponde a um processo do ponto de vista de um operador de uma fábrica, e S_{ge} é um outro SED dotado de controle que representa o mesmo processo do ponto de vista de um gerente. O sistema S_{ge} é uma abstração de S_{op}, obtida pelo canal de informação Inf_{og}, escreve-se $S_{ge} = Inf_{og}(S_{op})$. Seja f_{ge} um supervisor projetado para S_{ge} utilizando informação do canal Inf_{ge}, escreve-se a malha fechada para o gerente como sendo f_{ge}/S_{ge}. O supervisor f_{ge} exerce controle *virtual* sobre S_{ge}, isto é, o canal de controle Con_{ge} não existe efetivamente (representado pelo tracejado). O controle é exercido *de fato* pelo supervisor do operador f_{op}, que controla S_{op} utilizando os canais de controle Con_{op}, informação Inf_{op} e comando Com_{go}, este último a informar a diretiva de controle que está em vigor no momento para f_{ge}. Deseja-se que o controle exercido pelo operador faça com que a malha fechada do operador f_{op}/S_{op}, ao ser reportada ao gerente, $Inf_{og}(f_{op}/S_{op})$, siga o mais próximo possível a malha fechada virtual f_{ge}/S_{ge}.

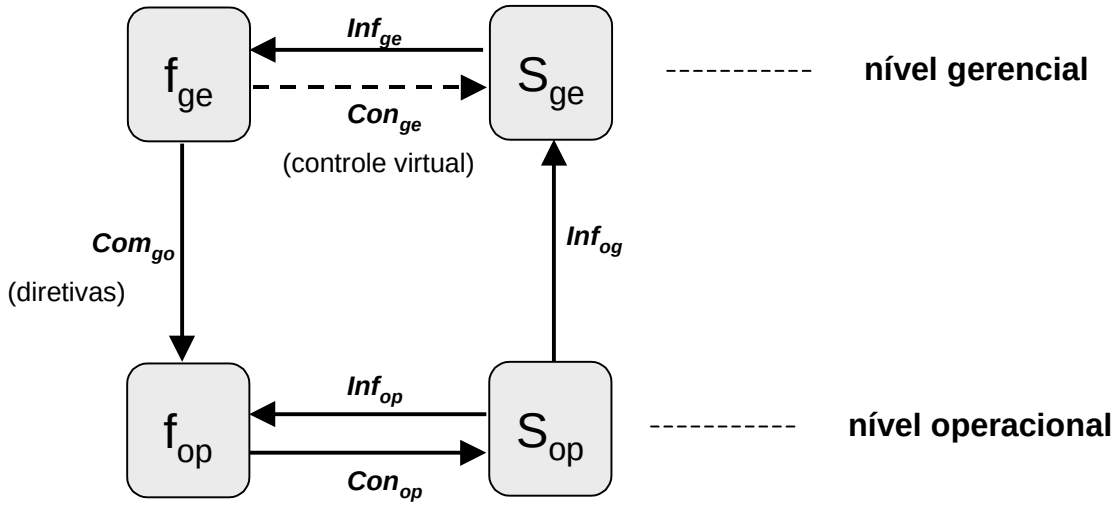


Figura 3.1: Esquema de supervisão hierárquica.

Quando todo comportamento realizável por supervisor não bloqueante para o

gerente f_{ge}/S_{ge} for imagem, através do canal de informação, de um comportamento realizável por supervisor não bloqueante para o operador f_{op}/S_{op}, denotado $Inf_{og}(f_{op}/S_{op}) = f_{ge}/S_{ge}$, diz-se que há *consistência hierárquica* entre S_{op} e S_{ge} (Zhong e Wonham 1990). Quando, adicionalmente à condição de consistência hierárquica, todos os comportamentos realizáveis por supervisor não bloqueante para o operador possuam imagens, através do canal de informação, que correspondam a comportamentos realizáveis por supervisor não bloqueante para o gerente, diz-se que há *consistência hierárquica forte* entre S_{op} e S_{ge} (Wong e Wonham 1996a).[1] A consistência hierárquica propriamente dita permite que haja comportamentos que sejam diagnosticados como sendo não realizáveis por supervisor não bloqueante para o gerente, mas que sejam imagens de comportamentos realizáveis por supervisor não bloqueante para o operador. Isto corresponde a uma perda de detalhamento de informação de controle do operador para o gerente. Já a consistência hierárquica forte garante uma correspondência completa entre comportamentos realizáveis por supervisor não bloqueante para o operador e o gerente.

O *problema de controle hierárquico*, consiste na determinação do modelo do gerente com o suficiente refinamento de informação para que ocorra a consistência hierárquica em alguma de suas gradações. O projeto do esquema de supervisão hierárquica consiste em, partindo do sistema do operador S_{op} e de um alfabeto para o gerente, calcular o sistema do gerente S_{ge}, conforme o tipo de consistência hierárquica desejado, o que corresponde ao projeto do canal de informação Inf_{og}. Segue-se o projeto do supervisor para o gerente f_{ge}, em função das especificações feitas neste nível. Por fim, realiza-se projeto do supervisor para o operador f_{op}.

Nas seções que seguem, são apresentadas as principais abordagens para o problema de controle hierárquico encontradas na literatura.

[1] Condição nomeada *hierarchical consistency* por Wong e Wonham (1996a).

3.2 Abordagem que considera apenas o comportamento gerado dos sistemas

Nesta seção apresentam-se os principais resultados da abordagem de Zhong e Wonham (1990) para o controle hierárquico de SEDs.

Para o esquema de supervisão hierárquica da figura 3.1, seja Σ o alfabeto do operador S_{op}. Representa-se S_{op} por um par $(L_{S_{op}}, \Gamma_{S_{op}})$, onde $L_{S_{op}}$ é uma linguagem prefixo-fechada em Σ, e $\Gamma_{S_{op}} \subseteq 2^{\Sigma}$ é uma estrutura de controle. O alfabeto do operador Σ é particionado num subconjunto controlável Σ_c e um subconjunto não controlável Σ_{nc}. A estrutura de controle $\Gamma_{S_{op}}$ corresponde então a um conjunto de entradas de controle $\gamma \subseteq \Sigma$, a significar um possível conjunto de eventos habilitados em S_{op}, as entradas de controle $\gamma \in \Gamma_{S_{op}}$ são tais que $\Sigma_{nc} \subseteq \gamma$ (vide subseção 2.5.2).

Seja T o alfabeto para o gerente, não necessariamente contido no alfabeto do operador e possivelmente distinto. Modela-se o canal de informação Inf_{og} por um *mapa repórter* definido a seguir.

Definição 3.1 (Mapa Repórter (Zhong e Wonham 1990)) *Um mapa repórter é uma função $\theta : L_{S_{op}} \rightarrow T^*$ definida recursivamente por $\theta(\epsilon) = \epsilon$ e :*

$$\theta(s\sigma) \;=\; \begin{cases} \theta(s) & ou \\ \theta(s)\tau & \end{cases} \tag{3.1}$$

onde ϵ representa a palavra vazia, $s \in L_{S_{op}}$, $\sigma \in \Sigma$ e $\tau \in T$.

O mapa repórter, com base na seqüência de eventos s gerada por S_{op}, gera a seqüência de eventos $\theta(s)$ em T^*. A cada novo evento gerado pelo operador, o mapa repórter ou *silencia* ou informa ao gerente a ocorrência de um novo evento $\tau \in T$, diz-se então que o mapa repórter *vocaliza* o evento τ. O mapa repórter possui a propriedade de ser *causal*, isto é, dados s e $s' \in L_{S_{op}}$, se s é prefixo de s', então $\theta(s)$ é prefixo de $\theta(s')$ (Zhong e Wonham 1990).

Define-se o *evento silencioso* τ_0, com $\tau_0 \notin \Sigma$ e $\tau_0 \notin T$, como sendo um evento fictício gerado quando, à ocorrência de um novo evento no nível operacional, o mapa repórter silencia, isto é, não gera a informação da ocorrência de um novo evento no gerente.

Representa-se o comportamento do operador mais o canal de informação por um autômato de Moore, vide seção 2.4.7,

$$G_{op} \quad = \quad (\Sigma, Q, \delta, q_0, Q_m, T \cup \{\tau_0\}, w) \tag{3.2}$$

onde, Σ é o alfabeto de entrada, $T \cup \{\tau_0\}$ é o alfabeto de saída, Q é o conjunto de estados, $\delta : Q \times \Sigma \rightarrow Q$ é a função de transição, parcial e determinista, q_0 é o estado inicial, Q_m é o conjunto de estados marcados, e $w : Q \rightarrow T \cup \{\tau_0\}$ é a função de saída (Zhong e Wonham 1990). Define-se G_{op} de tal forma que $L_{G_{op}} = L_{G_{op},m} = L_{S_{op}}$, isto é, suas linguagens gerada e marcada são iguais à linguagem gerada por S_{op}, e a função de saída w é definida recursivamente como $w(q_0) = \tau_0$ e

$$w(\hat{\delta}(q_0, s\sigma)) \quad = \quad \begin{cases} \tau_0 & \text{se } \theta(s\sigma) = \theta(s) \\ \tau & \text{se } \theta(s\sigma) = \theta(s)\tau \end{cases} \tag{3.3}$$

onde $s \in L_{S_{op}}$, $\sigma \in \Sigma$, $\tau \in T$ e $\hat{\delta}$ é a extensão da função de transição δ para palavras em Σ^*, ver equação (2.9) na seção 2.4.2. Assim, os estados de G_{op} em que a função de saída é diferente de τ_0 correspondem às palavras geradas por S_{op} em que o mapa repórter notifica a ocorrência de um novo evento para o gerente.

Adicionalmente, representa-se o SED S_{op} mais o canal de informação Inf_{og} pelo par (G_{op}, Γ_{op}), onde G_{op} é o autômato de Moore que representa o comportamento de S_{op} mais o canal de informação e Γ_{op} é a estrutura de controle de S_{op}.

Exemplo 3.1 (Autômato de Moore) *Considere o autômato de Moore G_{op} da figura 3.2, a representar o comportamento de um sistema no nível operacional S_{op}, mais o canal de informação Inf_{og}. Adicionalmente a estrutura de controle do sistema Γ_{op} é representada sobre o diagrama de transição, sendo os eventos controláveis identifica-*

dos pelas transições cujas setas são cortadas por uma linha. Da figura 3.2, extrai-se a seguinte informação sobre G_{op}*:* $\Sigma = \{a, b, c, d, e\}$*,* $Q = \{0, 1, 2, 3, 4\} = Q_m$*,* $q_0 = 0$*,* $T = \{A, B\}$*, e sobre a estrutura de controle* Γ_{op}*:* $\Sigma_c = \{c, e\}$ *e* $\Sigma_{nc} = \{a, b, d\}$*. Observe que, enquanto o operador gera seqüências do tipo* $abcde\ldots$*, o mapa repórter associado gera seqüências do tipo* $ABA\ldots$*.*

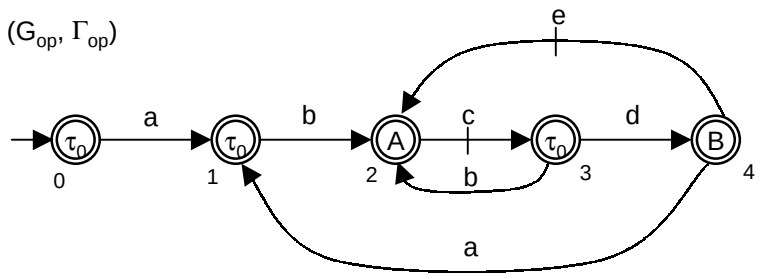


Figura 3.2: Autômato de Moore mais estrutura de controle.

Uma vez estabelecidas as representações do SED do operador e do canal de informação, parte-se para a definição do comportamento e da estrutura de controle do SED para o gerente S_{ge} no esquema de supervisão hierárquica da figura 3.1.

Para introduzir o comportamento do gerente, define-se a θ-imagem de uma linguagem $E_{op} \subseteq L_{S_{op}}$ por:

$$\theta(E_{op}) \;=\; \{t \in T^* : (\exists s \in E_{op})\ t = \theta(s)\}. \tag{3.4}$$

A θ-imagem da linguagem E_{op} possui a propriedade de preservar prefixos, isto é, $\theta(\overline{E_{op}}) = \overline{\theta(E_{op})}$ (Wong e Wonham 1996a). Assim, representa-se o comportamento do gerente S_{ge} pela linguagem $L_{S_{ge}} = \theta(L_{S_{op}})$, isto é, $L_{S_{ge}}$ é a θ-imagem da linguagem $L_{S_{op}}$. Repare que, pela propriedade comentada acima, como $L_{S_{op}}$ é prefixo-fechada, então $L_{S_{ge}}$ é prefixo-fechada.

Exemplo 3.2 (Comportamento do gerente) *A figura 3.3 representa a construção de um autômato* G_{ge} *que marca a linguagem do SED do gerente correspondente ao*

operador e canal de informação representados pelo autômato de Moore G_{op} da figura 3.2. A construção de G_{ge} se dá em três etapas. Primeiramente constrói-se o autômato G_θ a partir de G_{op} substituindo-se as etiquetas de transição de G_{op} pelas etiquetas de saída dos respectivos estados onde chegam as transições. Em seguida, faz-se a projeção de G_θ no alfabeto do gerente T, $G'_{ge} = proj(G_\theta, \{A, B\})$ (subseção 2.4.6). O autômato G'_{ge} é possivelmente não determinista. Por fim, obtém-se o autômato G_{ge} por construção do equivalente determinista e mínimo de G'_{ge} (subseção 2.4.4). O autômato G_{ge} é tal que $L_{G_{ge}} = L_{G_{ge},m} = \theta(L_{G_{op}})$, onde θ é o mapa repórter associado ao autômato de Moore G_{op}.

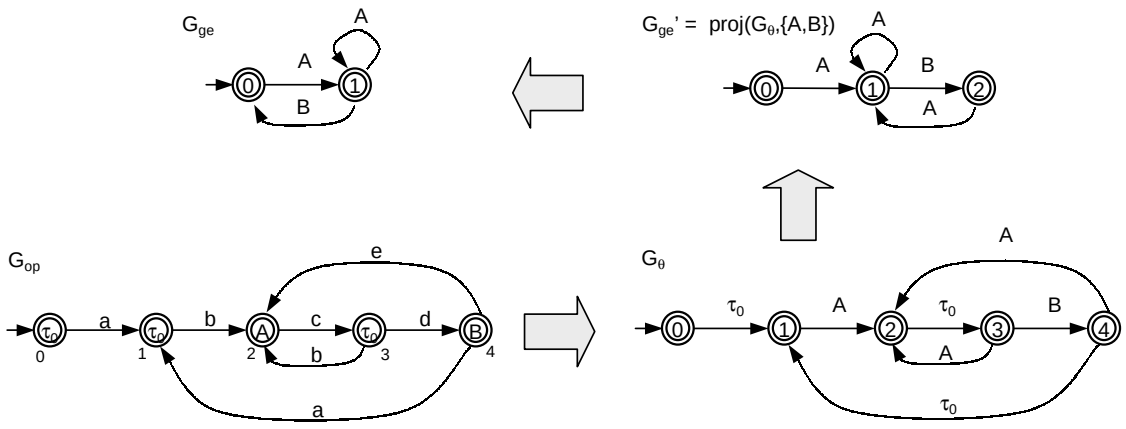


Figura 3.3: Modelo do comportamento do gerente.

Considere a seguinte discussão para a estrutura de controle do gerente, ilustrada por G_{op} e G_{ge} nas figuras 3.2 e 3.3. Suponha que o gerente desabilita um evento em G_{ge} por intermédio da proibição de que se alcance um estado no sistema do operador cuja saída é o evento em questão. Na figura 3.2, observe que as saídas A ou B podem ou não ser desabilitadas em G_{op} dependendo do estado corrente. Por exemplo, após o estado 4 de G_{op} pode-se impedir a transição para o estado 2, e conseqüente vocalização da saída A, por desabilitação do evento e. Entretanto, não se pode impedir a transição para o estado 2 a partir dos estados 0, 1 e 3, pois os correspondentes eventos destas transições são não controláveis. Portanto, neste exemplo, o nível gerencial não possui

estrutura de controle de acordo com a abordagem RW, ou seja, não se pode particionar T em T_c (eventos controláveis) e T_{nc} (eventos não controláveis). Neste caso, diz-se que S_{op} e θ não possuem *consistência de controle*[2] (Zhong e Wonham 1990).

Em (Zhong e Wonham 1990), propõe-se uma forma de refinamento de G_{op} para se definir uma estrutura de controle consistente no nível gerencial. O refinamento se dá por modificação da estrutura de transição de G_{op}, extensão e partição do alfabeto T para que existam subconjuntos de eventos controláveis e não controláveis. Em (Zhong e Wonham 1990) usa-se a *árvore de alcançabilidade* de $L_{S_{op}}$ para definir o conceito de consistência de controle e explicar o procedimento de refinamento. Por julgar que tal opção exige a apresentação e a explicação de um formalismo adicional, opta-se por uma forma de apresentação simplificada, baseada em material que encontrado em (Wong e Wonham 1998). Considere então o seguinte desenvolvimento.

Primeiramente define-se um *mapa vocal*[3] $\hat{w} : L_{S_{op}} \rightarrow T \cup \{\tau_0\}$ como sendo uma extensão do mapa de saída de G_{op} para palavras em $L_{S_{op}}$, ou seja,

$$\hat{w}(s) \quad = \quad w(\hat{\delta}(q_0, s)) \tag{3.5}$$

para $s \in L_{S_{op}}$ (Zhong e Wonham 1990). Representa-se $\hat{w}$ por w quando o contexto permitir.

Define-se uma *palavra vocal* de S_{op} como sendo uma palavra s em $L_{S_{op}}$ tal que $w(s) \neq \tau_0$ ou a palavra vazia ϵ. Define-se a *linguagem vocal* $L_{S_{op},voc} \subseteq L_{S_{op}}$ como sendo a linguagem de todas as palavras vocais em S_{op}, isto é,

$$L_{S_{op},voc} \quad = \quad \{\epsilon\} \cup \{s \in L_{S_{op}} : w(s) \neq \tau_0\}. \tag{3.6}$$

As palavras geradas por S_{op} que não são vocais, são ditas *palavras silenciosas* de S_{op}. As palavras vocais são as palavras geradas por S_{op} nas quais há uma atualização do sistema S_{ge}, seja pela inicialização, com a palavra vazia, seja pela geração de um novo

[2]No original em inglês *output-control-consistency*.
[3]No original em inglês, *tail map*.

evento pelo mapa repórter. No autômato de Moore G_{op}, os estados correspondentes às palavras vocais são chamados *estados vocais* de G_{op}. Por exemplo, para G_{op} na figura 3.2, os estados vocais são 0, 2 e 4.

Para o evento τ em T, um *trecho silencioso* correspondente a τ é uma seqüência não vazia de eventos u em Σ^+ para a qual existe uma palavra vocal s de S_{op} tal que: (i) su é uma palavra vocal de S_{op}, (ii) $w(su) = \tau$, e (iii) para todo prefixo estrito e não vazio de u, diga-se u', a palavra su' é silenciosa (Wong e Wonham 1998). Define-se formalmente o conjunto de trechos silenciosos correspondentes ao evento τ em T por:

$$\begin{aligned}\mathcal{X}_\tau \;=\; \{u \in \Sigma^+ : (\exists s \in L_{S_{op},voc}) \text{ tal que } \; & (su \in L_{S_{op}}) \text{ e } (w(su) = \tau) \text{ e} \\ & (((\forall u' \in \Sigma^+) \text{ tal que } u' < u)\; w(su') = \tau_0)\}\end{aligned} \tag{3.7}$$

Por exemplo, para o autômato de Moore da figura 3.2, os trechos silenciosos correspondentes ao evento A são ab, cb e e. Para o evento τ em T, diz-se que um trecho silencioso u em $\mathcal{X}_\tau$ é *não controlável* quando u é composto apenas por eventos não controláveis, isto é, $u \in \Sigma^+_{nc}$, e diz-se que u é *controlável* quando há algum evento controlável em u, isto é, $u \in \Sigma^+ - \Sigma^+_{nc}$.

Definição 3.2 (Consistência de Controle (Zhong e Wonham 1990)) *O par (S_{op}, θ) possui consistência de controle se o alfabeto do gerente T puder ser particionado num subconjunto de eventos não controláveis (T_{nc}) e num subconjunto de eventos controláveis (T_c) da seguinte forma, para $\tau \in T$,*

- *Se $\tau \in T_{nc}$, então, para todo $u \in \mathcal{X}_\tau$, $u \in \Sigma^+_{nc}$, e*
- *Se $\tau \in T_c$, então, para todo $u \in \mathcal{X}_\tau$, $u \in \Sigma^+ - \Sigma^+_{nc}$.*

Com a consistência de controle, se um evento $\tau \in T$ é não controlável então todos os seus trechos silenciosos são não controláveis, caso contrário, se τ é controlável então todos os seus trechos silenciosos são controláveis. Por análise do autômato de Moore na figura 3.2, o único trecho silencioso correspondente ao evento B é cd, com c controlável,

assim, considera-se o evento B como controlável. Entretanto o mesmo não pode ser feito para o evento A, pois este possui por trechos silenciosos cb e e, que são controláveis, e ab, que é não controlável

Zhong e Wonham (1990) propõem um procedimento que, dado um SED do operador com canal de informação (G_{op}, Γ_{op}), possivelmente sem consistência de controle, e um alfabeto do gerente T, constrói-se um novo sistema do operador e respectivo canal de informação $(G_{op,ext}, \Gamma_{op})$ e um novo alfabeto do gerente T_{ext} com consistência de controle. Basicamente o procedimento modifica as etiquetas dos eventos τ em T para τ_{nc} (instância não controlável) ou τ_c (instância controlável) em T_{ext} e modifica a estrutura de transição de G_{op}, para que todos os trechos silenciosos correspondentes aos eventos em T_{ext} sejam ou não controláveis ou controláveis. Na figura 3.4, representa-se o sistema do operador $(G_{op,ext}, \Gamma_{op})$ e o sistema do gerente correspondente (G_{ge}, Γ_{ge}) resultantes da aplicação do procedimento de consistência de controle para o sistema da figura 3.2. No processamento para garantir a consistência de controle estrita com G_{op} com n estados e m transições, o autômato $G_{op,ext}$ possui até $2n$ estados, sendo o procedimento de complexidade $\mathcal{O}(nm^2)$ (Zhong e Wonham 1990).

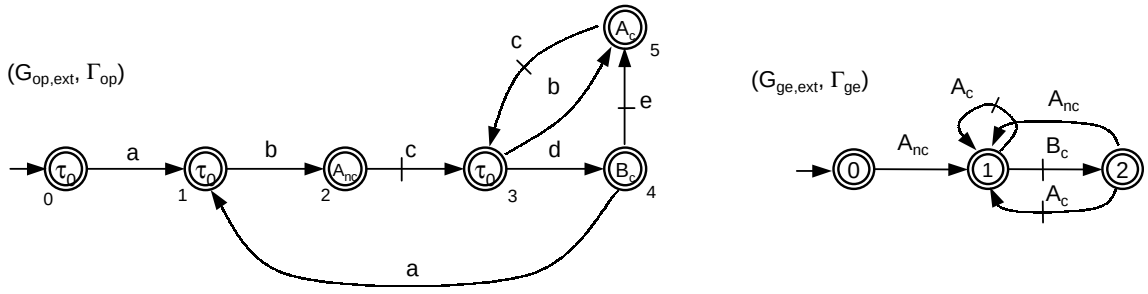


Figura 3.4: Consistência de controle.

Sejam o sistema do operador S_{op} e mapa repórter θ com consistência de controle, e o sistema do gerente S_{ge} correspondente, isto é, $\theta(L_{S_{op}}) = L_{S_{ge}}$ e o alfabeto do gerente T particionado em um subconjunto não controlável (T_{nc}) e um subconjunto controlável (T_c). Seja também $E_{ge} \subseteq L_{S_{ge}}$ controlável e.r.a. $L_{S_{ge}}$ e T_{nc} e prefixo-fechada. Assim,

existe um supervisor f_{ge} para S_{ge} tal que $L_{f_{ge}/S_{ge}} = E_{ge}$, vide subseção 2.5.4. Define-se f_{ge} como sendo $f_{ge}(t) = \Sigma_{E_{ge}}(t)$, para $t \in L_{S_{ge}}$ (Ramadge e Wonham 1989). Como visto na seção 3.1, o controle exercido pelo supervisor f_{ge} é virtual, isto é, o controle é de fato implementado por um supervisor f_{op} para o operador (vide o esquema de supervisão hierárquica representado na figura 3.1). Define-se o supervisor f_{op} para o operador por

$$\begin{aligned} f_{op}(s) \;=\; \Sigma - \{\sigma \in \Sigma_c : (\exists u \in \Sigma_{nc}^*) \text{ tal que } \; & (s\sigma u \in L_{S_{op}}) \text{ e} \\ & (((\forall u' \in \Sigma^+) \text{ tal que } u' < u)\ w(su') = \tau_0) \text{ e} \\ & (w(s\sigma u) \notin f_{ge}(\theta(s)))\} \end{aligned} \tag{3.8}$$

para $s \in L_{S_{op}}$ (Zhong e Wonham 1990). O supervisor f_{op} é tal que, para uma dada palavra s do operador, proíbe a ocorrência de todos os eventos controláveis que iniciam cadeias de eventos não controláveis que levam à vocalização de eventos não admitidos em $\overline{E_{ge}}$ após $\theta(s)$.

Para introduzir o principal resultado em relação a consistência de controle, define-se, para $E_{ge} \subseteq L_{S_{ge}}$, a θ-imagem inversa $\theta^{-1}(E_{ge})$ por

$$\theta^{-1}(E_{ge}) \;=\; \{s \in L_{S_{op}} : (\exists t \in E_{ge})\ \theta(s) = t\}, \tag{3.9}$$

isto é, $\theta^{-1}(E_{ge})$ representa a linguagem de todas as palavras s em $L_{S_{op}}$ tais que $\theta(s)$ pertence a E_{ge} (Zhong e Wonham 1990). Para a palavra $t \in L_{S_{ge}}$, denota-se $\theta^{-1}(\{t\})$ por $\theta^{-1}(t)$ e usa-se a denominação de θ-imagem inversa de t.

Proposição 3.1 (Consistência de controle (Zhong e Wonham 1990)) *Se (S_{op}, θ) possui consistência de controle, então, para toda linguagem controlável e prefixo-fechada $E_{ge} \subseteq L_{S_{ge}}$, o supervisor f_{op} para S_{op}, equação (3.8), é tal que $L_{f_{op}/S_{op}} = \sup C_{(L_{S_{op}}, \Sigma_{nc})}(\theta^{-1}(E_{ge}))$.*

Na proposição 3.1, a linguagem $\sup C_{(L_{S_{op}}, \Sigma_{nc})}(\theta^{-1}(E_{ge}))$ representa a máxima linguagem controlável e.r.a. $L_{S_{op}}$ e Σ_{nc} contida em $\theta^{-1}(E_{ge})$, vide subseção 2.5.4. Como,

para $E_{ge} \subseteq L_{S_{ge}}$, $\theta(\theta^{-1}(E_{ge})) = E_{ge}$, e, para $E_{op} \subseteq L_{S_{op}}$, $\sup C_{(L_{S_{op}},\Sigma_{nc})}(E_{op}) \subseteq E_{op}$, a proposição 3.1 implica $\theta(L_{f_{op}/S_{op}}) \subseteq E_{ge}$, condição chamada *consistência hierárquica de baixo nível*[4] por Zhong e Wonham (1990).

Exemplo 3.3 (Supervisão hierárquica com consistência de controle) *Para os sistemas $S_{op,ext}$ e $S_{ge,ext}$ na figura 3.4, considere a especificação gerencial E_{ge} marcada pelo autômato representado na figura 3.5. Verifica-se que a especificação E_{ge} é prefixo-fechada e controlável e.r.a. $L_{S_{ge,ext}}$ e $T_{ext,nc}$. Pela proposição 3.1, o resultado da supervisão hierárquica, representada na figura 3.5, corresponde à linguagem E_{op} no nível do operador, marcada pelo autômato também representado na figura 3.5. Verifica-se que $\theta_{ext}(E_{op}) \subset E_{ge}$, onde θ é o mapa repórter associado, configurando-se assim uma condição de consistência hierárquica de baixo nível para o esquema de supervisão hierárquica.*

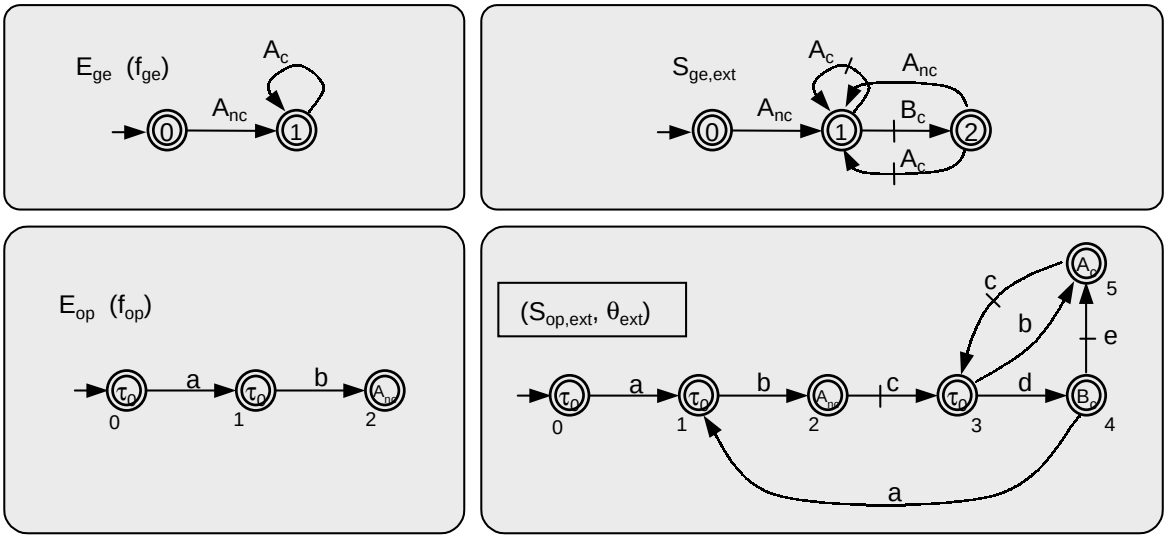


Figura 3.5: Supervisão hierárquica com consistência de controle.

A consistência de controle garante que o alfabeto do gerente pode ser particionado em subconjuntos de eventos controláveis e não controláveis. Entretanto, permanece

[4]No original em inglês, *low level hierarchical consistency.*

um problema estrutural que impede garantir a igualdade do comportamento esperado no gerente (representado por E_{ge} no desenvolvimento anterior) e o comportamento em malha fechada do operador visto pelo gerente (representado pela linguagem $\theta(L_{f_{op}/S_{op}})$). Tal igualdade corresponde à condição de *consistência hierárquica* neste formalismo, vide discussão na seção 3.1. O problema estrutural citado anteriormente resume-se ao fato de que, a desabilitação de um evento pelo gerente acarreta à desabilitação indesejada de um outro evento gerencial, em função do interrelacionamento das formas de desabilitação dos dois eventos no nível do operador. Voltando ao exemplo da figura 3.5, repare que para desabilitar o evento B_c a partir do estado 2 de $G_{op,ext}$ deve-se impedir que $G_{op,ext}$ evolua até o estado 4, o que é feito impedindo-se a ocorrência do evento c na transição (2, c, 3). Entretanto, esta ação acarreta a desabilitação indesejada do evento A_c no estado 5 de $G_{op,ext}$. Zhong e Wonham (1990) associam esse problema estrutural do mapa repórter à presença de *palavras vocais parceiras*[5] no sistema do operador.

Definição 3.3 (Palavras vocais parceiras (Zhong e Wonham 1990)) *Duas palavras vocais s_1 e s_2 de S_{op} são ditas parceiras se*

1. *corresponderem a distintos eventos do gerente,*
2. *corresponderem a trechos silenciosos controláveis que começam na mesma palavra vocal, e*
3. *seus trechos silenciosos correspondentes dividem um mesmo segmento inicial, diga-se $s'\sigma$, com $s' \in \Sigma^*$ e $\sigma \in \Sigma_c$, e pelo menos um dos trechos continua numa palavra em Σ^*_{nc}.*

Exemplo 3.4 (Palavras vocais parceiras) *Na figura 3.6, $\alpha\sigma\alpha\beta\alpha$ e $\alpha\sigma\alpha\alpha\mu$ são palavras vocais parceiras pois: (1) correspondem a eventos distintos para o gerente, quais sejam τ_1 e τ_2 respectivamente, (2) correspondem a trechos silenciosos, $\sigma\alpha\beta\alpha$ e $\sigma\alpha\alpha\mu$*

[5]Originalmente denominadas *partner nodes*, *nós parceiros*, em alusão aos nós da árvore de alcançabilidade de $L_{S_{op}}$.

respectivamente, que são controláveis e começam na mesma palavra vocal, qual seja α*, e (3) seus trechos silenciosos correspondentes dividem um mesmo segmento inicial,* $\sigma \in \Sigma_c$*, e o trecho* $\sigma\alpha\alpha\mu$ *termina na seqüência* $\alpha\alpha\mu$*, que só possui eventos não controláveis. Veja que, ao se desabilitar* τ_2 *após* τ *implica desabilitar* τ_1*.*

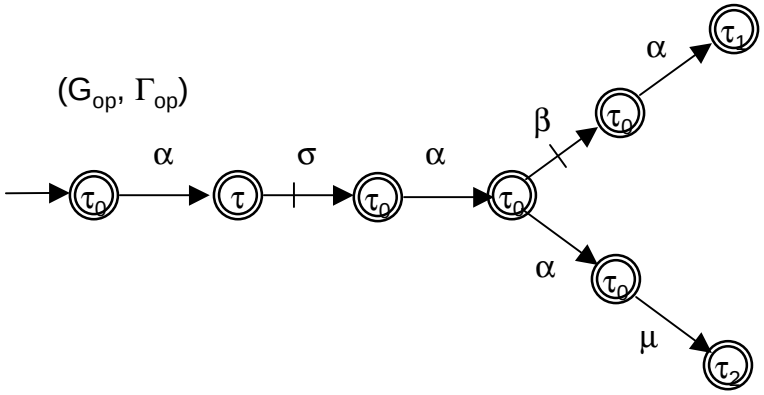


Figura 3.6: Palavras vocais parceiras.

Definição 3.4 (Consistência de controle estrita (Zhong e Wonham 1990)) *Diz-se que* S_{op} *e* θ *possuem consistência de controle estrita*[6] *se possuem consistência de controle e* S_{op} *não possui palavras vocais parceiras.*

Dado um sistema sem consistência de controle estrita, pode-se torná-lo consistente pela vocalização dos estados que seguem os eventos controláveis que causam a existência das palavras parceiras, seguida da reconsideração da controlabilidade dos eventos do gerente (Zhong e Wonham 1990). Por exemplo, na figura 3.7, ao se criar o novo evento τ_3, as palavras vocais $\alpha\sigma\alpha\beta\alpha$ e $\alpha\sigma\alpha\alpha\mu$ deixam de ser parceiras. Na reconsideração da controlabilidade dos eventos gerenciais, τ_1 e τ_3 ficam eventos controláveis, τ permanece não controlável, e τ_2 passa de controlável a não controlável. No procedimento de Zhong e Wonham (1990), com G_{op} com n estados e m transições, o sistema resultante com

[6]No original em inglês, *strictly-output-control-consistency.*

consistência de controle estrita, diga-se $G_{op,novo}$, pode possuir até $4n$ estados, sendo o procedimento de complexidade $\mathcal{O}(n^2m^2)$.

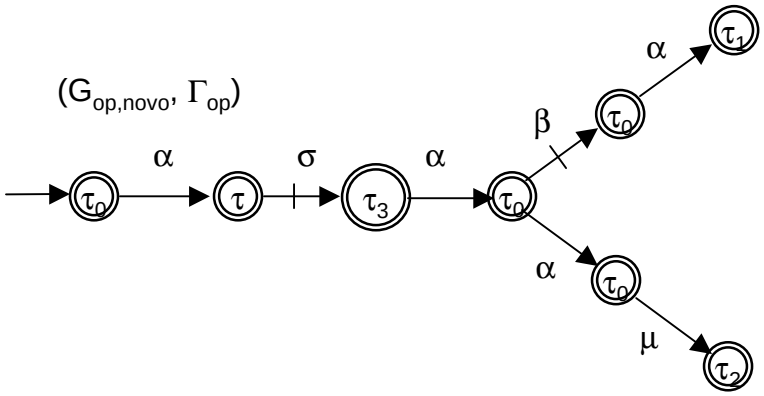


Figura 3.7: Correção das palavras vocais parceiras.

Zhong e Wonham (1990) implementam os procedimentos para tornar um sistema com consistência de controle estrita numa ferramenta computacional para o projeto e análise de SED dotados de controle chamada TCT (Wonham 2002a).

Proposição 3.2 (Consistência hierárquica (Zhong e Wonham 1990)) *Se* (S_{op}, θ) *possui consistência de controle estrita, então, dada* $E_{ge} \subseteq L_{S_{ge}}$ *controlável e prefixo-fechada, o supervisor* f_{op} *para* S_{op}*, equação (3.8), é tal que* $\theta(L_{f_{op}/S_{op}}) = E_{ge}$*.*

A proposição 3.2 assegura que *a consistência de controle estrita implica consistência hierárquica* para o problema de controle hierárquico formulado para linguagens prefixo-fechadas.

Exemplo 3.5 (Consistência de controle estrita) *Na figura 3.8,* $G_{op,novo}$ *é obtido de* $G_{op,ext}$ *na figura 3.4, por aplicação do procedimento de construção de um sistema com consistência de controle estrita de Zhong e Wonham (1990). Veja que o evento* C_c *foi criado para eliminação de palavras vocais parceiras, e a controlabilidade dos eventos do*

gerente foi modificada de acordo com as condições de consistência de controle. Verifica-se que há consistência hierárquica entre $S_{op,novo}$ e $S_{ge,novo}$ de acordo com a proposição 3.2.

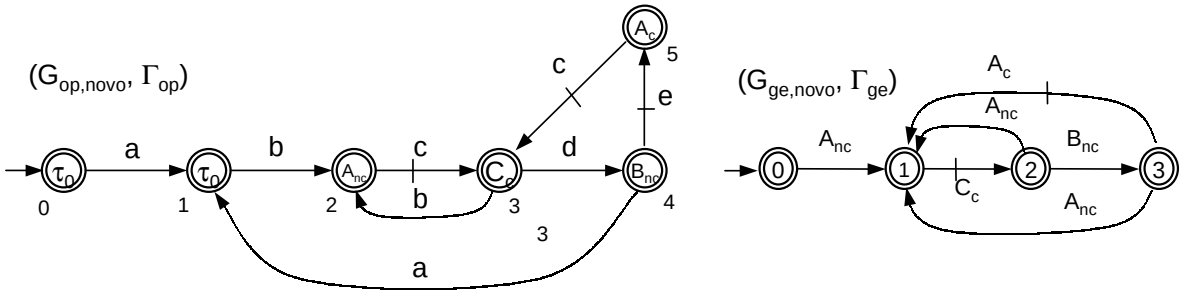


Figura 3.8: Consistência de controle estrita.

Exemplo 3.6 (Não possui consistência hierárquica forte) *Sejam o sistema do operador S_{op} e o mapa repórter θ na figura 3.9, onde é possível se verificar que há consistência de controle estrita, o que implica, pela proposição 3.2, que há consistência hierárquica entre S_{op} e S_{ge}, o sistema do gerente. Seja então a especificação E_{ge} marcada pelo autômato representado na figura 3.9. Verifica-se que E_{ge} não é controlável e.r.a. $L_{S_{ge}}$ e T_{nc} (o conjunto de eventos não controláveis para o gerente), pois não há, por exemplo, como impedir a ocorrência de $A1$ após a primeira ocorrência de AB em S_{ge}. Considere então a especificação E_{op} marcada pelo autômato representado na figura 3.9. Verifica-se que E_{op} é controlável e.r.a. $L_{S_{op}}$ e Σ_{nc} (o conjunto de eventos não controláveis para o operador), prefixo-fechada e que $\theta(E_{op}) = E_{ge}$. Dessa forma, E_{ge} é implementado via um supervisor f_{op} para o operador tal que $L_{f_{op}/S_{op}} = E_{op}$. Ou seja, existe um supervisor f_{op} para o operador tal que $\theta(L_{f_{op}/S_{op}}) = E_{ge}$. Assim, conclui-se que não há consistência hierárquica* **forte** *entre S_{op} e S_{ge} na figura 3.9, vide seção 3.1.*

Pelo exemplo 3.6, ilustra-se que as condições para consistência hierárquica para a abordagem de Zhong e Wonham (1990) não garantem consistência hierárquica forte.

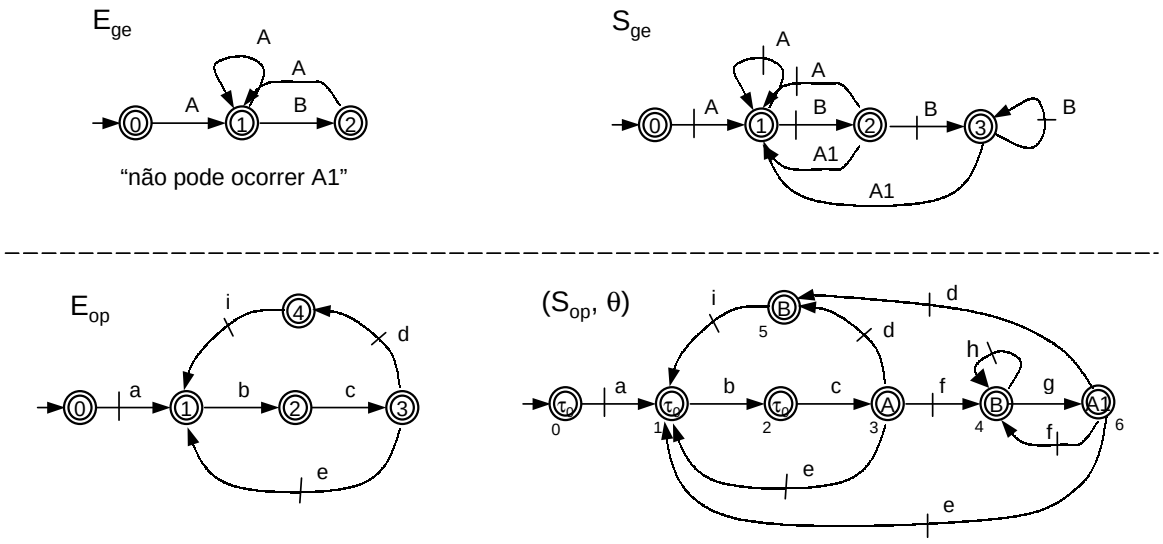


Figura 3.9: Não possui consistência hierárquica forte.

3.3 Abordagem que considera o comportamento marcado dos sistemas

Nesta seção apresentam-se os resultados simplificados da abordagem iniciada por Wong e Wonham (1996a) para o controle hierárquico de SEDs considerando-se o seu comportamento marcado.

A abordagem de Wong e Wonham (1996a) baseia-se em estruturas algébricas sofisticadas cuja apresentação e explanação demandam a inclusão de muito formalismo algébrico neste trabalho. Assim, optou-se por apresentar uma abordagem simplificada, originalmente presente em (Wong e Wonham 1998), correspondente à instanciação dos resultados principais de Wong e Wonham (1996a) para sistemas modelados segundo a abordagem RW. Uma outra razão desta escolha é que Wong e Wonham (1996a) não propõem os resultados na forma de estruturas computáveis, permanecendo os mesmos no plano conceitual e abstrato.

Para o esquema de supervisão hierárquica na figura 3.1, seja o SED para o operador S_{op} sobre o alfabeto Σ representado pela tripla $(L_{S_{op}}, L_{S_{op},m}, \Gamma_{S_{op}})$, onde $L_{S_{op}}$ é

a linguagem gerada, a representar todas as seqüências de eventos em Σ que S_{op} pode gerar, $L_{S_{op},m}$ é a linguagem marcada, a representar todas as seqüências de eventos em $L_{S_{op}}$ que definem tarefas completas de S_{op}, e $\Gamma_{S_{op}}$ é a estrutura de controle de S_{op}, definida pela partição do alfabeto Σ em um subconjunto controlável (Σ_c) e um subconjunto não controlável (Σ_{nc}), vide subseção 2.5.2. Sejam T o alfabeto para o gerente e o mapa repórter $\theta : L_{S_{op}} \rightarrow T^*$ a representar o canal de informação, conforme a definição 3.1. Seja o SED para o gerente S_{ge} representado pela tripla $(L_{S_{ge}}, L_{S_{ge},m}, \Gamma_{S_{ge}})$, onde a linguagem gerada é definida por $L_{S_{ge}} = \theta(L_{S_{op}})$, a linguagem marcada é definida por $L_{S_{ge},m} = \theta(L_{S_{op},m})$, e a estrutura de controle $\Gamma_{S_{ge}}$ é definida pela partição de T em subconjuntos controlável (T_c) e não controlável (T_{nc}) em função de consistência de controle estrita para S_{op} e θ.

Neste esquema de supervisão hierárquica surge o problema de bloqueio para o supervisor do nível do gerente, descrito a seguir. Considera-se nesta abordagem o controle supervisório de SEDs segundo a abordagem de RW original, sem o caso do supervisor poder desmarcar as tarefas do sistema em malha fechada, vide subseção 2.5.3. Seja $E_{ge} \subseteq L_{S_{ge},m}$ uma linguagem que seja realizável por um supervisor não bloqueante f_{ge} no nível gerencial. Na abordagem RW, a linguagem E_{ge} é $L_{S_{ge},m}$-fechada e controlável e.r.a. $L_{S_{ge}}$ e T_{nc} (Ramadge e Wonham 1989). O supervisor f_{ge} é definido por $f_{ge}(t) = \Sigma_{E_{ge}}(t)$, para $t \in L_{S_{ge}}$. Então, seguindo o esquema de supervisão hierárquica da figura 3.1, define-se o supervisor f_{op} para o operador conforme a equação (3.8). Pelas proposições 3.1 e 3.2, o supervisor f_{op} é tal que $L_{f_{op}/S_{op}} = \sup C_{(L_{S_{op}},\Sigma_{nc})}(\theta^{-1}(\overline{E_{ge}}))$ e $\theta(L_{f_{op}/S_{op}}) = \overline{E_{ge}}$. Por outro lado, a linguagem marcada em malha fechada para o supervisor do operador é definida por $L_{f_{op}/S_{op},m} = L_{f_{op}/S_{op}} \cap L_{S_{op},m}$, ou seja, $L_{f_{op}/S_{op},m} = \sup C_{(L_{S_{op}},\Sigma_{nc})}(\theta^{-1}(\overline{E_{ge}})) \cap L_{S_{op},m}$. É possível provar que $L_{f_{op}/S_{op}} \supseteq \overline{L_{f_{op}/S_{op},m}}$, isto é, o supervisor do operador pode ser bloqueante (Wong e Wonham 1996a).

Exemplo 3.7 (Bloqueio do supervisor para o operador – caso 1) *Considere o esquema de supervisão hierárquica representado na figura 3.10, onde S_{op} é o SED do operador e S_{ge} é o SED do gerente. Verifica-se que S_{op} e o mapa repórter associado*

θ possuem consistência de controle estrita. Entretanto, considere a especificação E_{ge} marcada pelo autômato também representado na figura 3.10. Verifica-se que E_{ge} é contida em $L_{S_{ge},m}$, $L_{S_{ge},m}$-fechada e controlável e.r.a. $L_{S_{ge}}$ e T_{nc} (o alfabeto não controlável do gerente). Assim, existe um supervisor não bloqueante f_{ge} para S_{ge} tal que $L_{f_{ge}/S_{ge},m} = E_{ge}$. Entretanto, considere o autômato H_{op} na figura 3.10. O autômato H_{op} representa os comportamentos em malha fechada para o supervisor do operador, definido conforme a equação (3.8), isto é, $L_{H_{op}} = L_{f_{op}/S_{op}}$, a linguagem gerada por H_{op}, e $L_{H_{op},m} = L_{f_{op}/S_{op},m}$, a linguagem marcada por H_{op}. Verifica-se então que f_{op} é bloqueante, isto é, $L_{f_{op}/S_{op}} \supset \overline{L_{f_{op}/S_{op},m}}$, pois a palavra b está em $L_{f_{op}/S_{op}}$, mas não está no prefixo de $L_{f_{op}/S_{op},m}$.

Em (Wong e Wonham 1996a) identificou-se uma propriedade estrutural do mapa repórter cuja ausência pode causar o bloqueio do supervisor para o operador.

Definição 3.5 (Mapa repórter observador (Wong e Wonham 1996a)) *O mapa repórter θ é um observador[7] se e somente se, para toda palavra s em $L_{S_{op}}$ e todo evento gerencial τ em T, se $\theta(s)\tau$ pertence a $\theta(L_{S_{op}})$, então existe uma palavra não vazia u em Σ^+ tal que su está em $L_{S_{op}}$ e $\theta(su)$ é igual a $\theta(s)\tau$.*

Interpreta-se a definição de observador como a seguir: toda palavra do operador cuja imagem correspondente no gerente possui um dado conjunto ativo de eventos, deve ser possível de ser estendida, por intermédio de trechos silenciosos, para palavras que vocalizem aquele exato conjunto de eventos. Na figura 3.10, o mapa repórter associado θ não é um observador, pois para a palavra do operador $s = b$ e o evento do gerente $\tau = B$, tem-se $\theta(s)\tau = AB \in L_{S_{ge}}$. Entretanto, para toda palavra u em Σ^+ tal que $su \in L_{S_{op}}$, tem-se $\theta(su) = AC \neq AB$.

Observa-se, na figura 3.10, se o gerente pudesse diferenciar se o estado corrente em S_{op} é 1 ou 2 quando da ocorrência do evento A, o problema de bloqueio não ocorreria, pois saber-se-ia que após 1 só pode ocorrer B e após 2 só pode ocorrer C.

[7]No original em inglês, *observer reporter map.*

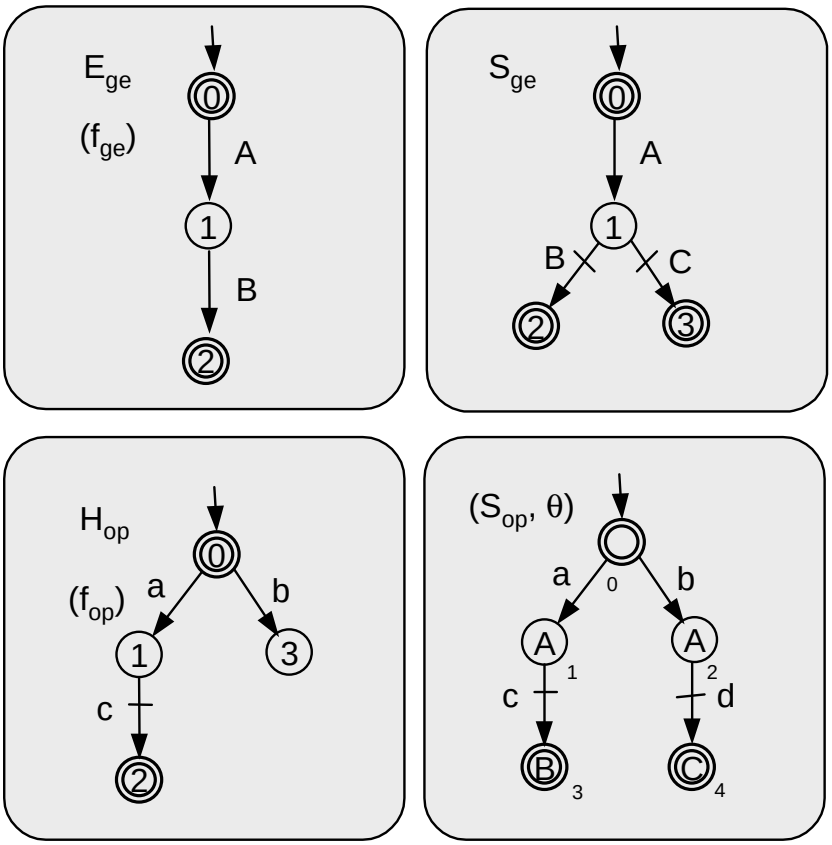


Figura 3.10: Bloqueio do supervisor para o operador – caso 1.

Dado um mapa repórter que não é um observador, existem algumas abordagens da literatura que propõem procedimentos para torná-lo um observador (Wong 1994, Guan 1997, Pu 2000). Em (Guan 1997) explora-se a seguinte propriedade para construção de um mapa repórter observador. Dada uma palavra $s \in L_{S_{op}}$ seja o *conjunto ativo de eventos gerenciais em* S_{op} *após* s definido por

$$\begin{aligned}\Sigma_{S_{op},voc}(s) \;=\; \{\tau \in T : (\exists u \in \Sigma^{+}) \text{ tal que } \quad & ((su \in L_{S_{op}}) \text{ e } (w(su) = \tau) \text{ e} \\ & (((\forall u' \in \Sigma^{+}) \text{ tal que } u' < u)\ w(su') = \tau_0))\}\end{aligned} \tag{3.10}$$

O conjunto $\Sigma_{S_{op},voc}(s)$ contém todos os eventos gerenciais τ que podem ser vocalizados após s em S_{op}, sendo que, desde a ocorrência de s até a ocorrência da palavra que vocaliza τ nenhum outro evento é vocalizado. Por exemplo, na figura 3.10, $\Sigma_{S_{op},voc}(\epsilon) = \{A\}$, $\Sigma_{S_{op},voc}(a) = \{B\}$ e $\Sigma_{S_{op},voc}(b) = \{C\}$.

Proposição 3.3 (Mapa repórter observador (Wong e Wonham 1996a)) *O mapa repórter* θ *é um observador se e somente se, para todo par de palavras* s_1 *e* s_2 *em* $L_{S_{op}}$*, se* $\theta(s_1) = \theta(s_2)$ *então* $\Sigma_{S_{op},voc}(s_1) = \Sigma_{S_{op},voc}(s_2)$.

A proposição 3.3 afirma que todas as palavras do operador que correspondem a uma mesma palavra no gerente devem possuir o mesmo conjunto ativo de eventos gerenciais. O procedimento de construção do observador de Guan (1997) cria novas etiquetas de eventos do gerente e renomeia algumas das etiquetas existentes, para assegurar a condição da proposição 3.3. Por exemplo, ao se aplicar o procedimento ao sistema da figura 3.10, uma nova instância do evento A, diga-se $A1$, deveria ser criada seja para o estado 1 ou para o estado 2, a fim de diferenciarem-se as palavras que correspondem à mesma palavra no nível gerencial e que possuem conjuntos ativos de eventos gerenciais distintos. Para ilustrar a complexidade da construção de um mapa repórter observador, no procedimento proposto por Guan (1997), considerando G_{op} com m transições, a complexidade é $\mathcal{O}(m^6)$. Guan (1997) não fornece uma estimativa do número de estados do autômato resultante do processo.

Um problema comum a todas as abordagens para construção de um mapa repórter observador resume-se à não disponibilidade de uma implementação computacional dos algoritmos, o que reduz a aplicação dos procedimentos puramente conceituais a sistemas de pequeno porte (Wong 1994, Guan 1997, Pu 2000).

A conjunção das condições de consistência de controle estrita e mapa repórter observador ainda não garantem consistência hierárquica com supervisor não bloqueante para o operador, como ilustrado pelo próximo exemplo.

Exemplo 3.8 (Bloqueio de supervisor para o gerente – caso 2) *Na figura 3.11 considere o esquema de supervisão hierárquica com os sistemas S_{op} (operador) e S_{ge} (gerente). Verifica-se que o par (S_{op}, θ), onde θ é o mapa repórter associado, possui consistência de controle estrita e θ é um observador. Considere a especificação $E_{ge} = \epsilon$, marcada pelo autômato na figura 3.11. Por um lado E_{ge} está contida na linguagem marcada de S_{ge}, é $L_{S_{ge},m}$-fechada e é controlável e.r.a. $L_{S_{ge}}$ e T_{nc} (os eventos não controláveis para o gerente). Assim, existe um supervisor não bloqueante para o gerente, diga-se f_{ge}, tal que $L_{f_{ge}/S_{ge},m} = E_{ge}$. Entretanto, dentro do esquema de supervisão hierárquica, considere o supervisor f_{op} correspondente a f_{ge}, calculado segundo a equação (3.8). O autômato H_{op}, representado na figura 3.11, gera as linguagens correspondentes a f_{op}, quais sejam, $L_{H_{op}} = L_{f_{op}/S_{op}}$ e $L_{H_{op},m} = L_{f_{op}/S_{op},m}$. Verifica-se que $L_{f_{op}/S_{op}} \supset \overline{L_{f_{op}/S_{op},m}}$, isto é, o supervisor f_{op} é bloqueante.*

Também em (Wong e Wonham 1996a) identifica-se o seguinte problema estrutural relacionado ao mapa repórter que causa bloqueio do supervisor do operador no esquema de supervisão hierárquica.

Definição 3.6 (Consistência de Marcação (Wong e Wonham 1996a)) *Existe consistência de marcação*[8] *entre S_{op} e S_{ge} se e somente se $\theta^{-1}(L_{S_{ge},m}) = L_{S_{op},m}$.*

Como $\theta(L_{S_{op},m}) = L_{S_{ge},m}$, a consistência de marcação expressa que toda palavra do operador cuja imagem no gerente é uma palavra marcada, deve ser marcada também.

[8]No original em inglês, *marking consistency.*

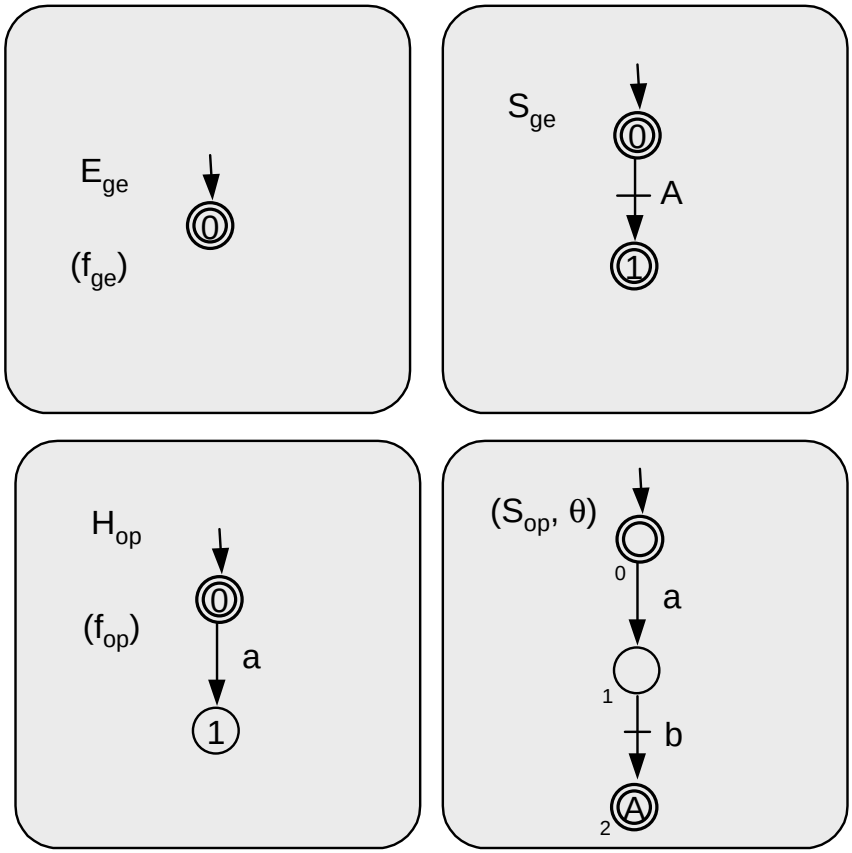


Figura 3.11: Bloqueio do supervisor para o operador – caso 2.

Verifica-se que não existe consistência de marcação entre S_{op} e S_{ge} na figura 3.11, pois $\theta^{-1}(\epsilon) = \{\epsilon, a\}$, e a palavra $s = a$ de S_{op} não é marcada.

Não se encontra na literatura um método para garantir a consistência de marcação. Entretanto, um procedimento geral consiste em modificar o mapa repórter para isolarem-se os estados marcados dos estados não marcados do operador por criação de novas etiquetas de eventos gerenciais.

Finalmente, a proposição 3.4 fornece um conjunto de condições suficientes para se obter consistência hierárquica forte para a formulação do problema de controle hierárquico considerando-se o comportamento marcado dos sistemas (Wong e Wonham 1996a, Wong e Wonham 1998).

Proposição 3.4 (Consistência hierárquica forte (Wong e Wonham 1996a)) *Num esquema de supervisão hierárquica de dois níveis, onde S_{op} é o sistema do operador, S_{ge} é o sistema do gerente e θ é o mapa repórter associado, se (S_{op}, θ) possui consistência de controle estrita, θ for um observador e há consistência de marcação entre S_{op} e S_{ge}, então há consistência hierárquica forte entre S_{op} e S_{ge}.*

Na proposição 3.4, a condição de consistência hierárquica forte entre S_{op} e S_{ge} expressa-se pelo fato de que, para toda linguagem E_{ge} contida em $L_{S_{ge},m}$, existe um supervisor não bloqueante f_{op} para S_{op}, definido conforme a equação (3.8), tal que $\theta(L_{f_{op}/S_{op},m}) = E_{ge}$ se e somente se, E_{ge} for $L_{S_{ge},m}$-fechada e controlável e.r.a. $L_{S_{ge}}$ e T_{nc} (o alfabeto não controlável do gerente). Isso decorra da definição de consistência hierárquica forte, subseção 3.1, e as condições para existência de um supervisor não bloqueante para a abordagem RW na seção 2.5.4.

Exemplo 3.9 (Consistência hierárquica forte) *Considere um esquema de supervisão hierárquica onde o sistema do operador S_{op} e o sistema do gerente S_{ge} são mostrados na figura 3.12. Verifica-se que os sistemas S_{op} e S_{ge} e o mapa repórter associado θ não atendem nenhuma das condições para garantir consistência hierárquica forte, a saber, consistência de marcação, consistência de controle estrita ou mapa repórter*

observador. Por exemplo, não há consistência de marcação entre S_{op} e S_{ge} pois, por exemplo, a palavra $abcf$ é não marcada para o operador, mas a palavra correspondente para o gerente, a saber AB, é marcada. Para assegurar consistência de marcação, vocaliza-se o evento C no estado 1 de S_{op}, a fim de isolar os estados marcados 0 e 5, e muda-se a etiqueta do evento gerencial vocalizado no estado 4 de S_{op} de B para $B1$, a fim de diferenciar-se da vocalização do estado marcado 5 de S_{op}. O sistema resultante destas modificações é $S_{op,ext}$, também representado na figura 3.12, com sistema do gerente resultante $S_{ge,ext}$. É possível verificar que o mapa repórter θ_{ext} é um observador, e o par $(S_{op,ext}, \theta_{ext})$ possui consistência de controle estrita, com eventos controláveis do gerente $T_{ext,c} = \{B, B1, C\}$ e eventos não controláveis do gerente $T_{ext,nc} = \{A\}$. Assim, conclui-se que há consistência hierárquica forte entre $S_{op,ext}$ e $S_{ge,ext}$.

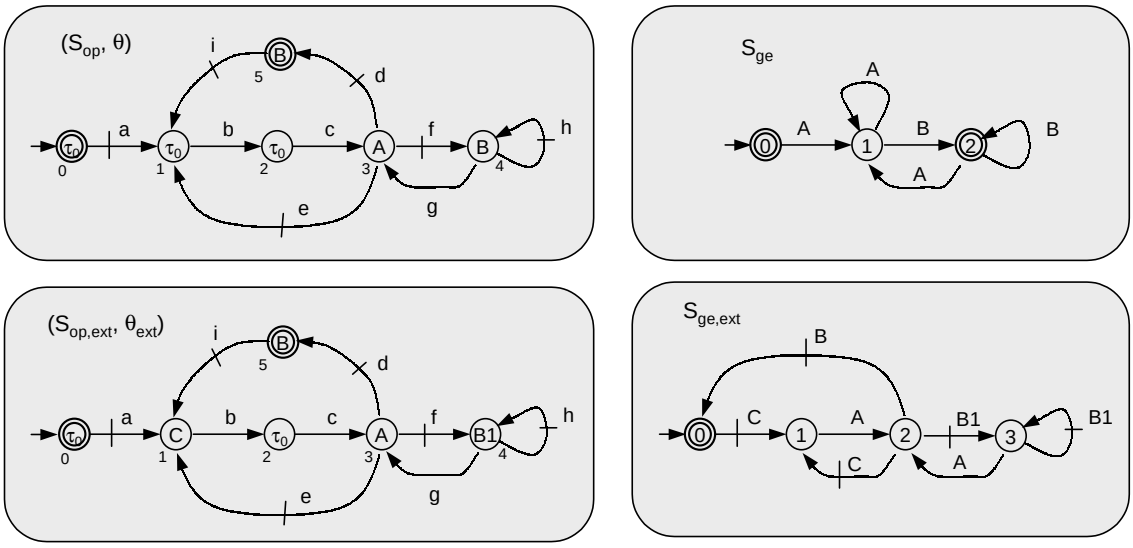


Figura 3.12: Exemplo de aplicação das condições para a consistência hierárquica forte.

Muito embora as condições para consistência hierárquica forte na abordagem de Wong e Wonham (1996a) e Wong e Wonham (1998) tenham sido conseguidas de forma simples e direta no exemplo 3.9, afirma-se que a obtenção de tais condições não é trivial para sistemas de grande porte. Não há nenhuma indicação da integração dos

diversos procedimentos que devem ser aplicados para garantir as três condições simultaneamente (Wong e Wonham 1996a, Wong e Wonham 1998). A ordem de aplicação dos procedimentos influi no resultado, em termos de número de eventos e número de estados do gerente, e a aplicação de um procedimento depois de outro pode invalidar as condições estabelecidas anteriormente. Ilustra-se este fato na figura 3.13, onde o SED $S_{op,ext,1}$ é obtido a partir de S_{op} da figura 3.12 pela aplicação do procedimento de consistência de controle estrita (os eventos $A1$ e B são controláveis e o evento A é não controlável). Em seguida, também representado na figura 3.13, está o SED $S_{op,ext,2}$, obtido de $S_{op,ext,1}$ por aplicação do procedimento de obtenção do mapa repórter observador. Veja que $S_{op,ext,2}$ não possui consistência de controle estrita, invalidando os resultados do procedimento anterior. Enfim, $S_{op,ext,3}$, também representado na figura 3.13 é obtido por modificação de $S_{op,ext,2}$ para obtenção de consistência de marcação. Verifica-se que agora $S_{op,ext,3}$ e o seu mapa repórter associado atendem às três propriedades, sendo os eventos não controláveis A e $A1$ e os eventos controláveis B, $B1$, C e D. Representa-se na figura 3.13 o sistema do gerente correspondente a $S_{op,ext,3}$, diga-se $S_{ge,ext,3}$.

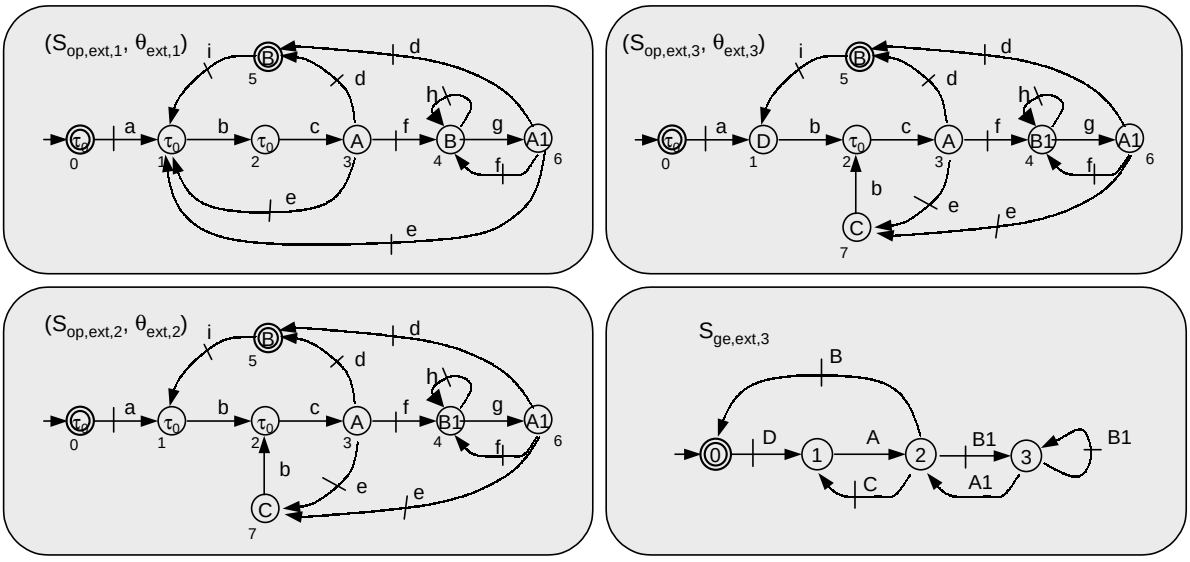


Figura 3.13: Passos alternativos para a consistência hierárquica forte.

Como regra de modelagem, indica-se que o procedimento para garantir a consistência de marcação seja feito antes de aplicar os outros procedimentos, quais sejam, consistência de controle estrita e observador. A aplicação dos outros procedimentos não vai afetar a consistência de marcação, pois os estados marcados estão isolados e diferenciados.

Um outro problema da aplicação conjunta dos procedimentos é a não existência de uma ferramenta computacional que integre os três procedimentos, sendo atualmente os cálculos realizados parte em computador parte à mão. Isso limita o porte dos sistemas que podem ser tratados na abordagem.

3.4 Abordagem baseada em abstrações consistentes

Nesta seção apresenta-se a abordagem de Pu (2000) para o problema de controle hierárquico de SEDs considerando-se o comportamento marcado.

Pu (2000) trabalha com uma generalização de um SED dotado de controle, que é analisada a seguir.

Definição 3.7 (SED controlado de Pu (2000)) *Um SED controlado S sobre o alfabeto Σ é uma tripla $(L_S, L_{S,m}, \mathcal{C}_S)$ onde:*

1. *$L_S \subseteq \Sigma^*$ define a linguagem gerada, uma linguagem prefixo-fechada que contém todas as palavras geradas pelo sistema,*

2. *$L_{S,m} \subseteq L_S$ define a linguagem marcada, que contém as palavras que representam as tarefas que o sistema completa, e* [9]

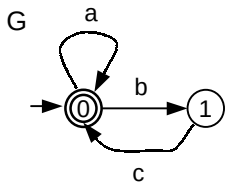
3. *$\mathcal{C}_S$ é uma função, denominada função de controle, que, para $s \in L_S$, associa um conjunto $\mathcal{C}_S(s) \subseteq 2^{\Sigma}$. Cada elemento $\gamma \in \mathcal{C}_S(s)$, denominado opção de controle, é um subconjunto do conjunto ativo de eventos em L_S após s considerados habilitados.*

[9] No original apresenta-se uma função de marcação $m_S : L_S \rightarrow \{0, 1\}$, onde para $s \in L_S$, se $m_S(s) = 1$, a palavra s é considerada marcada e, se $m_S(s) = 0$, a palavra s não é considerada marcada. Isso é equivalente a se definir uma linguagem marcada $L_{S,m} = \{s \in L_S : m_S(s) = 1\}$.

Um SED controlado de Pu (2000) S sobre o alfabeto Σ deve atender a dois requisitos, a saber, para toda palavra s em L_S, a função de controle $\mathcal{C}_S(s)$ deve ser *não vazia*, isto é, $\mathcal{C}_S(s) \neq \emptyset$, e *completa*, no sentido de que a união de todas as opções de controle disponíveis em $\mathcal{C}_S(s)$ deve ser igual ao conjunto ativo de eventos em L_S após s, isto é, $\bigcup_{\gamma \in \mathcal{C}_S(s)} \gamma = \Sigma_{L_S}(s)$.

Exemplo 3.10 (SED controlado de Pu (2000)) *Para $\Sigma = \{a, b, c\}$, considere o sistema S definido pelas linguagens $L_S = (a+bc)^*(\epsilon+b)$ e $L_{S,m} = (a+bc)^*$, e função de controle $\mathcal{C}_S(s) = \{\emptyset, \{a, b\}\}$, para $s \in (a+bc)^*$, e $\mathcal{C}_S(s) = \{\emptyset, \{c\}\}$, para $s \in (a+bc)^*b$.*

Pode-se representar um SED controlado de Pu (2000), diga-se S sobre o alfabeto Σ, por um par $(G, \mathcal{C})$, onde G é um autômato tal que $L_G = L_S$ e $L_{G,m} = L_{S,m}$, e $\mathcal{C}$ é uma tabela que associa um estado q de G ao resultado da função de controle $\mathcal{C}_S(s)$, para a palavra s correspondente a q (vide subseção 2.4.2 para definições relacionadas a autômatos e linguagens). Para o caso em que as linguagens L_S e $L_{S,m}$ são regulares e Σ é finito, o autômato G possui um número finito de estados e a tabela $\mathcal{C}$ possui um número finito de entradas. Na figura 3.14 representa-se o SED S do exemplo 3.10 por meio de um autômato G e uma tabela $\mathcal{C}$.



C

estado	opções de controle
0	∅ {a, b}
1	∅ {c}

Figura 3.14: Exemplo do SED controlado definido por Pu (2000).

Apresentam-se alguns resultados relativos à supervisão dos SEDs controlados definidos por Pu (2000). Considere o SED controlado S sobre o alfabeto Σ, definido conforme a abordagem de Pu (2000). Uma linguagem $E \subseteq L_S$ é dita *controlável em relação a* L_S

e $\mathcal{C}_S$ se, para toda palavra s em $\overline{E}$, $\Sigma_E(s) \in \mathcal{C}_S(s)$, isto é, o sistema gera o prefixo de E por meio da escolha adequada de opções de controle. Um supervisor é uma função f que associa a todo $s \in L_S$ uma opção de controle $f(s) \in \mathcal{C}_S(s)$. Representa-se o comportamento em malha fechada sob supervisão pelo par de linguagens $(L_{f/S}, L_{f/S,m})$, definidas da forma usual da abordagem RW (vide subseção 2.5.3). Um supervisor f para S é dito *não bloqueante* quando $\overline{L_{f/S,m}} = L_{f/S}$. Dada uma linguagem $E \subseteq L_{S,m}$ existe um supervisor não bloqueante f para S tal que $L_{f/S,m} = E$ se e somente se E for $L_{S,m}$-fechada e controlável e.r.a. L_S e $\mathcal{C}_S$. Quando, para todo $s \in L_S$, o conjunto $\mathcal{C}_S(s)$ for fechado em relação à união, o conjunto das linguagens $L_{S,m}$-fechadas e controláveis e.r.a L_S e $\mathcal{C}_S$ contidas na linguagem de especificação $E \subseteq L_S$, $C_{(L_S,L_{S,m},\mathcal{C}_S)}(E)$, é não vazio, fechado em relação à união e possui um único elemento supremo, a máxima linguagem $L_{S,m}$-fechada e controlável e.r.a L_S e $\mathcal{C}_S$, denotada $\sup C_{(L_S,L_{S,m},\mathcal{C}_S)}(E)$. Em (Pu 2000) propõe-se um algoritmo para obtenção da máxima linguagem $L_{S,m}$-fechada e controlável e.r.a L_S e $\mathcal{C}_S$ contida numa dada linguagem de especificação.

Considere agora o esquema de supervisão hierárquica da figura 3.1. Em (Pu 2000), define-se o SED do operador por um SED controlado S_{op} sobre o alfabeto Σ, e o canal de informação por um mapa repórter θ como na definição 3.1. De forma equivalente, representa-se o par (S_{op}, θ) por um autômato de Moore mais uma tabela para definir a função de controle dependente do estado, de forma semelhante ao exposto na seção 3.2 (Pu 2000). Introduz-se então o modelo gerencial por meio de uma *abstração consistente*, definida a seguir.

Primeiramente, tornam-se necessárias duas definições: dada uma palavra $t \in L_{S_{ge}}$, define-se por

$$\theta_{voc}^{-1}(t) \quad = \quad \{s \in L_{voc} : \theta(s) = t\}, \tag{3.11}$$

o conjunto de palavras vocais cuja θ-imagem é t, e dada uma palavra qualquer do operador $s \in L_{S_{op}}$, define-se por

$$\begin{aligned} L_{S_{op},voc}(s) \quad = \quad \{u \in \Sigma^+ : \quad & (su \in L_{S_{op}}) \text{ e } (w(su) \neq \tau_0) \text{ e} \\ & (((\forall u' \in \Sigma^+) \text{ tal que } u' < u)\ w(su') = \tau_0)\} \end{aligned} \tag{3.12}$$

o conjunto de palavras não vazias que, quando concatenadas com s, formam palavras vocais de S_{op}, e cujos prefixos estritos, quando concatenados com s, formam palavras silenciosas de S_{op}. Observe que $s \cdot L_{S,voc}(s)$, o conjunto de todas as palavras formadas pela concatenação de s com palavras em $L_{S,voc}(s)$, representam as próximas palavras vocais em S_{op} que ocorrem após s.

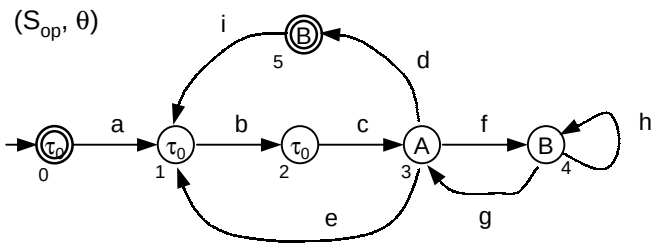
Definição 3.8 (Abstração Consistente (Pu 2000)) *Dados o sistema do operador S_{op} sobre o alfabeto Σ, o alfabeto do gerente T e o mapa repórter θ, uma abstração consistente de S_{op} em relação a θ é um SED controlado S_{ge} sobre T onde:*

1. $L_{S_{ge}} = \theta(L_{S_{op}})$,
2. *Para todo t em $L_{S_{ge}}$, se t pertence a $L_{S_{ge},m}$ então, para todo s em $\theta^{-1}_{voc}(t)$, tem-se s pertencente a $L_{S_{op},m}$, e*
3. *Para todo t em $L_{S_{ge}}$, se γ pertence a $\mathcal{C}_{S_{ge}}(t)$ então existe um supervisor não bloqueante f para S tal que, para todo s em $\theta^{-1}_{voc}(t)$, tem-se:*
   (a) $s \in L_{f/S_{op}}$, e
   (b) $w((s \cdot L_{S_{op},voc}(s)) \cap L_{f/S_{op}}) = \gamma$, *onde w denota o mapa vocal, equação (3.5).*

Interpreta-se a definição 3.8 como a seguir. O ítem 1 expressa que a linguagem gerada do gerente é a θ-imagem da linguagem gerada do operador. O ítem 2 expressa que a linguagem marcada do gerente é definida por palavras do gerente para as quais todas as palavras vocais correspondentes são marcadas. O ítem 3 define a função de controle para o gerente. Uma opção de controle válida em $t \in L_{S_{ge}}$ é um elemento do seu conjunto ativo de eventos que pode ser habilitado por controle. Tal controle corresponde à existência de um supervisor não bloqueante no operador, que, em malha fechada, admite todas as palavras vocais correspondentes à palavra do gerente, $s \in \theta^{-1}_{voc}(t)$, mais todas as próximas palavras vocais para estas palavras, $s' \in s \cdot L_{S,voc}(s)$, que vocalizam os eventos da opção de controle.

Dados o sistema do operador S_{op} sobre o alfabeto Σ, o alfabeto gerencial T e o mapa repórter θ, é possível que não exista uma abstração consistente de S_{op} e.r.a. θ (Pu 2000), como ilustrado pelo exemplo a seguir.

Exemplo 3.11 (Não existência de abstrações consistentes) *Considere o sistema do operador S_{op} com mapa repórter associado θ, representados na figura 3.15. Deseja-se determinar se existe uma abstração consistente de S_{op} e.r.a. θ. Seja a palavra $t = AB \in \theta(L_{S_{op}})$. Verifica-se que as palavras vocais do operador cuja θ-imagem é AB são $\theta^{-1}_{voc}(AB) = \{abcd, abcf\}$. O conjunto ativo de eventos gerencias após AB é $\Sigma_{\theta(L_{S_{op}})}(AB) = \{A, B\}$. Considere a seguinte análise das opções de controle possíveis para a palavra AB segundo a definição 3.8. As opções de controle $\emptyset$ e $\{B\}$ não são possíveis para AB pois não se pode proibir a ocorrência do evento g após o estado 4 de S_{op}, não sendo possível, por conseguinte, impedir-se a ocorrência de A após AB. A opção de controle $\{A, B\}$ também não é possível após AB, pois após a palavra $abcd$ não ocorre o evento B, violando-se assim a condição do item 3 da definição 3.8. Assim, como o conjunto de controles para $t = AB$ não pode ser completo, não existe uma abstração consistente de S_{op} e.r.a. θ.*



q	C(q)
0	∅, {a}
1	{b}
2	{c}
3	∅, {d}, {e}, {f}, {d,e}, {d,f}, {e,f}, {d,e,f}
4	{g}, {g, h}
5	∅ {i}

Figura 3.15: Não existe uma abstração consistente por Pu (2000).

Em (Pu 2000) identifica-se uma condição de existência para uma abstração consistente, analisada a seguir.

Definição 3.9 (Mapa repórter observador fraco (Pu 2000)) *O mapa repórter θ é dito um observador fraco se e somente se, para toda palavra vocal s em $L_{S_{op},voc}$ e para todo evento do gerente τ em T, se $\theta(s)\tau$ está em $\theta(L_{S_{op}})$, então existe uma palavra u em Σ^* tal que su está em $L_{S_{op}}$ e $\theta(su)$ é igual a $\theta(s)\tau$.*

O mapa repórter observador fraco difere do observador propriamente dito por exigir que possuam o mesmo conjunto ativo de eventos gerenciais apenas as palavras vocais com mesma θ-imagem, no lugar de todas as palavras com a mesma θ-imagem, para o caso do observador. Por exemplo, para o sistema da figura 3.15, o mapa repórter associado não é um observador fraco pois, por exemplo, as palavras vocais $abcd$ e $abcf$, que possuem a mesma θ-imagem AB, possuem distintos conjuntos ativos de eventos gerenciais, a saber $\{A\}$ e $\{A, B\}$, respectivamente.

Exemplo 3.12 (Diferença entre observador e o observador fraco) *Considere os pares (L, θ_i), para $i \in \{1, 2, 3\}$, representados na figura 3.16, onde L é uma linguagem prefixo-fechada e θ_i é um mapa repórter definido sobre esta linguagem. Verifica-se que o mapa θ_1 é observador e observador fraco. Já o mapa θ_2 é um observador fraco, mas não é um observador, pois as palavras ϵ, a e b com mesma θ-imagem ϵ possuem conjuntos ativos de eventos gerenciais $\{B, C\}$, $\{B\}$ e $\{C\}$, respectivamente. E o mapa θ_3 não é um observador fraco nem um observador, pois as palavras vocais a e b com mesma θ-imagem A possuem distintos conjuntos ativos de eventos gerenciais $\{B\}$ e $\{C\}$, respectivamente.*

Dado um projeto inicial de canal de informação, com mapa repórter não observador fraco, é necessário determinar se este mapa pode ser refinado até que se torne um observador fraco. Em (Pu 2000), não há indicação de construção de um observador fraco, exceto pela indicação de que se θ é um observador, então também é um observador fraco. Entretanto, ao se examinar a condição de observador fraco na definição 3.9, verifica-se que para um observador fraco todas as palavras vocais devem "ver"os mesmos conjuntos ativos de eventos gerenciais. Assim, um procedimento que renomeie

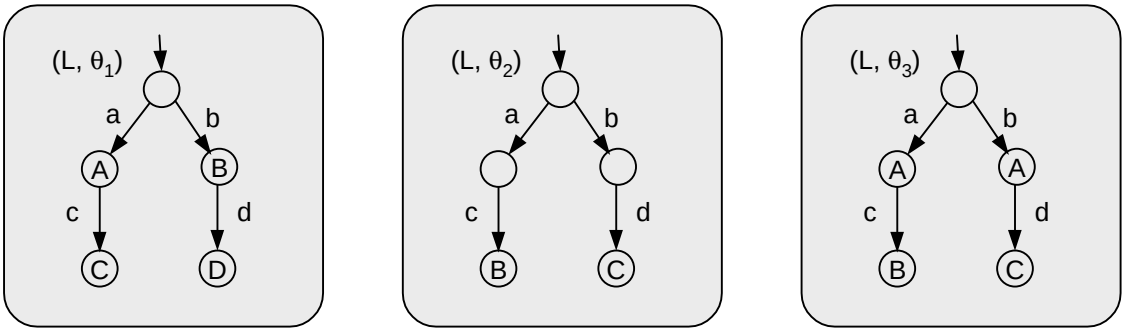


Figura 3.16: Diferença entre um observador e um observador fraco.

as etiquetas dos eventos gerenciais até que a condição de observador fraco seja atendida é uma possível solução. Falta, entretanto uma prova formal de convergência para um observador fraco para tal procedimento.

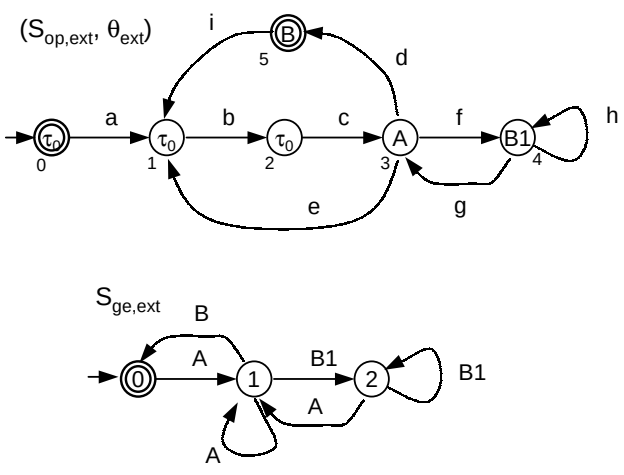
Proposição 3.5 (Existência de abstrações consistentes (Pu 2000)) *Existe uma abstração consistente de S_{op} e.r.a. θ se e somente se o mapa repórter θ for um observador fraco.*

A proposição 3.5 fornece uma condição necessária e suficiente para a existência de uma abstração consistente (Pu 2000).

Em (Pu 2000) apresentam-se condições de construção de uma abstração consistente, dado um sistema do operador e um mapa repórter observador fraco. Optou-se por não apresentar esses detalhes de construção neste trabalho.

Exemplo 3.13 (Abstração consistente de Pu (2000)) *Modifica-se o mapa repórter para o sistema da figura 3.15 de sorte que este seja um observador fraco, o resultado é o par $(S_{op,ext}, \theta_{ext})$ representado na figura 3.17. Em particular, basta modificar uma das etiquetas de evento B de um dos estados 4 ou 5 de S_{op}. O modelo do gerente $S_{ge,ext}$, também é representado na figura 3.17, construído como uma abstração consistente de $S_{op,ext}$ e.r.a. θ_{ext}, definição 3.8.*

A seguinte proposição situa as abstrações consistentes de Pu (2000) dentro do problema de controle hierárquico.



q	C(q)
0	∅, {a}
1	{b}
2	{c}
3	∅, {d}, {e}, {f}, {d,e}, {d,f}, {e,f}, {d,e,f}
4	{g}, {g, h}
5	∅ {i}

q`	C_{ge}(q`)
0	∅ {A}
1	{A} {B} {B1} {A, B} {A, B1} {B,B1} {A, B, B1}
2	{A} {A, B1}

Figura 3.17: Abstração consistente por Pu (2000).

Proposição 3.6 (Consistência hierárquica (Pu 2000)) *S_{ge} é uma abstração consistente de S_{op} e.r.a. θ se e somente se existe consistência hierárquica entre S_{op} e S_{ge}.*

O resultado da proposição 3.6 não é apresentado de forma explícita em (Pu 2000). Decorre da definição de consistência hierárquica na seção 3.1 da proposta de supervisor f_{op} para o operador correspondente a um supervisor f_{ge} para o gerente, dada a linguagem E_{ge}, a representar um comportamento realizável por supervisor não bloqueante para o gerente. Os detalhes do desenvolvimento são semelhantes ao que foi apresentado para a abordagem de Zhong e Wonham (1990) na seção 3.2, e não são apresentados.

Exemplo 3.14 (Não garante consistência hierárquica forte) *Considere os modelos do operador e do canal de informação (S_{op}, θ) e o modelo do gerente S_{ge}, representados na figura 3.18. Prova-se que o mapa repórter θ é um observador fraco e que S_{ge} é uma abstração consistente de S_{op} e.r.a. θ. Assim, pela proposição 3.6, há consistência hierárquica entre S_{op} e S_{ge}. Considere a especificação gerencial E_{ge}, marcada pelo autômato na figura 3.18. A especificação E_{ge} não é controlável e.r.a. $L_{S_{ge}}$ e $\mathcal{C}_{S_{ge}}$, pois a opção de controle $\emptyset$ não está disponível no estado 2 de S_{ge}. Entretanto, considere a especificação para o operador E_{op}, marcada pelo autômato representado na*

figura 3.18. Verifica-se que E_{op} é $L_{S_{op},m}$-fechada e controlável e.r.a. $L_{S_{op}}$ e $\mathcal{C}_{S_{op}}$. Assim, existe um supervisor não bloqueante f_{op} para o operador tal que $L_{f_{op}/S_{op},m} = E_{op}$. Por outro lado, verifica-se que $\theta(E_{op}) = E_{ge}$. Assim, a θ-imagem de E_{op} não é realizável por supervisor não bloqueante para o gerente. Com isso, mostra-se que não há consistência hierárquica forte entre S_{op} e S_{ge}.

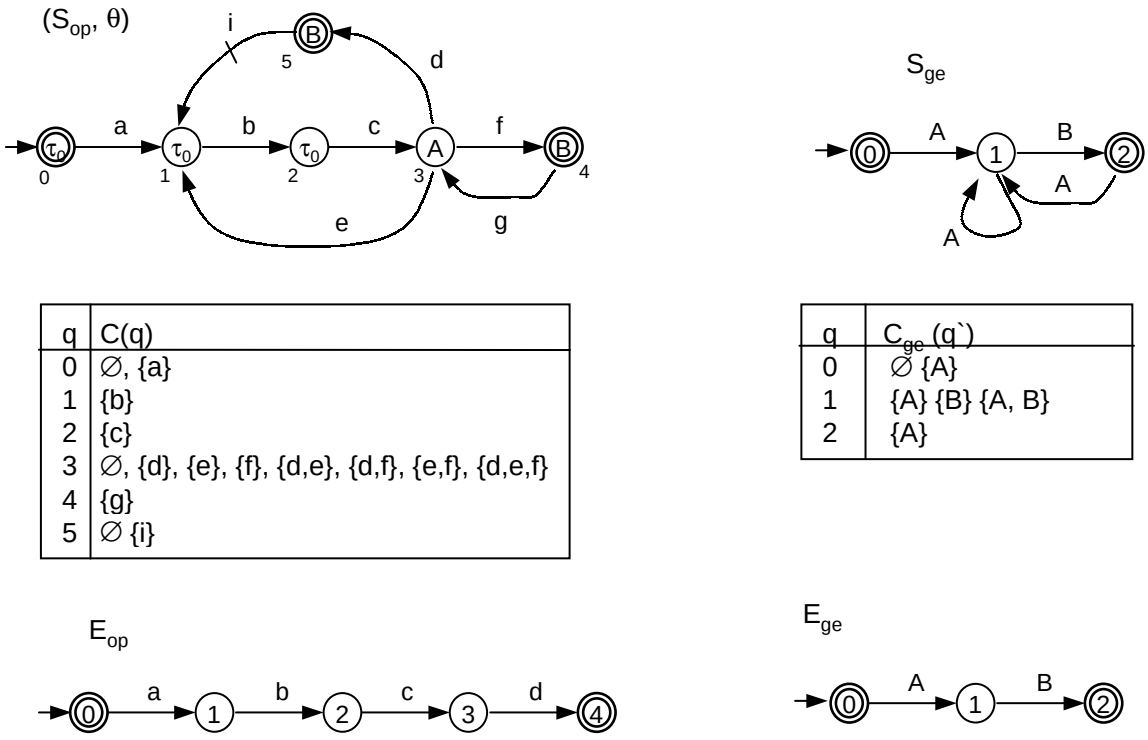


Figura 3.18: O modelo de Pu (2000) garante consistência hierárquica forte.

3.5 Discussão

Neste capítulo foram apresentadas as principais abordagens para o controle hierárquico de SEDs, ressaltando-se as principais vantagens e desvantagens de cada abordagem.

Na seção 3.1 enuncia-se o problema de controle hierárquico para um esquema de supervisão hierárquica de dois níveis que faz analogia a uma fábrica, onde associa-se o

nível inferior a um operador, e o nível superior a um gerente. O problema de controle hierárquico consiste em, dado um SED para o operador, calcular o canal de informação, que é o dispositivo pelo qual o SED do gerente é atualizado pelo SED do operador, de tal forma que haja consistência hierárquica entre os SEDs do operador e do gerente. Também na seção 3.1 são diferenciadas duas gradações para a consistência hierárquica, a saber, consistência hierárquica propriamente dita e consistência hierárquica forte, que são usadas como forma de comparação das abordagens. Na seção 3.2 estuda-se o problema de controle hierárquico com formulação para linguagens prefixo-fechadas segundo a abordagem de Zhong e Wonham (1990). Foram indicadas condições para obtenção de consistência hierárquica para o esquema de supervisão hierárquica proposto. Na seção 3.3 apresenta-se a formulação do problema de controle hierárquico para linguagens marcadas segundo a abordagem de Wong e Wonham (1996a), complementada por Wong e Wonham (1998). A solução computacional oferecida em (Wong e Wonham 1998) para sistemas modelados segundo a abordagem RW, permite a obtenção de um sistema do gerente com consistência hierárquica forte. As condições para construção de tal sistema são bastante conservadoras e, em geral, complexidade é introduzida no sistema original para garantia da consistência hierárquica forte, principalmente pela criação de eventos sem semântica para o gerente. Por fim, na seção 3.4 apresenta-se a abordagem por abstrações consistentes de Pu (2000), que lança mão de um modelo generalizado para o controle supervisório de SEDs. Como visto, o esquema de supervisão hierárquica de Pu (2000) garante apenas consistência hierárquica.

Um ponto comum das abordagens tratadas é que sempre forçam um refinamento do canal de informação, dada uma configuração inicial do mesmo. Alguns dos procedimentos para refinamento do canal de informação correspondem a condições conservadoras, como no caso da construção do mapa repórter observador (Wong 1994, Guan 1997, Wong e Wonham 2000) e da construção para garantir a consistência de marcação (Wong e Wonham 1996a). Adicionalmente, alguns dos procedimentos, além de modificarem as etiquetas originais do gerente, inserem novas etiquetas, a princípio sem semântica no *mundo do gerente*, como é o caso dos procedimentos de

consistência hierárquica forte (Zhong e Wonham 1990), construção de mapa repórter observador (Wong 1994, Guan 1997, Wong e Wonham 2000), e de consistência de marcação (Wong e Wonham 1996a). Tais procedimentos inserem complexidade na semântica de eventos para o sistema do gerente. Ao mesmo tempo, não há uma indicação para integração dos métodos computacionais que devem ser aplicados em conjunto, como a integração do cálculo de consistência de controle estrita, mapa repórter observador e consistência de marcação, que são necessários para garantir consistência hierárquica forte na abordagem de Wong e Wonham (1998). Por outro lado, algumas das abordagens, como em (Pu 2000), oferecem esquemas de supervisão hierárquica apenas com garantia de consistência hierárquica, não havendo garantia de se construir um modelo gerencial com consistência hierárquica forte. Este problema de não controle da gradação da consistência hierárquica pode gerar sistemas para gerente com muito pouca informação para síntese hierárquica de especificações.

No próximo capítulo trata-se de um SED dotado de controle denominado *SED com marcação flexível*, onde além da flexibilidade do controle de eventos, como no modelo proposto por Pu (2000), há flexibilidade de marcação de uma palavra conforme o conjunto de eventos habilitados escolhido. Como será visto no capítulo 4, o SED com marcação flexível permite a construção de uma hierarquia de SEDs com consistência hierárquica (nas suas duas formas) sem lançar mão de nenhuma das condições das abordagens anteriores. Com isto, será feita uma proposta de um método construtivo para o controle hierárquico de SEDs.

Capítulo 4

Controle supervisório de SEDs com marcação flexível

Neste capítulo descrevem-se os SEDs com marcação flexível e suas principais propriedades.[1]

Descreve-se a seguir a organização do capítulo. Na seção 4.1 introduz-se o SED com marcação flexível e são discutidas questões de modelagem. Na seção 4.2 apresenta-se um esquema de controle supervisório para o SED com marcação flexível e enunciam-se os principais resultados referentes à existência de supervisores. Na seção 4.3 propõe-se um algoritmo para síntese de supervisores para tais sistemas. Na seção 4.4 ilustra-se o método de síntese por intermédio de um exemplo. Por fim, a seção 4.5 contém alguns comentários finais.

4.1 SED com marcação flexível

Esta seção apresenta o SED com marcação flexível e discute algumas particularidades de modelagem.

Definição 4.1 (SED com marcação flexível) *Um SED com marcação flexível S*

[1]O material deste capítulo foi apresentado pela primeira vez em (Cury et al. 2001), tendo sido submetida uma versão completa para revista em (Cury et al. 2002).

sobre o alfabeto Σ *é um par* (L_S, Γ_S)*, onde* $L_S \subseteq \Sigma^*$ *é uma linguagem prefixo fechada e* Γ_S *é uma função que associa cada palavra* $s \in L_S$ *a um conjunto de controles* $\Gamma_S(s) \subseteq 2^\Sigma \times \{M, N\}$.

Para o SED com marcação flexível S sobre o alfabeto Σ, a linguagem L_S representa o conjunto de todas as palavras em Σ que S pode gerar, e, para a palavra s em L_S, $\Gamma_S(s)$ é um conjunto de controles $(\gamma, \#)$ em $2^\Sigma \times \{M, N\}$, onde γ é um possível conjunto de eventos habilitados após s e $\#$ é um indicador tal que, se $\#$ é igual a M, a palavra s é considerada *marcada*, a significar uma tarefa completa do sistema, e, se $\#$ é igual a N, a palavra s é considerada *não marcada*.

Exemplo 4.1 (SED com marcação flexível) *Como uma ilustração de um SED com marcação flexível, considere o alfabeto* $\Sigma = \{\alpha, \beta, \mu, \lambda\}$ *e o sistema* S *definido pela linguagem* $L_S = ((\alpha + \mu)\beta^*\lambda)^*(\epsilon + (\alpha + \mu)\beta^*)$*, onde se usa uma expressão regular para representar* L_S*, vide seção 2.4.4, e a estrutura de controle* Γ_S *é definida por:*

- $\Gamma_S(\epsilon) = \{(\emptyset, M),\ (\{\alpha\}, M)\}$,
- $\Gamma_S(s) = \{(\{\lambda, \beta\}, N)\}$, *para* $s \in (\alpha + \mu)\beta^*(\lambda(\alpha + \mu)\beta^*)^*$, *e*
- $\Gamma_S(s') = \{(\emptyset, N), (\{\alpha\}, N), (\{\alpha, \mu\}, M)\}$ *para* $s' \in ((\alpha + \mu)\beta^*\lambda)^+$.

Observe-se que o terceiro conjunto de controles acima caracteriza palavras que podem ser consideradas marcadas ou não marcadas, dependendo do controle selecionado. Isso é diferente do modelo usado na abordagem RW. Também difere do controle supervisório com marcação porque a presença de um controle (γ, M) *num conjunto de controles não implica a presença do controle* (γ, N) *com o mesmo conjunto de eventos habilitados* γ*. Por exemplo, no terceiro conjunto de controles acima, a ausência do controle* $(\{\alpha, \mu\}, N)$ *impede que se considere a palavra como não marcada quando o conjunto desejado de eventos habilitados for* $\{\alpha, \mu\}$.

Dado o SED com marcação flexível S sobre o alfabeto Σ, representa-se L_S por um autômato $G = (\Sigma, Q, \delta, q_0, Q_m)$, onde Q é o conjunto de estados, $\delta : Q \times \Sigma \to Q$ é a

função de transição (parcial e determinista), q_0 é o estado inicial, e Q_m é o conjunto de estados marcados. Faz-se G tal que $L_G = L_S$ e $L_{G,m} = \emptyset$, vide capítulo 2. Note que, para isso ocorrer tem-se que primeiro, para todo s em L_S, deve existir q em Q tal que $\hat{\delta}(q_0, s) = q$ (onde $\hat{\delta}$ é a extensão da função δ para palavras em Σ^*), e segundo, $Q_m = \emptyset$. Representa-se a estrutura de controle Γ por uma tabela T cujas entradas relacionam as palavras de S aos seus respectivos conjuntos de controle. Assim o par (G, Γ) é uma representação em estados do SED S. Para o caso em que L_S é uma linguagem regular e Σ possuir número finito de eventos tem-se G com número finito de estados (Hopcroft e Ullmann 1979), e com Σ finito, tem-se Γ com número finito de entradas por grupo de palavras. A figura 4.1(a) representa o autômato G e a tabela Γ para o SED S do exemplo 4.1.

Alternativamente representa-se o SED com marcação flexível S sobre o alfabeto Σ pelo par (G', Γ'), onde $G' = (\Sigma, Q', \delta', q'_0, Q'_m)$ é novamente um autômato e Γ' é uma função que associa cada estado q de G a um conjunto de controles $\Gamma'(q)$. Novamente pode-se escrever Γ' na forma de uma tabela. O par (G', Γ') deve ser tal que $L_{G'} = L_S$, $L_{G',m} = \emptyset$ e, para toda palavra s de S e estado q de G tais que $\hat{\delta}'(q_0, s) = q$, deve-se ter $\Gamma'(q) = \Gamma_S(s)$. Diz-se, ao se associar S ao par (G', Γ'), que S é representado por um autômato e uma estrutura de controle dependente do estado. Para o caso em que L_S é regular e Σ com número finito de eventos, G' possui número finito de estados e Γ' possui número finito de entradas por estado de G' (Hopcroft e Ullmann 1979). A figura 4.1(b) mostra a representação por autômato e estrutura de controle dependente do estado para o SED S do exemplo 4.1.

Definem-se, a seguir, algumas operações para os conjuntos de controle. Primeiro, para o conjunto $\{M, N\}$, defina-se a *ordem parcial* $\leq$ como sendo $N \leq M$, $N \leq N$ e $M \leq M$, o *ou lógico* $\vee$ como sendo $N \vee N = N$, $N \vee M = M$, $M \vee N = M$ e $M \vee M = M$, e o *e lógico* $\wedge$ como sendo $N \wedge N = N$, $N \wedge M = N$, $M \wedge N = N$ e $M \wedge M = M$. As operações $\leq$, $\vee$ e $\wedge$ definidas para o conjunto $\{M, N\}$ são equivalentes às operações correspondentes para o conjunto binário $\{1, 0\}$, com o M fazendo o papel de 1. Dado o alfabeto Σ, para os controles $(\gamma_1, \#_1)$ e $(\gamma_2, \#_2)$ em $2^\Sigma \times \{M, N\}$, definem-

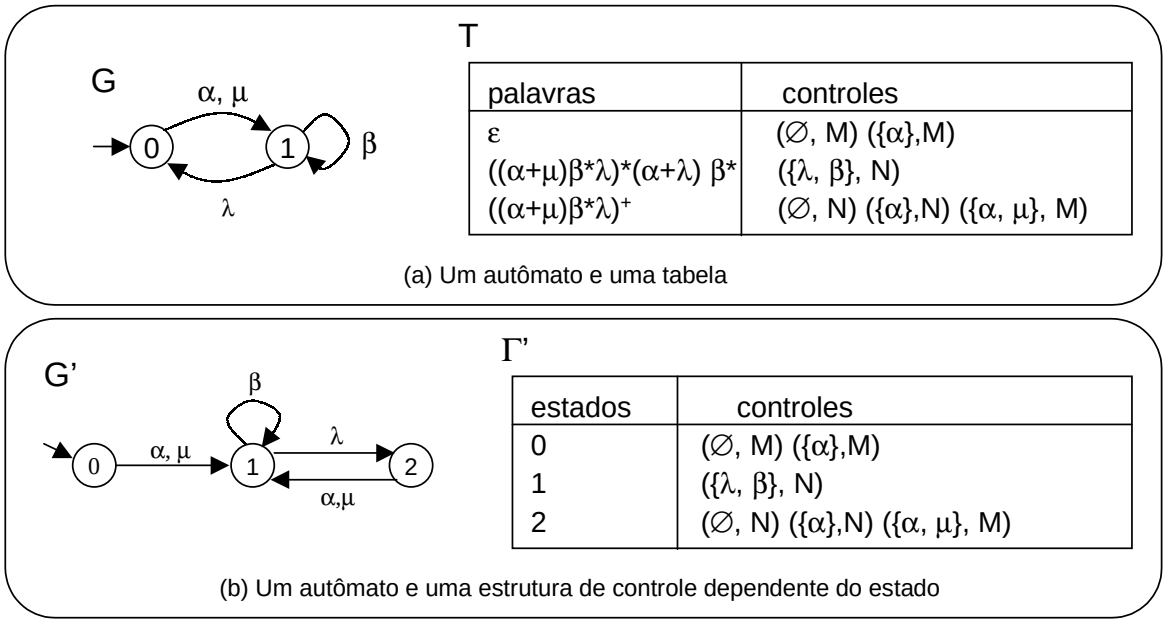


Figura 4.1: Diferentes representações de um SED com marcação flexível.

se as operações a seguir: *ordem parcial* $\subseteq$, como sendo $(\gamma_1, \#_1) \subseteq (\gamma_2, \#_2)$ se e somente se $\gamma_1 \subseteq \gamma_2$ e $\#_1 \leq \#_2$; e *união* $\cup$, como sendo $(\gamma_1, \#_1) \cup (\gamma_2, \#_2) = (\gamma_1 \cup \gamma_2, \#_1 \vee \#_2)$.

Definição 4.2 (Fechamento em relação à união) *Para o SED com marcação flexível S requer-se que, para todas as palavras s em L_S, o conjunto de controles $\Gamma_S(s)$ seja fechado em relação à união.*

Em termos práticos, o requisito na definição 4.2 afirma que a união de dois controles pertencentes ao conjunto de controles de uma dada palavra, também é um controle pertencente ao mesmo conjunto de controles. O requisito da definição 4.2 corresponde ao requisito usual para existência de um supervisor ótimo na literatura (Ramadge e Wonham 1989, Golaszewski e Ramadge 1987, Wong e Wonham 1996a, Li et al. 1998, Pu 2000), a ser empregado na próxima seção.

Os exemplos a seguir ilustram que alguns dos modelos significativos de SEDs dotados de controle na literatura podem ser expressos como casos particulares do modelo de um SED com marcação flexível.

Exemplo 4.2 (Abordagem RW padrão) *Para a abordagem RW padrão, vide seção 2.5, um SED S é representado por um par de linguagens $(L_S, L_{S,m})$ sobre um conjunto de eventos Σ, onde L_S contém todas as palavras que o sistema pode gerar e $L_{S,m}$ contém todas as palavras que correspondem a tarefas completas do sistema. O mecanismo de controle de eventos é definido pela partição do conjunto de eventos Σ em um conjunto de eventos controláveis (Σ_c) e um conjunto de eventos não controláveis (Σ_{nc}). A representação deste modelo como um SED de marcação flexível é o par (L_S, Γ_S), onde, para uma palavra s em L_S, $\Gamma_S(s) = \{(\gamma, \#) \in 2^{\Sigma} \times \{M, N\} : (\Sigma_{nc} \subseteq \gamma)\ e\ (\# = M,\ se\ s \in L_{S,m})\ e\ (\# = N,\ se\ s \in L_S - L_{S,m})\}$.*

Exemplo 4.3 (Controle supervisório com marcação) *O controle supervisório com marcação extende o modelo de SED da abordagem RW padrão ao permitir que o supervisor selecione que tarefas em malha aberta são consideradas tarefas em malha fechada, vide seção 2.5. A representação de um SED S com controle supervisório com marcação é semelhante à representação do SED da abordagem RW padrão, sendo que o conjunto de controles para uma palavra s gerada pelo sistema é definida como $\Gamma_S(s) = \{(\gamma, \#) \in 2^{\Sigma} \times \{M, N\} : (\Sigma_{nc} \subseteq \gamma)\ e\ (\# = N,\ se\ s \in L_S - L_{S,m})\}$.*

Exemplo 4.4 (SEDs com eventos forçáveis) *Os SEDs com eventos forçáveis, como introduzidos por Golaszewski e Ramadge (1987), estendem os SEDs da abordagem RW padrão pela definição de uma nova classe de eventos, chamados* eventos forçáveis. *De forma diferente aos eventos controláveis e não controláveis, que são gerados espontaneamente pelo sistema, a ocorrência dos eventos forçáveis é imposta ao sistema. O conjunto de eventos é então particionado em três subconjuntos, a saber $\Sigma = \Sigma_c \cup \Sigma_{nc} \cup \Sigma_f$, onde Σ_f é o conjunto de eventos forçáveis. A representação de um SED S com eventos forçáveis como pelo modelo de um SED com marcação flexível é similar à representação do SED da abordagem RW padrão, sendo que o conjunto de controles de uma palavra s gerada pelo sistema passa a ser representado por $\Gamma_S(s) = [\Gamma_f]_{\cup}$, onde $\Gamma_f = \{(\gamma, \#) \in 2^{\Sigma} \times \{M, N\} : (((\exists \sigma \in \Sigma_f) : \{\sigma\} = \gamma)\ ou\ (\Sigma_{nc} \subseteq \gamma))\ e(\# = N\ se\ s \in L_S - L_{S,m})\ e\ (\# = M\ se\ s \in L_{S,m})\}$ e $[A]_{\cup}$ denota o fechamento*

em relação à união do conjunto A.

Exemplo 4.5 (SEDs temporizados) *No controle supervisório de SEDs temporizados, conforme introduzido por Brandin e Wonham (1994), novas classes de eventos estendem o modelo para o SED da abordagem RW padrão. O tempo é representado pelo evento* tick*, que indica a evolução discreta no tempo do sistema, como o tique-taque de um relógio. Os outros eventos do sistema são chamados* eventos de atividade *(conjunto* Σ_{act}*). O conjunto total de eventos do sistema é então* $\Sigma = \Sigma_{act} \cup \{tick\}$*. Dos eventos de atividade, os* eventos proibitivos *(conjunto* Σ_{hib}*) são controláveis como na abordagem RW padrão. Os eventos não controláveis do sistema são então* $\Sigma_{nc} = \Sigma_{act} - \Sigma_{hib}$*. Um atributo adicional de um evento de atividade, além de ser controlável ou não controlável, é que este pode ser forçável em relação ao evento tick (conjunto* Σ_{for}*). Um evento forçável em relação ao evento tick pode* vencer o relógio*, no sentido de que, na presença de tal evento, o evento tick pode ser desabilitado. Quando não há eventos forçáveis em relação ao evento tick, este não pode ser desabilitado. A representação de um SED temporizado S como um SED com marcação flexível é semelhante à abordagem RW padrão, sendo que o conjunto de controles para a palavra gerada s do sistema passa a ser definido como* $\Gamma_S(s) = \{(\gamma, \#) \in 2^{\Sigma} \times \{M, N\} : ((\gamma \cap \Sigma_{for} = \emptyset \Rightarrow \Sigma_{nc} \cup \{tick\} \subseteq \gamma)$ *ou* $(\gamma \cap \Sigma_{for} \neq \emptyset \Rightarrow \Sigma_{nc} \subseteq \gamma))$ *e* $(\# = N$ *se* $s \in L_S - L_{S,m})$ *e* $(\# = M$ *se* $s \in L_{S,m})\}$.

De forma similar, os modelos de outras classes de SEDs dotados de controle encontrados na literatura podem ser expressos como casos particulares do modelo de um SED com marcação flexível, como por exemplo, os SEDs com reinicialização de Li et al. (1999), os modelos generalizados de Wong e Wonham (1996a), Li et al. (1998), Pu (2000), Sathaye e Krogh (1998), e o sistema discreto equivalente a um sistema híbrido de Cury et al. (1998).

Para mostrar as capacidades de modelagem de SEDs com marcação flexível, apresentam-se os dois exemplos a seguir.

Exemplo 4.6 (Máquina de três estados) *Representa-se na figura 4.2(a) o modelo segundo a abordagem RW de uma máquina com três estados genérica, vide exemplo*

2.3. Os estados da máquina são parada (P), trabalhando (T) e quebrada (Q), e é dirigida pelos seguintes eventos: início de operação (α), fim de operação (β), quebra (λ) e reparo (μ). O estado marcado é P, e os eventos não controláveis são o fim de operação β e a quebra λ. A máquina inicialmente encontra-se no estado P. A figura 4.2(b) representa o modelo da mesma máquina usando SEDs com marcação flexível. Adicionalmente, a figura 4.2(c) representa também a máquina como um SED com marcação flexível, sendo que considera-se o controle supervisório com marcação. Observa-se que a diferença dos modelos nas figuras 4.2(b) e 4.2(c) é a disponibilidade dos controles M e N para o estado marcado. Enquanto na figura 4.2(b) somente os controles do tipo M estão disponíveis para o estado marcado, pois na abordagem RW o supervisor não pode alterar a marcação, na figura 4.2(c) os controles do tipo M e N estão disponíveis, a significar que o supervisor agora pode escolher não marcar o estado em malha fechada.

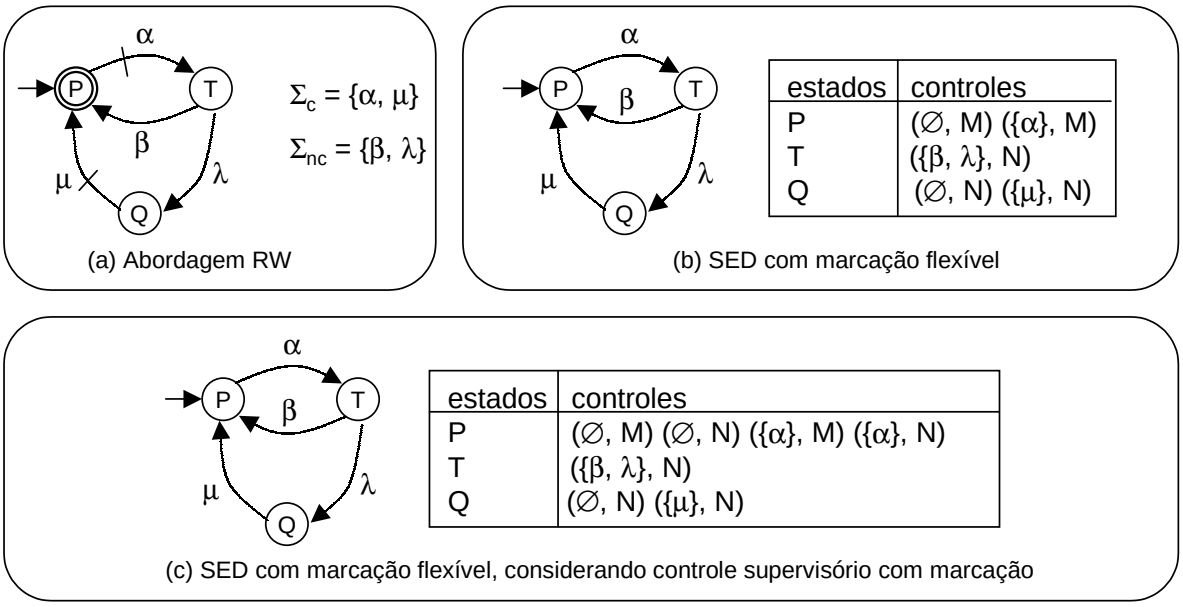


Figura 4.2: Diferentes representações de uma máquina de três estados.

Exemplo 4.7 (Um gato e um rato num labirinto) *Considere a seguinte extensão do problema clássico de um gato e um rato num labirinto (Wonham e*

Ramadge 1987). Como representado na figura 4.3, o labirinto consiste em 4 salas onde o gato e o rato estão confinados. Os animais deslocam-se de uma sala a outra através de uma rede privada de dutos, representados pelas linhas conectando as salas na figura 4.3. Os dutos são direcionais, indicados pelas setas, e o acesso do animal ao duto pode ser proibido por uma porta, indicada pelas linhas que cruzam as linhas que definem os dutos. Em alguns casos, o fechamento de um duto acarreta no fechamento de outro duto, como no caso em que o rato esteja na sala 3 e o fechamento do acesso à sala 1 também proíbe o acesso à sala 2. Para se manterem no labirinto, os animais devem ser alimentados, e a comida está disponível para os animais no interior de alguns dutos, indicados pelas estrelas na figura 4.3. Quando o acesso a um duto é proibido, também se impede o acesso à comida que está dentro do mesmo. O gato está inicialmente na sala 1 e o rato na sala 3. O problema de controle supervisório a ser tratado é projetar um supervisor para manter o gato e o rato vivos no labirinto. O supervisor nunca deve permitir que os dois estejam juntos numa mesma sala, quando supõe-se que o gato come o rato. Por outro lado, o acesso individual à comida deve ser garantido, senão os animais morrem de fome.

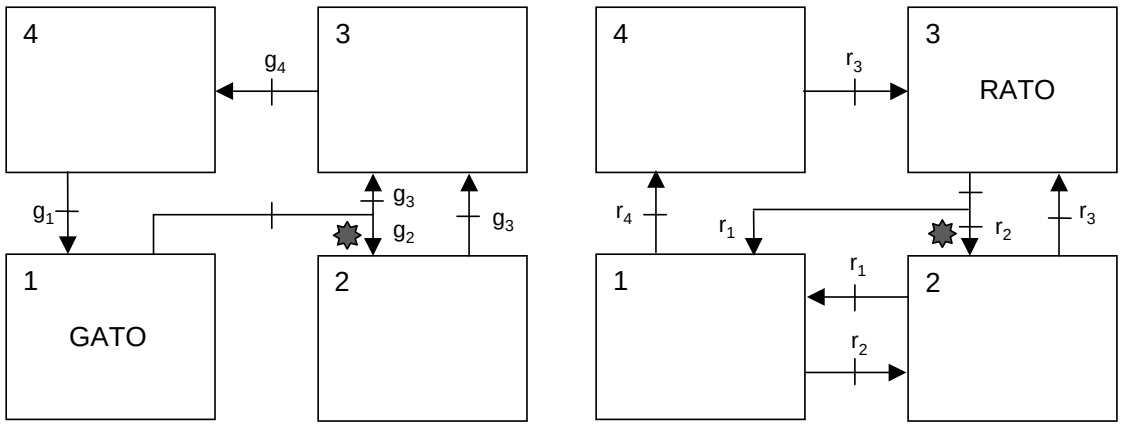
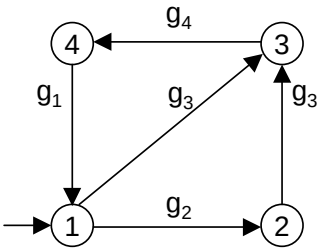


Figura 4.3: Um gato e um rato num labirinto.

Considere a representação do gato e do rato separadamente como SEDs com marcação flexível. Os comportamentos do gato e do rato são modelados por autômatos

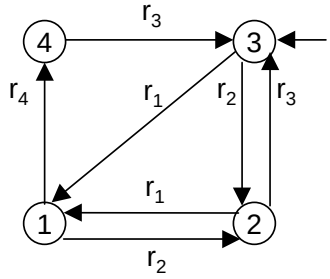
individuais na figura 4.4. Os eventos g_i e r_j indicam que o gato e o rato entraram nas salas i e j, respectivamente, para i e j em $\{1, 2, 3, 4\}$. A numeração dos estados dos autômatos indica o número da sala ocupada. Os conjuntos de controles para o gato e o rato são construídos por observação das opções de controle e marcação no labirinto da figura 4.3. Quando o duto que sai de uma sala pode ser fechado, o correspondente evento pode ser retirado de um controle disponível para o estado que corresponde à referida sala. Quando o acesso a um duto com comida é permitido por um controle, considera-se este controle como marcado.

GATO:



estados	controles
1	(∅,N) ({g_2}, M) ({g_2, g_3}, M)
2	(∅,N) ({g_3}, N)
3	(∅,N) ({g_4}, N)
4	(∅,N) ({g_1}, N)

RATO:



estados	controles
1	(∅,N) ({r_2}, N) ({r_4}, N) ({r_2, r_4}, N)
2	(∅,N) ({r_1}, N) ({r_3}, N) ({r_1, r_3}, N)
3	(∅,N) ({r_1}, N) ({r_1, r_2}, M)
4	(∅,N) ({r_3}, N)

Figura 4.4: Modelos por SED com marcação flexível para o gato e o rato.

Se o mesmo sistema for modelado pela abordagem RW, a rede de dutos deve ser detalhada para se descrever as opções de controle e marcação de forma apropriada. Os dutos são decompostos em dutos menores e surgem novas salas antes ocultas, como representado na figura 4.5. A proibição do acesso passa ser sala-a-sala exclusivamente, sem a interdependência existente no modelo não detalhado. Para detalhar os dutos para o gato, a sala 5 é criada, e, para detalhar os dutos para o rato, as salas 6 e 7 são criadas. Não há conexão entre a nova sala para o gato e as novas salas para o rato, e as salas comuns permanecem sendo 1, 2, 3 e 4.

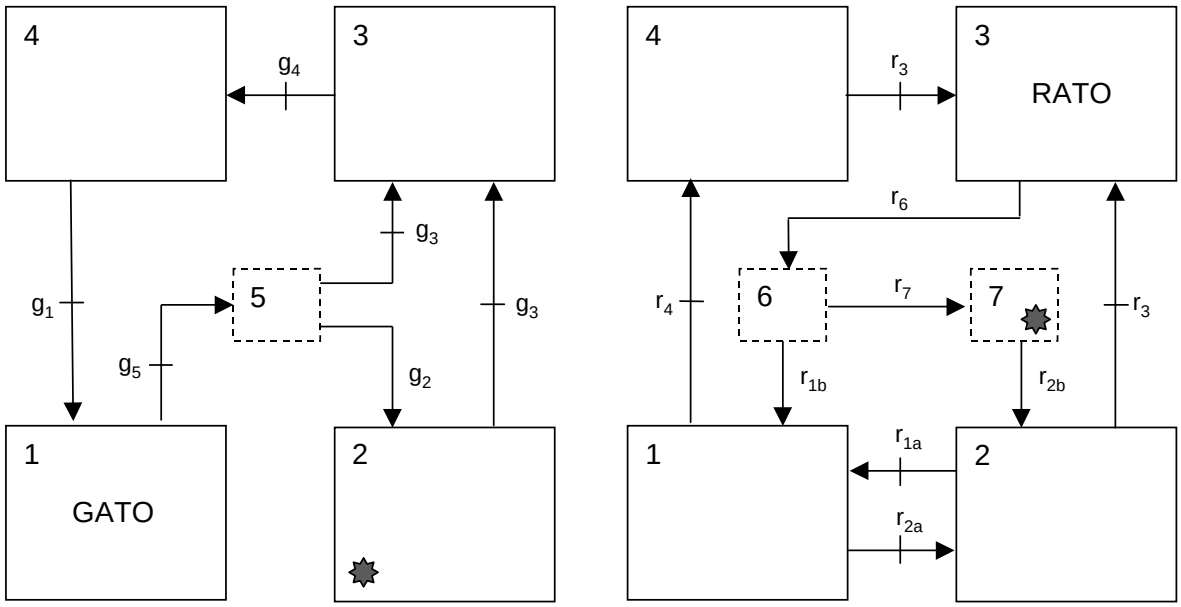


Figura 4.5: Uma possível descrição do problema para a abordagem RW.

A figura 4.6 representa os modelos da abordagem RW para o gato e o rato, baseados no detalhamento anterior. Um autômato com 5 estados e 6 transições é necessário para o gato, e um autômato com 6 estados e 9 transições é necessário para o rato. Observa-se que, para o modelo do rato, os eventos que correspondem a entrada nas salas 1 e 2 são discriminados cada um por um par de eventos (r_{1a}, r_{1b}, r_{2a} e r_{2b} na figura 4.5), correspondendo a instâncias controláveis e não controláveis dos mesmos.

O exemplo 4.7 ilustra três ações comuns para traduzir um sistema descrito por um SED com marcação flexível para o modelo da abordagem RW, quais sejam: criar novos estados e transições para detalhar a dinâmica subjacente, fazer a partição dos eventos em controláveis e não controláveis, e marcar os estados correspondentes ao completamento de tarefas.

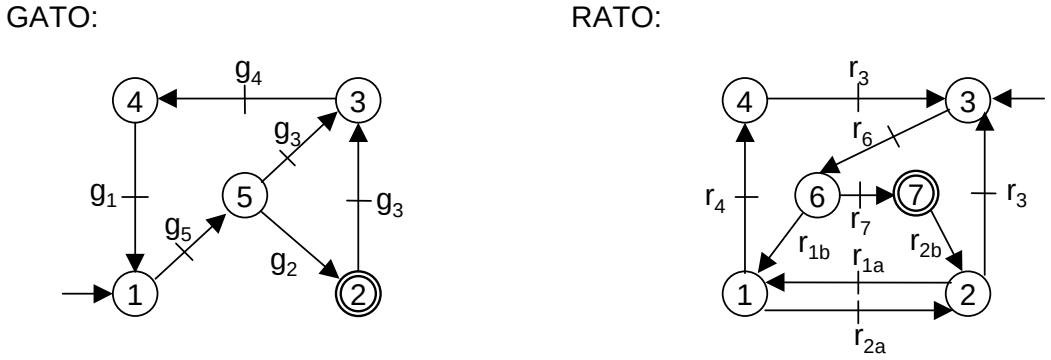


Figura 4.6: Modelos para o gato e o rato na abordagem RW.

4.2 Existência de supervisores

Nesta seção apresenta-se um esquema de controle supervisório para um SED com marcação flexível.

Definição 4.3 (Supervisor) *Para o SED com marcação flexível S sobre o alfabeto Σ, um supervisor para S é uma função $f : L_S \to 2^{\Sigma} \times \{M, N\}$ que associa a cada palavra s em L_S um controle $f(s) \in \Gamma_S(s)$.*

Para a palavra s em L_S, se $f(s) = (\gamma, \#) \in \Gamma_S(s)$, o conjunto ativo de eventos em L_S após s, $\Sigma_{L_S}(s)$, fica restrito a $\gamma \cap \Sigma_{L_S}(s)$, e considera-se s como marcada se $\#$ for igual a M, caso contrário, considera-se s como não marcada.

Definição 4.4 (Comportamento em malha fechada) *Para o SED com marcação flexível S sobre o alfabeto Σ, e o supervisor f para S, o sistema em malha fechada, denotado f/S, possui comportamento definido por duas linguagens, a saber, uma linguagem gerada, $L_{f/S}$, definida recursivamente por $\epsilon \in L_{f/S}$ e*

$$s\sigma \in L_{f/S} \iff ((s \in L_{f/S}) \text{ e } (s\sigma \in L_S) \text{ e } (\sigma \in \gamma, \text{ com } f(s) = (\gamma, \#))), \quad (4.1)$$

onde $\sigma \in \Sigma$ e $s \in \Sigma^*$, e a linguagem marcada em malha fechada, $L_{f/S,m}$, por

$$s\sigma \in L_{f/S,m} \iff (s \in L_{f/S}) \text{ e } (f(s) = (\gamma, M)). \quad (4.2)$$

A linguagem gerada em malha fechada $L_{f/S}$ contém as palavras em L_S permitidas por f sob supervisão, e é prefixo fechada. A linguagem marcada em malha fechada $L_{f/S,m}$, é uma linguagem contida em $L_{f/S}$ onde, para todas as palavras, o supervisor escolhe o indicador M. De forma geral, $\overline{L_{f/S,m}} \subseteq L_{f/S}$, isto é, pode haver palavras geradas em malha fechada que não sejam prefixos de palavras marcadas em malha fechada. Tais palavras não evoluem para completar uma tarefa do sistema, caracterizando uma situação chamada *bloqueio*.

Definição 4.5 (Supervisor não bloqueante) *Para o SED com marcação flexível S e o supervisor f para S, diz-se que f é não bloqueante se $\overline{L_{f/S,m}} = L_{f/S}$.*

Para um supervisor não bloqueante, todas as palavras geradas em malha fechada, são prefixos de tarefas completas. Dentro desta perspectiva, enuncia-se um problema de controle supervisório para os SEDs com marcação flexível.

Definição 4.6 (Problema de controle supervisório (PCS)) *Dados o SED com marcação flexível S e a linguagem $K \subseteq L_S$, encontre um supervisor não bloqueante f para S tal que $\emptyset \subset L_{f/S,m} \subseteq K$.*

Introduz-se o conceito de linguagem compatível, também referenciado como compatibilidade de uma linguagem, para trazer uma solução para o PCS para SEDs com marcação flexível.

Definição 4.7 (Linguagem compatível) *Dados o SED com marcação flexível S e a linguagem $K \subseteq L_S$, diz-se que K é (L_S, Γ_S)-compatível se e somente se*

- *para todo s em K, existir (γ, M) em $\Gamma_S(s)$ tal que $\gamma \cap \Sigma_{L_S}(s) = \Sigma_K(s)$, e*
- *para todo s em $\overline{K} - K$, existir (γ, N) em $\Gamma_S(s)$ tal que $\gamma \cap \Sigma_{L_S}(s) = \Sigma_K(s)$.*

Exemplo 4.8 (Linguagem compatível) *Para ilustrar o conceito de linguagem compatível, considere o SED com marcação flexível S do exemplo 4.1 e as linguagens K_1*

e K_2 marcadas pelos autômatos representados na figura 4.7. A linguagem K_1 é não (L_S, Γ_S)-compatível porque, para as palavras que terminam no estado 1 do autômato de K_1 não está disponível o controle $(\{\lambda\}, N)$ no estado correspondente de S (vide figura 4.1). A linguagem K_2 não é (L_S, Γ_S)-compatível porque primeiro, para as palavras que terminam no estado 1 do autômato de K_2 não está disponível o controle $(\{\beta, \lambda\}, M)$ no estado correspondente de S, e segundo, para as palavras que terminam no estado 2 do autômato de K_2, não está disponível o controle $(\{\alpha\}, M)$ no estado correspondente de S.

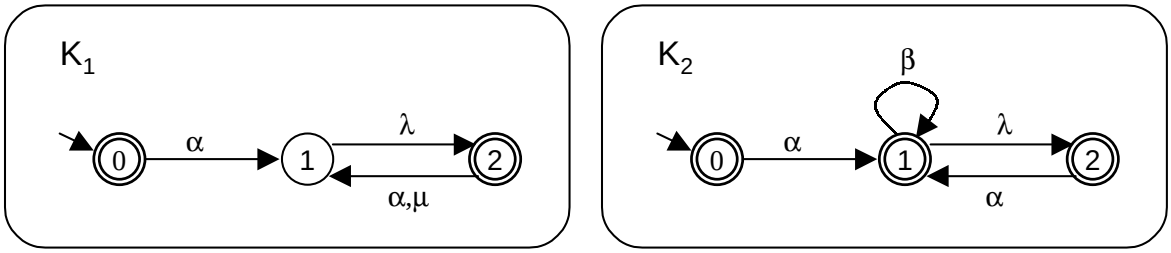


Figura 4.7: Linguagens de especificação para o exemplo 4.1.

Eventualmente se um SED com marcação flexível for definido por um autômato G e uma estrutura de controle dependente do estado Γ, diz-se que uma linguagem é (G, Γ)-compatível.

Teorema 4.1 (Existência de supervisores) *Dados o SED com marcação flexível S e a linguagem $K \subseteq L_S$, com K não vazio, existe um supervisor não bloqueante f para S tal que $L_{f/S,m} = K$ se e somente se K for (L_S, Γ_S)-compatível.*

Demonstração. (**Se**) Suponha que K seja (L_S, Γ_S)-compatível, e seja f definido por:

$$f(s) = \begin{cases} (\gamma, M) \in \Gamma_S(s), & \text{para } s \in K \\ (\gamma, N) \in \Gamma_S(s), & \text{para } s \in \overline{K} - K \\ \text{não definido}, & \text{caso contrário} \end{cases} \tag{4.3}$$

onde $\gamma \cap \Sigma_{L_S}(s) = \Sigma_K(s)$. Mostra-se primeiro que $L_{f/S} = \overline{K}$, e então que $L_{f/S,m} = K$

Prova-se que $L_{f/S} = \overline{K}$ por indução.

Base: Sabe-se que $\epsilon \in L_{f/S}$ e $\epsilon \in \overline{K}$.

Hipótese: Suponha que para $s \in L_S$, $s \in L_{f/S}$ e $s \in \overline{K}$.

Passo: Primeiro, seja $\sigma \in \Sigma$ tal que $s\sigma \in L_{f/S}$. Então, pela definição 4.4, $s\sigma \in L_S$ e $\sigma \in \gamma$, com $(\gamma, \#) = f(s) \in \Gamma_S(s)$, isto é, $\sigma \in \gamma \cap \Sigma_{L_S}(s)$. Pela equação (4.3), $\sigma \in \Sigma_K(s)$, e, conseqüentemente, $s\sigma \in \overline{K}$. Por outro lado, para $\sigma \in \Sigma$, seja $s\sigma \in \overline{K}$, isto é, $\sigma \in \Sigma_K(s)$. Como K é (L_S, Γ_S)-compatível, existe $(\gamma, \#) \in \Gamma_S(s)$ tal que $\gamma \cap \Sigma_{L_S}(s) = \Sigma_K(s)$. Pela equação (4.3), $(\gamma, \#) = f(s)$. Tem-se então $\sigma \in \gamma \cap \Sigma_{L_S}(s)$, com $(\gamma, \#) = f(s)$. Assim, $s\sigma \in L_S$ e $\sigma \in \gamma$, com $(\gamma, \#) = f(s)$. Pela definição 4.4, $s\sigma \in L_{f/S}$.

Assim, conclui-se $L_{f/S} = \overline{K}$.

Agora, como K é (L_S, Γ_S)-compatível, para todo s em K, existe (γ, M) em $\Gamma_S(s)$ tal que $\gamma \cap \Sigma_{L_S}(s) = \Sigma_K(s)$. Como $L_{f/S} = \overline{K}$, então $s \in L_{f/S}$. Pela definição 4.4, $f(s) = (\gamma, M)$ acima. Assim, $s \in L_{f/S,m}$. Por outro lado, seja s em $L_{f/S,m}$, então $s \in L_{f/S}$, e pela prova indutiva acima, $s \in \overline{K}$. Mas como $s \in L_{f/S,m}$, $f(s) = (\gamma, M) \in \Gamma_S(s)$, com $\gamma \cap \Sigma_{L_S}(s) = \Sigma_K(s)$. Mas, como K é (L_S, Γ_S)-compatível, $s \in K$. Assim, prova-se que $L_{f/S,m} = K$.

Finalmente, como $L_{f/S} = \overline{K}$ e $L_{f/S,m} = K$, tem-se que $L_{f/S} = \overline{L_{f/S,m}}$. Assim, f é não bloqueante.

(**Somente se**) Seja f um supervisor não bloqueante para S, prova-se que $K = L_{f/S,m}$ é (L_S, Γ_S)-compatível. Seja $s \in K$, pela definição 4.4, $f(s) = (\gamma, M) \in \Gamma_S(s)$ com $\gamma \cap \Sigma_{L_S}(s) = \Sigma_K(s)$. Por outro lado, seja $s \in \overline{K} - K$, isto é, s está em $L_{f/S}$ e não está em $L_{f/S,m}$, assim, também pela definição 4.4, $f(s) = (\gamma, N) \in \Gamma_S(s)$ com $\gamma \cap \Sigma_{L_S}(s) = \Sigma_K(s)$. Dessa forma, verifica-se que f atende às duas condições de (L_S, Γ_S)-compatibilidade, definição 4.7, e conclui-se que $K = L_{f/S,m}$ é (L_S, Γ_S)-compatível. □

Dados o SED com marcação flexível S e a linguagem $K \subseteq L_S$, o teorema 4.1 mostra que a condição de (L_S, Γ_S)-compatibilidade para K é necessária e suficiente

para que exista um supervisor não bloqueante f para S tal que $L_{f/S,m} = K$. Diz-se então que K é realizável por supervisor não bloqueante para S. Como mostrado pelo teorema 4.1, a condição de compatibilidade é equivalente à conjunção das condições de controlabilidade e fechamento em relação à linguagem marcada para a abordagem RW (vide seção 2.6.4).

Teorema 4.2 (Máxima linguagem compatível) *Dados o SED com marcação flexível S e a linguagem $K \subseteq L_S$, seja $C_{(L_S,\Gamma_S)}(K)$ o conjunto de linguagens (L_S, Γ_S)-compatíveis contidas em K, afirma-se o seguinte:*

1. *$C_{(L_S,\Gamma_S)}(K)$ é não vazio,*
2. *$C_{(L_S,\Gamma_S)}(K)$ é fechado em relação à união, e*
3. *$C_{(L_S,\Gamma_S)}(K)$ possui um único elemento supremo, denominado máxima linguagem (L_S, Γ_S)-compatível em K, obtido por*

$$\sup C_{(L_S,\Gamma_S)}(K) \quad = \quad \bigcup_{E \in C_{(L_S,\Gamma_S)}(K)} E. \tag{4.4}$$

Demonstração. Prova-se o ítem 1 pelo fato de que a linguagem vazia, $\emptyset$, é trivialmente (L_S, Γ_S)-compatível, o que pode ser verificado por inspeção da definição 4.7.

Para provar o ítem 2, sejam K_1 e K_2 linguagens contidas em K que sejam (L_S, Γ_S)-compatíveis. É suficiente provar que a linguagem $K_1 \cup K_2$ seja também (L_S, Γ_S)-compatível (observe que $K_1 \cup K_2$ também está contida em K).

Para $s \in K_1 \cup K_2$ podem ocorrer as seguintes situações:

1. Se $s \in K_i$ e $s \notin \overline{K_j}$, com i e j em $\{1, 2\}$ e $i \neq j$, como K_i é (L_S, Γ_S)-compatível, existe (γ_i, M) em $\Gamma_S(s)$ tal que $\gamma \cap \Sigma_{L_S}(s) = \Sigma_{K_i}(s)$, satisfazendo o primeiro requisito da condição de (L_S, Γ_S)-compatibilidade, definição 4.7.
2. Se $s \in K_i$ e $s \in \overline{K_j} - K_j$, com i e j em $\{1, 2\}$ e $i \neq j$, como K_i é (L_S, Γ_S)-compatível, existe (γ_i, M) em $\Gamma_S(s)$ tal que $\gamma \cap \Sigma_{L_S}(s) = \Sigma_{K_i}(s)$, e como K_j

é (L_S, Γ_S)-compatível, existe (γ_j, N) em $\Gamma_S(s)$ tal que $\gamma \cap \Sigma_{L_S}(s) = \Sigma_{K_j}(s)$. Novamente, pela definição 4.2, existe $(\gamma_i \cup \gamma_j, M)$ em $\Gamma_S(s)$ tal que $(\gamma_i \cup \gamma_j) \cap \Sigma_{L_S}(s) = (\gamma_i \cap \Sigma_{L_S}(s)) \cup (\gamma_j \cap \Sigma_{L_S}(s)) = \Sigma_{K_i}(s) \cup \Sigma_{K_j}(s) = \Sigma_K(s)$, satisfazendo o primeiro requisito da definição 4.7.

3. Se $s \in K_1$ e $s \in K_2$, como K_i é (L_S, Γ_S)-compatível, i em $\{1, 2\}$, existe (γ_i, M) em $\Gamma_S(s)$ tal que $\gamma \cap \Sigma_{L_S}(s) = \Sigma_{K_i}(s)$. Como o conjunto de controles para uma palavra de S é fechado em relação à união, o requisito na definição 4.2, existe $(\gamma_1 \cup \gamma_2, M)$ em $\Gamma_S(s)$ tal que $(\gamma_1 \cup \gamma_2) \cap \Sigma_{L_S}(s) = (\gamma_1 \cap \Sigma_{L_S}(s)) \cup (\gamma_2 \cap \Sigma_{L_S}(s)) = \Sigma_{K_1}(s) \cup \Sigma_{K_2}(s) = \Sigma_K(s)$, satisfazendo o primeiro requisito da definição 4.7.

Para s em $\overline{K_1 \cup K_2} - K_1 - K_2$, podem ocorrer as seguintes situações:

1. Se $s \in \overline{K_i} - K_i$ e $s \notin \overline{K_j}$, com i e j em $\{1, 2\}$ e $i \neq j$, como K_i é (L_S, Γ_S)-compatível, existe (γ_i, N) em $\Gamma_S(s)$ tal que $\gamma \cap \Sigma_{L_S}(s) = \Sigma_{K_i}(s)$, satisfazendo o segundo requisito da definição 4.7.

2. Se $s \in \overline{K_1} - K_1$ e $s \in \overline{K_2} - K_2$, como K_i é (L_S, Γ_S)-compatível, existe (γ_i, N) em $\Gamma_S(s)$ tal que $\gamma \cap \Sigma_{L_S}(s) = \Sigma_{K_i}(s)$. Também pela definição 4.2, existe $(\gamma_1 \cup \gamma_2, N)$ em $\Gamma_S(s)$ tal que $(\gamma_1 \cup \gamma_2) \cap \Sigma_{L_S}(s) = (\gamma_1 \cap \Sigma_{L_S}(s)) \cup (\gamma_2 \cap \Sigma_{L_S}(s)) = \Sigma_{K_1}(s) \cup \Sigma_{K_2}(s) = \Sigma_K(s)$, satisfazendo o segundo requisito da definição 4.7.

Pela enumeração de fatos acima, tem-se que $K_1 \cup K_2$ é (L_S, Γ_S)-compatível, isto é, $K_1 \cup K_2$ está em $C_{(L_S,\Gamma_S)}(K)$. Dessa forma, prova-se que $C_{(L_S,\Gamma_S)}(K)$ é fechado em relação à união.

Finalmente, prova-se o ítem 3 pelo seguinte argumento. Como, pelos ítens 1 e 2, o conjunto $C_{(L_S,\Gamma_S)}(K)$ é não vazio e fechado em relação à união, existe um elemento supremo para este conjunto, a máxima linguagem (L_S, Γ_S)-compatível contida em K. Obtém-se este elemento supremo pela união de todos os elementos do conjunto $C_{(L_S,\Gamma_S)}(K)$, o que corresponde exatamente à equação (4.4). □

Para um SED com marcação flexível S e uma linguagem $K \subseteq L_S$, a especificar um comportamento desejado para S, pode ser o caso que K não seja (L_S, Γ_S)-compatível,

isto é, não exista supervisor não bloqueante para S que realize o comportamento especificado por K. Neste caso, o teorema 4.2 prova que existe um conjunto de linguagens (L_S, Γ_S)-compatíveis contidas em K, tendo este conjunto uma única linguagem suprema, a significar o comportamento compatível e o mínimo restritivo ao comportamento de S que segue à especificação K.

Proposição 4.1 (Solução do PCS) *O PCS da definição 4.6 possui solução se e somente se* $\sup C_{(L_S,\Gamma_S)}(K) \neq \emptyset$.

Demonstração. (**Se**) Suponha que $\sup C_{(L_S,\Gamma_S)}(K) \neq \emptyset$, então, pelo teorema 4.1, existe um supervisor não bloqueante f para S tal que $\emptyset \neq L_{f/S,m} = \sup C_{(L_S,\Gamma_S)}(K) \subseteq K$. Assim, o PCS da definição 4.6 possui uma solução.

(**Somente se**) Suponha que o PCS da definição 4.6 possui uma solução, isto é, existe um supervisor não bloqueante f para S tal que $\emptyset \neq L_{f/S,m} \subseteq K$. Seja $K' = L_{f/S,m}$, então, pelo teorema 4.1, K' é (L_S, Γ_S)-compatível e não vazia. Assim, o conjunto $C_{(L_S,\Gamma_S)}(K)$ possui uma linguagem além da linguagem vazia e, pelo teorema 4.2, existe uma máxima linguagem (L_S, Γ_S)-compatível e não vazia contida em K. Dessa forma conclui-se que $\sup C_{(L_S,\Gamma_S)}(K) \neq \emptyset$. □

A partir da proposição 4.1, propõe-se uma forma de solução ótima para o PCS da definição 4.6. Dada uma linguagem de especificação de comportamento K para um SED com marcação flexível S, emprega-se a máxima linguagem (L_S, Γ_S)-compatível contida em K para implementar o supervisor que segue a K de forma ótima, no sentido de ser o mínimo restritivo para o comportamento de S. Assim, o cálculo da máxima linguagem (L_S, Γ_S)-compatível de uma dada linguagem é fundamental na síntese de supervisores para SEDs com marcação flexível. Na próxima seção trata-se de um algoritmo de cálculo da máxima linguagem compatível para o caso em que os sistemas e as especificações são expressas por autômatos de estados finitos.

Por fim, observa-se que os resultados desta seção são válidos independentemente da hipótese das linguagem serem regulares ou não.

4.3 Cálculo da máxima linguagem compatível

Apresenta-se nesta seção um algoritmo para calcular a máxima linguagem compatível contida numa dada linguagem em relação a um dado SED com marcação flexível.

Dado o SED com marcação flexível S e a linguagem $K \subseteq L_S$, deseja-se calcular a máxima linguagem (L_S, Γ_S)-compatível contida em K. Este cálculo é caracterizado pela busca do máximo ponto fixo de um operador sobre linguagens, como visto a seguir.

Definição 4.8 (Operador Ω) *Para o SED com marcação flexível S sobre Σ e a linguagem $A \subseteq L_S$, defina-se o operador $\Omega : 2^{L_S} \to 2^{L_S}$ por:*

$$\begin{aligned}
\Omega(A) \;=\; \{s \in A :\; & ((i)\exists(\gamma, M) \in \Gamma_S(s) : \gamma \cap \Sigma_{L_S}(s) \subseteq \Sigma_A(s)) \; e \\
& ((\forall s' \in \overline{A} \; e \; \forall \sigma \in \Sigma, \; com \; s'\sigma \leq s) : \\
& \quad ((ii)(s' \in A) \Rightarrow (\exists(\gamma, \#) \in \Gamma_S(s') : \\
& \quad\quad \gamma \cap \Sigma_{L_S}(s') \subseteq \Sigma_A(s') \; e \; \sigma \in \gamma)) \; e \\
& \quad ((iii)(s' \in \overline{A} - A) \Rightarrow (\exists(\gamma, N) \in \Gamma_S(s') : \\
& \quad\quad \gamma \cap \Sigma_{L_S}(s') \subseteq \Sigma_A(s') \; e \; \sigma \in \gamma)))\}.
\end{aligned} \tag{4.5}$$

O operador Ω corta palavras s da linguagem A que falham em verificar pelo menos uma de três condições, numeradas na equação (4.5). A condição (i) afirma que o sistema deve dispor de um controle marcado (γ, M) tal que $\gamma \cap \Sigma_{L_S}(s) \subseteq \Sigma_A(s)$, a significar que o controle restringe o sistema a gerar após s apenas os eventos que estejam previstos no conjunto de eventos ativos em A após s. A condição (ii) afirma que, para todo prefixo estrito de s que é elemento de A, diga-se $s' \in A$ com $s'\sigma \leq s$ e $\sigma \in \Sigma$, o sistema deve dispor de um controle $(\gamma, \#)$ tal que $\gamma \cap \Sigma_{L_S}(s') \subseteq \Sigma_A(s')$ e $\sigma \in \gamma$, esta última a significar que o controle escolhido garante que o sistema pode evoluir de s' a s, assegurado pela permissão de se ocorrer σ após s. Finalmente a condição (iii) afirma que, para todo prefixo de s que é elemento do prefixo de A mas não de A, diga-se $s' \in \overline{A} - A$ com $s'\sigma \leq s$ e $\sigma \in \Sigma$, o sistema deve dispor de um controle não marcado (γ, N) tal que $\gamma \cap \Sigma_{L_S}(s') \subseteq \Sigma_A(s')$ e $\sigma \in \gamma$.

Ilustra-se a sucessiva aplicação do operador Ω pelo exemplo a seguir.

Exemplo 4.9 (Aplicações sucessivas do operador Ω) *Considere o SED com marcação flexível sobre* $\Sigma = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ *representado pelo par* (G, Γ) *na figura 4.8(a) e a linguagem* A_1 *marcada pelo autômato na figura 4.8(b). O objetivo é ilustrar sucessivas aplicações do operador* Ω*, equação (4.5), a* A_1 *e às linguagens resultantes. A aplicação de* Ω *a* A_1 *elimina as palavras* α_1 *e* $\alpha_1\alpha_2$ *de* A_1*, ambas por não atenderem à primeira condição do operador* Ω*. O resultando é* $A_2 = \Omega(A_1)$*, representado na figura 4.8(c). A aplicação de* Ω *a* A_2 *agora elimina a palavra* $\alpha_1\alpha_3$*, em função do prefixo* α_1 *não atender à segunda condição do operador* Ω*. O resultado é* $A_3 = \Omega(A_2) = \{\epsilon\}$*, representada na figura 4.8(d). Finalmente, aplicando-se* Ω *a* A_3 *tem-se* $\Omega(A_3) = A_3$*, e aplicações subseqüentes produzem o mesmo resultado.*

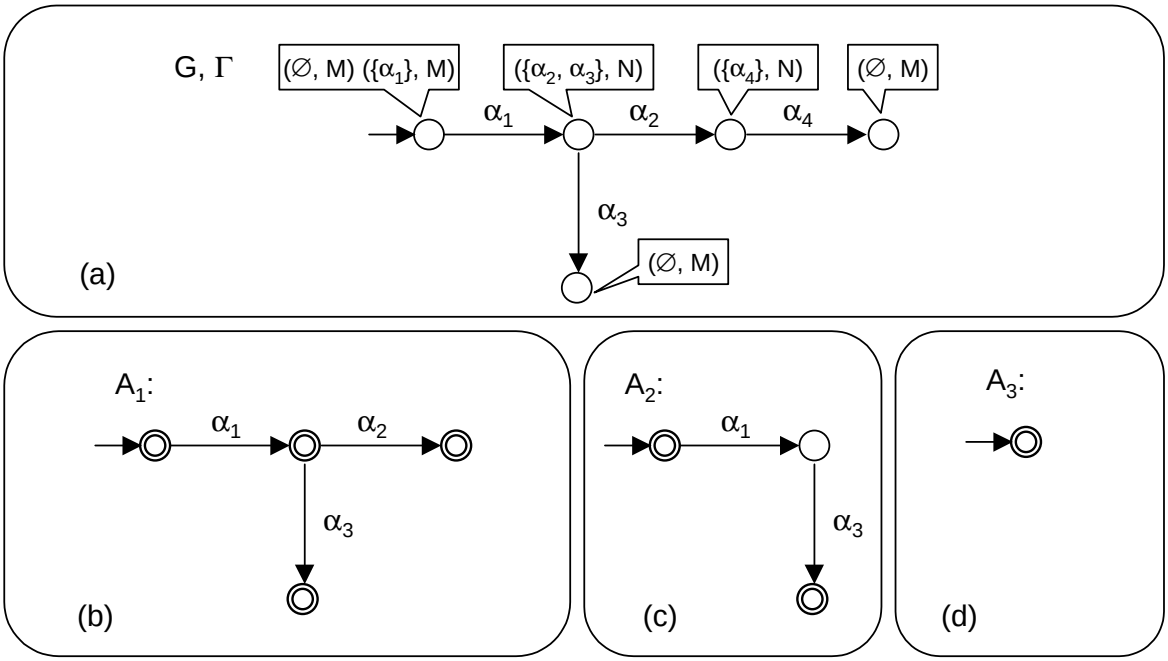


Figura 4.8: Aplicações sucessivas do operador Ω.

Para o operador Ω, definido na equação (4.5), diz-se que a linguagem $A \subseteq L_S$ é um *ponto fixo* de Ω quando $\Omega(A) = A$. Observa-se que a linguagem vazia, ∅, é um ponto fixo de Ω de forma trivial.

Proposição 4.2 (Pontos fixos e linguagens compatíveis) *Para o SED com marcação flexível S e a linguagem $K \subseteq L_S$, os pontos fixos do operador Ω contidos em K são as linguagens (L_S, Γ_S)-compatíveis contidas em K.*

Demonstração. Prova-se que, para toda linguagem $A \subseteq K$, tem-se que $\Omega(A) = A$ se e somente se A for (L_S, Γ_S)-compatível.
(**Se**) Pela definição 4.7, se A é (L_S, Γ_S)-compatível, então, para todo $s \in A$,

- existe (γ, M) em $\Gamma_S(s)$ tal que $\gamma \cap \Sigma_{L_S}(s) = \Sigma_A(s)$, e
- para todo $s' \in \overline{A}$ e $\sigma \in \Sigma$ com $s = s'\sigma$, tem-se
  - se $s' \in A$, então existe (γ, M) em $\Gamma_S(s')$ tal que $\gamma \cap \Sigma_{L_S}(s') = \Sigma_A(s')$, ou
  - se $s' \in \overline{A} - A$, então existe (γ, N) em $\Gamma_S(s')$ tal que $\gamma \cap \Sigma_{L_S}(s') = \Sigma_A(s')$.

A partir do conjunto de equações acima e a definição de Ω, equação (4.5), $\Omega(A) = A$, isto é, A é um ponto fixo de Ω.
(**Somente se**) Seja $A \subseteq K$ um ponto fixo de Ω, então $A - \Omega(A) = \emptyset$. Considere as seguintes afirmações:

1. Para todo s em A, existe (γ, M) em $\Gamma_S(s)$ tal que $\gamma \cap \Sigma_{L_S}(s) = \Sigma_A(s)$. Suponha que não exista s' em A tal que $s < s'$, isto implica $\Sigma_A(s) = \emptyset$. Como A é ponto fixo de Ω, existe (γ, M) em $\Gamma_S(s)$ tal que $\gamma \cap \Sigma_{L_S}(s) \subseteq \Sigma_A(s)$, então $\gamma \cap \Sigma_{L_S}(s) = \emptyset$ e a igualdade é válida. Suponha agora que exista s' em A tal que $s < s'$. Por contradição, suponha que, para todo (γ, M) em $\Gamma_S(s)$ tal que $\gamma \cap \Sigma_{L_S}(s) \subseteq \Sigma_A(s)$ tenha-se a condição de *estritamente contido*, isto é, $\gamma \cap \Sigma_{L_S}(s) \subset \Sigma_A(s)$ (nota: como s está em A não se consideram os controles do tipo N). Sob a hipótese anterior, existe σ em $\Sigma_A(s)$ tal que $\sigma \notin \gamma$, para todo (γ, M) acima citado. Como $\sigma \in \Sigma_A(s)$, existe s' em A tal que $s < s\sigma \leq s'$. Assim, para todo (γ, M) em $\Gamma_S(s)$ tal que $\gamma \cap \Sigma_{L_S}(s) \subseteq \Sigma_A(s)$, tem-se $\sigma \notin \gamma \cap \Sigma_{L_S}(s)$, ou seja $\sigma \notin \gamma$. Isso contradiz a hipótese que A seja um ponto fixo de Ω, porque equivale a dizer que existe s' em A para o qual existe $s\sigma \leq s'$, com s e $s\sigma$ em A, e não existindo (γ, M) em $\Gamma_S(s)$ tal que $\gamma \cap \Sigma_{L_S}(s) \subseteq \Sigma_A(s)$ e $\sigma \in \gamma$.

2. Para todo s em $\overline{A} - A$, existe (γ, N) em $\Gamma_S(s)$ tal que $\gamma \cap \Sigma_{L_S}(s) = \Sigma_A(s)$. Seja s em $\overline{A} - A$, pelo fato de que A é um ponto fixo de Ω, existe (γ, N) em $\Gamma_S(s)$ tal que $\gamma \cap \Sigma_{L_S}(s) \subseteq \Sigma_A(s)$. Suponha que para todo (γ, N) acima a igualdade seja estrita, isto quer dizer que $\gamma \cap \Sigma_{L_S}(s) \subset \Sigma_A(s)$. Seja então $\sigma \in \Sigma$ tal que $\sigma \in \Sigma_A(s)$ e, para todo (γ, N) acima $\sigma \notin \gamma$. Este fato contradiz que A seja um ponto fixo de Ω.

Assim, pelas duas afirmações acima, se $A \subseteq K$ é um ponto fixo de Ω, então A é (L_S, Γ_S)-compatível, o que completa a demonstração. □

Proposição 4.3 (Máximo ponto fixo de Ω) *Para o SED com marcação flexível S e a linguagem $K \subseteq L_S$, a máxima linguagem (L_S, Γ_S)-compatível contida em K é o máximo ponto fixo do operador Ω contido em K.*

Demonstração. Uma vez que as linguagens (L_S, Γ_S)-compatíveis contidas em K são os pontos fixos de Ω contidos em K, a máxima linguagem (L_S, Γ_S)-compatível contida em K é o máximo ponto fixo de Ω contido em K. □

Para o SED com marcação flexível S, a linguagem $K \subseteq L_S$ e o operador Ω, definido na equação (4.5), seja $\{K\}$ a seqüência de linguagens gerada pela sucessiva aplicação do operador Ω a K e às linguagens resultantes. Define-se $\{K\}$ formalmente por:

$$\begin{aligned} K_0 &= K \\ K_{i+1} &= \Omega(K_i), \end{aligned} \tag{4.6}$$

onde i é um número natural.

Proposição 4.4 (Convergência da seqüência) *Para o SED com marcação flexível S e a linguagem $K \subseteq L_S$, a seqüencia $\{K\}$, equação (4.6), converge para a máxima linguagem (L_S, Γ_S)-compatível contida em K.*

Demonstração. Para provar que a seqüência $\{K\}$ converge para a máxima linguagem (L_S, Γ_S)-compatível contida em K, prova-se primeiro que a seqüência converge para

um ponto fixo do operador Ω contido em K que, pela proposição 4.2, é uma linguagem (L_S, Γ_S)-compatível contida em K. Em seguida, prova-se que aplicações sucessivas do operador Ω iniciadas por K não "passam por cima"de um ponto fixo do operador Ω contido em K. Assim, demonstra-se que o primeiro ponto fixo de Ω encontrado em aplicações sucessivas de Ω a partir de K é o máximo ponto fixo que, pela proposição 4.3, é a máxima linguagem (L_S, Γ_S)-compatível contida em K.

O operador Ω é *monótono* (Wonham 2002b), isto é, para todo $A \subseteq L_S$, $\Omega(A) \subseteq A$, pois, como visto anteriormente $A - \Omega(A) \supseteq \emptyset$. Assim, a seqüência de linguagens $\{K\}$, geradas por sucessivas aplicações de Ω começando por K, converge para um ponto fixo de Ω contido em K.

Considere as linguagens K_i e K_{i+1} da seqüência $\{K\}$, com i um natural arbitrário, tais que $K_{i+1} = \Omega(K_i)$. Defina-se $L_b(A) = A - \Omega(A)$, para $A \subseteq L_S$. $L_b(A)$ é a linguagem que contém as palavras de A que foram *podadas* por aplicação de Ω. Prova-se então a seguir que se K_i não é um ponto fixo de Ω, então para todo $A \subseteq K$ tal que $K_{i+1} \subset A \subseteq K_i$, A não é um ponto fixo de Ω. Sabe-se que $K_i - K_{i+1} = K_i - \Omega(K_i) = L_b(K_i)$, assim qualquer $A \subseteq K$ tal que $K_{i+1} \subset A \subseteq K_i$ é obtido a partir de K_{i+1} por adição de palavras em $L_b(K_i)$. Entretanto, qualquer linguagem A construída por adição de palavras em $L_b(K_i)$ a K_{i+1} falha em ser um ponto fixo de Ω, pois as palavras inseridas violam as regras que tornariam $A - \Omega(A)$ vazio. Assim, nenhum $A \subseteq K$ tal que $K_{i+1} \subset A \subseteq K_i$ é um ponto fixo de Ω.

Conclui-se que a seqüência $\{K\}$ converge para a máxima linguagem (L_S, Γ_S)-compatível contida em K. □

Proposição 4.5 (Número finito de passos) *Para o SED com marcação flexível S e a linguagem $K \subseteq L_S$, se L_S e K forem regulares e Σ for finito, então a seqüência $\{K\}$, equação (4.6), converge em um número finito de passos.*

Demonstração. Três relações de equivalência são definidas para a demonstração desta proposição. Sejam s e s' palavras em L_S, defina-se a relação de equivalência R_{L_S} por $s \equiv s' \pmod{R_{L_S}} \iff L_S/s = L_S/s'$, onde $L_S/s = \{u \in \Sigma^* : su \in L_S\}$ é a linguagem

em L_S após s. As classes de equivalência de R_{L_S} são compostas pelas palavras cujas extensões em L_S são as mesmas. Defina-se também a relação de equivalência R_{Γ_S} por $s \equiv s' \pmod{R_{\Gamma_S}} \iff \Gamma_S(s) = \Gamma_S(s')$. As classes de equivalência de R_{Γ_S} são compostas pelas palavras que possuem o mesmo conjuntos de controles. Por fim, defina-se a relação de equivalência R_K por $s \equiv s' \pmod{R_K} \iff K/s = K/s'$. De forma similar a R_{L_S}, as classes de equivalência de R_K são compostas pelas palavras cujas extensões em K são as mesmas.

Para palavras em L_S, seja R a relação de equivalência definida pela intersecção das classes de equivalência de R_{L_S}, R_{Γ_S} e R_K. Para a palavra s em L_S, defina-se $[s]$ como sendo a classe de equivalência de s em relação a R. Pela definição de R, para todo s e s' em L_S, se $s' \in [s]$ o seguinte é verdade:

- $\Sigma_{L_S}(s') = \Sigma_{L_S}(s)$,
- $\Gamma_S(s') = \Gamma_S(s)$ e
- $\Sigma_K(s) = \Sigma_K(s')$.

Mais ainda, se Σ for finito e L_S e K forem regulares, a relação de equivalência R é de posto finito, isto é, possui um número finito de classes de equivalência (Hopcroft e Ullmann 1979).

Defina-se o autômato $C = (\Sigma, Z, \eta, z_0, Z_m)$ por:

- $Z = \{[s] : s \in \overline{K}\}$,
- para $s \in \overline{K}$ e $\sigma \in \Sigma$,

$$\eta([s], \sigma) = \begin{cases} [s\sigma] & \text{se } s\sigma \in \overline{K} \\ \text{não definido} & \text{caso contrário,} \end{cases} \tag{4.7}$$

- $z_0 = [\epsilon]$, e
- $Z_m = \{[s] : s \in K\}$.

Os estados do autômato C, elementos do conjunto Z, correspondem às classes de equivalência das palavras em $\overline{K}$ em relação à R. Por sua vez, os estados marcados do autômato C, elementos de Z_m, correspondem às classes de equivalência das palavras em K em relação a R. Por outro lado, os estados não marcados do autômato C, elementos de $Z - Z_m$, correspondem às palavras em $\overline{K} - K$ em relação a R. Assim, conclui-se que $L_C = \overline{K}$ e $L_{C,m} = K$. Ainda, se K e L_S forem regulares e Σ for finito, C é de estados finitos.

Relaciona-se a aplicação do operador Ω à linguagem K a operações sobre o autômato C. Seja s em K e considere as três condições para se eliminar s, vide equação (4.5):

1. Se s for eliminado por violação da condição (i), então todo $s' \in [s]$ também é eliminado. Isso corresponde a se desmarcar o estado $[s]$ de C, isto é, remover o estado do conjunto Z_m.

2. Se s for eliminado por violação da condição (ii), então existe $s'\sigma \leq s$ que viola à alguma das condições impostas em (ii), com $\sigma \in \Sigma$ e s' e $s'\sigma$ em $\overline{K}$. Assim, todo $s'' \in [s']$ é tal que $s''\sigma \leq s'''$ com $s''' \in [s]$, com s''' também violando a alguma das condições impostas por (ii). Mais ainda, isso também implica que toda palavra $s'' \in K$ tal que $s''\sigma \leq s'''$ também deve ser eliminada após a aplicação de Ω. Isso corresponde a se eliminar a transição $([s], \sigma, [s\sigma])$ no autômato C.

3. Num raciocínio similar, a eliminação da palavra que violar a condição (iii) também resulta na eliminação de uma transição de C.

Deduz-se assim que a aplicação de Ω a K é equivalente a se aplicar um conjunto de operações de desmarcação de estados e eliminação de transições ao autômato C. A tomada do componente *trim* do autômato resultante leva a um sub-autômato de C cuja linguagem marcada é $\Omega(K)$. Sendo K e L_S regulares e Σ finito, correspondendo a C de estados finitos, a aplicação sucessiva deste procedimento converge necessariamente em um número finito de passos, uma vez que o procedimento apenas elimina estados e transições de C. □

Da proposição 4.4, sabe-se que a seqüência $\{K\}$, equação (4.6), converge à máxima linguagem (L_S, Γ_S)-compatível contida em K. Já da proposição 4.5, sabe-se que, com a condição adicional de L_S e K regulares e Σ finito, essa convergência se dá em um número finito de aplicações do operador Ω.

No que segue, apresenta-se um possível algoritmo que implementa a geração da seqüência de linguagens $\{K\}$. Seja o SED com marcação flexível S sobre o alfabeto Σ representado pelo par (G, Γ), onde G é um autômato de estados finitos e Γ é uma estrutura de controle dependente do estado, vide seção 4.2. Considere a linguagem $K \subseteq L_S$ marcada pelo autômato de estados finitos B, isto é, $L_B = \overline{K}$ e $L_{B,m} = K$. O algoritmo 4.1 calcula a máxima linguagem (L_S, Γ_S)-compatível contida em K baseado na geração da seqüência de linguagens $\{K\}$. No algoritmo 4.1, para os autômatos E e F, $E \| F$ denota a composição síncrona dos autômatos E e F, $trim(E)$ é a componente *trim* do autômato E, e para o estado x do autômato E, $\Sigma_E(x)$ denota o conjunto ativo de eventos em E no estado x (vide capítulo 2).

No passo 1 do algoritmo 4.1, o autômato C_0 é construído pela obtenção do componente *trim* da composição síncrona dos autômatos G e B. Assim cada estado z de C_0 corresponde a um único par (x, y), onde x é um estado de G e y é um estado de B. Isso também vale para o autômato C_i, com $i \in \{1, \ldots, n\}$, uma vez que C_i é um subautômato de C_0, isto é, um autômato construído pela remoção de estados e transições de C_0. No passo 2 do algoritmo 4.1, para cada estado $z = (x, y)$ de C_i faz-se uma busca pelo máximo controle $(\gamma, \#) \in \Gamma'(x)$ atende às condições $\gamma \cap \Sigma_G(x) \subseteq \Sigma_{C_i}(z)$, $\# = M$ se z for marcado, e $\# = N$ se z for não marcado. Se pelo menos um controle de $\Gamma'(x)$ atende às condições anteriores, sabe-se que, pelo fechamento em relação à união dos controles de um SED com marcação flexível, definição 4.2, existe um máximo controle em $\Gamma'(x)$ que também atende às condições. Essa é a razão de se buscar por um controle máximo no passo 2. Dependendo do resultado da busca no passo 2, proceder com a remoção de estados do conjunto de estados marcados e de transições de C_i, seguidas da obtenção do componente *trim* do resultado. Copia-se então o autômato obtido para C_{i+1}. No passo 3, verifica-se se nenhum estado foi desmarcado ou removido e nenhuma

entradas:

- $G = (\Sigma, X, \delta, x_0, X_m)$ e Γ' autômato e estrutura de controle que representam o sistema S.
- $B = (\Sigma, Y, \psi, y_0, Y_m)$ autômato que marca a especificação K.

passo 1:

- Marcar todos os estados de G.
- Fazer $C_0 = (\Sigma, Z_0, \eta_0, z_{0,0}, Z_{m,0}) \leftarrow trim(B \| G)$.
- Fazer $i \leftarrow 0$.

passo 2:

- Para $z = (x, y) \in Z_{i,m}$ encontre o máximo $(\gamma, \#)$ em $\Gamma'(x)$ tal que $\gamma \cap \Sigma_G(x) \subseteq \Sigma_{C_i}(z)$ e $\# \leq M$.
  - Se tal $(\gamma, \#)$ não existe, remova z de $Z_{m,i}$ e remova todas as transições que partem de z.
  - Se tal $(\gamma, \#)$ existe:
    - Se $\# = N$, remova z de $Z_{m,i}$.
    - Remova todas as transições que partem de z com eventos em $\Sigma_{C_i}(z) - (\gamma \cap \Sigma_G(x))$.
- Para $z = (x, y) \in Z_i - Z_{i,m}$ encontre o máximo $(\gamma, \#)$ em $\Gamma'(x)$ tal que $\gamma \cap \Sigma_G(x) \subseteq \Sigma_{C_i}(z)$ e $\# = N$.
  - Se tal (γ, N) não existe, remova todas as transições que partem de z.
  - Se tal (γ, N) existe, remova todas as transições que partem de z com eventos em $\Sigma_{C_i}(z) - (\gamma \cap \Sigma_G(x))$.
- Obtenha a componente *trim* do autômato resultante e copie para C_{i+1}.

passo 3:

- Se $C_{i+1} = C_i$, parar e fazer $n \leftarrow i$, senão fazer $i \leftarrow i + 1$ e volte ao passo 2.

saída:

- Autômato C_n tal que $L_{C_n,m} = \sup C_{(L_S,\Gamma_S)}(K)$.

Algoritmo 4.1: Calcula a máxima linguagem compatível.

transição foi removida no passo 2, isto é, $C_{i+1} = C_i$. Se for esse o caso, o algoritmo termina suas iterações, caso contrário, retorna-se ao passo 2 para se processar C_{i+1}.

Considere que, no algoritmo 4.1, G possui n estados, B possui m estados, e Σ possui e eventos. O algoritmo manipula autômatos obtidos pelo corte de estados e transições do autômato $G\|B$, que possui, no máximo, nm estados. No pior caso, a cada passo do algoritmo, examina-se todo o conjunto de estados do autômato e elimina-se apenas um estado. Isso resulta nm passos ao todo para o algoritmo, sendo que a cada passo analisam-se, no máximo, nm estados. Para cada passo do algoritmo e para cada estado tratado no passo, faz-se uma busca por um controle que seja o máximo de um conjunto de controles. Esta busca é uma operação de até $\log_2 p$ passos para uma lista ordenada com p elementos (Hopcroft e Ullmann 1979). No caso do conjunto de controles tratado no algoritmo 4.1, limita-se p a 2^{e+1}, correspondendo ao maior conjunto de controles possível para o sistema, $2^{\Sigma} \times \{M, N\}$. Assim, estima-se a complexidade o algoritmo 4.1 na ordem de $\mathcal{O}((e+1)m^2n^2)$.

4.4 Exemplo de aplicação

Esta seção apresenta um exemplo de aplicação para SEDs com marcação flexível.

Considere o exemplo do gato e do rato num labirinto, exemplo 4.7. A tarefa é projetar um supervisor para manter tanto o gato quanto o rato vivos no labirinto de salas. Primeiro nunca deve ser permitido que os dois animais ocupem a mesma sala ao mesmo tempo, quando supõe-se que o gato come o rato. Por outro lado, deve ser sempre permitido o acesso dos animais às suas comidas, caso contrário, os animais morrem de fome. A lei de controle deve ser ótima, a significar que deve ser o menos restritiva o possível ao movimento dos animais.

O SED com marcação flexível que representa o comportamento conjunto do gato e do rato no labirinto é representado por um par (G, Γ), onde G é um autômato e Γ é uma estrutura de controle dependente do estado, obtidos a seguir. O autômato G é obtido pela composição síncrona dos autômatos para o gato e o rato, representados

na figura 4.4. O autômato G possui 16 estados e 48 transições, e não é apresentado. Dado um estado (i, j) de G, onde a numeração (i, j) indica as salas ocupadas pelo gato e pelo rato, respectivamente, constrói-se o conjunto de controles $\Gamma(i, j)$ como a seguir. Um controle $(\gamma, \#)$ está em $\Gamma(i, j)$ se, e somente se, γ contém um possível conjunto de eventos permitidos de ocorrer após o estado (i, j), $\# = M$ quando γ contém um evento que corresponde à permissão do acesso à comida, e, caso contrário, $\# = N$.[2] Por exemplo, para o estado $(1, 3)$ de G o controle correspondente é:

$$\begin{aligned}\Gamma(1,3) \;=\; & \{(\emptyset, N), (\{r_1\}, N), (\{r_1, r_2\}, M), (\{g_2\}, M), (\{g_2, r_1\}, M), \\ & (\{g_2, r_1, r_2\}, M), (\{g_2, g_3\}, M), (\{g_2, g_3, r_1\}, M), (\{g_2, g_3, r_1, r_2\}, M)\}.\end{aligned}$$

A especificação de coordenação $K \subseteq L_G$ é obtida a partir do autômato G, eliminando-se os estados que correspondem ao gato e o rato ocuparem a mesma sala, marcando-se os estados onde seja possível dar acesso à comida, e obtendo-se o componente *trim* do autômato resultante. Por aplicação do algoritmo 4.1, determina-se a máxima linguagem (G, Γ)-compatível contida em K, marcada pelo autômato representado na figura 4.9. Observe que, a marcação dos estados $(1, 3)$ e $(4, 3)$ na figura 4.9, cuja numeração representa a sala ocupada pelos animais como em G, correspondem aos controles $(\{g_2\}, M)$ e $(\{r_1, r_2, g_1\}, M)$, a significar que se permite o acesso à comida para o gato e o rato, respectivamente. Para maior entendimento da política de controle adotada, também são representados em cinza claro na figura 4.9 os estados que correspondem a situações evitadas para o movimento do gato e do rato dentro do labirinto.

O método de síntese de supervisores para SEDs com marcação flexível permite a solução do problema de controle supervisório preservando-se o nível de abstração natural de descrição do sistema. Se a abordagem RW for empregada para se resolver o mesmo problema, o comportamento nos dutos deve ser detalhado, como mostra o

[2] O SED com marcação flexível que representa o comportamento conjunto do gato e do rato no labirinto pode ser obtido alternativamente por aplicação do produto síncrono (vide seção 6.2) aos SEDs que representam os comportamentos isolados do gato e do rato, representados na figura 4.4.

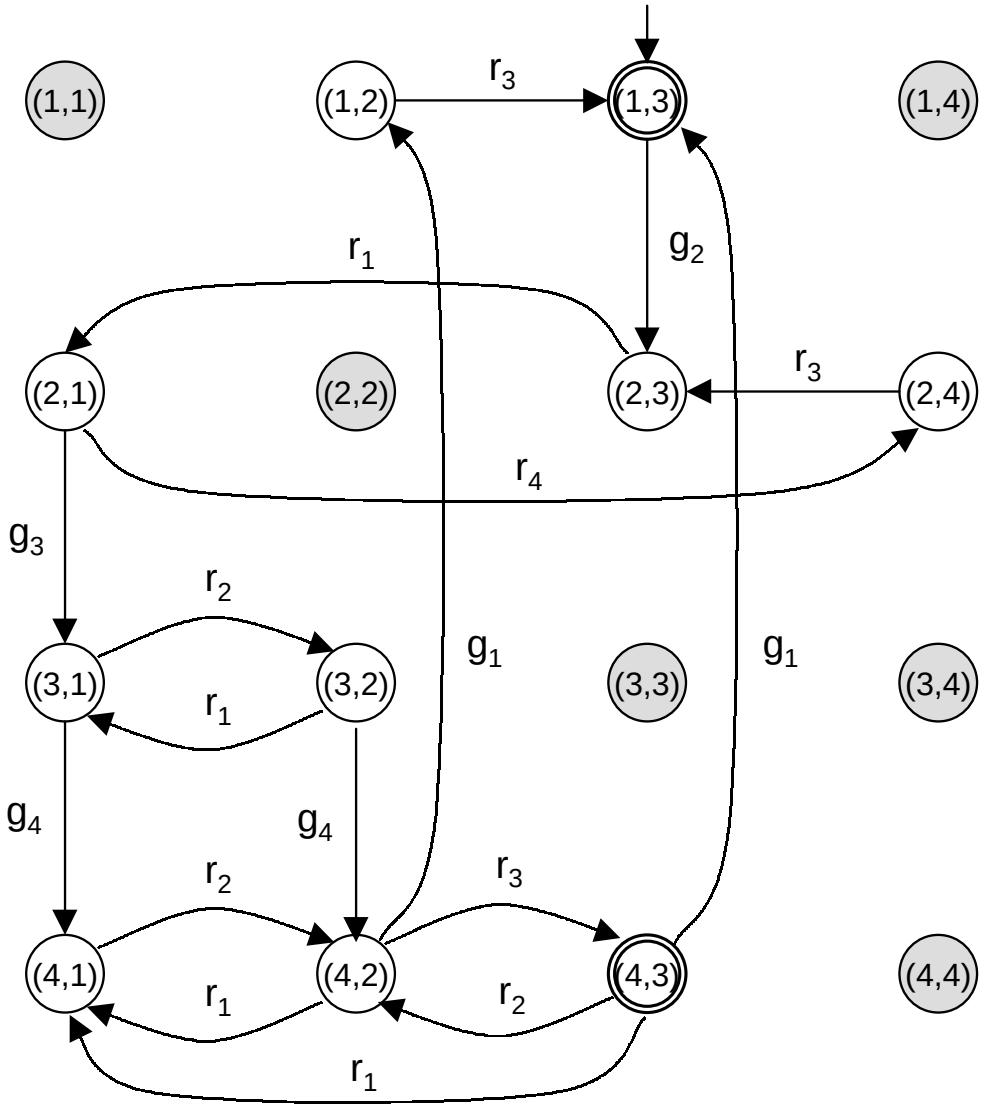


Figura 4.9: Máxima linguagem (G, Γ)-compatível para o gato e o rato.

exemplo 4.7. O autômato que representa o comportamento conjunto do gato e do rato no labirinto possui 30 estados e 81 transições na abordagem RW. Aplicando-se o método de síntese de supervisores da abordagem RW, um supervisor com 20 estados e 38 transições é obtido. Prova-se que a política de controle dos supervisores calculados na abordagem RW e na abordagem apresentada neste trabalho, figura 4.9, são equivalentes, em termos de ocupação das salas comuns do labirinto e de acesso à comida. Assim, os supervisores calculados para o mesmo sistema são mais complexos na abordagem RW, em termos de número de estados, que na abordagem proposta neste trabalho. Por outro lado, se os modelos generalizados de Wong e Wonham (1996a), Li et al. (1998) ou Pu (2000) fossem aplicados para resolver o mesmo problema, os supervisores ainda possuiriam maior número de estados que na abordagem deste trabalho, em termos de número de estados ou transições. Isso se dá porque tais abordagens não tratam da flexibilidade da marcação que ocorre na descrição do sistema como se trata na abordagem deste trabalho.

4.5 Conclusões

Este capítulo apresenta um tipo de SED dotado de controle, chamado SED com marcação flexível, onde o principal destaque é uma estrutura de controle que define um conjunto de controles para cada palavra gerada pelo sistema, cada controle, por sua vez, possui dois atributos: um para a ação de se permitir a ocorrência de eventos, e outro para a ação de marcação da palavra.

O exemplo do gato e do rato apresentado no exemplo 4.7 e na seção 4.4 sugere uma aplicação natural do modelo proposto: sistemas descritos em um alto nível de abstração onde a informação relativa à marcação está encapsulada. No exemplo, o sistema do gato e do rato foi modelado tanto na abordagem proposta quanto na abordagem RW, sendo que esta abordagem necessitou entrar nos detalhes da rede de dutos subjacente ao labirinto. A existência destes dois níveis de abstração para o mesmo problema remete ao controle hierárquico de SEDs, onde um gerente cuida de diretivas de controle gerenciais,

de amplo horizonte temporal, abstratas e estratégicas, enquanto um operador cuida de ações de controle operacionais, de curto horizonte temporal, detalhadas ou táticas. Como será visto no capítulo 5, o SED com marcação flexível aplica-se naturalmente ao controle hierárquico de SEDs, trazendo uma solução simples, elegante e direta ao problema de controle hierárquico. De forma diferente do visto no capítulo 3, o uso do SED com marcação flexível permite uma solução totalmente independente das diversas condições impostas por abordagens similares, vistas no capítulo 3.

Comparativamente à abordagem RW, a abordagem por SEDs com marcação flexível necessita formas de representação adicionais para a estrutura de controle. Assim, tem-se o potencial problema de se adicionar complexidade ao modelo por causa da estrutura de controle do sistema. Isso pode ser contornado explorando-se formas alternativas e compactas de representação da estrutura de controle, como, por exemplo, codificar o controle num número inteiro. A forma de implementação do SED com marcação flexível na ferramenta para controle hierárquico de SEDs é descrita na seção A.4.

Alguns desenvolvimentos e extensões são considerados para os SEDs com marcação flexível. O primeiro é a consideração da composição de tais sistemas, explorada no capítulo 6. O segundo é o controle modular de tais sistemas, aplicado então ao controle hierárquico (Torrico e Cury 2002a). E o terceiro é inspirado no problema do gato e do rato apresentado. Neste problema, tratou-se de um sistema composto, onde se deveria garantir que as tarefas dos sistemas componentes sejam completadas independentemente. Isso é diferente da composição na abordagem RW onde se define uma tarefa completa do sistema composto apenas quando todos os sistemas componentes completam suas tarefas (Ramadge e Wonham 1989). Assim, propõe-se estudar a definição da marcação de um sistema composto discriminando-se as tarefas para os sistemas componentes (de Queiroz 2002).

Capítulo 5

A construção de hierarquias de controle consistentes para SEDs

Neste capítulo, propõe-se um esquema de supervisão hierárquica para sistemas a eventos discretos onde os sistemas são descritos pelo SED com marcação flexível apresentado no capítulo 4.[1]

A organização deste capítulo é a seguinte. Na seção 5.1 faz-se uma revisão do problema de controle hierárquico dentro da perspectiva deste trabalho. Nas seções 5.2, 5.3 e 5.4 faz-se a análise de uma proposta de um esquema de supervisão hierárquica de dois níveis, a saber: na seção 5.2 introduz-se uma forma de decomposição do sistema do operador, na seção 5.3 propõe-se o sistema do gerente, e, na seção 5.4, apresentam-se os resultados principais relacionados à consistência hierárquica. Com base no esquema de supervisão hierárquica proposto nas seções anteriores, na seção 5.5 propõe-se um método para supervisão hierárquica, e, na seção 5.6, faz-se uma análise da complexidade do método proposto. Na seção 5.7 apresenta-se um exemplo ilustrativo baseado no problema de controle discreto de célula de manufatura real. Por fim, a seção 5.8 contém considerações finais sobre a abordagem.

[1] O material deste capítulo corresponde ao trabalho de da Cunha e Cury (2002a), tendo sido apresentado, em forma preliminar, por da Cunha e Cury (2002b).

5.1 Revendo a formulação do problema de controle hierárquico

Nesta seção tratam-se de alguns aspectos da visão deste trabalho para o problema de controle hierárquico, apresentado na seção 3.1.

Considera-se, como no capítulo 3, um esquema de supervisão hierárquica que adota a metáfora de uma fábrica para descrição dos níveis, sendo o nível inferior associado ao nível de um operador, e o nível superior associado ao nível de um gerente. Na figura 5.1, S é um SED com marcação flexível sobre o alfabeto Σ, a representar a visão de um dado sistema para operador. O SED com marcação flexível P sobre o alfabeto T representa o mesmo sistema do ponto de vista de um gerente. O funcionamento do esquema de supervisão hierárquico é o mesmo que o descrito no capítulo 3. O SED P é, na verdade, a imagem de S através do canal de informação Inf_{og}. O supervisor do gerente f_{ge} é um supervisor que observa eventos gerados por P pelo canal de informação Inf_{ge} e virtualmente aplica um controle no canal de controle Con_{ge}. Na verdade, o controle de f_{ge} é transmitido como uma diretiva, através do canal de comando Com_{ge} para um supervisor do operador f_{op}. Este, por sua vez, observando os eventos gerados por S pelo canal de informação Inf_{op} e considerando as diretivas de f_{ge}, aplica controle em S através do canal de controle Con_{op}. Deseja-se que a malha fechada por Com_{go}, Inf_{op} e Con_{op}, e Inf_{og} seja equivalente ao controle virtual aplicado pelo supervisor gerencial em Con_{ge}. Como foi visto na seção 3.1, este problema é mapeado nas condições de consistência hierárquica e consistência hierárquica forte.

Uma hipótese que se faz neste trabalho é considerar que o alfabeto do gerente como sendo um subconjunto de eventos relevantes do alfabeto operador, isto é, $T = \Sigma_r$, com $\Sigma_r \subseteq \Sigma$. Prova-se, informalmente, que que esta opção de modelagem possui o mesmo poder de expressão que a formulação original com a argumentação a seguir. Considere o SED do operador S sobre o alfabeto Σ e o alfabeto T, cujos eventos são distintos dos eventos em Σ, e cada evento de T é definido pela ocorrência de uma palavra em S. É sempre possível modificar S, Σ e T por renomeação de eventos

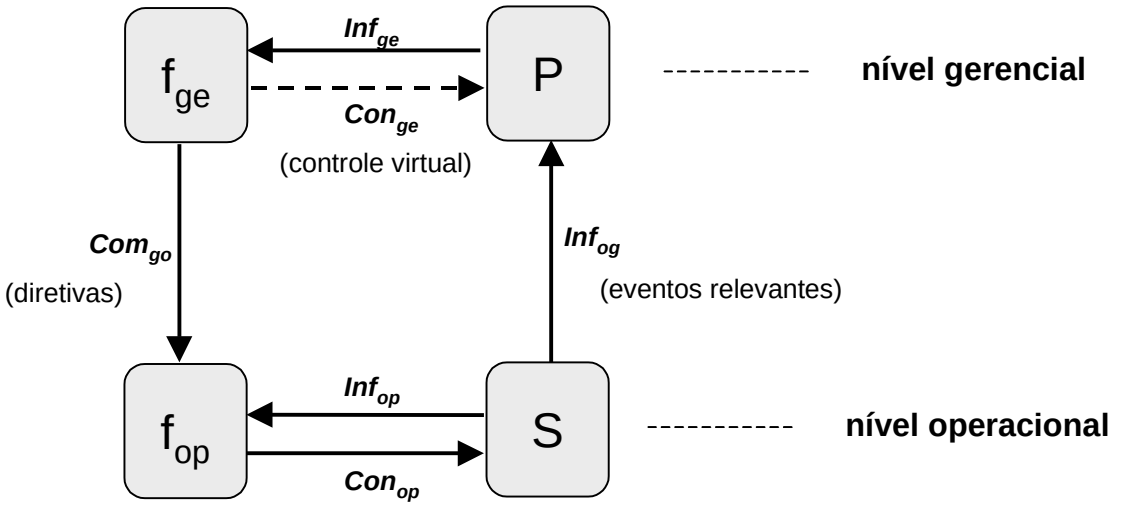


Figura 5.1: Esquema de supervisão hierárquica de dois níveis.

para se obter um novo SED para o operador S', com alfabeto Σ', e um alfabeto de eventos relevantes $\Sigma'_r \subseteq \Sigma'$ tais que todas as ocorrências de eventos em T observadas a partir de ocorrências de palavras em S sejam mapeadas em ocorrências em S' de eventos em Σ'_r. A hipótese acima é motivada pela observação de que a maioria dos exemplos encontrados na literatura, o alfabeto do gerente é um subconjunto do alfabeto do operador (vide os exemplos apresentados por Zhong e Wonham (1990), Wong e Wonham (1996a), Wong e Wonham (1996b), Wong e Wonham (1998), Wong et al. (1998), Guan (1997) Park e Lim (2001), Pu (2000), Wong et al. (2000), e Hubbard e Caines (2002), entre outros).

Uma conseqüência interessante da escolha acima é a simplificação do formalismo. Não é necessário usar autômatos de Moore para representar o sistema do operador mais o canal de informação. O mapa repórter passa a fazer a mesma ação que uma projeção no alfabeto do gerente, sendo que o domínio do mapa repórter permanece sendo a linguagem de S. Entretanto, parte da nomenclatura original para tratamento do problema, onde se faz referência às saídas do autômato de Moore, tais como mapa vocal, palavra vocal ou vocalização são mantidas por coerência aos trabalhos anteriores.

Nas seções que seguem, apresenta-se um modelo para este esquema de supervisão hierárquica de sorte a se obter consistência hierárquica entre S e P.

5.2 Decomposição do sistema do operador

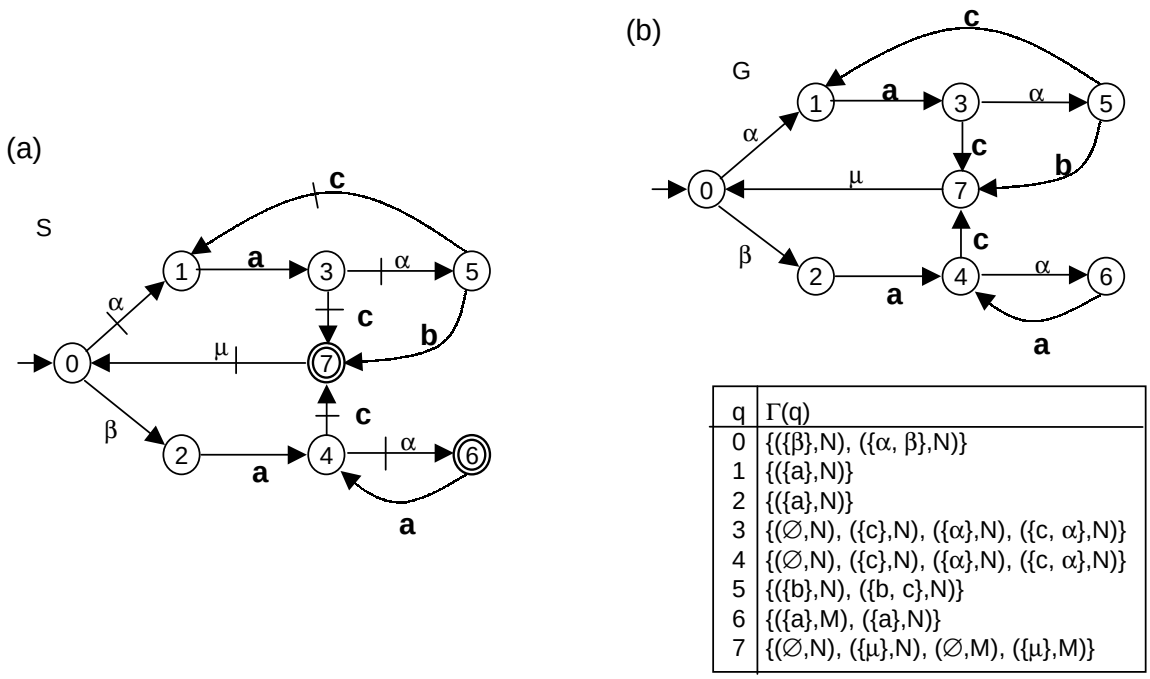
Esta seção apresenta uma decomposição do sistema do operador, para o esquema de supervisão hierárquica de dois níveis apresentado.

Para o esquema de supervisão hierárquico de dois níveis proposto na seção 5.1, onde S é um SED com marcação flexível a representar o sistema do operador, sobre o alfabeto Σ, e $\Sigma_r \subseteq \Sigma$ é o conjunto de eventos relevantes para o gerente, faz-se uma revisão de alguma notação necessária referente ao capítulo 3. Seja $\theta : L_S \to \Sigma_r^*$ o mapa repórter associado a L_S e Σ_r, definição 3.1, cuja ação é apagar eventos $\Sigma - \Sigma_r$ de palavras geradas por S. Seja $w : L_S \to \Sigma_r \cup \{\tau_0\}$ o mapa vocal associado a L_S e Σ_r, equação (3.5), com τ_0 sendo o evento silencioso. Enquanto o mapa repórter relaciona palavras s em L_S a palavras $\theta(s)$ em Σ_r^*, o mapa vocal associa palavras s em L_S ao seu último evento, se este for um evento relevante, ou o evento silencioso τ_0, caso contrário. No contexto deste capítulo, as palavras vocais de S são, além da palavra vazia ϵ, as palavras geradas por S que terminem por eventos relevantes.

Considere a representação de estados do SED com marcação flexível S pelo par (G, Γ), vide seção 4.1, onde G é um autômato e Γ é uma estrutura de controle dependente do estado. Definem-se os estados vocais de S como sendo os estados do autômato G que são finais de transições envolvendo eventos relevantes, mais o estado inicial de G. A cada palavra vocal de S corresponde um estado vocal de S, enquanto a um estado vocal de S corresponde possivelmente um conjunto de palavras vocais de S.

Exemplo 5.1 (Ilustração de SED com marcação flexível para o operador) *Considere uma ilustração de SED com marcação flexível S na figura 5.2. O SED S é dotado de estrutura de controle e marcação RW padrão, indicada na figura 5.2(a), e considere o caso do controle supervisório com marcação (Wonham 2002b). A tradução de S para um SED com marcação flexível é representada na figura 5.2(b), onde G*

é o autômato e Γ *é a estrutura de controle (vide exemplo 4.6). Considere-se como conjunto de eventos relevantes* $\Sigma_r = \{a, b, c\}$*, o conjunto de estados vocais de* S *é* $\{0, 1, 3, 4, 7\}$.



q	Γ(q)
0	{({β},N), ({α, β},N)}
1	{({a},N)}
2	{({a},N)}
3	{(∅,N), ({c},N), ({α},N), ({c, α},N)}
4	{(∅,N), ({c},N), ({α},N), ({c, α},N)}
5	{({b},N), ({b, c},N)}
6	{({a},M), ({a},N)}
7	{(∅,N), ({μ},N), (∅,M), ({μ},M)}

Figura 5.2: Exemplo ilustrativo de SED com marcação flexível para o operador.

A escolha do exemplo acima é motivada pelo fato deste ser um exemplo em que são violadas todas as condições impostas pelas as abordagens anteriores para o controle hierárquico. Em (Wong e Wonham 1996a), duas condições impostas para a consistência hierárquica são que o mapa repórter seja um observador e que haja consistência de marcação (vide capítulo 3). Pode-se verificar que o mapa repórter associado para S e Σ_r do exemplo 5.1 não é um observador, pois por exemplo, os estados equivalentes 3 e 4 possuem conjuntos ativos de eventos relevantes em S distintos, a saber $\{b, c\}$ e $\{a, b\}$, respectivamente. Também pode-se verificar que, para o mapa repórter no exemplo 5.1, nenhum grupo de estados θ-equivalentes possui somente estados marcados, de

forma que, para não violar a consistência de marcação, nenhum estado do sistema do gerente poderia ser marcado. Já em (Pu 2000) a condição imposta para a consistência hierárquica é que o mapa repórter seja um observador fraco. Pode-se verificar que o sistema do exemplo 5.1 também viola a condição de observador fraco, e como ilustração, pode-se usar o mesmo caso dos estados 3 e 4. Como será visto mais adiante neste capítulo, a utilização do método que se propõe neste trabalho permite obter o sistema do gerente com consistência hierárquica sem consideração de nenhuma das condições tratadas nas outras abordagens para controle hierárquico de SEDs.

Uma forma de decomposição do sistema do operador S é proposta abaixo.

Definição 5.1 (Subsistema) *Para a palavra s em L_S, o subsistema de s em S é um SED com marcação flexível $S(s)$ sobre o alfabeto Σ tal que*

$$L_{S(s)} = \{u \in \Sigma^* : (su \in L_S) \ e \ (((\forall u' \in \Sigma^+) \ tal \ que \ u' < u) \ w(su') = \tau_0)\} \quad (5.1)$$

e, para $u \in L_{S(s)}$,

$$\Gamma_{S(s)}(u) = \begin{cases} \Gamma_S(su) & se \ w(su) = \tau_0 \ ou \ u = \epsilon \\ \{(\emptyset, M)\} & caso \ contrário. \end{cases} \quad (5.2)$$

O subsistema $S(s)$ é uma cópia de uma porção do sistema S, onde a palavra inicial corresponde a s e todos os possíveis comportamentos de S após s são copiados, até a ocorrência de um novo evento relevante. Uma tarefa para o subsistema corresponde a uma tarefa original do sistema ou à ocorrência de um evento relevante.

Considere a representação de estados do SED com marcação flexível S pelo par (G, Γ), onde G é um autômato e Γ é uma estrutura de controle dependente do estado, vide seção 4.1. Sejam a palavra s em L_S e q o estado de G correspondente a s, o SED com marcação flexível $S(q)$, representado pelo par (G_q, Γ_q), resultante do algoritmo 5.1, é uma representação de estados do subsistema $S(s)$. No algoritmo 5.1, constrói-se o autômato G_q por uma busca recursiva nos estados de G, começando de q, e copiando estados e transições de G para G_q, parando-se quando for encontrado um estado visitado

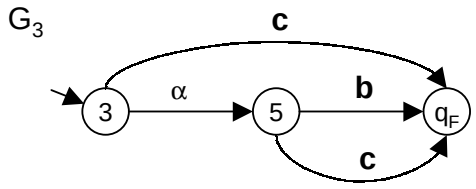
ou um evento relevante. No caso de se encontrar um evento relevante, uma transição para o estado fictício q_F é criada.

entradas
Autômato $G = (\Sigma, Q, \delta, q_0, \emptyset)$ e
Estrutura de controle $\Gamma : Q \rightarrow 2^{2^{\Sigma} \times \{M,N\}}$
Conjunto de eventos relevantes $\Sigma_r \subseteq \Sigma$
Estado $q \in Q$
início
$Q_q \leftarrow \{q, q_F\}$
$Q' \leftarrow \{q\}$
$\Gamma_q(q) \leftarrow \Gamma(q)$
$\Gamma_q(q_F) \leftarrow \{(\emptyset, M)\}$
enquanto $Q' \neq \emptyset$ **faça**
  selecione e retire q' de Q'.
  para todo $q'' \in Q$ e $\sigma \in \Sigma$ tal que $\delta(q', \sigma) = q''$ **faça**
    se $\sigma \notin \Sigma_r$ **então**
      se $q'' \notin Q_q$ **então**
        $Q_q \leftarrow Q_q \cup \{q''\}$
        $Q' \leftarrow Q' \cup \{q''\}$
        $\delta_q(q', \sigma) \leftarrow q''$
        $\Gamma_q(q'') \leftarrow \Gamma(q'')$
      senão
        $\delta_q(q', \sigma) \leftarrow q''$
      fim se
    senão
      $\delta_q(q', \sigma) \leftarrow q_F$
    fim se
  fim para
fim enquanto
fim
saídas
Autômato $G_q = (\Sigma, Q_q, \delta_q, q, \emptyset)$ e
Estrutura de controle $\Gamma_q : Q_q \rightarrow 2^{2^{\Sigma} \times \{M,N\}}$

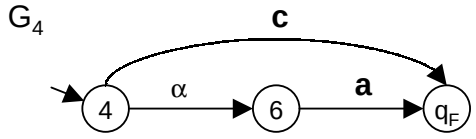
Algoritmo 5.1: Calcula o subsistema de um estado

Exemplo 5.2 (Construção de subsistemas) *Para o sistema do exemplo 5.1, os subsistemas para os estados 3 e 4 são mostrados na figura 5.3.*

Para a próxima definição, recordam-se algumas notações do capítulo 3. Para a palavra s em L_S, $L_{S,voc}(s)$ denota o conjunto de palavras não vazias de $L_{S(s)}$ que,



q	Γ_3 (q)
3	{(∅,N), ({c},N), ({α},N), ({c, α},N)}
5	{({b},N), ({b, c},N)}
q_F	{(∅,M)}



q	Γ_4 (q)
4	{(∅,N), ({c},N), ({α},N), ({c, α},N)}
6	{({a},M), ({a},N)}
q_F	{(∅,M)}

Figura 5.3: Alguns subsistemas para o exemplo 5.1.

quando concatenadas com s formam uma palavra vocal de S, equação (3.12); e $\Sigma_{S,voc}(s)$ é o conjunto ativo de eventos gerenciais em S após s, aqui referenciado conjunto ativo de eventos relevantes em S após s, equação (3.10). Dadas estas definições, associa-se um conjunto de controles envolvendo eventos relevantes à palavra s em L_S a seguir.

Definição 5.2 (Conjunto de controles vocais) *Para a palavra s em L_S, o conjunto de controles vocais de s é um conjunto $\Gamma_{S,voc}(s) \subseteq 2^{\Sigma_{S,voc}(s)} \times \{M, N\}$ tal que $(\gamma, \#) \in \Gamma_{S,voc}(s)$ se, e somente se, existir uma linguagem não vazia $K \subseteq L_{S(s)}$ tal que:*

1. *K seja $(L_{S(s)}, \Gamma_{S(s)})$-compatível,*
2. *para todo $u \in L_{S,voc}(s)$, $w(su) \in \gamma$ se e somente se $u \in K$,*
3. *se $\# = N$, $K - L_{S,voc}(s) = \emptyset$, e*
4. *se $\# = M$, $K - L_{S,voc}(s) \neq \emptyset$.*

Um controle vocal válido $(\gamma, \#)$ para a palavra s em L_S corresponde a uma linguagem K não-vazia e $(L_{S(s)}, \Gamma_{S(s)})$-compatível tal que:

- toda palavra u em $L_{S,voc}(s)$ está em K se e somente se terminar por evento em γ,

- se o indicador # for N, então K só contém palavras que terminem em eventos relevantes, e
- se o indicador # for M, então K contém uma palavra que não termina em um evento relevante.

Restringe-se o conjunto de controles vocais ao conjunto ativo de eventos relevantes em S após s, $\Sigma_{S,voc}(s)$, ao se fazer $\Gamma_{S,voc}(s) \subseteq 2^{\Sigma_{S,voc}(s)} \times \{M, N\}$.

Exemplo 5.3 (Controles vocais) *A figura 5.4 mostra alguns controles vocais válidos para os estados 3 e 4 do sistema do exemplo 5.1, e as respectivas linguagens. Observa-se que o controle* $(\{c\}, N)$ *não é válido para o estado 3 porque a linguagem correspondente viola a regra 2 da definição 5.2.*

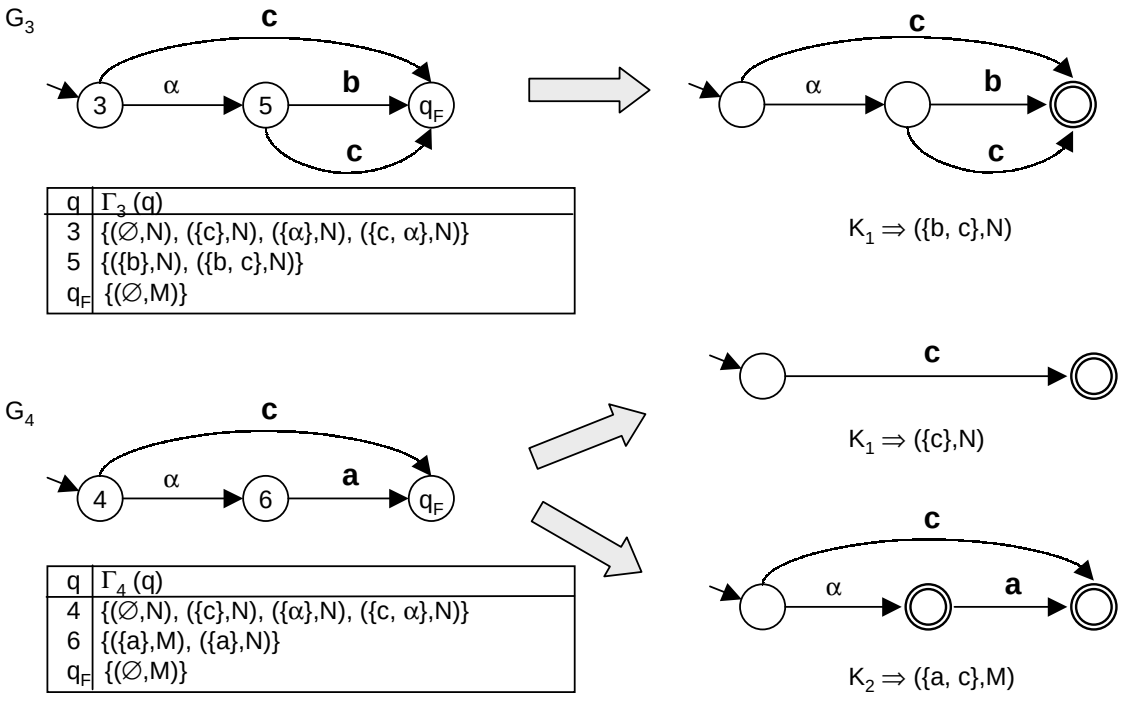


Figura 5.4: Exemplos de controles vocais.

Proposição 5.1 (Fechamento em relação à união) *Para a palavra s em L_S, o conjunto de controles vocais de s, $\Gamma_{S,voc}(s)$ é fechado em relação à união.*

Demonstração. Cada controle vocal válido $(\gamma, \#) \in \Gamma_{S,voc}(s)$ corresponde a uma linguagem $(L_{S(s)}, \Gamma_{S(s)})$-compatível. Como o conjunto das linguagens $(L_{S(s)}, \Gamma_{S(s)})$-compatíveis contidas em $L_{S(s)}$ é fechado em relação à união de linguagens, proposição 4.2, o conjunto $\Gamma_{S,voc}(s)$ é fechado em relação à união de controles. □

A proposição 5.1 afirma que o conjunto de controles vocais é fechado em relação à união, conforme o requisito da definição 4.2 para conjuntos de controles em SEDs com marcação flexível.

O cálculo do conjunto de controles vocais de s em L_S corresponde à solução de um conjunto de problemas de controle supervisório para o subsistema $S(s)$. Defina-se a linguagem de especificação

$$E_s(\gamma, M) = L_{S(s)} - \{u \in L_{S,voc}(s) : w(su) \notin \gamma\} \tag{5.3}$$

para o controle (γ, M). A linguagem $E_s(\gamma, M)$ é prefixo-fechada e obtida de $L_{S(s)}$ pelo corte das palavras que terminem por eventos relevantes que não estejam em γ. Defina-se também a linguagem de especificação

$$E_s(\gamma, N) \;=\; L_{S,voc}(s) - \{u \in L_{S,voc}(s) : w(su) \notin \gamma\} \tag{5.4}$$

para o controle (γ, N). A linguagem $E_s(\gamma, N)$ é construída a partir de $L_{S,voc}(s)$ cortando-se as palavras que terminem por eventos relevantes que não estejam em γ. Note-se que a diferença entre $E_s(\gamma, M)$ e $E_s(\gamma, N)$ é que a primeira contém palavras que não terminam por eventos relevantes.

Proposição 5.2 (Cálculo do conjunto de controles vocais) *Para a palavra s em L_S e o controle $(\gamma, \#) \in 2^{\Sigma_{S,voc}(s)} \times \{M, N\}$, se $K = \sup C_{(L_{S(s)}, \Gamma_{S(s)})}(E_s(\gamma, \#))$ é não vazio e satisfaz aos requisitos 2 e 4 da definição 5.2, então o controle $(\gamma, \#)$ está no conjunto $\Gamma_{S,voc}(s)$.*

Demonstração. Para o controle (γ, M), a especificação $E_s(\gamma, M)$ representa o comportamento menos restritivo de $S(s)$ que habilita os eventos em γ e que marca palavras de $S(s)$ que não terminem por eventos relevantes. Se K acima definido é vazio, a especificação $E_s(\gamma, M)$ e nenhuma de suas sublinguagens podem ser realizadas em $S(s)$ por meio de controle supervisório não bloqueante. Por outro lado, se K for não vazia e atender às condições 2 e 4 da definição 5.2, o controle (γ, M) está em $\Gamma_{S,voc}(s)$. Um raciocínio similar traz o mesmo resultado para o controle não marcado (γ, N). O teste da condição 3 da definição 5.2 não é necessário para $E_s(\gamma, N)$ porque, da equação (5.4), $E_s(\gamma, N) - L_{S,voc}(s) = \emptyset$. □

Considere o SED com marcação flexível S representado pelo par (G, Γ), onde G é um autômato e Γ é uma estrutura de controle dependente do estado, vide seção 4.1. Para o estado q de G, calcula-se o conjunto de controles vocais de q, denotado $\Gamma_{S,voc}(q)$, aplicando-se o algoritmo 5.2. É necessário explicar alguma notação para o entendimento do algoritmo 5.2. Para o estado q de G, $\Sigma_{S,voc}(q)$ é o conjunto ativo de eventos relevantes em S no estado q, obtido por separação das etiquetas de eventos de transições de G_q que envolvam eventos relevantes; $L_{S,voc}(q)$ é o conjunto de palavras geradas por G_q que terminem em eventos relevantes, obtido separando-se as palavras que terminam no estado fictício criado para G_q (o estado q_F); para o controle (γ, N), a especificação $E_q(\gamma, N)$ é obtida apagando-se transições de G_q cujas etiquetas sejam eventos relevantes que não estejam em γ e marcando-se o estado fictício q_F; para o controle (γ, M), a especificação $E_q(\gamma, M)$ é obtida apagando-se as transições de G_q cujas etiquetas sejam eventos relevantes que não estejam em γ e marcando-se *todos* os estados do autômato resultante. Observa-se que o algoritmo 5.2 é uma aplicação direta da proposição 5.2.[2]

Exemplo 5.4 (Cálculo dos conjuntos de controles vocais) *Aplicando-se o algoritmo 5.2 para o cálculo dos conjuntos de controles vocais para os estados 3 e 4 do sistema S do exemplo 5.2 tem-se* $\Gamma_{S,voc}(3) = \{(\{b\}, N), (\{b, c\}, N)\}$ *e* $\Gamma_{S,voc}(4)$

[2]As definições formais de $\Sigma_{S,voc}(q)$, $L_{S,voc}(q)$, $\Gamma_{S,voc}(q)$, $E_q(\gamma, M)$ e $E_q(\gamma, N)$ para o estado q são extensões óbvias das definições correspondentes para a formulação em termos de linguagens.

entradas
Autômato $G = (\Sigma, Q, \delta, q_0, \emptyset)$ e
Estrutura de controle $\Gamma : Q \rightarrow 2^{2^{\Sigma} \times \{M,N\}}$
Conjunto de eventos relevantes $\Sigma_r \subseteq \Sigma$
Estado $q \in Q$
início
calcule o subsistema de q, $G_q = (\Sigma, Q_q, \delta_q, q, \emptyset)$ e $\Gamma_q : Q_q \rightarrow 2^{2^{\Sigma} \times \{M,N\}}$
$\Gamma_{S,voc}(q) \leftarrow \emptyset$
para todo $(\gamma, \#) \in 2^{\Sigma_{S,voc}(q)} \times \{M, N\}$ **faça**
  calcule $K = \sup C_{(G_q,\Gamma_q)}(E_q(\gamma, \#))$
  se $(K \neq \emptyset)$ **então**
    se para todo $u \in L_{S,voc}(q)$, se u termina por um evento relevante em γ se e somente se $u \in K$ **então**
      se $(\# = N)$ ou $(\# = M$ e $K - L_{S,voc}(q) \neq \emptyset)$ **então**
        $\Gamma_{S,voc}(q) \leftarrow \Gamma_{S,voc}(q) \cup \{(\gamma, \#)\}$
      fim se
    fim se
  fim se
fim para
fim
saída
Conjunto de controles vocais $\Gamma_{S,voc}(q)$

Algoritmo 5.2: Calcula o conjunto de controles vocais de um estado

$= \{(\{c\}, N), (\{a\}, N), (\{a\}, M), (\{a, c\}, N), (\{a, c\}, M)\}$.

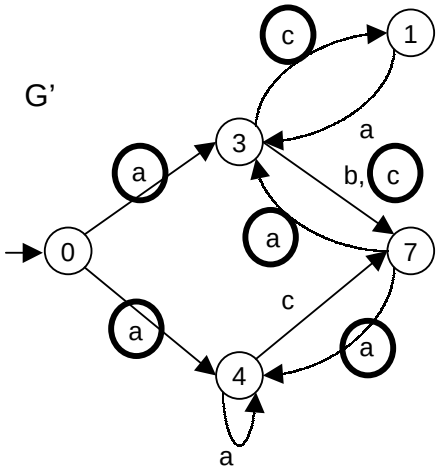
E isso completa a análise do sistema do operador. Na próxima seção trata-se do sistema do gerente para o esquema de supervisão hierárquico proposto.

5.3 Sistema do gerente

Nesta seção apresenta-se o modelo do gerente para o esquema de supervisão hierárquica de dois níveis proposto.

Dados o sistema do operador S sobre o alfabeto Σ, e o conjunto de eventos relevantes $\Sigma_r \subseteq \Sigma$, com mapa repórter associado θ, observa-se que o sistema do gerente é atualizado apenas na inicialização do sistema do operador e na ocorrência de um evento relevante, então reportado por θ. Da definição das palavras vocais de S, vide capítulo 3, as atualizações do gerente correspondem à geração de palavras vocais pelo operador. Assim, uma definição natural para o sistema do gerente é fazer a linguagem gerada como sendo a θ-imagem da linguagem vocal do operador e fazer os conjuntos de controle das palavras do gerente iguais aos conjuntos de controles vocais das palavras vocais correspondentes. O problema é que pode haver mais de uma palavra vocal correspondente a uma dada palavra do sistema do gerente. Na representação por autômatos, este problema faz com que as transições do autômato para o gerente sejam não deterministas.

Exemplo 5.5 (Não determinismo para o gerente) *O autômato G' na figura 5.5 é a projeção de G do exemplo 5.1 sobre o alfabeto $\Sigma_r = \{a, b, c\}$, denotado $G' = proj(G, \Sigma_r)$, vide capítulo 2. Como o interesse é apenas na linguagem gerada por G, faz-se o conjunto de estados marcados de G' vazio. Na figura 5.5, mostram-se também os conjuntos de controles vocais $\Gamma_{S,voc}(q)$, calculados para S e Σ_r, cada um correspondendo a um estado q de G'. O par $(G', \Gamma_{S,voc})$ é uma potencial opção para sistema do gerente. O problema é que o autômato G' é não determinista, como pode ser constatado pelas transições $(0, a, 3)$ e $(0, a, 4)$, ou $(3, c, 1)$ e $(3, c, 7)$ em G'.*



q	$\Gamma_{S,voc}$ (q)
0	{({a},N)}
1	{({a},N)}
3	{({b},N), ({a, b},N)}
4	{({c},N), ({a},N), ({a},M), ({a, c},N), ({a, c},M)}
7	{(∅,M), ({a},N), ({a},M)}

Figura 5.5: Autômato não determinista para o gerente.

Consideram-se duas possíveis soluções para o problema do não determinismo na construção do sistema do gerente. A primeira é definir conjuntos de controle para as palavras do gerente como uma função dos controles vocais das palavras vocais correspondentes, solução considerada por da Cunha e Cury (2002b). A segunda solução é eliminar o não determinismo pela modificação de etiquetas de eventos do operador, solução considerada por Torrico e Cury (2002a) e da Cunha e Cury (2002b). Ambas as soluções são apresentadas a seguir.

Para a definição a seguir, recorda-se que, para a palavra $t \in \theta(L_S)$, $\theta_{voc}^{-1}(t)$ é o conjunto de palavras vocais de S correspondentes a t, equação (3.11).

Definição 5.3 (Sistema do gerente) *Define-se o sistema do gerente como sendo o SED com marcação flexível P sobre o alfabeto Σ_r para o qual $L_P = \theta(L_S)$ e, para t em L_P,*

$$\begin{aligned}\Gamma_P(t) \;=\; \{(\gamma,\#) \in 2^{\Sigma_r} \times \{M,N\} :\; & (\forall s \in \theta_{voc}^{-1}(t))\ (\exists(\gamma',\#') \in \Gamma_{S,voc}(s))\ \textit{tais que} \\ & ((\gamma \cap \Sigma_{S,voc}(s) = \gamma')\ e\ (\#' = \#))\}\end{aligned} \tag{5.5}$$

Observe na equação (5.5) que, para cada controle de uma palavra t de P, deve

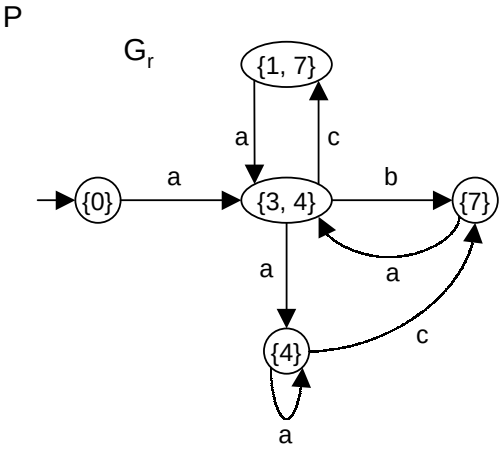
haver um controle vocal correspondente para toda palavra vocal s de S cuja θ-imagem é t. O efeito do controle em t deve ser o mesmo que o efeito virtual do controle vocal correspondente para o subsistema $S(s)$, isto é, mesmos eventos relevantes habilitados e mesmo atributo de marcação.

Seja o sistema do operador S representado pelo par (G, Γ), onde G é um autômato e Γ é uma estrutura de controle dependente do estado, vide seção 4.1. Para o conjunto de eventos relevantes $\Sigma_r \subseteq \Sigma$, o sistema representado pelo par (G_r, Γ_r), saída do algoritmo 5.3, é a representação de estados do SED P correspondente da definição 5.3. O algoritmo 5.3 é uma aplicação direta da definição 5.3. No algoritmo 5.3 usa-se a construção do autômato determinista equivalente de um dado autômato não determinista, vide seção 2.5.4.

entradas
Autômato $G = (\Sigma, Q, \delta, q_0, \emptyset)$ e
Estrutura de controle $\Gamma : Q \rightarrow 2^{2^{\Sigma} \times \{M,N\}}$
Conjunto de eventos relevantes $\Sigma_r \subseteq \Sigma$
início
calcule Q_{voc}, conjunto de estados vocais de G
para todo $q \in Q_{voc}$ **faça**
  calcule o subsistema para q, $G_q = (\Sigma, Q_q, \delta_q, q, \emptyset)$ e $\Gamma_q : Q_q \rightarrow 2^{2^{\Sigma} \times \{M,N\}}$
  calcule o conjunto de controles vocais de q, $\Gamma_{S,voc}(q)$
  $\psi(q, \sigma) \leftarrow \emptyset$
  para todo $\sigma \in \Sigma_r$, $q' \in Q_{voc}$ e $q'' \in Q_q$ tal que $\delta_q(q'', \sigma) = q'$ **faça**
    $\psi(q, \sigma) \leftarrow \psi(q, \sigma) \cup \{q'\}$
  fim para
fim para
calcule o equivalente determinista de $(\Sigma_r, Q_{voc}, \psi, q_0, \emptyset)$, $(\Sigma_r, Q^D_{voc}, \psi^D, q^D_0, \emptyset)$
para todo $q^D \in Q^D_{voc}$ **faça**
  calcule $\Gamma_r(q^D)$ a partir de $\Gamma_{S,voc}(q)$, para $q \in q^D$ {usa-se a equação (5.5)}
fim para
fim
saídas
SED P:
Autômato $G_r = (\Sigma_r, Q^D_{voc}, \psi^D, q^D_0, \emptyset)$ e
Estrutura de controle $\Gamma_r : Q^D_{voc} \rightarrow 2^{2^{\Sigma_r} \times \{M,N\}}$

Algoritmo 5.3: Calcula o sistema do gerente

Exemplo 5.6 (Sistema do gerente) *Considerando-se como conjunto de eventos relevantes* $\Sigma_r = \{a, b, c\}$ *, o sistema* P *da figura 5.6 é o sistema do gerente correspondente ao sistema* S *no exemplo 5.1 por aplicação do algoritmo 5.3. O autômato* G_r *é o equivalente determinista ao autômato* G' *da figura 5.5. Os estados indicados na figura 5.6 são os estados equivalentes encontrados na figura 5.5. A estrutura de controle resultante,* Γ_r*, é também representada na figura 5.6. Observe que o sistema do operador possui estrutura de controle padrão RW, entretanto o sistema do gerente possui a forma geral da estrutura de controle do SED com marcação flexível.*



q	Γ_r (q)
{0}	{({a},N)}
{1, 7}	{({a},N)}
{3, 4}	{({a, b},N), ({b, c},N), ({a, b, c},N)}
{4}	{({c},N), ({a},N), ({a},M), ({a, c},N), ({a, c},M)}
{7}	{(∅,M), ({a},N), ({a},M)}

Figura 5.6: SED do gerente.

As duas proposições seguintes apresentam propriedades para o sistema do gerente P proposto na definição 5.3.

Proposição 5.3 (Controle correspondente) *Para* t *em* L_P*,* $(\gamma_r, \#_r)$ *em* $\Gamma_P(t)$ *e* s *em* $\theta^{-1}_{voc}(t)$ *existe um único controle* $(\gamma'_r, \#'_r)$ *em* $\Gamma_{S,voc}(s)$ *tal que* $\gamma'_r = \gamma \cap \Sigma_{S,voc}(s)$ *e* $\#'_r = \#_r$*. O referido controle é calculado pela fórmula*

$$(\gamma_r, \#_r)^{\uparrow} \quad = \quad \sup\{(\gamma'_r, \#'_r) \in \Gamma_{S,voc}(v) : (\gamma'_r \subseteq \gamma_r) \ e \ (\#'_r \leq \#_r)\}. \tag{5.6}$$

Demonstração. Da definição de $\Gamma_P(t)$, equação (5.5), e pelo fato de que $\Gamma_{S,voc}(s) \subseteq 2^{\Sigma_{S,voc}(s)} \times \{M, N\}$ na definição 5.2. □

Proposição 5.4 (Fechamento em relação à união dos controles do gerente) *Para o SED com marcação flexível P, definição 5.3, e todo t em L_P, $\Gamma_P(t)$ é fechado em relação à união.*

Demonstração. Primeiramente, prova-se que, pela definição de $\Gamma_P(t)$, equação (5.5), para t em L_P, $(\gamma_r, \#_r)$ em $\Gamma_P(t)$, s_i em $\theta_{voc}^{-1}(t)$, com i variando de 1 a n, sendo n o número de elementos em $\theta_{voc}^{-1}(t)$, e $(\gamma_i, \#_i)$ sendo os controles correspondentes em s que seguem a $(\gamma_r, \#_r)$, então $\gamma = \gamma_1 \cup \ldots \cup \gamma_n$ e $\# = \#_1 = \ldots = \#_n$.

Sejam então $(\gamma_1, \#_1) \in \Gamma_P(t)$ e$(\gamma_2, \#_2) \in \Gamma_P(t)$. Escreve-se $(\gamma_k, \#_k)$, com k em $\{1, 2\}$, como sendo $(\gamma_k, \#_k) = (\gamma_k^1 \cup \ldots \cup \gamma_k^n, \#_k^1 \wedge \ldots \wedge \#_k^n)$, onde $(\gamma_k^i, \#_k^i) \in \Gamma_{S,voc}(s_i)$, $i \in \{1, \ldots n\}$, n é o número de elementos de $\theta_{voc}^{-1}(t)$, e $\#_k^1 = \ldots = \#_k^n = \#_k$. Escreve-se $(\gamma, \#) = (\gamma_1 \cup \gamma_2, \#_1 \vee \#_2) = ((\gamma_1^1 \cup \ldots \cup \gamma_1^n) \cup (\gamma_2^1 \cup \ldots \cup \gamma_2^n), (\#_1^1 \wedge \ldots \wedge \#_1^n) \vee (\#_1^1 \wedge \ldots \wedge \#_1^n)) = ((\gamma_1^1 \cup \gamma_2^1) \cup \ldots \cup (\gamma_1^n \cup \gamma_2^n), (\#_1^1 \vee \#_2^1) \wedge \ldots \wedge (\#_1^n \vee \#_2^n))$. É possível escrever a última expressão acima porque $\#_1^1 = \ldots = \#_1^n = \#_1$ e $\#_2^1 = \ldots = \#_2^n = \#_2$, conforme a equação (5.5).

Pela proposição 5.1, $\Gamma_{S,voc}(s_i)$ é fechado em relação à união de controles, então $(\gamma_1^i \cup \gamma_2^i, \#_1^i \vee \#_2^i) \in \Gamma_{S,voc}(s_i)$. Assim, pela definição de $\Gamma_P(t)$, equação 5.5, $(\gamma, \#) \in \Gamma_P(t)$ e isto completa a prova. □

A proposição 5.4 mostra que o sistema do gerente P, definição 5.3, atende ao requisito do modelo apresentado no capítulo 4, definição 4.2.

Um caso especial do sistema do gerente surge quando o mapa repórter tem a seguinte propriedade.

Definição 5.4 (Mapa repórter determinista) *Dados o SED com marcação flexível S sobre o alfabeto Σ e o conjunto de eventos relevantes $\Sigma_r \subseteq \Sigma$, o mapa repórter θ associado a S e Σ_r é determinista se, para todo par de palavras vocais distintas de S, digam-se s_1 e s_2, $\theta(s_1) \neq \theta(s_2)$.*

Se o mapa repórter é determinista, todo par de palavras vocais distintas possuem θ-imagens também distintas. O mapa repórter determinista resulta num sistema do gerente com a seguinte propriedade.

Proposição 5.5 (Mapa repórter determinista) *Para o sistema do gerente P, definição 5.3, se o mapa repórter for determinista, para todo t em L_P existe uma única palavra vocal $s \in \theta_{voc}^{-1}(t)$, e ainda, $\Gamma_P(t) = \Gamma_{S,voc}(s)$.*

Demonstração. Se o mapa repórter for determinista, para todo $t \in L_P$, há apenas uma palavra vocal s tal que $\theta(s) = t$. Assim, s é o único elemento de $\theta_{voc}^{-1}(t)$, e, pela definição 5.3, $\Gamma_P(t) = \Gamma_{S,voc}(s)$, o que completa a prova. □

Dados o SED com marcação flexível S sobre o alfabeto Σ e o conjunto de eventos $\Sigma_r \subseteq \Sigma$, pode-se provar que é sempre possível modificar as etiquetas de eventos em Σ_r, de forma a se obter um alfabeto Σ', um sistema equivalente S' e um conjunto de eventos relevantes Σ'_r tais que o mapa repórter associado seja determinista. O conjunto de eventos Σ'_r é obtido de Σ_r pela criação de novas etiquetas de eventos a partir das etiquetas originais, o novo alfabeto do sistema é $\Sigma' = \Sigma'_r \cup (\Sigma - \Sigma_r)$, e o novo sistema S' é obtido substituindo as etiquetas de transições com eventos em Σ_r por etiquetas em Σ'_r.

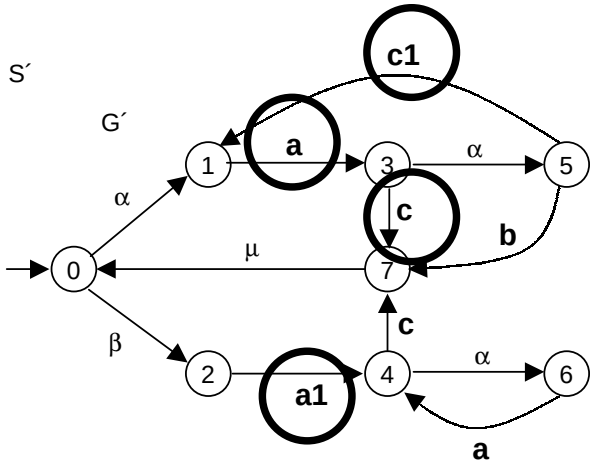
Considere a representação de estados para S dada pelo par (G, Γ), onde G é um autômato e Γ uma estrutura de controle dependente do estado, vide seção 4.1. O resultado da aplicação do algoritmo 5.4 a (G, Γ) e Σ_r, fornece o par (G', Γ'), a representar o sistema S', e Σ'_r tais que o mapa repórter associado a S' e Σ'_r é determinista. O algoritmo 5.4 torna a função ψ determinista por modificação das etiquetas de eventos. A função ψ define transições envolvendo triplas de estado vocal, evento relevante e estado vocal, e é usada para criar as transições no sistema do gerente. As novas etiquetas de evento são criadas como instâncias dos eventos relevantes originais, o que preserva a semântica de eventos do gerente. O autômato correspondente do gerente é determinista, como conseqüência.

Exemplo 5.7 (Mapa repórter determinista) *A figura 5.7 mostra o resultado da*

entradas
Autômato $G = (\Sigma, Q, \delta, q_0, \emptyset)$ e
Estrutura de controle $\Gamma : Q \to 2^{2^{\Sigma} \times \{M,N\}}$
Conjunto de eventos relevantes $\Sigma_r \subseteq \Sigma$
início
$G' \leftarrow G$
$\Gamma' \leftarrow \Gamma$
$\Sigma'_r \leftarrow \Sigma_r$
calcule os estados vocais de G', Q_{voc}
para todo $q \in Q_{voc}$ **faça**
  calcule o subsistema de q, $G'_q = (\Sigma, Q'_q, \delta'_q, q, \emptyset)$ e $\Gamma'_q : Q'_q \to 2^{2^{\Sigma} \times \{M,N\}}$
  $\psi(q, \sigma) \leftarrow \emptyset$
  para todo $\sigma \in \Sigma'_r$, $q' \in Q_{voc}$ e $q'' \in Q'_q$ tal que $\delta_q(q'', \sigma) = q'$ **faça**
    $\psi(q, \sigma) \leftarrow \psi(q, \sigma) \cup \{q'\}$
  fim para
  para todo $\sigma \in \Sigma'_r$ para o qual $\psi(q, \sigma)$ possui mais de um elemento **faça**
    Crie uma nova etiqueta de evento σ_i como instância σ
    $\Sigma'_r \leftarrow \Sigma'_r \cup \{\sigma_i\}$
    $\Sigma' \leftarrow \Sigma' \cup \{\sigma_i\}$
    para todo $(q', q'') \in Q_{voc} \times Q'_q$ para o qual $\delta'(q'', \sigma) = q'$ **faça**
      apague $\delta'(q'', \sigma)$ e defina $\delta'(q'', \sigma_i) \leftarrow q'$
      substitua a etiqueta σ por σ_i em $\Gamma'(q'')$
    fim para
  fim para
fim para
fim
saídas
Autômato $G' = (\Sigma', Q', \delta', q_0, \emptyset)$ e
Estrutura de controle $\Gamma' : Q' \to 2^{2^{\Sigma'} \times \{M,N\}}$
Conjunto de eventos relevantes modificado $\Sigma'_r \subseteq \Sigma'$

Algoritmo 5.4: Modifica o sistema do operador e o conjunto de eventos relevantes para tornar o mapa repórter determinista.

aplicação do algoritmo 5.4 para o sistema S do exemplo 5.2, sendo o sistema resultante S′. Observe-se que há um não determinismo envolvendo os estados 3 e 4 de S e o evento a, e uma nova instância a_1 é criada para o evento a. Outro não determinismo, agora envolvendo os estados 3 e 5 e o evento c é resolvido pela criação da instância c_1 para o evento c. A criação de instâncias preserva a semântica do evento para o gerente. O sistema do gerente resultante $P' = (G'_r, \Gamma'_r)$ é mostrado na figura 5.8, por aplicação direta do algoritmo 5.3. Repare que G'_r é determinista.



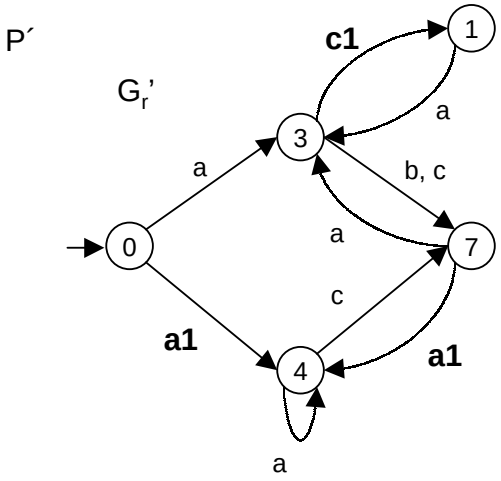
q	Γ´ (q)
0	{({β},N), ({α, β},N)}
1	{({a},N)}
2	{({**a1**},N)}
3	{(∅,N), ({c},N), ({α},N), ({c, α},N)}
4	{(∅,N), ({c},N), ({α},N), ({c, α},N)}
5	{({b},N), ({b, **c1**},N)}
6	{({a},M), ({a},N)}
7	{(∅,N), ({μ},N), (∅,M), ({μ},M)}

Figura 5.7: Modificação para tornar o mapa repórter determinista

Revendo as duas soluções para o não determinismo no gerente mencionadas no início desta seção, a primeira solução corresponde à construção conservadora do sistema do gerente na definição 5.3 (da Cunha e Cury 2002b). A segunda, trata-se da transformação do mapa repórter em determinista (da Cunha e Cury 2002b, Torrico e Cury 2002a).

5.4 Consistência hierárquica

Esta seção apresenta os principais resultados referentes à consistência hierárquica do esquema de supervisão hierárquica de dois níveis proposto.



q	Γ_r' (q)
0	{({a1},N), ({a, a1},N)}
1	{({a},N)}
3	{({c},N), ({b},N), ({b, c1},N), ({b, c},N) ({b, c, c1},N)}
4	{({c},N), ({a},N), ({a},M), ({a, c},N), ({a, c},M)}
7	{(∅,M), ({a1},N), ({a1},M), ({a, a1},N), ({a, a1},M)}

Figura 5.8: Sistema do gerente resultante.

Considere o esquema de supervisão hierárquica de dois níveis formado pelo sistema do operador S sobre o alfabeto Σ e o sistema do gerente P sobre o alfabeto $\Sigma_r \subseteq \Sigma$, construído conforme a definição 5.3 (vide figura 5.1). Seja uma linguagem para o gerente $K_r \subseteq \Sigma_r^*$ que seja (L_P, Γ_P)-compatível. Assim, pelo teorema 4.1, existe um supervisor não bloqueante f_{ge} para P tal que $L_{f_{ge}/P,m} = K_r$. Também pelo teorema 4.1, define-se o supervisor do gerente $f_{ge} : L_P \rightarrow 2^{\Sigma_r} \times \{M, N\}$ como sendo:

$$f_{ge}(t) = \begin{cases} (\gamma_r, M) \in \Gamma_P(t) & \text{para } t \in K_r \\ (\gamma_r, N) \in \Gamma_P(t) & \text{para } t \in \overline{K_r} - K_r \\ \text{indefinido} & \text{caso contrário} \end{cases} \tag{5.7}$$

onde $\gamma_r \cap \Sigma_{L_P}(t) = \Sigma_{K_r}(t)$.

O supervisor do operador a ser proposto f_{op} é projetado de tal forma que o sistema do operador é visto como operando num subsistema correspondente à mais recente palavra vocal gerada. Na ocorrência de um novo evento relevante, considera-se que o sistema passa a se comportar como outro subsistema correspondente à nova palavra vocal gerada. Assim, decompõe-se a palavra do operador $s \in L_S$ como sendo $s = vu$,

onde $v = pre_{voc}(s) = \sup\{v' \in L_{S,voc} : v' \leq v\}$ é chamado prefixo vocal de s, o maior prefixo de s que é uma palavra vocal, e $u \in L_{S(v)} - L_{S,voc}(v)$.

Considere a ocorrência da palavra vocal v em S, correspondendo à ocorrência de um novo evento relevante. Neste ponto, o sistema do gerente é atualizado e o supervisor para o gerente seleciona uma nova diretiva de controle, diga-se $(\gamma_r, \#_r) = f_{ge}(t)$, onde $t = \theta(v)$, e a comunica ao supervisor do operador. O supervisor do operador faz duas ações. A primeira é encontrar o controle vocal correspondente à diretiva de controle $(\gamma_r, \#_r)$, diga-se $(\gamma'_r, \#'_r)$, encontrado aplicando-se a equação (5.6) da proposição 5.3. Segundo, o supervisor do operador passa a aplicar a política de controle correspondente ao controle vocal $(\gamma'_r, \#'_r)$ para o subsistema $S(v)$, dada por

$$E_v(\gamma_r, \#_r)^{\uparrow} \;=\; \sup C_{(L_{S(v)},\Gamma_{S(v)})}(E_v(\gamma'_r, \#'_r)) \;=\; \sup C_{(L_{S(v)},\Gamma_{S(v)})}(E_v(\gamma_r, \#_r)) \tag{5.8}$$

previamente calculada, quando da determinação do conjunto de controles vocais de v, $\Gamma_{S,voc}(v)$.

Defina-se então o supervisor do operador $f_{op} : L_S \to 2^{\Sigma} \times \{M, N\}$ para a palavra $s \in L_S$ como sendo:

$$f_{op}(s) = \begin{cases} (\gamma, M) \in \Gamma_{S(v)}(u) & \text{para } u \in E_v(\gamma_r, \#_r)^{\uparrow} \\ (\gamma, N) \in \Gamma_{S(v)}(u) & \text{para } u \in \overline{E_v(\gamma_r, \#_r)^{\uparrow}} - E_v(\gamma_r, \#_r)^{\uparrow} \\ \text{não definido} & \text{caso contrário} \end{cases} \tag{5.9}$$

onde $s = vu$, $v = pre_{voc}(s)$, $u \in L_{S(v)} - L_{S,voc}(v)$, $(\gamma_r, \#_r) = f_{ge}(\theta(s))$, e $\gamma \cap \Sigma_{L_{S(v)}}(u) = \Sigma_{E_v(\gamma_r,\#_r)^{\uparrow}}(u)$.

A partir das definições dos supervisores acima, são enunciados duas propriedades da malha fechada do operador f_{op}/S.

Proposição 5.6 (Comportamento em malha fechada do operador) *Para o esquema de supervisão hierárquica proposto, a linguagem gerada em malha fechada, $L_{f_{op}/S} \subseteq L_S$, e a linguagem marcada em malha fechada, $L_{f_{op}/S,m} \subseteq L_{f_{op}/S}$, são tais*

que, para s em L_S:

1. *$s \in L_{f_{op}/S}$ se e somente se $u \in \overline{E_v(\gamma_r, \#_r)^\uparrow} - L_{S,voc}(v)$, e*

2. *$s \in L_{f_{op}/S,m}$ se e somente se $u \in E_v(\gamma_r, \#_r)^\uparrow - L_{S,voc}(v)$,*

onde $s = vu$, $v = pre_{voc}(s)$, $u \in L_{S(v)} - L_{S,voc}(v)$ e $(\gamma_r, \#_r) = f_{ge}(\theta(s))$.

Demonstração. Os ítens 1 e 2 da proposição decorrem da definição do supervisor f_{op} na equação 5.9. □

Proposição 5.7 (Palavras vocais habilitadas na supervisão do operador) *No esquema de supervisão hierárquica proposto, para $t \in \Sigma_r^*$, se t está em K_r, então para todo $s \in \theta_{voc}^{-1}(t)$, tem-se $s \in L_{f_{op}/S}$.*

Demonstração. A proposição decorre em função do ítem 2 da definição dos controles vocais, definição 5.2, e da definição do sistema do gerente, definição 5.3. A partir destas duas definições conclui-se que duas palavras vocais equivalentes segundo o mapa repórter não podem ser discriminadas a partir de uma diretiva de controle do gerente, e que se uma palavra vocal está habilitada, todas as outras palavras equivalentes segundo o mapa repórter estão habilitadas. □

A proposição 5.7 afirma que se o supervisor do gerente habilita a palavra t em P, o supervisor do operador correspondente também habilita *toda* palavra vocal correspondente a t em S. Os três lemas seguintes são propriedades usadas para provar a consistência hierárquica para o esquema de supervisão hierárquica proposto.

Lema 5.1 (Não bloqueio) *Para o esquema de supervisão hierárquica proposto, se $K_r \subseteq \Sigma_r^*$ é (L_P, Γ_P)-compatível, então f_{op} é não bloqueante.*

Demonstração. Prova-se que $L_{f_{op}/S} = \overline{L_{f_{op}/S,m}}$. Primeiramente, por definição, $L_{f_{op}/S,m} \subseteq L_{f_{op}/S}$, então $\overline{L_{f_{op}/S,m}} \subseteq L_{f_{op}/S}$.

Agora seja $s \in L_{f_{op}/S}$, e decomponha-se $s = vu$, onde $v = pre_{voc}(s)$, $u \in \overline{E_v(\gamma_r, \#_r)^\uparrow} - L_{S,voc}(v)$, e $(\gamma_r, \#_r) = f_{ge}(\theta(s))$. Como $u \in \overline{E_v(\gamma_r, \#_r)^\uparrow} - L_{S,voc}(v)$,

há $u' \in E_v(\gamma_r, \#_r)^\uparrow$ tal que $u \leq u'$. Então, a palavra vu' ou é uma palavra vocal ou uma palavra silenciosa.

Se vu' é silenciosa, $u' \in E_v(\gamma_r, \#_r)^\uparrow - L_{S,voc}(v)$, então pela proposição 5.6, $vu' \in L_{f_{op}/S,m}$ e $s = vu \in \overline{L_{f_{op}/S,m}}$.

Se vu' é uma palavra vocal de S, então $vu' \in L_{S,voc}$, $w(vu') = \tau' \neq \tau_0$ e $\theta(vu') = t\tau' \in \overline{K_r}$. Pode-se sempre estender $vu'u'' \ldots$ e $t\tau'\tau'' \ldots$, onde $\tau'' = w(vu'u'')$, até se encontrar $v_1 \in L_{S,voc}$ e $t_1 \in K_r$ tais que $s = vu \leq v_1$ e $t_1 = \theta(v_1)$, como ilustrado pela figura 5.9.

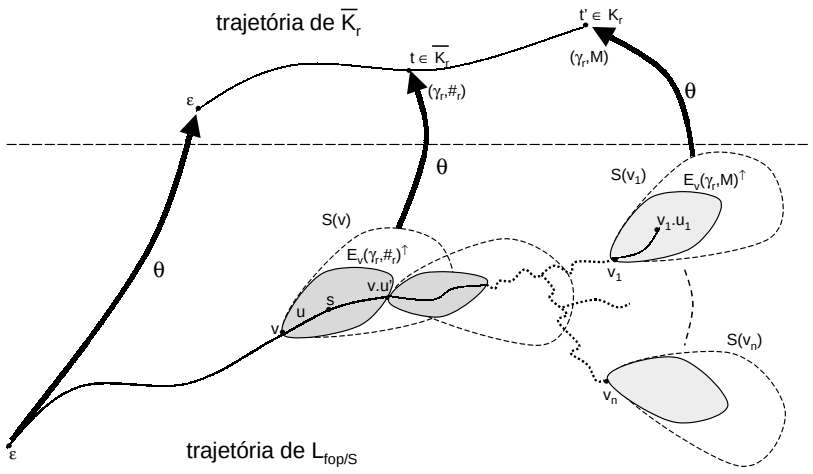


Figura 5.9: O supervisor do operador é não bloqueante

Como $t_1 \in K_r$ e K_r é (L_P, Γ_P)-compatível, então $f_{ge}(t_1) = (\gamma_r, M) \in \Gamma_P(t_1)$. Pela proposição 5.3, para todo $v_i \in \theta^{-1}_{voc}(t_1)$, com $i = 1 \ldots n$ e $n = |\theta^{-1}_{voc}(t)|$, há $(\gamma_{r,i}, M) \in \Gamma_{S,voc}(v_i)$ tal que $\gamma_r = \bigcup_i \gamma_{r,i}$. Então, para v_1 acima, há $u_1 \in E_{v_1}(\gamma_r, M)^\uparrow$ tal que v_1u_1 é silencioso. Pela proposição 5.6, $v_1u_1 \in L_{f_{op}/S,m}$. E, ainda $s = vu \leq vu' \leq v_1 \leq v_1u_1$, tem-se $s = vu \in \overline{L_{f_{op}/S,m}}$. Assim, $L_{f_{op}/S} \subseteq \overline{L_{f_{op}/S,m}}$ e isso completa a prova. □

Lema 5.2 (Compatibilidade) *Para o esquema de supervisão hierárquica proposto, se $K_r \subseteq \Sigma_r^*$ é (L_P, Γ_P)-compatível, então $L_{f_{op}/S,m}$ é (L_S, Γ_S)-compatível.*

Demonstração. Seja $s \in L_{f_{op}/S,m}$ e escreva $s = vu$, onde $v = pre_{voc}(s)$, $u \in E_v(\gamma_r, \#_r)^{\uparrow} - L_{S,voc}(v)$ e $(\gamma_r, \#_r) = f_{ge}(\theta(s))$. Como $E_v(\gamma_r, \#_r)^{\uparrow}$ é $(L_{S(v)}, \Gamma_{S(v)})$-compatível, existe $(\gamma, M) \in \Gamma_{S(v)}(u)$ para o qual $\gamma \cap \Sigma_{L_{S(v)}}(u) = \Sigma_{E_v(\gamma_r,\#_r)^{\uparrow}}(u)$. Mas, $u \notin L_{S,voc}$, isto é, $w(su) = \tau_0$ ou $u = \epsilon$, e, pela definição do subsistema $S(v)$, definição 5.1, $\Gamma_{S(v)}(u) = \Gamma_S(vu)$. Assim, existe $(\gamma, M) \in \Gamma_S(vu)$ tal que $\gamma \cap \Sigma_{L_S}(vu) = \Sigma_{L_{f_{op}/S,m}}(u)$.

Seja $s \in \overline{L_{f_{op}/S,m}} - L_{f_{op}/S,m}$ e escreva $s = vu$, onde $v = pre_{voc}(s)$, $u \in \overline{E_v(\gamma_r, \#_r)^{\uparrow}} - E_v(\gamma_r, \#_r)^{\uparrow}$ e $(\gamma_r, \#_r) = f_{ge}(\theta(s))$. Da equação acima, $u \notin L_{S,voc}(v)$, isto é, $w(su) = \tau_0$ ou $u = \epsilon$. Como $E_v(\gamma_r, \#_r)^{\uparrow}$ é $(L_{S(v)}, \Gamma_{S(v)})$-compatível, existe $(\gamma, N) \in \Gamma_{S(v)}(u)$ tal que $\gamma \cap \Sigma_{L_{S(v)}}(u) = \Sigma_{E_v(\gamma_r,\#_r)^{\uparrow}}(u)$. Pela definição do subsistema $S(v)$, definição 5.1, e como $u \notin L_{S,voc}(v)$, tem-se $\Gamma_{S(v)}(u) = \Gamma_S(vu)$. Assim, existe $(\gamma, N) \in \Gamma_S(vu)$ tal que $\gamma \cap \Sigma_{L_S}(vu) = \Sigma_{L_{f_{op}/S,m}}(u)$.

Pelo desenvolvido acima, conclui-se que $L_{f_{op}/S,m}$ é (L_S, Γ_S)-compatível. □

Lema 5.3 (Imagem para o gerente do supervisor do operador) *Para o esquema de supervisão hierárquica proposto, se $K_r \subseteq \Sigma_r^*$ é (L_P, Γ_P)-compatível, então $\theta(L_{f_{op}/S,m}) = K_r$.*

Demonstração. Prova-se que $\theta(L_{f_{op}/S,m}) = K_r$ primeiro provando-se que $\overline{\theta(L_{f_{op}/S,m})} = \overline{K_r}$, e então provando-se que $\theta(L_{f_{op}/S,m}) \subseteq K_r$ e $K_r \subseteq \theta(L_{f_{op}/S,m})$.

Prova de que $\overline{\theta(L_{f_{op}/S,m})} = \overline{K_r}$. Primeiro, observe-se que $\overline{\theta(L_{f_{op}/S,m})} = \theta(\overline{L_{f_{op}/S,m}}) = \theta(L_{f_{op}/S})$, uma vez que θ preserva prefixos, vide capítulo 3, e pelo lema 5.1. Prova-se que $\theta(L_{f_{op}/S}) = \overline{K_r}$ por indução.

Base: $\epsilon \in \theta(L_{f_{op}/S})$ e $\epsilon \in \overline{K_r}$.

Hipótese: Para $t \in \Sigma_r^*$, considera-se que $t \in \theta(L_{f_{op}/S})$ e $t \in \overline{K_r}$.

Passo: Primeiramente, para $\tau \in \Sigma_r$, faça-se $t\tau \in \theta(L_{f_{op}/S})$. Pela proposição 5.7, como $t \in \theta(L_{f_{op}/S})$, então, para todo $v \in \theta_{voc}^{-1}(t)$, tem-se $v \in L_{f_{op}/S}$. Como $t \in \theta(L_{f_{op}/S})$, então, para todo $s \in L_{f_{op}/S}$ tal que $\theta(s) = t$, pode-se escrever $s = vu$, onde $v = pre_{voc}(s)$, $u \in \overline{E_v(\gamma_r, \#_r)^{\uparrow}} - L_{S,voc}(v)$ e $(\gamma_r, \#_r) = f_{ge}(t)$. Como $t\tau \in \theta(L_{f_{op}/S})$, há $v \in L_{f_{op}/S}$ e $u \in L_{S,voc}(v)$ tais que $\theta(vu) = t\tau$ e $u \in E_v(\gamma_r, \#_r)^{\uparrow}$. Este fato é ilustrado

pela figura 5.10. Pelo desenvolvimento acima, tem-se que $\tau \in \gamma_r$. Como $t \in \overline{K_r}$ e K_r é (L_P, Γ_P)-compatível, $f_{ge}(t) = (\gamma_r, \#_r) \in \Gamma_P(t)$ e $\gamma_r \cap \Sigma_{L_P}(t) = \Sigma_{K_r}(t)$. E tem-se que $t\tau \in \overline{K_r}$, e conclui-se que $\theta(L_{f_{op}/S}) \subseteq \overline{K_r}$.

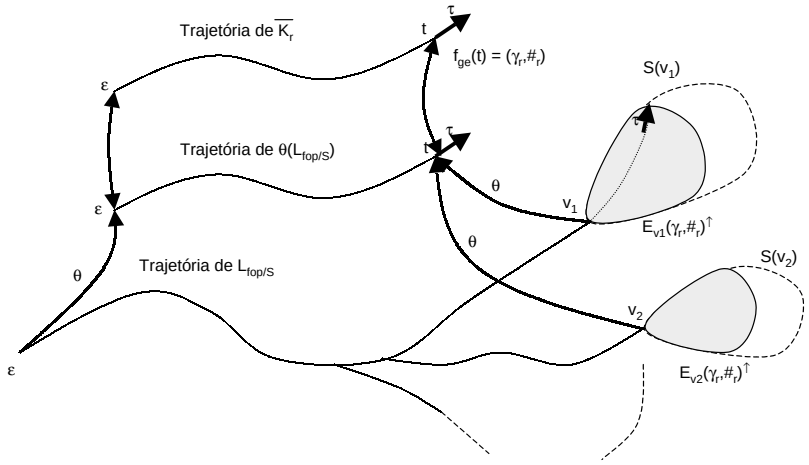


Figura 5.10: Imagem para o gerente do supervisor do operador

Agora, para $\tau \in \Sigma_r$, faça-se $t\tau \in \overline{K_r}$. Como $t \in \overline{K_r}$ e K_r é (L_P, Γ_P)-compatível, então $f_{ge}(t) = (\gamma_r, \#_r) \in \Gamma_P(t)$, com $\gamma_r \cap \Sigma_{L_P}(t) = \Sigma_{K_r}(t)$. Mas, $t\tau \in \overline{K_r}$, então tem-se $\tau \in \gamma_r$. Como $t \in \theta(L_{f_{op}/S})$, pode-se escrever que, para todo $s \in L_{f_{op}/S}$ tal que $\theta(s) = t$, tem-se $s = vu$, para $v = pre_{voc}(s)$, e $u \in \overline{E_v(\gamma_r, \#_r)^{\uparrow}} - L_{S,voc}(v)$. Também com $t \in \theta(L_{f_{op}/S})$ e pela proposição 5.7, para todo $v \in \theta_{voc}^{-1}(t)$, tem-se $v \in L_{f_{op}/S}$. Como $\tau \in \gamma_r$, então, para algum $v \in \theta_{voc}^{-1}(t)$, existe $u \in E_v(\gamma_r, \#_r)^{\uparrow}$ tal que $\theta(vu) = t\tau$. Então, tem-se que $vu \in L_{f_{op}/S}$ e $t\tau = \theta(vu) \in \theta(L_{f_{op}/S})$. Assim, conclui-se que $\overline{K_r} \subseteq \theta(L_{f_{op}/S})$.

Com os dois pontos acima, tem-se $\theta(L_{f_{op}/S}) = \overline{K_r}$ e $\overline{\theta(L_{f_{op}/S,m})} = \overline{K_r}$, o que conclui a primeira parte da prova.

Prova de que $\theta(L_{f_{op}/S,m}) \subseteq K_r$. Primeiro, seja $t \in \theta(L_{f_{op}/S,m})$, então há $s \in L_{f_{op}/S,m}$ tal que $\theta(s) = t$. Pela proposição 5.6, existe $u \in E_v(\gamma_r, \#_r)^{\uparrow} - L_{S,voc}(v)$ tal que $s = vu$, $v = pre_{voc}(s)$ e $(\gamma_r, \#_r) = f_{ge}(t)$. Assim, $E_v(\gamma_r, \#_r)^{\uparrow} - L_{S,voc}(v) \neq \emptyset$, e, pela definição 5.2, tem-se $\#_r = M$. Mas como $t \in \theta(L_{f_{op}/S,m})$, tem-se $t \in \overline{\theta(L_{f_{op}/S,m})}$, e, pela primeira parte do lema, $t \in \overline{K_r}$. Como K_r é (L_P, Γ_P)-compatível, e, pela definição de f_{ge}, equação (5.7), $t \in K_r$. Segundo, seja $t \in \overline{\theta(L_{f_{op}/S,m})} - \theta(L_{f_{op}/S,m})$, então, para todo

$v \in \theta_{voc}^{-1}(t)$, tem-se $v \in L_{f_{op}/S} - L_{f_{op}/S,m}$ e $E_v(\gamma_r, \#_r)^{\uparrow} - L_{S,voc}(v) = \emptyset$, onde $(\gamma_r, \#_r) = f_{ge}(t)$. Novamente, pela definição 5.2, tem-se $\#_r = N$. Como $t \in \overline{\theta(L_{f_{op}/S,m})}$, então $t \in \overline{K_r}$. E, como K_r é (L_P, Γ_P)-compatível, tem-se $t \in \overline{K_r} - K_r$. E isso completa a prova de que $\theta(L_{f_{op}/S,m}) \subseteq K_r$.

Prova de que $K_r \subseteq \theta(L_{f_{op}/S,m})$. Primeiro, seja $t \in K_r$. Como K_r é (L_P, Γ_P)-compatível, $f_{ge}(t) = (\gamma_r, M) \in \Gamma_P(t)$, com $\gamma_r \cap \Sigma_{L_P}(t) = \Sigma_{K_r}(t)$. Mas como $t \in K_r$, tem-se que $t \in \overline{K_r}$ e $t \in \theta(L_{f_{op}/S})$. E como $f_{ge}(t) = (\gamma_r, M)$, então, para todo $v \in \theta_{voc}^{-1}(t)$, tem-se $E_v(\gamma_r, M)^{\uparrow} - L_{S,voc}(v) \neq \emptyset$. Pela proposição 5.7, para todo $v \in \theta_{voc}^{-1}(t)$, tem-se $v \in L_{f_{op}/S}$. E, pela proposição 5.6, existe $u \in E_v(\gamma_r, M)^{\uparrow} - L_{S,voc}(v)$. Assim, $vu \in L_{f_{op}/S,m}$ e $\theta(vu) = t$. E conclui-se que $t \in \theta(L_{f_{op}/S,m})$. Segundo, seja $t \in \overline{K_r} - K_r$. Como K_r é (L_P, Γ_P)-compatível, $f_{ge}(t) = (\gamma_r, N) \in \Gamma_P(t)$, onde $\gamma_r \cap \Sigma_{L_P}(t) = \Sigma_{K_r}(t)$. Como $t \in \overline{K_r} - K_r$, então $t \in \theta(L_{f_{op}/S})$, pelo fato de que $\theta(L_{f_{op}/S}) = \overline{K_r}$. Mas, para todo $v \in \theta_{voc}^{-1}(t)$, tem-se $v \in L_{f_{op}/S}$ e $E_v(\gamma_r, N)^{\uparrow} - L_{S,voc}(v) = \emptyset$. Então, para todo $vu \in L_{f_{op}/S}$ tal que $\theta(vu) = t$, tem-se $u \in \overline{E_v(\gamma_r, N)^{\uparrow}} - E_v(\gamma_r, N)^{\uparrow}$. Assim, $vu \in L_{f_{op}/S} - L_{f_{op}/S,m}$ e $t = \theta(vu) \in \theta(L_{f_{op}/S}) - \theta(L_{f_{op}/S,m}) = \overline{\theta(L_{f_{op}/S,m})} - \theta(L_{f_{op}/S,m})$.

Com isso, tem-se que $\theta(L_{f_{op}/S,m}) = K_r$, e isso completa a prova. □

Teorema 5.1 (Consistência hierárquica) *Para o SED com marcação flexível S sobre o alfabeto Σ e o SED com marcação flexível P sobre o alfabeto $\Sigma_r \subseteq \Sigma$, se P é construído conforme a definição 5.3, então há consistência hierárquica entre S e P.*

Demonstração. Prova-se o teorema mostrando-se que, dado o sistema do operador S e o sistema do gerente P, se uma especificação para o gerente $K_r \subseteq \Sigma_r^*$ é (L_P, Γ_P)-compatível, então existe um supervisor não bloqueante para o operador, diga-se f_{op}, tal que $\theta(L_{f_{op}/,m}) = K_r$. Assim, a prova consiste na proposta do supervisor f_{op}, feita na equação (5.9), e os lemas 5.1, 5.2 e 5.3. □

Exemplo 5.8 (Consistência hierárquica) *Pelo teorema 5.1, há consistência hierárquica entre os sistemas S e P dos exemplos 5.1 e 5.6, respectivamente. Considere a especificação E_1 na figura 5.11(a). Pode-se mostrar que* $\sup C_{(L_P,\Gamma_P)}(E_1)$ *é igual a*

K_1, mostrado na figura 5.11(b). Mostram-se também na figura 5.11(b) os controles que implementam K_1 em P. A figura 5.12 mostra o supervisor do operador correspondente à linguagem K_1 do gerente. Na figura 5.12, o supervisor do operador está decomposto em supervisores locais para os subsistemas de S, como na equação (5.9). A notação $S(i)$ @ $(\gamma, \#)$ na figura 5.12 expressa que o supervisor do operador está operando como um supervisor local para o subsistema $S(i)$, onde i é o índice do estado de S, e aplicando uma política de controle que implementa o controle vocal $(\gamma, \#)$. Note-se que no estado 3 na figura 5.12, o supervisor implementa o controle vocal $(\{b, c\}, N)$, enquanto que no estado 4, o supervisor implementa o controle vocal $(\{c\}, N)$, ambos correspondem ao mesmo estado do gerente e receberam a mesma diretiva gerencial $(\{b, c\}, N)$ neste ponto. Isso expressa a seleção do controle vocal correspondente indicada na proposição 5.3.

Por fim, considere a especificação E_2 na figura 5.11(c). É possível mostrar que a linguagem $\sup C_{(L_P,\Gamma_P)}(E_2)$ é vazia. Entretanto, também pode-se mostrar que a linguagem $\sup C_{(L_S,\Gamma_S)}(E_2 \| L_S)$ é não vazia, o que quer dizer que há uma sublinguagem do operador que atende à especificação E_2. Isso decorre do fato de que não há consistência hierárquica forte entre S e P.

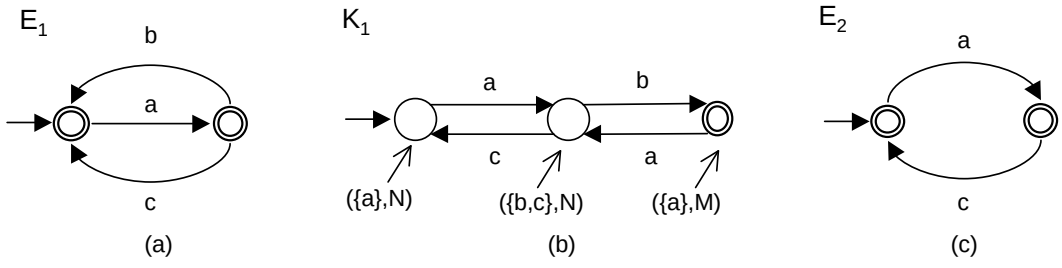


Figura 5.11: Algumas especificações para o gerente.

O objetivo do desenvolvimento a seguir é encontrar condições para a construção de

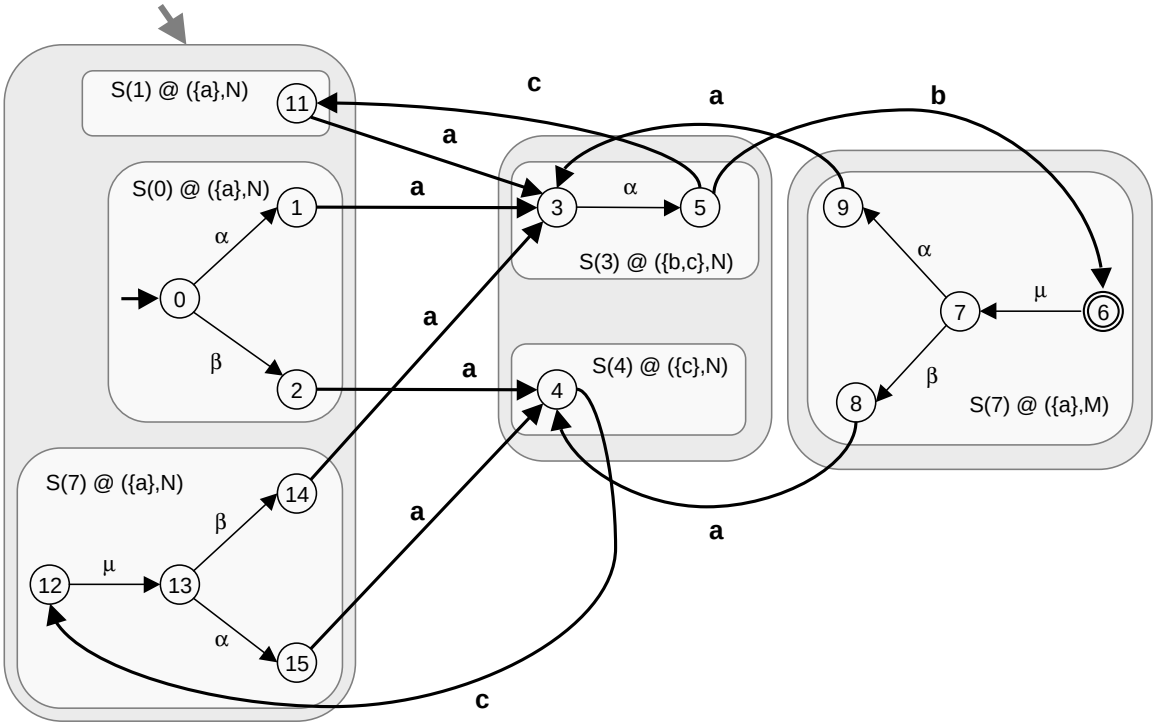


Figura 5.12: Supervisão do operador decomposta em subsistemas.

um esquema de supervisão hierárquica com consistência hierárquica forte.

Lema 5.4 (Imagem para o gerente de linguagem compatível do operador) *Para o esquema de supervisão hierárquica proposto, se o mapa repórter for determinista e $K \subseteq \Sigma^*$ for (L_S, Γ_S)-compatível, então $\theta(K)$ é (L_P, Γ_P)-compatível.*

Demonstração. Seja $t \in \overline{\theta(K)}$ e $v \in \theta_{voc}^{-1}(t) \cap \overline{K}$. Defina-se a linguagem $K(v)$, vide figura 5.13, por :

$$K(v) \quad = \quad \{u \in L_{S(v)} : ((vu \in K) \text{ e } (u \notin L_{S,voc}(v))) \text{ ou } ((vu \in \overline{K}) \text{ e } (u \in L_{S,voc}(v)))\} \tag{5.10}$$

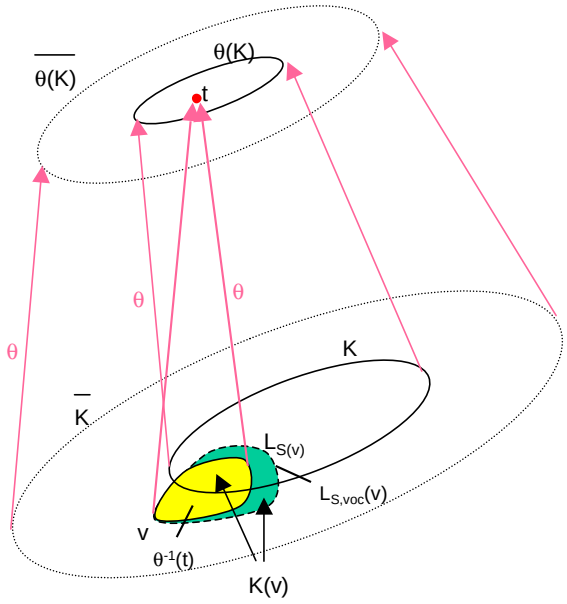


Figura 5.13: Ilustração para a linguagem $K(v)$.

Enumeram-se, a seguir, algumas propriedades da linguagem $K(v)$:

1. Para $t \in \overline{\theta(K)}$ e $v \in \theta_{voc}^{-1}(v) \cap \overline{K}$, $K(v)$ é $(L_{S(v)}, \Gamma_{S(v)})$-compatível, provado no seguinte raciocínio. Seja $u \in K$. Se $vu \in \overline{K}$ e $u \in L_{S,voc}(v)$ então, pela definição da estrutura de controle do subsistema, equação (5.2), existe $(\emptyset, M) \in \Gamma_{S(v)}(u)$ para o qual $\emptyset \cap \Sigma_{L_{S(v)}}(u) = \Sigma_{K(v)}(u)$. Se $vu \in K$ e $u \notin L_{S,voc}(v)$, novamente

pela equação (5.2) e como K é (L_S, Γ_S)-compatível, existe $(\gamma, M) \in \Gamma_S(vu)$ tal que $\gamma \cap \Sigma_{L_S}(vu) = \Sigma_K(vu)$. Mas, $u \notin L_{S,voc}(v)$, então $(\gamma, M) \in \Gamma_{S(v)}(u)$ tal que $\gamma \cap \Sigma_{L_{S(v)}}(u) = \Sigma_{K(v)}(u)$. Agora, seja $u \in \overline{K(v)} - K(v)$, então $vu \in \overline{K}$, $vu \notin L_{S,voc}$ e $vu \notin K$. Como K é (L_S, Γ_S)-compatível, existe $(\gamma, N) \in \Gamma_S(vu)$ para o qual $\gamma \cap \Sigma_{L_S}(vu) = \Sigma_K(vu)$. E, como $vu \notin L_{S,voc}$, pela equação (5.2), existe $(\gamma, N) \in \Gamma_{S(v)}(u)$ tal que $\gamma \cap \Sigma_{L_{S(v)}}(u) = \Sigma_{K(v)}(u)$.

2. Se o mapa repórter é determinista e para $t \in \overline{\theta(K)}$ e $v \in \theta^{-1}_{voc}(v) \cap \overline{K}$, se $u \in L_{S,voc}(v)$ é tal que $w(vu) \in \Sigma_{\theta(K)}(t)$, então $u \in K(v)$, como se prova a seguir. Como o mapa repórter é determinista, há um único $v \in \theta^{-1}_{voc}(t)$ e $u \in L_{S,voc}(v)$ tal que $v \in \overline{K}$ e $w(vu) = \tau \in \Sigma_{\theta(K)}(t)$. Como $\Sigma_{\theta(K)}(t) = \{\tau \in \Sigma_r : t\tau \in \overline{\theta(K)}\}$, tem-se que $v \in \overline{K}$ e, como conseqüência, $u \in K(v)$.

3. Se o mapa repórter é determinista, para $t \in \theta(K)$ e $v \in \theta^{-1}_{voc}(v) \cap \overline{K}$, então $K(v) - L_{S,voc}(v) \neq \emptyset$, provado a seguir. Como o mapa repórter é determinista, se $t \in \theta(K)$, existe um único $v \in \theta^{-1}_{voc}(t) \cap \overline{K}$. Ainda, para todo $s \in \theta^{-1}(t)$, tem-se $v = pre_{voc}(s)$. Se $t \in \theta(K)$, existe $s \in K$ tal que $\theta(s) = t$. Escrevendo-se $s = vu$, tem-se que $u \notin L_{voc,S}$ e $su \in K$. Assim, $K(v) - L_{S,voc}(v) = \{u \in L_{S(v)} : (vu \in K)$ e $(u \notin L_{S,voc}(v))\} \neq \emptyset$.

4. Como o mapa repórter é determinista, e para $t \in \overline{\theta(K)} - \theta(K)$ e $v \in \theta^{-1}_{voc}(t) \cap \overline{K}$, tem-se $K(v) - L_{S,voc}(v) \neq \emptyset$, provado a seguir. Como o mapa repórter é determinista, se $t \in \overline{\theta(K)} - \theta(K)$, há um único $v \in \theta^{-1}_{voc}(t) \cap \overline{K}$. Ainda, para todo $s \in \theta^{-1}(t)$, tem-se $v = pre_{voc}(s)$. Agora, como $t \in \overline{\theta(K)} - \theta(K)$, para todo $s \in \theta^{-1}_{voc}(t)$, então $s \in \overline{K} - K$. Escreva-se $s = vu$, tem-se que $vu \notin K$ e, como conseqüência, $K(v) - L_{S,voc}(v) = \{u \in L_{S(v)} : (vu \in K)$ e $(u \in L_{S,voc}(v))\} = \emptyset$.

A partir das propriedades 1, 2, 3 e 4 acima, conclui-se o seguinte:

- Seja $t \in \theta(K)$. Pelas propriedades 1, 2 e 3, e pela definição 5.2, o controle (γ_r, M) está em $\Gamma_{S,voc}(v)$ para $v \in \theta^{-1}_{voc}(t)$, onde $\gamma_r = \Sigma_{\theta(K)}(t)$. Como o mapa repórter

é determinista, e pela proposição 5.5, tem-se que $(\gamma_r, M) \in \Gamma_P(t)$. Assim, há $(\gamma_r, M) \in \Gamma_P(t)$ tal que $\gamma_r \cap \Sigma_{L_P}(t) = \Sigma_{\theta(K)}(t)$.

- Seja $t \in \overline{\theta(K)} - \theta(K)$. Pelas propriedades 1, 2 e 4, e pela proposição 5.2, o controle (γ_r, N) está em $\Gamma_{S,voc}(v)$ para $v \in \theta_{voc}^{-1}(t)$, onde $\gamma_r = \Sigma_{\theta(K)}(t)$. Novamente, como o mapa repórter é determinista, e pela proposição 5.5, tem-se que $(\gamma_r, N) \in \Gamma_P(t)$. Assim, há $(\gamma_r, N) \in \Gamma_P(t)$ tal que $\gamma_r \cap \Sigma_{L_P}(t) = \Sigma_{\theta(K)}(t)$.

Conclui-se então que $\theta(K)$ é (L_P, Γ_P)-compatível. □

Observe que uma condição prévia necessária para o lema 5.4 é que o mapa repórter seja determinista. Isto se dá porque, quando o mapa repórter é não determinista, o conjunto de controle de uma palavra do gerente $\Gamma_P(\cdot)$, equação (5.5), desconsidera a informação de alguns controles vocais para as palavras vocais correspondentes.

Teorema 5.2 (Consistência hierárquica forte) *Para o SED com marcação flexível S sobre o alfabeto Σ e o SED com marcação flexível P sobre o alfabeto $\Sigma_r \subseteq \Sigma$, se P é construído conforme a definição 5.3 e o mapa repórter associado for determinista, então há consistência hierárquica forte entre S e P.*

Demonstração. Pelo teorema 5.1 e o lema 5.5, conclui-se que, dado $K_r \subseteq \Sigma_r^*$, K_r é (L_P, Γ_P)-compatível se, e somente se, há uma linguagem $K \subseteq \Sigma^*$ que seja (L_S, Γ_S)-compatível e tal que $\theta(K) = K_r$. Assim, há consistência hierárquica forte entre S e P. □

Exemplo 5.9 (Consistência hierárquica forte) *Pelo teorema 5.2, há consistência hierárquica forte entre os sistemas S' e P' do exemplo 5.7. Considere então a tradução da especificação E_2 da figura 5.11(c) para o alfabeto modificado do exemplo 5.7, especificação E_2', representada na figura 5.14. Pode-se provar que* $\sup C_{(L_{P'}, \Gamma_{P'})}(E_2')$ *é não vazia. Pode-se verificar ainda que o mapa repórter correspondente não é um observador e também que não há consistência de marcação, assim violando as condições para consistência hierárquica forte de Wong e Wonham (1996a).*

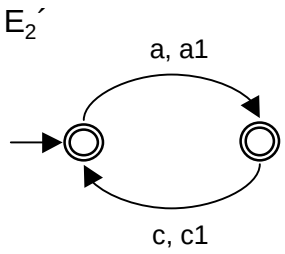


Figura 5.14: Especificação gerencial modificada.

Como último tópico desta seção, explicita-se a natureza da linguagem resultante da supervisão, para o operador, do esquema de supervisão hierárquica proposto. Na proposição 5.8 abaixo, para o SED com marcação flexível S sobre o alfabeto Σ, o conjunto de eventos relevantes $\Sigma_r \subseteq \Sigma$, o SED com marcação flexível P sobre o alfabeto Σ_r, construído pela definição 5.3, e a linguagem $K_r \subseteq L_P$, $\theta^{-1}(K_r)$ é a linguagem que contém todas as palavras em L_S cuja θ-imagem está em K_r, equação (3.9).

Proposição 5.8 (Supervisor monolítico equivalente à supervisão hierárquica) *Para o esquema de supervisão hierárquica proposto, se* $K_r \subseteq \Sigma_r^*$ *é* (L_P, Γ_P)*-compatível, então o supervisor* f_{op} *definido na equação (5.9) é tal que* $L_{f_{op}/S,m} = \sup C_{(L_S,\Gamma_S)}(\theta^{-1}(K_r))$.

Demonstração. Primeiro, pelo lema 5.4, como $\theta(L_{f_{op}/S,m}) = K_r$, então $L_{f_{op}/S,m} \subseteq \theta^{-1}(K_r)$. Segundo, pelo lema 5.2, $L_{f_{op}/S,m}$ é (L_S, Γ_S)-compatível. Terceiro, prova-se que, para todo $K \subseteq \Sigma^*$ que seja (L_S, Γ_S)-compatível e tal que $K \subseteq \theta^{-1}(K_r)$, tem-se $K \subseteq L_{f_{op}/S,m}$. Para isso, considere $t \in \theta(K)$, então $t \in K_r$. Seja também $f_{ge}(t) = (\gamma_r, \#_r) \in \Gamma_P(t)$. Defina-se $v \in \theta^{-1}_{voc}(t)$ e considere-se a linguagem $K(v)$, equação (5.10). Seja $u \in K(v)$, consideram-se dois casos: (i) se $\theta(vu) = t \in K_r$, então $u \in E_v(\gamma_r, \#_r) = L_{S(s)} - \{u \in L_{S,voc}(v) : w(vu) \notin \gamma_r\}$, e (ii) se $\theta(vu) = t\tau \in \overline{K_r}$, então $u \in E_v(\gamma_r, \#_r)$ uma vez que $\gamma_r \cap \Sigma_{L_P}(t) = \Sigma_{K_r}(t)$. Pelo enumerado acima tem-se que $K(v) \subseteq E_v(\gamma_r, \#_r)$. Pelo fato de que K é (L_S, Γ_S)-compatível, $K(v)$ é

$(L_{S(v)}, \Gamma_{S(v)})$-compatível. Pelo lema 5.2, tem-se $K(v) \subseteq \sup C_{(L_{S(v)},\Gamma_{S(v)})}(E_v(\gamma_r, \#_r)) = E_v(\gamma_r, \#_r)^{\uparrow}$. Finalmente, seja $s \in K$ e decomponha-se $s = vu$, onde $v = pre_{voc}(s)$ e $u \in K(v) - L_{S,voc}(v)$. Como $K(v) \subseteq E_v(\gamma_r, \#_r)^{\uparrow}$, onde $(\gamma_r, \#_r) = f_{ge}(t)$ e $t = \theta(s) \in K_r$, tem-se $u \in E_v(\gamma_r, \#_r)^{\uparrow} - L_{S,voc}(v)$. Pela proposição 5.6, $s = vu \in L_{f_{op}/S,m}$. Assim, conclui-se que $K \subseteq L_{f_{op}/S,m}$. A partir dos três pontos acima, conclui-se que $L_{f_{op}/S,m} = \sup C_{(L_S,\Gamma_S)}(\theta^{-1}(K_r))$. □

Pela proposição 5.8, a linguagem resultante no operador para o esquema de supervisão hierárquica proposto é ótima, no sentido de restringir o mínimo possível o comportamento sob supervisão do operador para que se obtenha o comportamento desejado no gerente.

5.5 Método para supervisão hierárquica

Apresenta-se nesta seção um método para supervisão hierárquica de SEDs baseado no presente desenvolvimento.

1. Dados o sistema do operador S sobre o alfabeto Σ, selecione os eventos relevantes para o controle hierárquico $\Sigma_r \subseteq \Sigma$.

2. Construa o sistema do gerente:

   (a) com consistência hierárquica, usando diretamente o algoritmo 5.3, ou

   (b) com consistência hierárquica forte, primeiro tornando o mapa repórter determinista, algoritmo 5.4, e segundo, usando o algoritmo 5.3.

3. Execute a síntese no nível gerencial, algoritmo 4.1, e encontre o supervisor do gerente, f_{ge}, equação (5.7).

4. Construa o supervisor correspondente do operador f_{op}, equação (5.9). Observe que o supervisor do operador é construído simplesmente por combinação dos supervisores locais para os subsistemas que implementam as políticas de con-

trole operacionais correspondentes às diretivas gerenciais, calculados previamente quando da construção do modelo do gerente.

Embora o método acima dirija-se a um esquema de supervisão hierárquica de dois níveis, este é válido também para hierarquias de vários níveis. Primeiro porque decompõe-se o problema de controle de uma hierarquia de vários níveis em problemas locais para dois níveis consecutivos. Segundo, porque o modelo para SEDs com marcação flexível é usado tanto para modelar o sistema do gerente quanto o do operador.

O método proposto foi implementado numa ferramenta computacional para controle hierárquico de SEDs, vide apêndice A.

5.6 Análise de complexidade

Nesta seção apresenta-se a análise de complexidade para o método de supervisão hierárquica proposto.

Para a análise de complexidade, considere que o alfabeto Σ do operador possui e eventos, o sistema do operador S é representado por um autômato G com n estados, o conjunto de eventos relevantes Σ_r possui r eventos, sendo $r \leq e$. Assume-se que o número de estados vocais de G é limitado por n, o número de estados de G. Analisa-se a complexidade em duas etapas, quais sejam, a construção do sistema do gerente e a síntese gerencial. A realização por supervisor no operador da política de controle sintetizada para o gerente não é considerada nesta análise. A razão para isso é que este passo consiste na aplicação, a partir das diretivas do gerente, das políticas de controle locais para os subsistemas do operador, calculadas previamente na construção da estrutura de controle do gerente.

Primeiramente analisa-se a complexidade da construção do sistema do gerente. Para o caso da consistência hierárquica, esta análise comporta a construção do autômato e da estrutura de controle do gerente. A construção do autômato do gerente corresponde a dois passos, como indicado no algoritmo 5.3. O primeiro passo é a projeção do autômato do operador no conjunto de eventos relevantes, operação com complexidade

$\mathcal{O}(n^2)$ (Hopcroft e Ullmann 1979), resultando num autômato não determinista de até n estados (tal como G' no exemplo 5.5). O segundo passo é tornar o autômato resultante determinista, um procedimento com complexidade, no pior caso $\mathcal{O}(2^n)$ (Hopcroft e Ullmann 1979), resultando num sistema com p estados, sendo p limitado por 2^n. A construção da estrutura de controle consiste numa seqüência de testes para cada estado vocal de G, cada teste para verificar se um controle vocal é válido, vide algoritmo 5.2. O número de testes para cada estado é limitado por 2^{r+1}, uma vez que pode haver até r eventos para cada estado. Como no algoritmo 5.2 considera-se apenas o conjunto ativo de eventos relevantes para os estados, e o número de eventos ativos para um dado estado é possivelmente menor que r, o número de testes por estado é potencialmente menor que a estimativa acima. O teste de validade de um controle, corresponde um procedimento de síntese de máxima linguagem compatível seguido de alguns testes, vide algoritmo 5.3. Cada síntese envolve um subsistema, com tamanho limitado por n estados, e uma especificação construída pelo corte de transições do referido subsistema, vide equações (5.3) e (5.4). Assim, a complexidade de cada teste é $\mathcal{O}(en^4)$, vide seção 4.3. Assim, a complexidade completa de construção da estrutura de controle é $\mathcal{O}(2^{r+1}en^5)$, sendo o tamanho do conjunto de controles para cada estado do gerente limitado a 2^{r+1} controles.

Considere-se então a complexidade da construção do sistema do gerente para o caso da consistência hierárquica forte. Neste caso, a análise comporta as etapas de modificação do mapa repórter para torná-lo determinista, a construção do autômato, e a construção da estrutura de controle. A modificação do mapa repórter para torná-lo determinista é um procedimento que, para cada estado vocal de G, constrói a função de transição do gerente, e modifica as etiquetas de eventos para tornar a função de transição determinista, vide algoritmo 5.4. Então, a complexidade deste procedimento é $\mathcal{O}(n^2)$, a mesma que a construção de n subsistemas. O sistema do operador resultante possui n estados, mas o número de eventos relevantes muda para q, limitado por n^2r, a corresponder a uma nova instância de evento relevante para cada estado vocal. O número de eventos do operador passa a ser $e - r + q$. A construção do autômato

do gerente corresponde apenas à projeção do autômato do operador modificado sobre o novo conjunto de eventos relevantes, sendo o autômato resultante já determinista. Assim, esta etapa possui complexidade $\mathcal{O}(n^2)$, tendo o sistema resultante tamanho limitado por n. A construção da estrutura de controle do gerente é equivalente ao caso da consistência hierárquica pura. A diferença é que o número de eventos relevantes é q e o número de eventos do operador é $e - r + q$. A complexidade desta etapa fica sendo então $\mathcal{O}(2^{nr+1}(e - r + q)n^5)$.

Para a análise da complexidade da síntese gerencial, considera-se uma especificação reconhecida por um autômato de m estados. Para o caso da consistência hierárquica, o procedimento de síntese possui complexidade $\mathcal{O}(rp^2m^2)$, e o supervisor resultante possui número de estados limitado por pm. Para o caso de consistência hierárquica forte, a especificação deve ser traduzida para as novas instâncias de eventos relevantes criadas, uma operação que corresponde a uma busca no espaço de estados a partir do estado inicial, modificando-se as etiquetas de transição correspondentes, uma operação de complexidade $\mathcal{O}(m)$ (Hopcroft e Ullmann 1979). Para este caso, a síntese possui complexidade $\mathcal{O}(qn^2m^2)$, e o supervisor resultante possui espaço de estados limitado por nm.

A tabela 5.1 traz um sumário das complexidades envolvidas no método de síntese hierárquica proposto.

Tabela 5.1: Complexidades para o método de síntese hierárquica.

Operação ou tamanho do sistema	Consistência hierárquica	Consistência hierárquica forte
Construção do autômato do gerente	$\mathcal{O}(2^n)$	$\mathcal{O}(2n^2)$
Número de estados do autômato do gerente	$p \leq 2^n$	n
Número de eventos no alfabeto do gerente	r	$q \leq n^2 r$
Construção da estrutura de controle do gerente	$\mathcal{O}(2^{r+1}en^5)$	$\mathcal{O}(2^{q+1}(e - r + q)n^5)$
Síntese gerencial	$\mathcal{O}(rp^2m^2)$	$\mathcal{O}(qn^2m^2)$
Número de estados do supervisor gerencial	pm	nm

Um dos problemas do método de supervisão hierárquica é o número de estados potencial do autômato do gerente, que pode ser maior que o autômato do operador.

Este problema decorre do procedimento de se tornar um autômato determinista e é comum a todas as abordagens de controle hierárquico (Hopcroft e Ullmann 1979). Dentre os trabalhos pesquisados, Wong (1998) trata desta problemática. Segundo Wong (1998), o número de estados do gerente é garantidamente menor que o número de estados do operador quando o mapa repórter for um observador.

Um problema particular a esta abordagem é a complexidade exponencial do cálculo da estrutura de controle do gerente, bem como o número resultante de controles por estado. Propõe-se então uma estratégia para contornar esta complexidade. Ao invés de se calcular a estrutura de controle do gerente no momento da construção do sistema, constrói-se apenas o autômato. Os conjuntos de controle são então calculados concomitantemente aos procedimentos de síntese em que se usa o sistema do gerente. A cada passo do procedimento de síntese, vide algoritmo 4.1, pesquisa-se se um dado estado do gerente possui um dado controle. Consulta-se então uma base de dados com a informação dos controles para o referido estado do gerente, possivelmente incompleta. Se não for encontrado o controle desejado, recorre-se a um procedimento que verifica se este controle é válido para o estado do gerente. O procedimento requer que volte-se ao sistema do operador e façam-se testes se o controle é válido considerando-se todos os estados vocais correspondentes ao estado do gerente em questão, o teste corresponde ao exposto na proposição 5.2. A resposta deste procedimento é se o controle é válido para o estado do gerente ou o maior controle válido para o referido estado que se aproxime do controle desejado. A segunda resposta do procedimento corresponde à busca do controle sub-ótimo na lista de controles do estado, feita no algoritmo 4.1. Por fim, atualiza-se a base de dados de controles para os estados do gerente com o resultado deste procedimento.

Finalmente, afirma-se que o método de síntese hierárquica é mais complexo que o procedimento de síntese monolítica para *uma* especificação, por exemplo, para os parâmetros desta seção a complexidade do procedimento de síntese monolítica é $\mathcal{O}(n^2m^2)$. Entretanto apontam-se duas vantagens potenciais. A primeira é que, para diferentes procedimentos de síntese gerencial, a complexidade total será menor, pa-

gando assim o preço da construção do gerente, que só é feito uma vez. A segunda é que o supervisor resultante é decomposto hierarquicamente em supervisores menores. Isso é equivalente a se comparar um programa de computador monolítico e um programa em subrotinas.

5.7 Exemplo – célula de manufatura

Esta seção tem por objetivo ilustrar o método de supervisão hierárquica proposto com um exemplo de síntese de controle discreto para uma célula de manufatura. Por intermédio deste exemplo, ilustra-se também a decomposição hierárquica de um problema de controle supervisório, a comparação entre o uso da consistência hierárquica e da consistência hierárquica forte na solução de um problema, e a problemática da complexidade computacional para sistemas de grande porte.

A célula de manufatura, conforme descrita em (de Queiroz e Cury 2002b), é composta por uma mesa circular de quatro posições (M_0), vide figura 5.15, onde peças metálicas são furadas e testadas, e quatro outros equipamentos, a saber, uma esteira de entrada (M_1), uma máquina furadeira (M_2), um equipamento de teste (M_3), e um robô manipulador (M_4). Um controlador lógico-programável (CLP) comanda a célula conforme a seguinte seqüência:

1. a esteira avança até que uma peça seja posta em P_1,
2. a mesa gira 90 graus,
3. a peça é furada em P_2,
4. a mesa gira 90 graus,
5. a peça é testada em P_3,
6. a mesa gira 90 graus, e
7. o robô manipulador retira a peça de P_4 e deposita em um dos armazéns de saída.

Adicionalmente, o equipamento de teste classifica a peça em dois tipos, digam-se tipos 1 e 2, e o manipulador deposita a peça em armazéns de saída distintos de acordo com a classificação. Não há sensores que informem a presença de peças especificamente nas distintas posições da mesa.

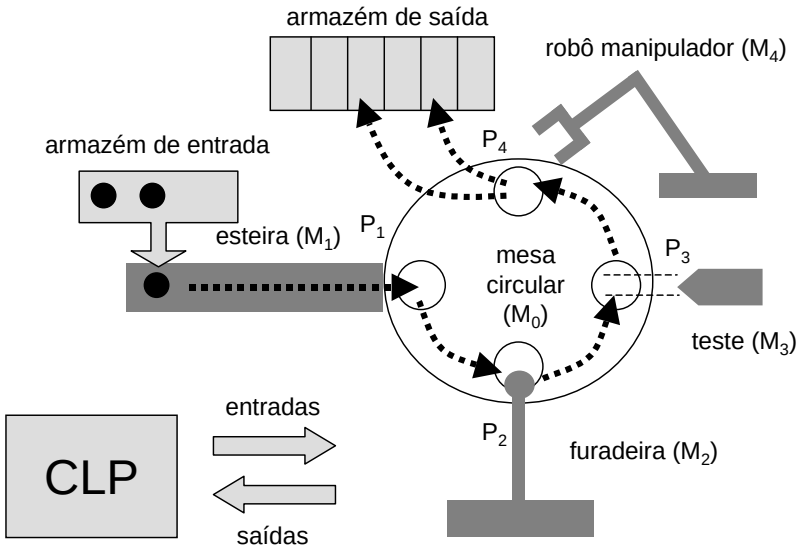


Figura 5.15: Célula de manufatura sobre mesa giratória.

O programa original fornecido pelo fabricante da célula permite a operação seqüencial para apenas uma peça por vez na mesa. Esta estratégia de controle é ineficiente em termos de utilização dos equipamentos da célula de manufatura, pois os equipamentos passam a maior parte do tempo ociosos, quando poderiam estar operando concorrentemente. Em (de Queiroz e Cury 2002b) utiliza-se a teoria de controle supervisório para gerar um programa de controle discreto para a célula. A solução encontrada é ótima em termos de utilização da célula, e a mesa passa a comportar até quatro peças ao mesmo tempo.

A seguir descrevem-se os modelos da teoria de controle supervisório para os componentes da célula. Os modelos são de alto nível, no sentido de que não entram em detalhes dos comandos e das etapas de operação dos equipamentos. Focaliza-se apenas na coordenação do funcionamento concorrente dos equipamentos. Os modelos apre-

sentados são uma extensão dos modelos em (de Queiroz e Cury 2002b), ao incluírem o processamento da informação de teste da peça. Para sintetizar a lógica de coordenação, considere que a operação dos sistemas M_i, com $i \in \{0, 1, 2, 3, 4\}$, inicia com o evento a_i, com $i \in \{0, 1, 2, 3\}$, e finaliza com o evento b_i, $i \in \{0, 1, 2, 4\}$. O equipamento de teste M_3 finaliza a operação com os eventos b_{31}, a significar que a peça é do tipo 1, e b_{32}, a significar que a peça é do tipo 2. O robô manipulador M_4 inicia a operação com o evento a_{41}, se for direcionado para o armazém das peças do tipo 1, e com o evento a_{42}, se for direcionado para o armazém das peças do tipo 2. Os autômatos G_i, representados na figura 5.16, modelam os comportamentos dos equipamentos M_i, com $i \in \{0, 1, 2, 3, 4\}$, respectivamente.

Para gerar a lógica de controle, expressam-se as especificações de coordenação desejadas na forma de linguagens marcadas por autômatos. A especificação E_a, marcada pelo autômato representado na figura 5.16, trata de prevenir que a mesa gire sem que haja uma peça bruta em P_1, uma peça furada em P_2 ou uma peça testada em P_3. As especificações E_{bi}, com $i \in \{1, 2, 3, 4\}$, marcadas pelos autômatos representados na figura 5.16, tratam da exclusão mútua do giro da mesa com o funcionamento dos outros equipamentos, a saber, a esteira de entrada (E_{b1}), a máquina furadeira (E_{b2}), o equipamento de teste (E_{b3}) e o robô manipulador (E_{b4}). As especificações E_{ci}, com $i \in \{1, 2, 3\}$, marcadas pelos autômatos representados na figura 5.16, tratam do fluxo de peças entre as posições consecutivas da mesa, a saber, peças brutas entre P_1 e P_2 (E_{c1}), peças furadas entre P_2 e P_3 (E_{c2}) e peças testadas entre P_3 e P_4 (E_{c3}). As especificações E_{di}, com $i \in \{1, 2\}$, marcadas pelos autômatos representados na figura 5.16, tratam do direcionamento do manipulador com base no resultado do teste da peça que chega em P_4, a saber, se a peça for do tipo 1, é direcionada para o armazém 1 (E_{d1}), e se for do tipo 2, é direcionada para o armazém 2 (E_{d2}). A especificação E_e, marcada pelo autômato representado na figura 5.16, trata do controle de qualidade para o funcionamento da mesa. Supõe-se que peças do tipo 2 são peças furadas de forma incorreta. A especificação E_e expressa que se deve cessar a entrada de peças na mesa se forem produzidas duas peças do tipo 2 consecutivamente. Trata-se de uma especificação de

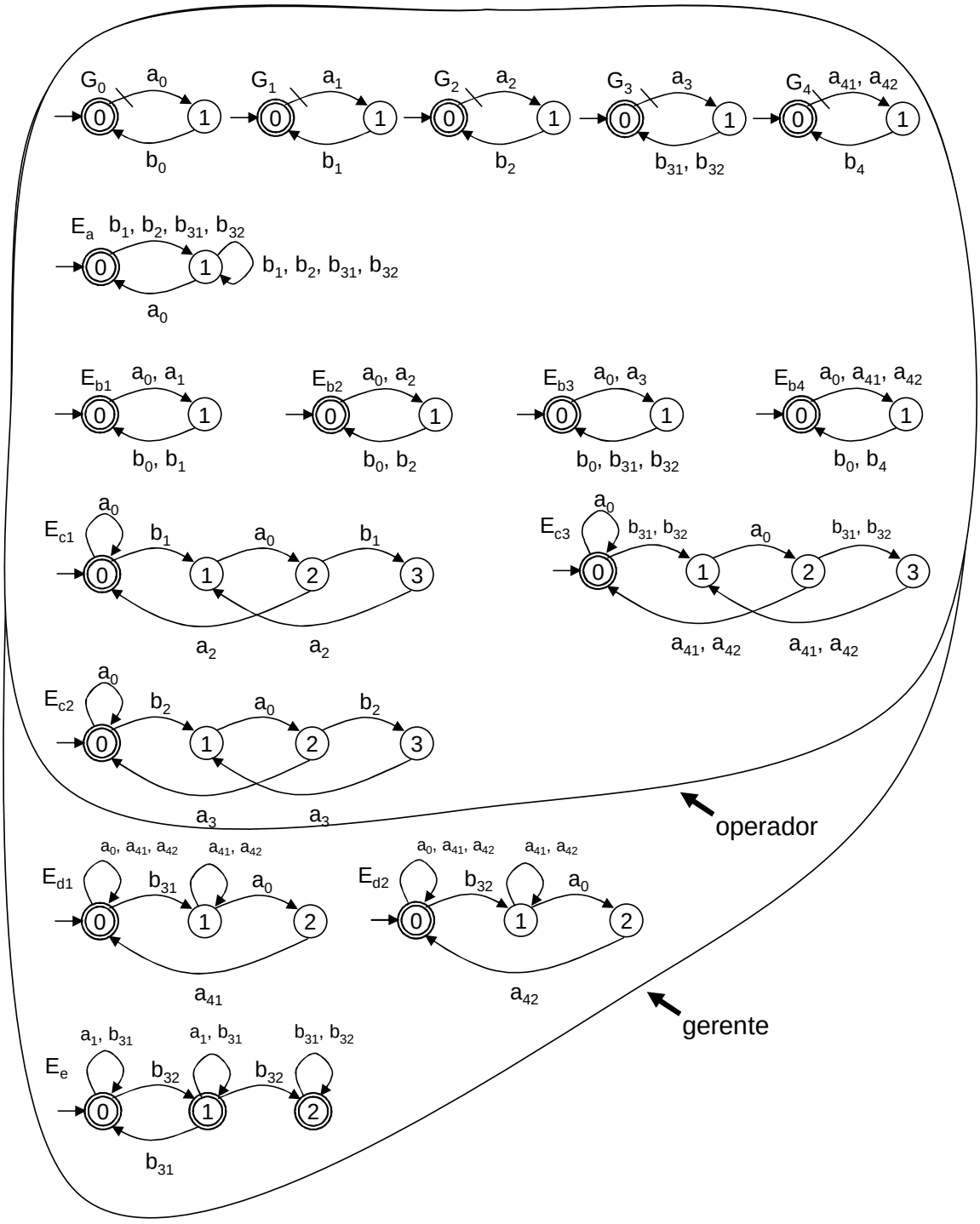


Figura 5.16: Autômatos para a célula de manufatura.

controle de qualidade da produção, e para-se a operação da célula para manutenção, por exemplo.

Considere a síntese do supervisor monolítico para este problema de controle da célula, segundo a abordagem RW padrão. A planta $G = ||_{i=0}^{4} G_i$ possui 32 estados e 192 transições sobre alfabeto $\Sigma = \{a_0, b_0, a_1, b_1, a_2, b_2, a_3, b_{31}, b_{32}, a_{41}, a_{42}, b_4\}$, ao todo 12 eventos. Consideram-se controláveis os eventos de início de operação, a_i, com $i \in \{0, 1, 2, 3, 41, 42\}$, e, não controláveis os eventos de fim de operação, b_i, com $i \in \{0, 1, 2, 31, 32, 4\}$. A especificação de coordenação total E, é obtida pelo produto síncrono das especificações E_a, $E_b = \|_{i=1}^{4} E_{bi}$, $E_c = \|_{i=1}^{3} E_{ci}$, $E_d = \|_{i=1}^{2} E_{ci}$, e E_e. A especificação E é marcada por um autômato de 810 estados e 2014 transições. A máxima linguagem controlável contida em $E\|L_m(G)$ e.r.a. G e Σ_{nc}, $K = \sup C_{(G,\Sigma_{nc})}(E\|L_m(G))$, é reconhecida por um autômato de 327 estados e 712 transições. A linguagem K corresponde ao supervisor ótimo, isto é, o supervisor que restringe o sistema o mínimo possível e ainda atende às especificações de coordenação. Analisando o comportamento do supervisor, observa-se que este admite que até 4 peças estejam ocupando a mesa ao mesmo tempo. Entretanto, o tamanho do supervisor monolítico, tanto em número de estados quanto de transições, dificulta a implementação em um controlador do tipo CLP. Duas razões para isto são, primeiro, o tamanho limitado e possivelmente pequeno de memória disponível, e segundo, a inexistência de um programa de tradução automática do supervisor na forma de um autômato em um programa reconhecido ou executado pelo CLP. Esta transcrição não automatizada do supervisor na lógica de controle pode introduzir erros de programação. Em (de Queiroz e Cury 2002b) propõe-se um método de implementação de programas de controle discreto baseados nos supervisores calculados na teoria de controle supervisório, onde os problemas acima são tratados. Em particular, usa-se em (de Queiroz e Cury 2002b) a teoria de controle modular local (de Queiroz e Cury 2002a).

Nesta seção propõe-se uma abordagem por supervisão hierárquica para o controle da célula. Para a definição do esquema de supervisão hierárquica de dois níveis, dividem-se as especificações de coordenação em um grupo associado a um nível operacional, e

outro associado a um nível gerencial, conforme representado na figura 5.16. O nível operacional contém as especificações de coordenação entre os componentes M_0, M_1, M_2, M_3 e M_4 que tratam das operações que são realizadas sobre uma peça desde sua entrada até sua retirada da mesa. A especificação operacional E' é obtida pelo produto síncrono de E_a, E_b e E_c, representadas na figura 5.16, e é marcada por um autômato de 296 estados e 918 transições, não mostrado. O nível gerencial trata de tomada de decisões sobre o comportamento da mesa a partir do resultado do teste, assim trata do direcionamento da peça classificada para o armazém correspondente e do controle de qualidade de produção. A especificação gerencial resultante E_{ge} é o produto síncrono de E_d e E_e, ambas na figura 5.16, e é reconhecida por um autômato de 15 estados e 47 transições, não mostrado. Assim, propõe-se o esquema de supervisão hierárquica representado na figura 5.17. Os eventos do operador relevantes para as especificações gerenciais são $\Sigma_r = \{a_0, a_1, b_{31}, b_{32}, a_{41}, a_{42}\}$.

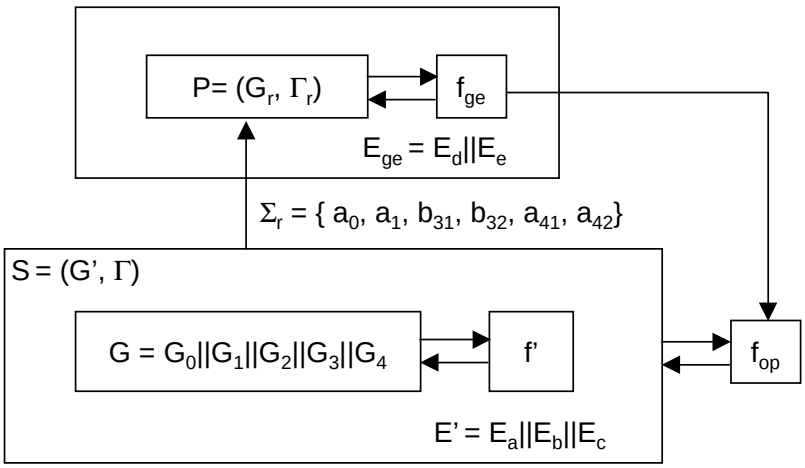


Figura 5.17: Esquema de supervisão hierárquica para a célula de manufatura

O problema de controle operacional é resolvido calculando-se a máxima linguagem controlável e.r.a. G e Σ_{nc} contida em $E'\|L_{G,m}$, $K' = \sup C_{(G,\Sigma_{nc})}(E'\|L_{G,m})$, marcada por um autômato de 151 estados e 434 transições, não mostrado. A linguagem K' é implementada pelo supervisor f' na figura 5.17. Define-se o sistema em malha fechada no nível operacional como sendo $S = (G', \Gamma)$, onde G' é um autômato e Γ é uma estrutura

de controle dependente do estado. O autômato G' é tal que $L_{G'} = \overline{K_{op}}$ e $L_{G',m} = \emptyset$, e é obtido a partir do autômato que marca K_{op}, e Γ é construído pela transcrição da estrutura de controle padrão RW definida por Σ_{nc} e K_{op} para a estrutura de controle do capítulo 4 (vide exemplo 4.6). O sistema gerencial $P = (G_r, \Gamma_r)$ que possui consistência hierárquica com S e Σ_{ge} é obtido pelo algoritmo 5.3. O autômato G_r possui 24 estados e 59 transições, e Γ_r é uma estrutura de controle dependente do estado que não corresponde a um modelo RW padrão, possuindo, no máximo, 6 controles por estado. O supervisor para o problema de controle gerencial corresponde à máxima linguagem (G_r, Γ_r)-compatível contida em $E_{ge} \| L_{G_r}$, $K_{ge} = \sup C_{(G_r,\Gamma_r)}(E_{ge} \| L_{G_r})$, reconhecida por um autômato de 29 estados e 39 transições. O esquema de supervisão hierárquica corresponde ao supervisor gerencial f_{ge}, implementado por um autômato que marque a linguagem K_{ge}, e o supervisor do operador correspondente f_{op}, construído por concatenação dos supervisores para subsistemas correspondentes, vide equação (5.9), calculados previamente quando da construção de G_r e Γ_r.

O esquema de supervisão hierárquica tem a vantagem de poder ser implementado em processadores separados. Por exemplo, f_{ge} pode ser implementado em um computador que cuide não só o funcionamento do CLP da célula em questão, como também de outras células de manufatura na mesma fábrica. Enquanto que f_{op} pode ser implementado no CLP da célula na forma de supervisores separados para cada subsistema de S, correspondendo a diferentes estágios da operação, ativados pelo comando do supervisor que está no computador remoto.

Calcula-se o supervisor monolítico correspondente ao esquema de supervisão hierárquica. De acordo com a proposição 5.8, este supervisor corresponde à linguagem $K_{op} = \sup C_{(L_{G'},\Gamma)}\ (K_{ge} \| L_{G'})$, marcada por um autômato de 85 estados e 121 transições. Ao se comparar K_{op} e K, a linguagem para o supervisor monolítico, verifica-se que $K_{op} \subset K$, isto é, o resultado da supervisão hierárquica é mais restritivo que a supervisão monolítica. Por exemplo, ao analisar-se o comportamento da célula apenas por entrada e saída de peças, enquanto K admite até quatro peças ao mesmo tempo na mesa, K_{op} admite somente até duas peças. Isto se dá porque o esquema de su-

pervisão hierárquica não foi construído para haver consistência hierárquica forte. No caso de se desejar otimalidade no resultado, deve-se construir o esquema de supervisão hierárquica com consistência hierárquica forte, mas, como é visto a seguir, o preço a pagar é o aumento da complexidade computacional de construção do sistema do gerente.

Pode-se verificar que o mapa repórter associado a $L_{G'}$ e Σ_r é não determinista. Para obter o esquema de supervisão hierárquica com consistência hierárquica forte, o primeiro passo é executar o procedimento de tornar o mapa repórter determinista, algoritmo 5.4. Por aplicação deste algoritmo, foram gerados $S' = (G'', \Gamma')$ e Σ'_r tais que o mapa repórter associado é determinista. O novo conjunto de eventos relevantes Σ'_r possui 222 eventos. O algoritmo 5.4 não minimiza o número de novas instâncias de eventos geradas, nem para o sistema como um todo nem para um dado estado vocal. O novo conjunto de eventos relevantes criado é tal que a modificação do conjunto ativo de eventos relevantes para um dado estado é suficiente para garantir o determinismo do mapa repórter. A construção do sistema gerencial P' correspondente a S' e Σ'_r é computacionalmente complexa em virtude do número de instâncias de eventos criadas. O problema é gerado pela complexidade exponencial da construção da estrutura de controle do gerente em função do número de eventos relevantes. Por exemplo, encontram-se para um dado estado vocal, um conjunto ativo de eventos relevantes com 30 eventos, o que corresponde a um teste de 2^{31} controles. O problema não é só a complexidade do cálculo, mas o potencial número de controles possíveis para um dado estado. Uma tentativa de cálculo foi realizada num computador pessoal de configuração Pentium III, com 1GHz de relógio, 256Mb de RAM e com sistema operacional Linux, o cálculo foi terminado pelo sistema por violar condições de gerenciamento de memória depois de três dias corridos ocupando a máquina o tempo todo (a máquina ficou isolada só para executar o cálculo). Por esta experiência, conclui-se que a melhor forma de trabalhar com a consistência hierárquica forte para sistemas de grande porte é obterem-se os controles do nível gerencial concomitantemente aos procedimentos de síntese gerencial, conforme sugerido no final da seção 5.6. Ainda que se minimize o

número de instâncias de eventos criadas no procedimento de tornar o mapa repórter determinista, o número de controles por estado no nível gerencial permanece variando exponencialmente com o número de eventos relevantes.

Ao se comparar esta abordagem hierárquica com a abordagem modular em (de Queiroz e Cury 2002b) para este problema da célula de manufatura, verifica-se uma desvantagem na abordagem hierárquica. A desvantagem reside na ordem de grandeza dos supervisores obtidos e o resultado não ótimo da supervisão hierárquica quando não há consistência hierárquica forte. De forma geral, a abordagem modular local em (de Queiroz e Cury 2002b) gera supervisores pequenos para sistemas cujos componentes e especificações possam ser decompostos em módulos, e quando a condição de modularidade local é aceita, como é o caso deste exemplo de célula de manufatura (de Queiroz e Cury 2002b). Entretanto, afirma-se que a supervisão hierárquica é uma opção razoável quando não se pode decompor o sistema em módulos de acordo com as especificações e componentes, ou quando as condições de modularidade não são atendidas. E ainda, sistemas realmente de grande porte, onde hierarquias de controle surjam naturalmente, são uma potencial aplicação da supervisão hierárquica. Como ilustração, sugere-se o cenário onde se deve coordenar um conjunto de células de manufatura do tipo da que é apresentada neste exemplo, no caso realista de uma fábrica com manufatura flexível.

Os cálculos deste exemplo de controle hierárquico para uma célula de manufatura foram realizados utilizando-se a ferramenta para controle hierárquico de SEDs desenvolvida neste trabalho, vide apêndice A. O leitor interessado em reproduzir o exemplo pode acessar os arquivos referentes ao exemplo na página `http://www.das.ufsc.br/~aecc`.

5.8 Conclusões

Este capítulo apresenta uma abordagem de controle hierárquico para SEDs. Pela aplicação do SED com marcação flexível, um esquema de supervisão hierárquica de dois níveis com consistência hierárquica é proposto. A extensão necessária para se ter

consistência hierárquica forte também é proposta, e por fim enuncia-se um método para construção de hierarquias de controle consistentes para SEDs. Aspectos computacionais dos algoritmos propostos são tratados, e ilustra-se a abordagem com um exemplo baseado numa aplicação real de uma célula de manufatura.

As vantagens da abordagem apresentada, quando comparada com as abordagens relacionadas na literatura, como Zhong e Wonham (1990), Wong e Wonham (1996a), Pu (2000) e Hubbard e Caines (2002), são:

- a proposta de um esquema de supervisão hierárquica de dois níveis sem a necessidade das condições impostas pelas outras abordagens;
- a existência de um método para a construção de um esquema de supervisão hierárquica com consistência hierárquica, sendo inclusive desenvolvida uma ferramenta computacional;
- a disponibilidade de métodos para consistência hierárquica e consistência hierárquica forte;
- a proposta de decomposição do supervisor para o operador em supervisores menores para os subsistemas; e
- a preservação da semântica dos eventos do nível gerencial no esquema de supervisão hierárquica de dois níveis, mesmo no caso da renomeação de eventos.

O preço a pagar é a utilização do SED com marcação flexível, que não segue ao padrão da abordagem RW, mas que mostra-se bem apropriado para sistemas com decomposição hierárquica. Em contrapartida, o uso do SED com marcação flexível permite o tratamento de hierarquias de mais de dois níveis.

Considere agora o problema de se construir um esquema de supervisão hierárquica para um sistema composto, isto é, um sistema formado por diversos componentes que evoluam paralelamente, possivelmente com alguma sincronização de eventos. Na presente abordagem, para se construir um esquema de supervisão hierárquica para tal

sistema, deve-se construir primeiro o sistema composto, para então tratar da construção dos sistemas com nível de abstração mais alto. O problema é que o número de estados do sistema composto cresce exponencialmente com o número de componentes (Ramadge e Wonham 1989), acarretando aumento da complexidade da construção do esquema de supervisão hierárquica. É desejável então, construir-se a abstração do sistema composto sem ter de construir o sistema composto propriamente dito antes. É disso que trata o próximo capítulo. Algumas alternativas de construção de tal abstração de alto nível são consideradas, e um método inédito, baseado no raciocínio supor-garantir, é proposto para a construção de tal abstração por composição.

Capítulo 6

O controle hierárquico de sistemas compostos

Neste capítulo considera-se a construção de um esquema de controle hierárquico onde o sistema do operador é um sistema composto. Propõe-se um método baseado no raciocínio supor-garantir para se construir o sistema do gerente.[1]

A organização deste capítulo é a seguinte. Na seção 6.1 definem-se algumas novas operações para o SED com marcação flexível, dando-se ênfase à operação chamada *produto síncrono*, que corresponde à composição dos sistemas. Na seção 6.2 apresentam-se o problema de controle hierárquico sob a ótica de *abstrações consistentes*, um conceito útil para expressar a consistência hierárquica para sistemas compostos. Na seção 6.3 discutem-se os problemas principais da composição de abstrações consistentes e apresenta-se um resultado representativo da literatura traduzido para o contexto dos SEDs com marcação flexível. Na seção 6.4 introduz-se um método de composição de abstrações consistentes tendo como base o raciocínio de supor-garantir. Na seção 6.5 apresenta-se um exemplo ilustrativo para a abordagem proposta baseado numa aplicação real de piloto automático inteligente para automóveis. Finalmente, na seção 6.6 resumem-se as principais contribuições deste capítulo e tecem-se comentários sobre a abordagem proposta.

[1]Este capítulo corresponde a uma apresentação detalhada dos resultados de (da Cunha et al. 2002).

6.1 Algumas operações para os SEDs com marcação flexível

Nesta seção, apresentam-se três operações para os SEDs com marcação flexível que são utilizadas neste capítulo.

A primeira operação a ser definida chama-se produto síncrono e será aplicada como forma de composição de SEDs com marcação flexível.

Definição 6.1 (Produto síncrono) *Dados os SEDs com marcação flexível S_1 sobre o alfabeto Σ_1 e S_2 sobre o alfabeto Σ_2, o produto síncrono de S_1 e S_2, denotado $S_1 \| S_2$, é um SED com marcação flexível S sobre o alfabeto $\Sigma_1 \cup \Sigma_2$ para o qual $L_S = L_{S_1} \| L_{S_2}$ e, para $s \in L_S$,*

$$\begin{aligned}\Gamma_S(s) \;=\; \{(\gamma,\#) \in 2^{\Sigma_{L_S}(s)} \times \{M,N\} :\; & (\exists(\gamma_i,\#_i) \in \Gamma_{S_i}(p_i(s)),\ i \in \{1,2\}) \text{ tais que} \\ & (\gamma \cap (\Sigma_1 - \Sigma_2) = (\gamma_1 - \gamma_2) \cap \Sigma_{L_S}(s)) \text{ e} \\ & (\gamma \cap (\Sigma_2 - \Sigma_1) = (\gamma_2 - \gamma_1) \cap \Sigma_{L_S}(s)) \text{ e} \\ & (\gamma \cap (\Sigma_1 \cap \Sigma_2) = (\gamma_1 \cap \gamma_2) \cap \Sigma_{L_S}(s)) \text{ e} \\ & (\# = \#_1 \wedge \#_2)\}\end{aligned} \tag{6.1}$$

onde p_i denota a projeção de palavras de Σ^* em Σ_i^*, $i \in \{1,2\}$.

No produto síncrono $S = S_1 \| S_2$, a linguagem L_S é o produto síncrono das linguagens L_{S_1} e L_{S_2}, e o controle $(\gamma, \#)$ é válido para $s \in L_S$ se e somente se existirem controles $(\gamma_1, \#_1)$ e $(\gamma_2, \#_2)$ para as palavras correspondentes a s em S_1 e S_2, respectivamente, tais que: (i) para os eventos ativos em S que estejam em S_1 e que não estejam em S_2, o resultado da ação de controle de γ e γ_1 deve ser o mesmo; (ii) idem para os eventos ativos em S que estejam em S_2 e não estejam em S_1 e as ações de controle de γ e γ_2; (iii) para os eventos ativos em S compartilhados por S_1 e S_2, o resultado da ação de controle de γ_1, γ_2 e γ deve ser o mesmo; e (iv) o atributo de marcação $\#$ é o resultado do *e lógico* dos atributos $\#_1$ e $\#_2$.

Exemplo 6.1 (Produto síncrono) *Para ilustrar o produto síncrono considere o seguinte cenário: sejam S_1 um SED com marcação flexível sobre o alfabeto $\Sigma_1 = \{a, b, c, d\}$ e S_2 sobre o alfabeto $\Sigma_2 = \{a, b, c, e\}$. Sejam também s_1 em L_{S_1} e s_2 em L_{S_2} tais que os conjuntos ativo de eventos sejam $\Sigma_{L_{S_1}}(s_1) = \{a, b, d\}$ e $\Sigma_{L_{S_2}}(s_2) = \{a, c, e\}$, e os conjuntos de controles vocais sejam $\Gamma_{S_1}(s_1) = \{(\{a, b\},\ N), (\{a, b, d\}, N)\}$ e $\Gamma_{S_2}(s_2) = \{(\{\}, M),\ (\{a, e\}, M),\ (\{a, c, e\}, M)\}$. Para a palavra s em $S_1 \| S_2$ que corresponde ao par (s_1, s_2) o conjunto ativo de eventos é $\Sigma_{L_S}(s) = \{a, d, e\}$ e o conjunto de controles é $\Gamma_S(s) = \{(\{a, e\}, N), (\{a, e, d\}, N)\}$.*

Pode-se mostrar que o produto síncrono para os SEDs com marcação flexível aqui definido é associativo e comutativo, e é uma generalização do produto síncrono de SEDs da abordagem de RW. As operações definidas a seguir são usadas na aplicação do conceito de abstração consistente, apresentado na seção 6.2 para o controle hierárquico de sistemas compostos.

Definição 6.2 (Comparação) *Dados dois SEDs com marcação flexível S_1 e S_2 ambos sobre o alfabeto Σ, diz-se que S_1 está contido em S_2, denotado $S_1 \leq S_2$, se $L_{S_1} \subseteq L_{S_2}$ e, para todo $s \in L_{S_1}$ e $(\gamma_1, \#_1) \in \Gamma_{S_1}(s)$, existe $(\gamma_2, \#_2) \in \Gamma_{S_2}(s)$ tal que $\gamma_1 \cap \Sigma_{L_{S_1}}(s) = \gamma_2 \cap \Sigma_{L_{S_2}}(s)$ e $\#_1 = \#_2$.*

Para o caso particular em que $L_{S_1} = L_{S_2}$ e, para todo $s \in L_{S_i}$, com i igual a 1 ou 2, e $\Gamma_{S_i}(s) \subseteq 2^{\Sigma_{L_{S_i}}(s)} \times \{M, N\}$, a comparação entre os conjuntos de controles, na definição 6.2, simplifica-se para $\Gamma_{S_1}(s) \subseteq \Gamma_{S_2}(s)$, para todo $s \in L_{S_i}$. É óbvio provar-se que se $S_1 \leq S_2$ e $K \in \Sigma^*$ é (L_{S_1}, Γ_{S_1})-compatível, então K é (L_{S_2}, Γ_{S_2})-compatível.

Definição 6.3 (União) *Dados os SEDs com marcação flexível S_1 sobre o alfabeto Σ_1 e S_2 sobre o alfabeto Σ_2, a união $S_1 \cup S_2$ é um SED com marcação flexível S sobre o alfabeto $\Sigma = \Sigma_1 \cup \Sigma_2$, onde $L_S = L_{S_1} \cup L_{S_2}$, e, para todo $s \in L_S$, $\Gamma_S(s) = \Gamma_{S_1}(s)$, se $s \in L_{S_1} - L_{S_2}$; $\Gamma_S(s) = \Gamma_{S_2}(s)$, se $s \in L_{S_2} - L_{S_1}$; e $\Gamma_S(s) = [\Gamma_1(s) \cup \Gamma_2(s)]_\cup$, se $s \in L_{S_1} \cap L_{S_2}$, onde $[\Gamma]_\cup$ denota o fechamento em relação à união do conjunto $\Gamma \subseteq 2^\Sigma \times \{M, N\}$, definido por $[\Gamma]_\cup = \{(\gamma, \#) \in 2^\Sigma \times \{M, N\} : ((\gamma, \#) \in \Gamma)$ ou $((\exists (\gamma_1, \#_1)$ e $(\gamma_2, \#_2) \in \Gamma)$ tais que $(\gamma = \gamma_1 \cup \gamma_2)$ e $(\# = \#_1 \vee \#_2))\}$.*

Para o caso que $L_{S_1} = L_{S_2}$ e, para todo $s \in L_{S_i}$, com i igual a 1 ou 2, $\Gamma_{S_i}(s) \subseteq 2^{\Sigma_{L_{S_i}}(s)} \times \{M, N\}$, a união $S = S_1 \cup S_2$ simplifica-se para $L_S = L_{S_1} = L_{S_2}$ e, para todo $s \in L_S$, $\Gamma_S(s) = [\Gamma_{S_1}(s) \cup \Gamma_{S_2}(s)]_{\cup}$. É óbvio provar que $S_i \leq S_1 \cup S_2$, para $i \in \{1, 2\}$.

6.2 Abstrações consistentes em controle hierárquico

Esta seção introduz o conceito de abstração consistente como solução para se obter uma hierarquia com consistência hierárquica.

Definição 6.4 (Abstração consistente) *Dado um SED com marcação flexível S sobre o alfabeto Σ e o conjunto de eventos relevantes $\Sigma_r \subseteq \Sigma$, com mapa repórter associado θ, uma abstração consistente de S em relação a Σ_r é um SED com marcação flexível P sobre o alfabeto Σ_r para o qual: (i) $L_P = \theta(L_S)$ e (ii) para todo $t \in L_P$, se $(\gamma, \#) \in \Gamma_P(t)$, então, para todo $s \in \theta^{-1}_{voc}(t)$, existe $(\gamma', \#') \in \Gamma_{S,voc}(s)$ tal que $\gamma \cap \Sigma_{S,voc}(s) = \gamma'$ e $\# = \#'$.*

A definição 6.4 é uma extensão da definição de abstrações consistentes de Pu (2000) para os SEDs com marcação flexível, vide capítulo 3. As abstrações consistentes são usadas para se obter uma hierarquia com consistência hierárquica, como mostra a proposição 6.1.

Proposição 6.1 (Abstrações consistentes e consistência hierárquica) *Dados o SED com marcação flexível S sobre o alfabeto Σ, o conjunto de eventos relevantes $\Sigma_r \subseteq \Sigma$ e o SED com marcação flexível P sobre o alfabeto Σ_r, P é uma abstração consistente de S e.r.a. Σ_r se, e somente se, há consistência hierárquica entre S e P.*

Demonstração. (**Se**) Suponha que há consistência hierárquica ente S e P. Considere a seguinte expressão local para a consistência hierárquica: para todo subsistema $S(v)$, onde $v \in L_{S,voc}$, e $(\gamma_r, \#_r) \in \Gamma_P(\theta(v))$ defina-se $K = \sup C_{(L_{S(v)}, \Gamma_{S(v)})}(E_s(\gamma_r, \#_r))$.

Pela hipótese de consistência hierárquica, (i) K é não vazia e $(L_{S(v)}, \Gamma_{S(v)})$-compatível; (ii) para todo $u \in L_{S,voc}(v)$, se $w(vu) \in \gamma_r$, então $u \in K$; (iii) se $\#_r = N$, então $K - L_{S,voc}(v) = \emptyset$; e (iv) se $\#_r = M$, então $K - L_{S,voc}(v) \neq \emptyset$. Seja $(\gamma'_r, \#'_r)$ o controle em $\Gamma_{S,voc}(v)$ correspondente a K, isto é, $(\gamma'_r, \#'_r)$ é tal que $\gamma'_r = \gamma_r \cap \Sigma_{S,voc}(v)$ e $\#'_r = \#_r$. Assim, P é uma abstração consistente de S.

(**Somente se**) Pelo teorema 5.1. □

Proposição 6.2 (Existência de abstrações consistentes) *Para o SED com marcação flexível S sobre o alfabeto Σ e o conjunto de eventos relevantes $\Sigma_r \subseteq \Sigma$, o conjunto das abstrações consistentes de S e.r.a. Σ_r é não vazio e fechado a uniões arbitrárias de SEDs com marcação flexível.*

Demonstração. Primeiro, seja o SED com marcação flexível P sobre o alfabeto Σ_r definido por $L_P = \theta(L_S)$ e, para todo t em L_P, $\Gamma_P(t) = \emptyset$. Por inspeção da definição de abstração consistente, prova-se que P acima definido é uma abstração consistente de S e.r.a. Σ_r. Trata-se do caso trivial onde nenhum controle está disponível.

Segundo, sejam P_1 e P_2 abstração consistentes de S e.r.a. Σ_r. Defina-se $P = P_1 \cup P_2$. Então $L_P = L_{P_1} = L_{P_2} = \theta(L_S)$ e, para todo t em L_P, $\Gamma_P(t) = [\Gamma_{P_1}(t) \cup \Gamma_{P_2}(t)]_\cup$. Agora sejam $t \in L_P$ e $(\gamma_r, \#_r) \in \Gamma_P(t)$, considere os seguintes casos:

Caso 1 – Se $(\gamma_r, \#_r)$ está em $\Gamma_{P_1}(t)$ ou $\Gamma_{P_2}(t)$. Então, para todo $s \in \theta^{-1}_{voc}(t)$, existe $(\gamma'_r, \#'_r) \in \Gamma_{S,voc}(s)$ tal que $\gamma'_r = \gamma_r \cap \Sigma_{S,voc}(v)$ e $\#'_r = \#_r$.

Caso 2 – Se $(\gamma_r, \#_r)$ é tal que há $(\gamma_{r1}, \#_{r1}) \in \Gamma_{P_1}(t)$ e $(\gamma_{r2}, \#_{r2}) \in \Gamma_{P_2}(t)$ tais que $\gamma_r = \gamma_{r1} \cup \gamma_{r2}$ e $\#_r = \#_{r1} \vee \#_{r2}$. Uma vez que $(\gamma_{ri}, \#_{ri}) \in \Gamma_{P_i}(t)$, $i \in \{1, 2\}$, então, para todo $s \in \theta^{-1}_{voc}(t)$, há $(\gamma'_{ri}, \#'_{ri}) \in \Gamma_{S,voc}(s)$ tal que $\gamma'_{ri} = \gamma_{ri} \cap \Sigma_{S,voc}(s)$ e $\#'_{ri} = \#_{ri}$. Considere então para $s \in \theta^{-1}_{voc}(t)$ o controle $(\gamma'_r, \#'_r) = (\gamma'_{r1} \cup \gamma'_{r2}, \#'_{r1} \vee \#'_{r2})$. Pelo fechamento em relação à união do conjunto de controles vocais $\Gamma_{S,voc}(s)$, proposição 5.1, $(\gamma'_r, \#'_r) \in \Gamma_{S,voc}(s)$. Então, para todo $s \in \theta^{-1}_{voc}(t)$ existe $(\gamma'_r, \#'_r) \in \Gamma_{S,voc}(s)$ tal que $\gamma'_r = \gamma'_{r1} \cup \gamma'_{r2} = (\gamma_{r1} \cap \Sigma_{S,voc}(s)) \cup (\gamma_{r2} \cap \Sigma_{S,voc}(s)) = (\gamma_{r1} \cup \gamma_{r2}) \cap \Sigma_{S,voc}(s) = \gamma_r \cap \Sigma_{S,voc}(s)$ e $\#'_r = \#'_{r1} \wedge \#'_{r2} = \#_{r1} \wedge \#_{r2} = \#_r$.

Pela análise dos dois casos acima, P é uma abstração consistente de S e.r.a. Σ_r. □

A proposição 6.2 garante a existência de uma abstração consistente dado um sistema S e um conjunto de eventos relevantes Σ_r. Este resultado difere de Pu (2000) que vincula a existência de abstrações consistentes à condição de que o mapa repórter seja um observador fraco, vide capítulo 3. Esta condição não é necessária para a presente abordagem pela possibilidade de existência de um SED com marcação flexível com conjunto de controles vazio e a pela própria definição do conjunto de controles vocais, definição 5.2. A proposição 6.2 fornece também uma propriedade do conjunto de abstrações consistentes para um dado sistema e um conjunto de eventos relevantes que é explorada para se construir uma abstração consistente na proposição 6.3.

Proposição 6.3 (Construção de abstrações consistentes) *Para o SED com marcação flexível S sobre o alfabeto Σ e o conjunto de eventos relevantes $\Sigma_r \subseteq \Sigma$, com mapa repórter associado θ, existe uma máxima abstração consistente de S e.r.a. Σ_r, denotada $\langle S, \Sigma_r \rangle$, e dada por $L_{\langle S,\Sigma_r\rangle} = \theta(L_S)$ e por*

$$\begin{aligned}\Gamma_{\langle S,\Sigma_r\rangle}(t) \;=\; \{(\gamma,\#) \in 2^{\Sigma_r} \times \{M,N\} :\; & (\forall s \in \theta^{-1}_{voc}(t))\ (\exists(\gamma',\#') \in \Gamma_{S,voc}(s)) \textit{ tais que} \\ & ((\gamma \cap \Sigma_{S,voc}(s) = \gamma')\ \textit{e}\ (\#' = \#))\}.\end{aligned} \tag{6.2}$$

para todo t em $L_{\langle S,\Sigma_r\rangle}$.

Demonstração. Pelo fato do conjunto das máximas abstrações consistentes de S e.r.a. Σ_r ser não vazio e fechado em relação à união de SEDs com marcação flexível, proposição 6.2, existe uma máxima abstração consistente de S e.r.a. Σ_r, o supremo do conjunto, dada por $\langle S, \Sigma_r \rangle = \bigcup_i P_i$, onde P_i é uma abstração consistente de S e.r.a. Σ_r. Pela definição de abstração consistente, $L_{\langle S,\Sigma_r\rangle} = \theta(L_S)$ e

$$\begin{aligned}\Gamma_P(t) \;&=\; [\textstyle\bigcup_i \Gamma_{P_i}(t)]_\cup && (\text{para } P_i \text{ abstração consistente de } S \text{ e.r.a. } \Sigma_r) \\ &=\; \{(\gamma,\#) \in 2^{\Sigma_r} \times \{M,N\} : && (\forall s \in \theta^{-1}_{voc}(t))\ (\exists(\gamma',\#') \in \Gamma_{S,voc}(s)) \text{ tais que} \\ & && ((\gamma \cap \Sigma_{S,voc}(s) = \gamma') \text{ e } (\#' = \#))\},\end{aligned}$$

para todo t em $L_{\langle S,\Sigma_r\rangle}$. □

A máxima abstração consistente $\langle S, \Sigma_r \rangle$ é máxima no sentido de que para toda abstração consistente de S e.r.a. Σ_r, $P_i \leq \langle S, \Sigma_r \rangle$.

Exemplo 6.2 (Máxima abstração consistente) *Na figura 6.1, seja S um SED com marcação flexível sobre o alfabeto $\Sigma = \{a, b, c, \alpha, \beta, \mu\}$. Considere que S possui estrutura de controle padrão da abordagem RW com controle supervisório marcador (Wonham 2002b). O atributo de controle dos eventos e o atributo de marcação dos estados estão indicados na estrutura de transição da figura 6.1. Para $\Sigma_r = \{a, b, c\}$, pode-se verificar que P_1 e P_2 na figura 6.1 são abstrações consistentes de S e.r.a. Σ_r. Ainda, pode-se verificar que P_1 é a máxima abstração consistente de S e.r.a. Σ_r, isto é, $P_1 = \langle S, \Sigma_r \rangle$.*

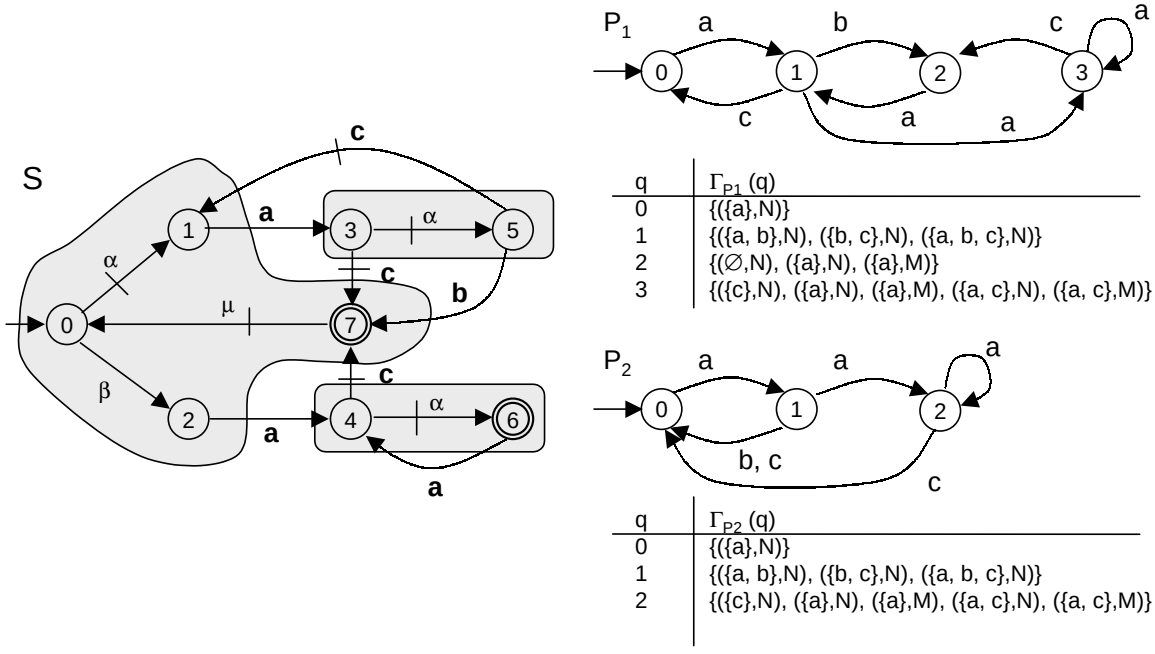


Figura 6.1: Exemplo de abstrações consistentes.

6.3 Composição de abstrações consistentes

Esta seção apresenta resultados para a construção de abstrações consistentes por composição.

O objetivo é construir uma abstração consistente para controle hierárquico de um sistema composto $S = S_1 \| \ldots \| S_n$ sem ter que se obter o sistema S pela composição propriamente dita. Considera-se o caso em que $n = 2$, na seção 6.6 serão tecidos comentários para $n > 2$. Sejam S_1 e S_2 SEDs com marcação flexível sobre o alfabeto Σ_1 e Σ_2, respectivamente, e $S = S_1 \| S_2$ o sistema composto resultante sobre o alfabeto $\Sigma = \Sigma_1 \cup \Sigma_2$. Considera-se que S é um *sistema produto* (Ramadge e Wonham 1989), isto é, $\Sigma_1 \cap \Sigma_2 = \emptyset$. Embora esta seja uma hipótese relativamente restritiva, os sistemas produto representam uma classe razoável dos sistemas do mundo real.

Sejam $\Sigma_r \subseteq \Sigma$ o conjunto de eventos relevantes e $\Sigma_{ir} = \Sigma_i \cap \Sigma_r$ o conjunto de eventos relevantes para cada sistema S_i, i sendo 1 ou 2. Considere o seguinte cenário de composição de abstrações consistentes para os SEDs S_1 e S_2: sejam P_1 uma abstração consistente de S_1 e.r.a. Σ_{1r}, P_2 uma abstração consistente de S_2 e.r.a. Σ_{2r}, e P a máxima abstração consistente de S e.r.a. Σ_r, isto é, $P = \langle S, \Sigma_r \rangle$. Alguma notação torna-se necessária neste ponto. Para $i, j \in \{1, 2\}$ com $i \neq j$, seja $p_i : \Sigma^* \to \Sigma_i^*$ a projeção de palavras sobre Σ em palavras sobre Σ_i; $l_i : \Sigma_r^* \to \Sigma_{ri}^*$ a projeção de palavras sobre Σ_r em palavras sobre Σ_{ri}; $\theta : L_S \to \Sigma_r^*$ o mapa repórter associado a S e Σ_r; e $\theta_i : L_{S_i} \to \Sigma_{ir}^*$ o mapa repórter associado a S_i e Σ_{ir}. As linguagens de P e $P_1 \| P_2$ são iguais, como pode ser visto fazendo-se $L_P = \theta(L_{S_1} \| L_{S_2}) = \theta(p_1^{-1}(L_{S_1}) \cap p_2^{-1}(L_{S_2}))$ $= l_1^{-1}(\theta_1(L_{S_1})) \cap l_2^{-1}(\theta_2(L_{S_2})) = L_{P_1} \| L_{P_2}$. Entretanto, pode haver $t \in L_P$ para o qual $(\gamma_r, \#_r) \in \Gamma_{P_1 \| P_2}(t)$ e $(\gamma_r, \#_r) \notin \Gamma_P(t)$. Assim, $P_1 \| P_2$ pode não ser uma abstração consistente de S e.r.a. Σ_r, como ilustrado pelo exemplo 6.3.

Exemplo 6.3 (Composição de abstrações consistentes) *Considere os SEDs S_1, S_2 e $S = S_1 \| S_2$ mostrados na figura 6.2. Os sistemas são dotados de estrutura de controle padrão RW, conforme indicado. A figura 6.2 também mostra P, a máxima abstração consistente de S e.r.a. $\Sigma_r = \{b, d\}$; P_1 e P_2, as máximas abstrações consis-*

tentes de S_1 e S_2 e.r.a. $\Sigma_{1r} = \{b\}$ e $\Sigma_{2r} = \{d\}$, respectivamente; e o produto síncrono $P_1 \| P_2$. Observe que $L_P = L_{P_1 \| P_2}$, mas, para o estado 0 de $P_1 \| P_2$, os controles $(\{d\}, M)$ e $(\{d\}, N)$ são válidos, enquanto não são válidos para o estado 0 de P. Assim, $P_1 \| P_2$ não é uma abstração consistente de S w.r.t. Σ_r.

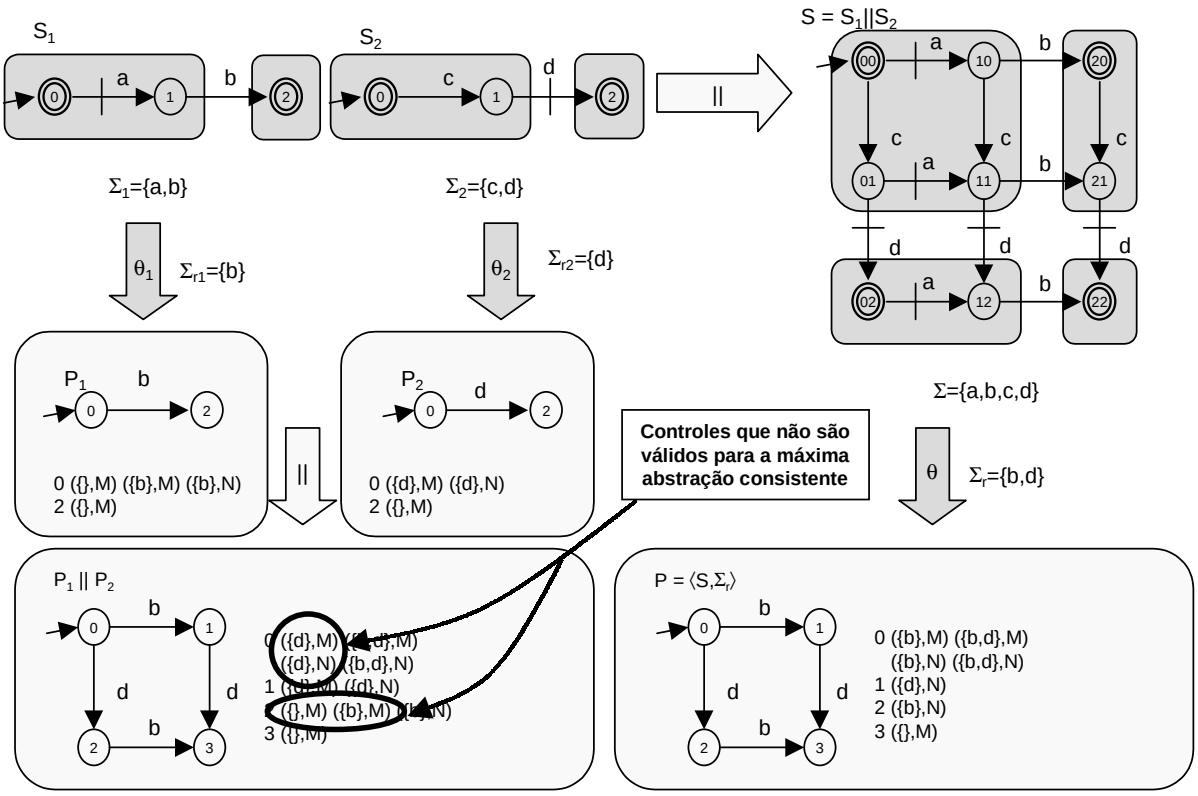


Figura 6.2: Exemplo de composição de abstrações consistentes.

Uma solução possível para este problema é encontrar uma classe de abstrações consistentes que possam ser compostas para gerar uma abstração consistente do sistema composto. Este é o caso das chamadas abstrações confiáveis (*reliable abstractions*), definidas pela primeira vez em (Pu 2000), e transcritas aqui para os SEDs com marcação flexível.

Definição 6.5 (Abstração confiável) *Dados o SED com marcação flexível S sobre o alfabeto Σ e o conjunto de eventos relevantes $\Sigma_r \subseteq \Sigma$, com mapa repórter associado*

θ, uma abstração confiável de S e.r.a. Σ_r é um SED com marcação flexível R sobre o alfabeto Σ_r tal que (i) $L_R = \theta(L_S)$, e (ii) para todo $t \in L_R$, se $(\gamma, \#) \in \Gamma_R(t)$, então, para todo $s \in \theta^{-1}(t)$, há $(\gamma', \#') \in \Gamma_{S,voc}(s)$ para o qual $\gamma \cap \Sigma_{S,voc}(s) = \gamma'$ e $\# = \#'$.

Uma abstração confiável é uma abstração consistente para a qual, se um controle de alto nível está disponível para uma cadeia de alto nível t, então o mesmo controle deve estar disponível no conjunto de controles vocais de *toda palavra do baixo nível correspondente a t*. Dois fatos óbvios relacionados às abstrações confiáveis são: primeiro, se R é uma abstração confiável de S e.r.a. Σ_r, então R é uma abstração consistente de S e.r.a. Σ_r e, segundo, há consistência hierárquica entre S e R.

Proposição 6.4 (Composição de abstrações confiáveis) *Se R_1 é uma abstração confiável de S_1 e.r.a. Σ_{1r} e R_2 é uma abstração confiável de S_2 e.r.a. Σ_{2r}, então $R = R_1 \| R_2$ é uma abstração confiável de $S = S_1 \| S_2$ e.r.a. $\Sigma_r = \Sigma_{r1} \cup \Sigma_{r2}$.*

Demonstração. Uma vez que $R = R_1 \| R_2$, escreve-se $L_R = L_{R_1} \| L_{R_2} = l_1^{-1}(\theta_1(L_{S_1})) \cap l_2^{-1}(\theta_2(L_{S_2})) = \theta(p_1^{-1}(L_{S_1}) \cap p_2^{-1}(L_{S_2})) = \theta(L_{S_1} \| L_{S_2}) = \theta(L_S)$, onde p_i, θ_i, l_i e θ são as funções definidas acima. Como $L_R = L_{R_1} \| L_{R_2}$, então para cada $t \in L_R$ existe um único par correspondente $(t_1, t_2) \in L_{R_1} \times L_{R_2}$ para o qual $t_1 = l_1(t)$ e $t_2 = l_2(t)$. Considerando-se que $\gamma_{ri} \subseteq 2^{\Sigma_{L_{R_i}}(t_i)}$, para $i \in \{1, 2\}$, para $t \in L_R$ e $(\gamma_r, \#_r) \in \Gamma_R(t)$ há $(\gamma_{r1}, \#_{r1}) \in \Gamma_{R_1}(t_1)$ e $(\gamma_{r2}, \#_{r2}) \in \Gamma_{R_2}(t_2)$ para os quais $\gamma_r = \gamma_{r1} \cup \gamma_{r2}$ e $\#_r = \#_{r1} \wedge \#_{r2}$, onde $(t_1, t_2) \in L_{R_1} \times L_{R_2}$ é o par correspondente a t.

Como R_i é uma abstração confiável de S_i e.r.a. Σ_{ri}, $i \in \{1, 2\}$, se $(\gamma_{ri}, \#_{ri}) \in \Gamma_{R_i}(t_i)$, então para todo $s_i \in \theta_i^{-1}(t_i)$ há $(\gamma'_{ri}, \#'_{ri}) \in \Gamma_{S_i,voc}(s_i)$ para o qual $\gamma_{ri} = \gamma'_{ri} \cap \Sigma_{S_i,voc}(s_i)$ e $\#'_{ri} = \#_{ri}$. Esta afirmação pode ser reescrita pelo uso da definição 5.2, dizendo: para todo $s_i \in \theta_i^{-1}(t_i)$ há uma linguagem não vazia $K_i \subseteq L_{S_i(s_i)}$ tal que: (i) K_i é $(L_{S_i(s_i)}, \Gamma_{S_i(s_i)})$-compatível; (ii) para todo $u \in L_{S_i,voc}(s_i)$, se $w_i(s_i u) \in \gamma_{ri}$ então $u \in K_i$; (iii) se $\#_{ri} = N$, então $K_i - L_{S_i,voc}(s_i) = \emptyset$; e (iv) se $\#_{ri} = M$, então $K_i - L_{S_i,voc}(s_i) \neq \emptyset$, onde $w_i : L_{S_i} \to \Sigma_{ri} \cup \{\tau_0, \tau_{init}\}$ é o mapa vocal associado a θ_i.

Como $S = S_1 \| S_2$, para cada $s \in L_S$ há um único par correspondente $(s_1, s_2) \in L_{S_1} \times L_{S_2}$ tal que $s_1 = p_1(s)$ e $s_2 = p_2(s)$. Com isso, para todo $t \in L_R$ e par correspondente

$(t_1, t_2) \in L_{R_1} \times L_{R_2}$, para $s \in \theta^{-1}(t)$ existe um par único $(s_1, s_2) \in \theta_1^{-1}(t_1) \times \theta_2^{-1}(t_2)$ para o qual $s_1 = p_1(s)$ e $s_2 = p_2(s)$. Para s em L_S, com par correspondente (s_1, s_2) em $L_{S_1} \times L_{S_2}$, e subsistemas $S(s)$, $S_1(s_1)$ e $S_2(s_2)$ enumeram-se os seguintes fatos. Primeiro, para todo $u \in L_{S(s)} - L_{S,voc}(s)$ existe um único par correspondente $(u_1, u_2) \in (L_{S_1(s_1)} - L_{S_1,voc}(s_1)) \times (L_{S_2(s_2)} - L_{S_2,voc}(s_2))$, onde $u_1 = p_1(u)$ e $u_2 = p_2(u)$. Segundo, como a composição $S = S_1 \| S_2$ é assíncrona, para todo $(\gamma, \#) \in \Gamma_{S(s)}(u)$ há $(\gamma_1, \#_1) \in \Gamma_{S_1(s_1)}(u_1)$ e $(\gamma_2, \#_2) \in \Gamma_{S_2(s_2)}(u_2)$ tais que $\gamma = \gamma_1 \cup \gamma_2$, $\# = \#_1 \wedge \#_2$. Terceiro, para u, u_1 e u_2 acima, $\Sigma_{L_{S(s)}}(u) = \Sigma_{L_{S_1(s_1)}}(u_1) \cup \Sigma_{L_{S_2(s_2)}}(u_2)$. Quarto, para todo $u \in L_{S,voc}(s)$, há um único par (u_1, u_2) em $[L_{S_1,voc}(s_1) \times (L_{S_2(s_2)} - L_{S_2,voc}(s_2))] \cup [(L_{S_1(s_1)} - L_{S_1,voc}(s_1)) \times L_{S_2,voc}(s_2)]$ tais que $u_1 = p_1(u)$ e $u_2 = p_2(u)$. E quinto, para todo $u_i \in L_{S_i,voc}(s_i)$, há $u \in L_{S,voc}(s)$ para o qual $p_i(u) = u_i$, com $i \in \{1, 2\}$, pela não sincronicidade de S_1 e S_2.

Depois desta enumeração inicial de fatos, considere $t \in L_R$, com par correspondente $(t_1, t_2) \in L_{R_1} \times L_{R_2}$, controle $(\gamma_r, \#_r) \in \Gamma_R(t)$, com par de controles correspondentes $(\gamma_{ri}, \#_{ri}) \in \Gamma_{R_i}(t_i)$, com $i \in \{1, 2\}$, palavra $s \in \theta^{-1}(t)$, com par correspondente $(s_1, s_2) \in \theta_1^{-1}(t_1) \times \theta_2^{-1}(t_2)$, e linguagens K_1 e K_2 para $S_1(s_1)$ e $S_2(s_2)$, respectivamente, que correspondem aos controles $(\gamma_{r1}, \#_{r1})$ e $(\gamma_{r2}, \#_{r2})$, defina-se a linguagem $K \subseteq L_{S(s)}$ como:

$$\begin{aligned} u \in K \iff & (u \in K_1 \| K_2) \text{ e } ((\forall u' < u)\ w(su') = \tau_0) \\ & (u \in \overline{K_1 \| K_2} - K_1 \| K_2) \text{ e } (w(su) \neq \tau_0) \text{ e } ((\forall u' < u)\ w(su') = \tau_0) \end{aligned} \tag{6.3}$$

onde $w : L_S \to \Sigma_r \cup \{\tau_0, \tau_{init}\}$ é o mapa vocal associado a θ. A linguagem $\overline{K}$ copia $\overline{K_1 \| K_2}$ desde ϵ até a ocorrência do primeiro evento relevante. As palavras de K são as palavras de $K_1 \| K_2$ que permaneceram em $\overline{K}$, mais as palavras de $\overline{K_1 \| K_2} - K_1 \| K_2$ que terminam com um evento relevante. A seguir, enumeram-se propriedades de K:

1. Como K_1 e K_2 são não vazias e o produto $K_1 \| K_2$ é assíncrono, então K é não vazia e $p_i(K) = K_i$, $i \in \{1, 2\}$.

2. Seja $u \in K$ e par correspondente $(u_1, u_2) \in K_1 \times K_2$, isto é, $u_i = p_i(u)$, para $i \in$

$\{1,2\}$. Se $u \in L_{S(s)} - L_{S,voc}(s)$, então $(u_1, u_2) \in (L_{S_1(s_1)} - L_{S_1,voc}(s_1)) \times (L_{S_2(s_2)} - L_{S_2,voc}(s_2))$. Como K_i é $(L_{S_i(s_i)}, \Gamma_{S_i(s_i)})$-compatível, há $(\gamma_i, M) \in \Gamma_{S_i(s_i)}(u_i)$ para o qual $\gamma_i \cap \Sigma_{L_{S_i(s_i)}}(u_i) = \Sigma_{K_i}(u_i)$. Pelas relações enumeradas para os subsistemas $S(s)$, $S_1(s_1)$ e $S_2(s_2)$, há $(\gamma, M) \in \Gamma_{S(s)}(u)$ tal que $\gamma \cap \Sigma_{L_{S(s)}}(u) = (\gamma_1 \cup \gamma_2) \cap (\Sigma_{L_{S_1(s_1)}} \cup \Sigma_{L_{S_1(s_1)}}) = (\gamma_1 \cap \Sigma_{L_{S_1(s_1)}}) \cup (\gamma_1 \cap \Sigma_{L_{S_1(s_1)}}) = \Sigma_{K_1}(u_1) \cup \Sigma_{K_2}(u_2) = \Sigma_K(u)$. O segundo passo do desenvolvimento acima é possível porque o produto $S = S_1 \| S_2$ é assíncrono, $\gamma_i \subseteq \Sigma_i$ e $\Sigma_{L_{S_i(s_i)}}(u_i) \subseteq \Sigma_i$, para $i \in \{1,2\}$. Agora, se $u \in L_{S,voc}(s)$, então u termina com evento relevante e existe $(\emptyset, M) \in \Gamma_{L_{S(s)}}(u)$ tal que $\emptyset \cap \Sigma_{L_{S(s)}}(u) = \Sigma_K(u)$. Assim, para todo $u \in K$ há $(\gamma, M) \in \Gamma_{S(s)}(u)$ tal que $\gamma \cap \Sigma_{L_{S(s)}}(u) = \Sigma_K(u)$.

3. Seja $u \in \overline{K} - K$, então $u \in L_{S(s)} - L_{S,voc}(s)$. Considere o par $(u_1, u_2) \in K_1 \| K_2 \subseteq (L_{S_1(s_1)} - L_{S_1,voc}(s_1)) \times (L_{S_2(s_2)} - L_{S_2,voc}(s_2))$ correspondente a u. Como $p_i(K) = K_i$, com $i \in \{1,2\}$, então $u_i \in \overline{K_i} - K_i$. E como K_i é $(L_{S_i(s_i)}, \Gamma_{S_i(s_i)})$-compatível, há $(\gamma_i, N) \in \Gamma_{S_i(s_i)}$ tal que $\gamma_i \cap \Sigma_{L_{S_i(s_i)}}(u_i) = \Sigma_{K_i}(u_i)$. Assim, há $(\gamma, N) = (\gamma_1 \cup \gamma_2, \#_1 \wedge \#_2) \in \Gamma_{S(s)}(u)$ tal que $\gamma \cap \Sigma_{L_{S(s)}}(u) = \Sigma_K(u)$. Assim, para todo $u \in \overline{K} - K$, existe $(\gamma, N) \in \Gamma_{S(s)}(u)$ tal que $\gamma \cap \Sigma_{L_{S(s)}}(u) = \Sigma_K(u)$.

4. Pelos ítens 2 e 3 acima, K é $(L_{S(s)}, \Gamma_{S(s)})$-compatível.

5. Para $u \in L_{S,voc}(s)$, seja $w(su) \in \gamma_r$. Então para $u_i \in L_{S_i,voc}(s_i)$ correspondente a u, com $i \in \{1,2\}$, $w_i(s_i u_i) \in \gamma_{ri}$. Então, pelas propriedades de K_i, $u_i \in K_i$. Então, pela definição de K, $u \in K$.

6. Se $\#_r = N$, então $\#_{1r} = N$ ou $\#_{2r} = N$. Se $\#_{ir} = N$, então $K_i - L_{S_i,voc}(s_i) = \emptyset$, com $i \in \{1,2\}$. Então para todo $u \in K \subseteq K_1 \| K_2$, u termina com um evento relevante, então $K - L_{S,voc}(s) = \emptyset$.

7. Se $\#_r = M$, então $\#_{r1} = \#_{r2} = M$. Então há $u_i \in K_i$ para o qual $u_i \notin L_{S_i,voc}(s_i)$, para $i \in \{1,2\}$. Então há $u \in K$ tal que $p_i(u) = u_i$ e $u \notin L_{S,voc}(s)$. Então, $K - L_{S,voc}(s) \neq \emptyset$.

Pelas propriedades enumeradas para a linguagem K e pela definição 5.2, existe um controle em $\Gamma_{S,voc}(s)$ correspondente a K, diga-se $(\gamma'_r, \#'_r)$, tal que $\gamma_r = \gamma'_r \cap \Sigma_{S,voc}(s)$ e $\#'_r = \#_r$.

Assim, para todo t em L_R, se $(\gamma_r, \#_r)$ está em $\Gamma_R(t)$, então para todo s em $\theta^{-1}(t)$ existe um controle $(\gamma'_r, \#'_r)$ em $\Gamma_{S,voc}(s)$ para o qual $\gamma'_r = \gamma_r \cap \Sigma_{S,voc}(s)$ e $\#'_r = \#_r$. Assim, R é uma abstração confiável de S e.r.a. Σ_r, o que completa a prova. □

A proposição 6.4 afirma que a composição de abstrações confiáveis é também uma abstração confiável para o sistema composto. Assim, há consistência hierárquica entre a composição das abstrações confiáveis dos sistemas componentes e o sistema composto.

Proposição 6.5 (Existência de abstrações confiáveis) *Para o SED com marcação flexível S sobre o alfabeto Σ e o conjunto de eventos relevantes $\Sigma_r \subseteq \Sigma$, com mapa repórter associado θ, o conjunto das abstrações confiáveis de S e.r.a. Σ_r é não vazio e fechado a uniões arbitrárias de SEDs com marcação flexível.*

Demonstração. O sistema trivial P, onde $L_P = \theta(L_S)$ e, para todo $t \in L_P$, $\Gamma_P(t) = \emptyset$, é uma abstração confiável de S e.r.a. Σ_r. A prova de que as abstrações confiáveis são fechadas em relação à união é análoga à prova apresentada na proposição 6.2. □

A proposição 6.5 garante a existência de uma abstração confiável dado um sistema S e um conjunto de eventos relevantes Σ_r. Novamente, este resultado difere do resultado correspondente em (Pu 2000) que vincula a existência de abstrações confiáveis à condição de que o mapa repórter seja um observador. Esta condição não é necessária pela mesma razão construtiva do presente modelo que permite existência de uma abstração consistente (vide seção 6.2). A proposição 6.5 também fornece uma propriedade do conjunto de abstrações confiáveis para um dado sistema e um conjunto de eventos relevantes que é explorada para se construir uma abstração confiável na proposição 6.6 a seguir.

Proposição 6.6 (Construção de abstrações confiáveis) *Para o SED com marcação flexível S sobre o alfabeto Σ e o conjunto de eventos relevantes $\Sigma_r \subseteq \Sigma$,*

com mapa repórter associado θ, existe a máxima abstração confiável de S e.r.a. Σ_r, denotada $[S, \Sigma_r]$, e definida por $L_{[S,\Sigma_r]} = \theta(L_S)$ e por

$$\begin{aligned}\Gamma_{[S,\Sigma_r]}(t) \;=\; \{(\gamma, \#) \in 2^{\Sigma_r} \times \{M, N\} :\; & (\forall s \in \theta^{-1}(t))\ (\exists(\gamma', \#') \in \Gamma_{S,voc}(s))\ \textit{tais que} \\ & ((\gamma \cap \Sigma_{S,voc}(s) = \gamma')\ \textit{e}\ (\#' = \#))\}.\end{aligned} \tag{6.4}$$

para todo $t \in L_{[S,\Sigma_r]}$.

Demonstração. Pela proposição 6.5, dados S e Σ_r, a máxima abstração confiável de S e.r.a. Σ_r existe e é dada por $R = [S, \Sigma_r] = \bigcup_i R_i$, onde R_i é uma abstração confiável de S e.r.a. Σ_r. Escreve-se R como sendo $L_R = L_{R_i} = \theta(L_S)$ e

$$\begin{aligned}\Gamma_R(t) \;&=\; [\textstyle\bigcup_i \Gamma_{R_i}(t)]_{\cup} = && (\text{para } R_i \text{ abstração confiável de } S \text{ e.r.a. } \Sigma_r) \\ &=\; \{(\gamma, \#) \in 2^{\Sigma_r} \times \{M, N\} :\; && (\forall s \in \theta^{-1}(t))\ (\exists(\gamma', \#') \in \Gamma_{S,voc}(s)) \text{ tais que} \\ & && ((\gamma \cap \Sigma_{S,voc}(s) = \gamma') \text{ e } (\#' = \#))\}.\end{aligned}$$

para todo $t \in L_R$. □

Exemplo 6.4 (Composição de abstrações confiáveis) *Considere a construção das abstrações confiáveis para os sistemas S_1, S_2 e S do exemplo 6.3, repetidos na figura 6.3. Na figura 6.3 $R_1 = [S_1, \Sigma_{r1}]$, $R_2 = [S_2, \Sigma_{r2}]$ e $R = [S, \Sigma_r]$. Note-se que $R_1 \| R_2 \leq R$, isto é, $R_1 \| R_2$ é uma abstração confiável de S e.r.a. Σ_r. Entretanto, ao observar-se o número de controles disponíveis para cada estado de $R_1 \| R_2$ pode-se notar que $R_1 \| R_2$ possui informação pobre para síntese de alto nível.*

Como ilustrado pelo exemplo 6.4, de forma geral, uma abstração confiável corresponde a uma abstração consistente grosseira para o sistema, o que acarreta pouca informação disponível para síntese no alto nível. Na próxima seção busca-se uma alternativa na construção de abstrações consistentes dos componentes do sistema a fim de se obter, por composição no alto nível, uma abstração consistente do sistema composto mais refinada que a abstração confiável.

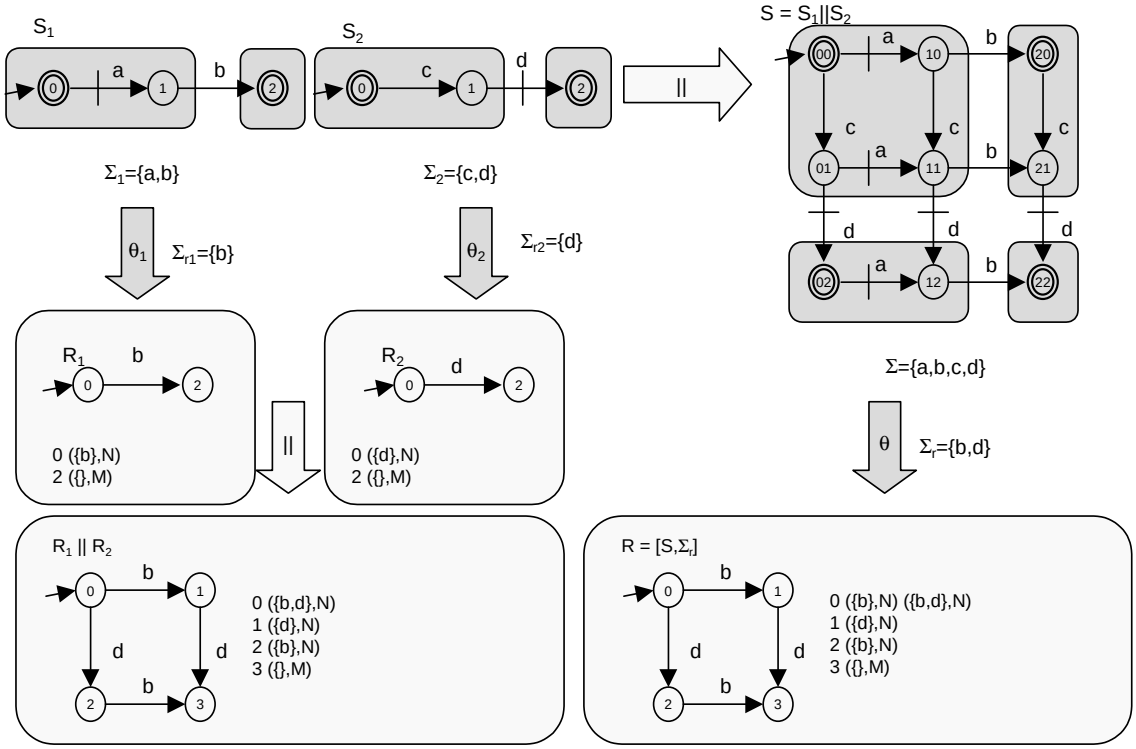


Figura 6.3: Composição de abstrações confiáveis.

6.4 Uma abordagem pelo raciocínio supor-garantir

Propõe-se, nesta seção, uma abordagem alternativa para o controle hierárquico de sistemas compostos baseada no raciocínio supor-garantir.

O *raciocínio supor-garantir* é utilizado em abordagens onde há, de forma geral, um sistema formado por diversos sistemas menores componentes e se deseja verificar ou sintetizar propriedades para o sistema total a partir de propriedades dos componentes, exemplos de tais abordagens são Stark (1985), Alur e Henzinger (1999) e Maier (2001).

Para ilustrar a aplicação do raciocínio supor-garantir, faz-se uma rápida digressão aos resultados da verificação composicional de sistemas em (Maier 2001). Nesta abordagem, considera-se um sistema S como sendo um conjunto. A composição de dois sistemas, S_1 e S_2, corresponde à interseção $S_1 \cap S_2$. Considera-se também uma propriedade como sendo um conjunto P, e a conjunção de duas propriedades, digam-se P_1 e P_2 como sendo a interseção $P_1 \cap P_2$. O objetivo é verificar que $S_1 \cap S_2 \subseteq P_1 \cap P_2$, a significar que $S_1 \cap S_2$ satisfaz a $P_1 \cap P_2$. Se for possível verificar-se que $S_1 \subseteq P_1$ e $S_2 \subseteq P_2$, cumpre-se o objetivo pela aplicação da *regra de prova composicional*

$$((S_1 \subseteq P_1) \text{ e } (S_2 \subseteq P_2)) \;\; \Rightarrow \;\; (S_1 \cap S_2 \subseteq P_1 \cap P_2), \tag{6.5}$$

ilustrada por diagramas de Venn representados na figura 6.4.

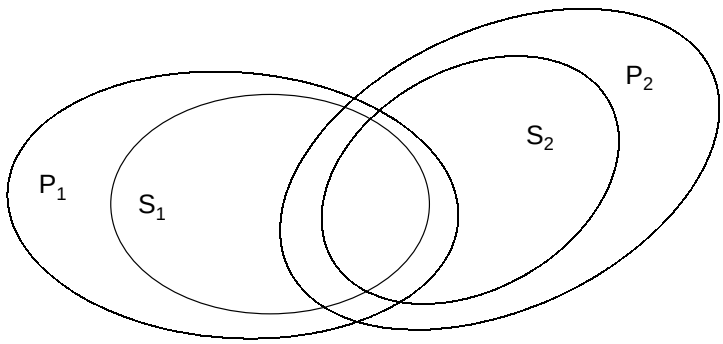


Figura 6.4: Regra de prova composicional.

Entretanto, pode ser o caso em que, sob a hipótese de que o ambiente de S_1 se

comporta como P_2, garante-se a propriedade Q_1 para S_1 e seu ambiente, denotado $S_1 \cap P_2 \subseteq Q_1$. O mesmo se afirma para S_2, com ambiente P_1 e atendendo a propriedade Q_2, denotado $S_2 \cap P_1 \subseteq Q_2$. Prova-se então que $S_1 \cap S_2$ satisfaz a $Q_1 \cap Q_2$ usando-se a *regra de prova supor-garantir*

$$((S_1 \cap P_2 \subseteq Q_1) \text{ e } (S_2 \cap P_1 \subseteq Q_2) \text{ e } (S_1 \subseteq P_1) \text{ e } (S_2 \subseteq P_2)) \;\Rightarrow\; (S_1 \cap S_2 \subseteq Q_1 \cap Q_2), \tag{6.6}$$

ilustrada na figura 6.5.

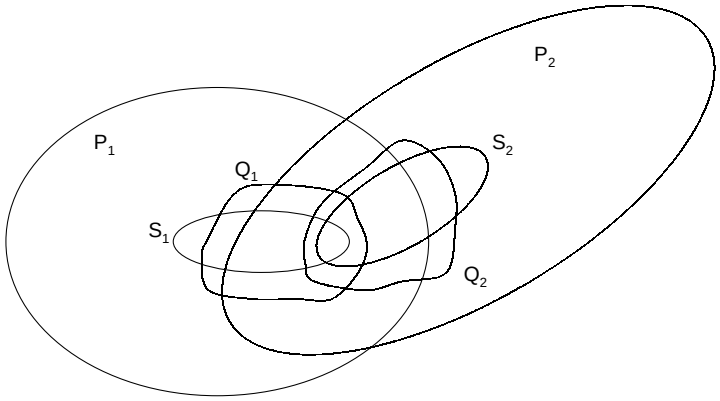


Figura 6.5: Regra de prova supor-garantir.

Observe a necessidade das condições adicionais $S_i \subseteq P_i$, $i \in \{1, 2\}$, para quebrar a circularidade da regra, pois as duas primeiras hipóteses garantem apenas que $(S_1 \cap S_2) \cap (P_1 \cap P_2) \subseteq Q_1 \cap Q_2$. Basicamente dizer $S_i \subseteq P_i$ significa que P_i funciona como uma aproximação do comportamento de S_i, usada no caso como ambiente de S_j, com i e j em $\{1, 2\}$ e $i \neq j$. De forma geral, a condição de quebra de circularidade do raciocínio é dependente do modelo em questão, veja, por exemplo, Stark (1985) e Alur e Henzinger (1999).

De volta às abstrações consistentes de SEDs com marcação flexível, por observação da equação (6.5), pode-se provar que as abstrações consistentes não atendem ao raciocínio composicional, enquanto as abstrações confiáveis atendem. Propõe-se, a seguir, um método baseado no raciocínio supor-garantir para se obterem abstrações consisten-

tes de sistemas compostos a partir da composição de abstrações consistentes modificadas dos seus componentes.

No sistema composto $S = S_1 \| S_2$, S_1 serve de ambiente para S_2 e vice versa. Primeiro, seja P_i a máxima abstração consistente de S_i e.r.a. Σ_{ri}, isto é, $\langle S_i, \Sigma_{ri} \rangle$, para $i \in \{1, 2\}$, e construa-se o SED com marcação flexível P_i' a partir de P_i fazendo-se $L_{P_i'} = L_{P_i}$ e, para todo $t \in L_{P_i'}$, $\Gamma_{P_i'}(t) = 2^{\Sigma_{L_{P_i'}}(t)} \times \{M, N\}$. As palavras geradas por P_i' são as mesmas geradas por P_i. Por outro lado, a cada palavra t gerada por P_i', o conjunto $\Gamma_{P_i'}(t)$ admite todo controle possível envolvendo os eventos ativos após t. Segundo, considera-se P_i' como sendo ambiente de S_j, com $j \in \{1, 2\}$ e $j \neq i$, e constrói-se o sistema expandido $S_j' = S_j \| P_i'$, que corresponde ao sistema original S_j mais seu ambiente P_i'. Terceiro, constrói-se a máxima abstração consistente do sistema expandido S_j' e.r.a. Σ_r, $Q_j = \langle S_j', \Sigma_r \rangle$, e o sistema composto $Q = Q_1 \| Q_2$. O desenvolvimento a seguir prova que Q, assim construído, é igual à máxima abstração consistente de S e.r.a. Σ_r.

Lema 6.1 (Composição de S_i') $S = S_1' \| S_2'$.

Demonstração. Para o desenvolvimento seguinte, introduz-se alguma notação. Para $i, j \in \{1, 2\}$ e $i \neq j$, seja $\theta_{ie} : L_{S_i'} \to \Sigma_r^*$ o mapa repórter que apaga eventos em $\Sigma_i - \Sigma_{ir}$ de palavras em $L_{S_i'} \subseteq (\Sigma_i \cup \Sigma_{jr})^*$; $q_j : (\Sigma_i \cup \Sigma_{jr})^* \to \Sigma_i^*$ a projeção de palavras sobre $\Sigma_i \cup \Sigma_{jr}$ em palavras sobre Σ_i; $h_i : (\Sigma_i \cup \Sigma_{jr})^* \to \Sigma_{jr}^*$ a projeção de palavras sobre $\Sigma_i \cup \Sigma_{jr}$ em palavras sobre Σ_{jr}; $m_i : \Sigma^* \to (\Sigma_j \cup \Sigma_{ir})^*$ a projeção de palavras sobre Σ em palavras sobre $\Sigma_j \cup \Sigma_{ir}$; e $\tilde{\theta}_{ie} : (\Sigma_i \cup \Sigma_{jr})^* \to \Sigma_r^*$ a projeção associada ao mapa repórter θ_{ie}.

Primeiro, prova-se que $L_{S_1'} \| L_{S_2'} = L_S$. Note-se que $L_{S_i'} = L_{S_i} \| L_{P_j'} = q_i^{-1}(L_{S_i}) \cap h_j^{-1}(\theta_j(L_{S_j}))$, onde $i, j \in \{1, 2\}$ e $i \neq j$. Assim,

$$
\begin{aligned}
L_{S_1'}\|L_{S_2'} &= m_1^{-1}(L_{S_1}) \cap m_2^{-1}(L_{S_2}) \\
&= m_1^{-1}(q_1^{-1}(L_{S_1}) \cap h_2^{-1}(\theta_2(L_{S_2}))) \cap m_2^{-1}(q_2^{-1}(L_{S_2}) \cap h_1^{-1}(\theta_1(L_{S_1}))) \\
&= m_1^{-1}(q_1^{-1}\ (L_{S_1})) \cap m_1^{-1}(h_2^{-1}(\theta_2(L_{S_2})))) \cap m_2^{-1}(q_2^{-1}(L_{S_2})) \cap m_2^{-1}(h_1^{-1}(\theta_1(L_{S_1})))) \\
&= p_1^{-1}(L_{S_1}) \cap p_2^{-1}(\theta_2^{-1}(\theta_2(L_{S_2}))) \cap p_1^{-1}(\theta_1^{-1}(\theta_1(L_{S_1}))) \cap p_2^{-1}(L_{S_2}) \\
&= p_1^{-1}(L_{S_1}) \cap p_2^{-1}(L_{S_2}) \\
&= L_{S_1}\|L_{S_2} = L_S.
\end{aligned}
\tag{6.7}
$$

Segundo, prova-se a igualdade da estruturas de controle de $S_1'\|S_2'$ e S. Defina-se $S' = S_1'\|S_2'$. Do desenvolvimento anterior, sabe-se que $L_{S'} = L_S$. Prova-se a seguir que, para todo s em $L_{S'}$, $\Gamma_{S'}(s) = \Gamma_S(s)$. Para todo s em $L_{S'}$, $\Gamma_{S'}(s) = \{(\gamma, \#) \in 2^{\Sigma_{L_{S'}}(s)} \times \{M, N\} : (\exists(\gamma_i, \#_i) \in \Gamma_{S_i'}(s_i'),\ s_i' = m_i(s),\ i \in \{1,2\})$ tal que $(\gamma \cap (\Sigma_1 - \Sigma_{1r}) = (\gamma_1 - \gamma_2) \cap \Sigma_{L_{S'}}(s))$ e $(\gamma \cap (\Sigma_2 - \Sigma_{2r}) = (\gamma_2 - \gamma_1) \cap \Sigma_{L_{S'}}(s))$ e $(\gamma \cap \Sigma_r = (\gamma_1 \cap \gamma_2) \cap \Sigma_{L_{S'}}(s))$ e $(\# = \#_1 \wedge \#_2)\}$. Justifica-se o uso dos alfabetos acima da seguinte forma: denote-se o alfabeto do SED com marcação flexível A por $\Sigma(A)$, e note-se que $\Sigma(S_1') = \Sigma_1 \cup \Sigma_{2r}$ e $\Sigma(S_2') = \Sigma_2 \cup \Sigma_{1r}$. Assim, escreve-se $\Sigma(S_1') - \Sigma(S_2') = \Sigma_1 - \Sigma_{1r}$, $\Sigma(S_2') - \Sigma(S_1') = \Sigma_2 - \Sigma_{2r}$, e $\Sigma(S_1') \cap \Sigma(S_2') = \Sigma_r$.

Como $S_i' = S_i\|P_j'$, $i, j \in \{1,2\}$ e $i \neq j$, afirma-se o seguinte: se $(\gamma_i, \#_i) \in \Gamma_{S_i'}(s_i')$, então há $(\gamma_{ii}, \#_{ii}) \in \Gamma_{S_i}(s_i)$, com $s_i = q_i(s_i')$, e há $(\gamma_{jr}, \#_{jr}) \in 2^{\Sigma_{L_{P_j'}}(t_j)} \times \{M, N\}$, com $t_j = h_j(s_i')$, tais que $\gamma_i = \gamma_{ii} \cap \gamma_{jr}$ e $\#_i = \#_{ii} \wedge \#_{jr}$. Note-se também que $\gamma_i \subseteq \Sigma_i$ e $\gamma_{ir} \subseteq \Sigma_{ir}$ no desenvolvimento acima. A partir da decomposição dos controles anterior, reescrevem-se algumas das expressões para o conjunto de controles $\Gamma_{S'}(s)$: $\gamma_i - \gamma_j = (\gamma_{ii} \cup \gamma_{jr}) \cap (\gamma_{jj} \cup \gamma_{ir})^c = (\gamma_{ii} \cap \gamma_{ir}^c \cap \gamma_{jj}^c) \cup (\gamma_{jr} \cap \gamma_{ir}^c \cap \gamma_{jj}^c) = \gamma_{ii} - \gamma_{ir}$, com $i, j \in \{1,2\}$ e $i \neq j$. Escreve-se também $\gamma_1 \cap \gamma_2 = (\gamma_{11} \cup \gamma_{2r}) \cap (\gamma_{22} \cup \gamma_{1r}) = (\gamma_{11} \cap \gamma_{1r}) \cup (\gamma_{11} \cap \gamma_{22}) \cup (\gamma_{2r} \cap \gamma_{1r}) \cup (\gamma_{2r} \cap \gamma_{22}) = (\gamma_{11} \cap \gamma_{1r}) \cup (\gamma_{2r} \cap \gamma_{22})$. Então, o conjunto de controles $\Gamma_{S'}(s)$ é reescrito como $\Gamma_{S'}(s) = \{(\gamma, \#) \in 2^{\Sigma_{L_{S'}}(s)} \times \{M, N\} : (\exists(\gamma_{ii}, \#_{ii}) \in \Gamma_{S_i}(s_i),\ s_i = q_i(m_i(s)))$ e $(\exists(\gamma_{jr}, \#_{jr}) \in \Gamma_{P_j'}(t_j),\ t_j = h_j(m_i(s)))$ tais que $(\gamma \cap (\Sigma_i - \Sigma_{ir}) = (\gamma_{ii} - \gamma_{ir}) \cap \Sigma_{L_{S'}}(s))$ e $(\gamma \cap \Sigma_r = (\cup(\gamma_{ii} \cap \gamma_{ir})) \cap \Sigma_{L_{S'}}(s))$ e

$(\# = \wedge(\#_{ii} \wedge \#_{jr})),\ i,j \in \{1,2\},\ i \neq j\}$.

Considere a seguinte caracterização de $\Sigma_{L_{S'}}(s)$. Para a palavra $s \in L_{S'} = L_S$, definam-se as palavras correspondentes $s'_i = m_i(s) \in L_{S'_i}$, $s_i = q_i(m_i(s)) \in L_{S_i}$ e $t_j = h_j(m_i(s)) \in L_{P_j}$, com $i,j \in \{1,2\}$ e $i \neq j$. Como o produto $S'_i = S_i \| P'_j$, com $i,j \in \{1,2\}$ e $i \neq j$, é assíncrono, então $\Sigma_{L_{S'_i}}(s'_i) = \Sigma_{L_{S_i}}(s_i) \cap \Sigma_{L_{P'_j}}(t_j)$. Conclui-se também para as palavras s_i e t_i acima definidas ($i \in \{1,2\}$) tem-se: primeiro $\theta_i(s_i) = t_i$, segundo, $\Sigma_{L_{S_i}}(s_i) \cap \Sigma_{ir} \subseteq \Sigma_{S_i,voc}(s_i) \subseteq \Sigma_{P'_i}(t_i)$, e terceiro, $\Sigma_{P'_i}(t_i) \cap \Sigma_{L_{S_i}}(s_i) = \Sigma_{L_{S_i}}(s_i)$. Assim, afirma-se que $\alpha \in \Sigma_{L_S}(s)$ se e somente se, para $i \in \{1,2\}$: (i) $\alpha \in \Sigma_{L_{S_i}}(s_i) \cap (\Sigma_i - \Sigma_{ir})$ ou (ii) $\alpha \in (\Sigma_{L_{S_i}}(s_i) \cap \Sigma_{L_{P'_i}}(t_i)) \cap \Sigma_{ir}$. Do raciocínio acima, conclui-se que os eventos envolvidos na sincronização de S'_1 e S'_2 no produto $S'_1 \| S'_2$ são os eventos em Σ_r. Dessa forma, escreve-se $\Sigma_{L_S}(s) = (\Sigma_{L_{S_1}}(s_1) \cap (\Sigma_1 - \Sigma_{1r}))$ $\cup(\Sigma_{L_{S_2}}(s_2) \cap (\Sigma_2 - \Sigma_{2r})) \cup((\Sigma_{L_{S_1}}(s_1) \cap \Sigma_{L_{P'_1}}(t_1)) \cap \Sigma_{1r}) \cup((\Sigma_{L_{S_2}}(s_2) \cap \Sigma_{L_{P'_2}}(t_2)) \cap \Sigma_{2r})$ $= \Sigma_{L_{S_1}}(s_1) \cup \Sigma_{L_{S_2}}(s_2)$. Assim, o conjunto de controles $\Gamma_{S'}(s)$ é reescrito como $\Gamma_{S'}(s)$ $= \{(\gamma,\#) \in 2^{\Sigma_{L_{S'}}(s)} \times \{M,N\} : (\exists(\gamma_{ii},\#_{ii}) \in \Gamma_{S_i}(s_i),\ s_i = q_i(m_i(s)))$ e $(\exists(\gamma_{jr},\#_{jr}) \in \Gamma_{P'_j}(t_j),\ t_j = h_j(m_i(s)))$ tais que $(\gamma \cap (\Sigma_i - \Sigma_{ir}) = (\gamma_{ii} - \gamma_{ir}) \cap \Sigma_{L_{S_i}}(s_i))$ e $(\gamma \cap \Sigma_r = \cup((\gamma_{ii} \cap \gamma_{ir}) \cap \Sigma_{L_{S_i}}(s_i)))$ e $(\# = \wedge(\#_{ii} \wedge \#_{jr})),\ i,j \in \{1,2\},\ i \neq j\}$.

Por inspeção da expressão acima, os controles válidos são aqueles para os quais $\gamma_{ii} \cap \Sigma_{ir} = \Sigma_{ir}$ e $\#_{ir} = \#_i$, para $i \in \{1,2\}$. Assim, reescreve-se o conjunto de controles $\Gamma_{S'}(s)$ como $\Gamma_{S'}(s) = \{(\gamma,\#) \in 2^{\Sigma_{L_{S'}}(s)} \times \{M,N\} : (\exists(\gamma_{ii},\#_{ii}) \in \Gamma_{S_i}(s_i),\ s_i = q_i(m_i(s)),$ $i \in \{1,2\})$ tal que $(\gamma = (\gamma_{11} \cup \gamma_{22}) \cap \Sigma_{L_{S'}})$ e $(\# = \#_{11} \wedge \#_{22})\}$. Da última expressão para o conjunto $\Gamma_{S'}(s)$, conclui-se que, para todo $s \in L_{S'}$, tem-se $\Gamma_{S'}(s) = \Gamma_S(s)$, e isso conclui a prova. □

O lema 6.1 afirma que o produto síncrono dos sistemas expandidos S'_1 e S'_2 é igual ao sistema composto S. Este resultado é usado para relacionar palavras em S a palavras de S'_1 e S'_2.

Lema 6.2 (Composição das linguagens vocais) $L_{S,voc} = L_{S'_1,voc} \| L_{S'_2,voc}$.

Demonstração. Primeiro, note-se que $L_{S'_1,voc} \| L_{S'_2,voc} = m_1^{-1}(L_{S'_1,voc}) \cap m_2^{-1}(L_{S'_2,voc})$, onde o efeito da função m_i^{-1} é adicionar *selfloops* com os eventos em $\Sigma_j - \Sigma_{jr}$ à linguagem

em que é aplicada, com $i, j \in \{1, 2\}$ e $i \neq j$.

Para todo $s \in L_{S_{voc}}$, seja $s'_i = m_i(s) \in L_{S'_i}$, com $i \in \{1, 2\}$. Como s termina por um evento relevante e o efeito de m_i é apagar eventos em $\Sigma_j - \Sigma_{jr}$ de palavras em Σ^*, com $j \in \{1, 2\}$ e $j \neq i$, s'_i termina com o mesmo evento relevante que s. Então, $s'_i \in L_{S'_i,voc}$ e para todo $s \in L_{S,voc}$ há um único par $(s'_1, s'_2) \in L_{S'_1,voc} \times L_{S'_2,voc}$ para o qual $m_i(s) = s'_i$. Conclui-se assim que $L_{S,voc} \subseteq L_{S'_1,voc} \| L_{S'_2,voc}$.

Para todo $s \in L_{S'_1,voc} \| L_{S'_2,voc}$, é fato que $s \in L_{S'_1} \| L_{S'_2} = L_S$. Por outro lado, s corresponde a um par $(s'_1, s'_2) \in L_{S'_1,voc} \times L_{S'_2,voc}$ tal que $m_i(s) = s'_i$, com $i \in \{1, 2\}$. Como $s'_i \in L_{S'_i,voc}$, s'_i termina com um evento relevante e $m_1^{-1}(s'_1) \cap m_2^{-1}(s'_2)$ possui apenas palavras que terminam por eventos relevantes. Com isso conclui-se que $s \in m_1^{-1}(s'_1) \cap m_2^{-1}(s'_2)$ termina por um evento relevante e $s \in L_{S,voc}$. Assim, conclui-se que $L_{S,voc} \supseteq L_{S'_1,voc} \| L_{S'_2,voc}$, e isso completa a prova. □

O lema 6.2 permite relacionar as linguagens vocais dos sistemas expandidos à linguagem vocal do sistema composto. Usando-se o lema 6.2, pode-se relacionar cada palavra vocal de S, diga-se $s \in L_{S,voc}$, a um único par de palavras vocais em S'_1 e S'_2, diga-se $(s'_1, s'_2) \in L_{S'_1,voc} \times L_{S'_2,voc}$.

Lema 6.3 (Composição de subsistemas) *Sejam $s \in L_S$ e o par correspondente $(s'_1, s'_2) \in L_{S'_1} \times L_{S'_2}$, isto é, $s'_i = m_i(s)$ para i igual a 1 ou 2, então: (i) $S'_1(s'_1) \| S'_2(s'_2) = S(s)$; (ii) $L_{S'_1,voc}(s'_1) \| L_{S'_2,voc}(s'_2) = L_{S,voc}(s)$, e (iii) $\Sigma_{S'_1,voc}(s'_1) = \Sigma_{S'_2,voc}(s'_2) = \Sigma_{S,voc}(s)$.*

Demonstração. Primeiro, escrevem-se as definições de $S(s)$ e $S'_i(s'_i)$, com $i \in \{1, 2\}$, de acordo com a definição 5.1. Para $S(s)$, $L_{S(s)} = \{u \in \Sigma^* : (su \in L_S)$ e $(((\forall u' \in \Sigma^+)$ tal que $u' < u)$ $w(su') = \tau_0)\}$ e, para, todo $u \in L_{S(s)}$, $\Gamma_{S(s)}(u) = \Gamma_S(su)$ se $w(su) = \tau_0$ ou $u = \epsilon$, e $\{(\emptyset, M)\}$, caso contrário, onde $w : L_S \to \Sigma_r \cup \{\tau_0\}$ é o mapa vocal associado a θ, vide equação (3.5). Para $S'_i(s'_i)$, com $i \in \{1, 2\}$, $L_{S'_i(s'_i)} = \{u_i \in (\Sigma_i \cup \Sigma_{jr})^* : (s'_i u_i \in L_{S'_i})$ e $(((\forall u'_i \in \Sigma_i^+)$ tal que $u'_i < u_i)$ $w_{ie}(s'_i u'_i) = \tau_0)\}$ e, para todo $u_i \in L_{S'_i(s'_i)})$, $\Gamma_{S'_i(s'_i)}(u_i) = \Gamma_{S'_i}(s'_i u_i)$, se $w_{ie}(s'_i u_i) = \tau_0$ ou $u_i = \epsilon$, e $\{(\emptyset, M)\}$, caso contrário, onde $w_{ie} : L_{S'_i} \to \Sigma_r \cup \{\tau_0\}$ é o mapa vocal associado a θ_{ie}. Os fatos a seguir

decorrem do exame das expressões para $S(s)$ e $S'_i(s'_i)$.

Seja $S(s'_1, s'_2) = S'_1(s'_1) \| S'_2(s'_2)$, prova-se que $S(s'_1, s'_2) = S(s)$. A igualdade das linguagens $L_{S(s'_1,s'_2)} = L_{S'_1(s'_1)} \| L_{S'_2(s'_2)} = m_1^{-1}(L_{S'_1(s'_1)}) \cap m_2^{-1}(L_{S'_2(s'_2)}) = L_{S(s)}$ decorre da mesma forma que se iguala $L_S = L_{S'_1} \| L_{S'_2}$. A igualdade dos controles $\Gamma_{S(s'_1,s'_2)}(u) = \Gamma_{S(s)}(u)$ decorre pela razão que, para todo $s \in L_S$ e par correspondente $(s'_1, s'_2) \in L_{S'_1} \times L_{S'_2}$, $\Gamma_{S'_1 \| S'_2}(s) = \Gamma_S(s)$. Os ítens (ii) e (iii) são derivados da mesma forma. □

O lema 6.3 traz relações para os subsistemas das palavras correspondentes de S, S'_1 e S'_2, que são usadas no teorema a seguir. Enfim, o teorema 6.1 afirma que o produto síncrono das abstrações Q_1 e Q_2 é exatamente igual à máxima abstração consistente de S e.r.a. Σ_r.

Teorema 6.1 (Supor-garantir) $Q_1 \| Q_2 = \langle S_1 \| S_2, \Sigma_r \rangle$.

Demonstração. Seja $Q = Q_1 \| Q_2$ e $P = \langle S_1 \| S_2, \Sigma_r \rangle$, prova-se que $L_Q = L_P$ e, para todo t em L_Q, $\Gamma_Q(t) = \Gamma_P(t)$.

Primeiro, note-se que $L_Q = L_{Q_1} \cap L_{Q_2}$, pois Q_1 e Q_2 possuem o mesmo alfabeto. Prova-se a seguir que $L_{Q_1} = L_P$: $L_{Q_1} = \theta_{1e}(L_{S_1} \| L_{P'_2}) = \theta_{1e}(q_1^{-1}(L_{S_1}) \cap h_2^{-1}(\theta_2(L_{S_2})))$ $= \tilde{\theta}_{1e}(q_1^{-1}(L_{S_1})) \cap \tilde{\theta}_{1e}(h_2^{-1}(\theta_2(L_{S_2}))) = l_1^{-1}(\theta_1(L_{S_1})) \cap l_2^{-1}(\theta_2(L_{S_2})) = L_{P_1} \| L_{P_2} = L_P$. Similarmente pode-se provar que $L_{Q_2} = L_P$, e, como conseqüência, $L_Q = L_{Q_1} \cap L_{Q_2} = L_P = \theta(L_S)$.

Segundo, como $L_Q = L_{Q_1} = L_{Q_2}$, para todo $t \in L_Q$, $\Sigma_{L_Q}(t) = \Sigma_{L_{Q_1}}(t) = \Sigma_{L_{Q_2}}(t)$. Porém, pela definição da estrutura de controle de Q, para todo $t \in L_Q$ e $(\gamma_r, \#_r) \in \Gamma_Q(t)$, $\gamma_r \subseteq 2^{\Sigma_{L_Q}(t)} \times \{M, N\}$. Por outro lado, pela definição da estrutura de controle de Q_i, $i \in \{1, 2\}$, para todo $t \in L_{Q_i}$ e $(\gamma_r, \#_r) \in \Gamma_{Q_i}(t)$, $\gamma_r \subseteq 2^{\Sigma_{L_{Q_i}}(t)} \times \{M, N\}$. Então, pela definição do produto síncrono, para todo $t \in L_Q$, $\Gamma_Q(t) = \Gamma_{Q_1}(t) \cap \Gamma_{Q_2}(t)$.

Pela definição 6.4, afirmar que $t \in L_P$ e $(\gamma_r, \#_r) \in \Gamma_P(t)$ é equivalente a afirmar que, para todo $s \in \theta_{voc}^{-1}(t)$, há $(\gamma'_r, \#'_r) \in \Gamma_{S,voc}(s)$ tal que $\gamma_r = \gamma'_r \cap \Sigma_{S,voc}(s)$ e $\#_r = \#'_r$. Pela definição 5.2, isto é equivalente a dizer que, para todo $s \in \theta_{voc}^{-1}(t)$, há uma linguagem não vazia $K \subseteq L_{S(s)}$ tal que (i) K é $(L_{S(s)}, \Gamma_{S(s)})$-compatível; (ii) para todo $u \in L_{S,voc}(s)$, $w_e(su) \in \gamma_r$ se e somente se $u \in K$; (iii) se $\#_r = N$, então $K - L_{S,voc}(s) = \emptyset$; e (iv)

se $\#_r = M$, então $K - L_{S,voc}(s) \neq \emptyset$. Por sua vez, sejam $t \in L_Q$ e $(\gamma_r, \#_r) \in \Gamma_Q(t)$. Como visto acima, $\Gamma_Q(t) = \Gamma_{Q_1}(t) \cap \Gamma_{Q_2}(t)$, então $(\gamma_r, \#_r) \in \Gamma_{Q_i}(t)$, com $i \in \{1, 2\}$. Novamente, pela definição 6.4 isto é equivalente a afirmar que, para todo $s'_i \in \theta^{-1}_{ie,voc}(t)$, há $(\gamma'_r, \#'_r) \in \Gamma_{S'_i,voc}(s'_i)$ tal que $\gamma_r = \gamma'_r \cap \Sigma_{S'_i,voc}(s'_i)$ e $\#_r = \#'_r$. Da mesma forma, pela definição 5.2, reescreve-se a condição anterior como sendo, para todo $s'_i \in \theta^{-1}_{ie,voc}(t)$, há uma linguagem não vazia $K_i \subseteq L_{S'_i(s'_i)}$ tal que: (i) K_i é $(L_{S'_i(s'_i)}, \Gamma_{S'_i(s'_i)})$-compatível; (ii) para todo $u_i \in L_{S'_i,voc}(s'_i)$, $w_{ie}(s'_i u_i) \in \gamma_r$ se e somente se $u_i \in K_i$; (iii) se $\#_r = N$, então $K_i - L_{S'_i,voc}(s'_i) = \emptyset$; e (iv) se $\#_r = M$, então $K_i - L_{S'_i,voc}(s'_i) \neq \emptyset$.

Primeiro, sejam $t \in L_Q$, $(\gamma_r, \#_r) \in \Gamma_Q(t)$, $s \in \theta^{-1}_{voc}(t)$ e $(s'_1, s'_2) \in \theta^{-1}_{1e,voc}(t) \times \theta^{-1}_{2e,voc}(t)$ correspondentes a s (lema 6.1). Defina-se a linguagem $K = K_1 \| K_2$. A seguir, enumeram-se algumas propriedades de K:

1. $K \neq \emptyset$, uma vez que K_1 e K_2 são não vazios e os eventos de sincronização de K_1 e K_2 são os eventos relevantes.

2. A projeção $m_i(K) = K_i$ pela mesma razão que acima. Assim, para todo $u \in \overline{K}$ há um único par correspondente $(u_1, u_2) \in \overline{K_1} \times \overline{K_2}$ para o qual $m_i(u) = u_i$.

3. Sejam $u \in K$ e par correspondente $(u_1, u_2) \in K_1 \times K_2$. Como K_i é $(L_{S'_i(s'_i)}, \Gamma_{S'_i(s'_i)})$-compatível, para todo $u \in K_i$, há um controle $(\gamma_i, M) \in \Gamma_{S'_i(s'_i)}(u_i)$ tal que $\gamma_i \cap \Sigma_{LS'_i(s'_i)}(u_i) = \Sigma_{K_i}(u_i)$, com $i \in \{1, 2\}$. Considere o caso não trivial onde u não termina com um evento relevante. Para o caso trivial, uma vez que u termina com evento relevante, u está em $L_{S,voc}(s)$ e há o controle $(\emptyset, M) \in \Gamma_{S(s),voc}(u)$. Seja então $u \in K$ e $(\gamma, \#) \in \Gamma_{S(s)}(u)$ correspondendo a $(\gamma_i, \#_i) \in \Gamma_{S'_i(s'_i)}(u_i)$, isto é, $\gamma = \gamma_1 \cup \gamma_2$ e $\# = \#_1 \wedge \#_2$. Como $\gamma_i \subseteq \Sigma_i \cup \Sigma_{ir}$, escreve-se $\gamma = [\gamma_1 \cap (\Sigma_1 - \Sigma_{1r})] \cup [\gamma_2 \cap (\Sigma_2 - \Sigma_{2r})] \cup [\gamma_1 \cap \gamma_2 \cap \Sigma_r]$. Escreve-se então $\gamma \cap \Sigma_{L_{S(s)}}(u)$ $= (\gamma_1 \cup \gamma_2) \cap ((\Sigma_{L_{S'_1(s'_1)}}(u_1) \cap (\Sigma_1 - \Sigma_{1r})) \cap (\Sigma_{L_{S'_2(s'_2)}}(u_2) \cap (\Sigma_2 - \Sigma_{2r})) \cap (\Sigma_{L_{S'_1(s)}}(u_1) \cap \Sigma_{L_{S'_2(s'_2)}}(u_2) \cap \Sigma_r) = (\gamma_1 \cap \Sigma_{L_{S'_1(s'_1)}}(u_1) \cap (\Sigma_1 - \Sigma_{1r})) \cup (\gamma_2 \cap \Sigma_{L_{S'_2(s'_2)}}(u_2) \cap (\Sigma_2 - \Sigma_{2r}))$ $\cup ((\gamma_1 \cap \Sigma_{L_{S'_1(s'_1)}}(u_1)) \cap (\gamma_2 \cap \Sigma_{L_{S'_2(s'_2)}}(u_2)) \cap (\Sigma_1 - \Sigma_{1r})) = (\Sigma_{K_1}(u_1) \cap (\Sigma_1 - \Sigma_{1r}))$ $\cup (\Sigma_{K_2}(u_2) \cap (\Sigma_2 - \Sigma_{2r})) \cup (\Sigma_{K_1}(u_1) \cap \Sigma_{K_2}(u_2) \cap \Sigma_r) = \Sigma_K(u)$. Assim, para todo $u \in K$, há $(\gamma, M) \in \Gamma_{S(s)}(u)$ tal que $\gamma \cap \Sigma_{LS(s)}(u) = \Sigma_K(u)$.

4. Para todo $u \in \overline{K} - K$, há um único par correspondente $(u_1, u_2) \in [\overline{K_1} - K_1] \times [\overline{K_2} - K_2]$ tal que $m_i(u) = u_i$. Como K_i é $(L_{S_i'(s_i')}, \Gamma_{S_i'(s_i')})$-compatível, há um controle $(\gamma_i, N) \in \Gamma_{S_i'(s_i')}(u_i)$ tal que $\gamma_i \cap \Sigma_{L_{S_i'(s_i')}}(u_i) = \Sigma_{K_i}(u_i)$. Assim, para todo $u \in \overline{K} - K$, pelo mesmo argumento do ítem 3, há um controle $(\gamma, N) = (\gamma_1 \cup \gamma_2, N) \in \Gamma_{S(s)}$ tal que $\gamma \cap \Sigma_{L_{S(s)}}(u) = \Sigma_K(u)$.

5. Pelos ítens 3 e 4 acima, prova-se que K é $(L_{S(s)}, \Gamma_{S(s)})$-compatível.

6. Sejam $u \in L_{S,voc}(s)$ e o par correspondente $(u_1, u_2) \in L_{S_1',voc}(s_1') \times L_{S_2',voc}(s_2')$. Para todo $u_i \in L_{S_i',voc}(s_i')$, se $w_{ie}(s_i'u_i) \in \gamma_r$, então $u_i \in K_i$. Seja $u \in L_{S,voc}(s)$ e $w(su) \in \gamma_r$, então, para $u_i \in L_{S_i',voc}(s_i')$ correspondente a u, $w_{1e}(s_1'u_1) \in \gamma_r$ **e** $w_{2e}(s_2'u_2) \in \gamma_r$. Pela hipótese, $u_1 \in K_1$ **e** $u_2 \in K_2$. Assim, como $K = K_1 \| K_2$, $u \in K$.

7. Se $\#_r = N$, pela hipótese, $K_i - L_{S_i',voc}(s_i') = \emptyset$ para $i \in \{1, 2\}$. Então, todo $u_i \in K_i$, u_i termina num evento relevante. Como $K = K_1 \| K_2$, para todo $u \in K$, u também termina com um evento relevante. Assim, conclui-se que $K - L_{S,voc}(s) = \emptyset$.

8. Se $\#_r = M$, pela hipótese, $K_i - L_{S_i',voc}(s_i') \neq \emptyset$ para $i \in \{1, 2\}$. Então, há pelo menos um u_i em K_i que não termina com um evento relevante, para ambos $i = 1$ e $i = 2$. Seja (u_1, u_2) um par de tais palavras em $K_1 \times K_2$ e a palavra correspondente u em K. Assim u não termina com um evento relevante, como conseqüência, $K - L_{S,voc}(s) \neq \emptyset$.

Pela enumeração de propriedades de K acima, o controle $(\gamma_r, \#_r)$ está em $\Gamma_P(t)$ e isto completa a primeira parte da prova.

Segundo, sejam t em L_P, $(\gamma_r, \#_r)$ em $\Gamma_P(t)$, s em $\theta_{voc}^{-1}(t)$ e K em $L_{S(s)}$ correspondendo ao controle $(\gamma_r, \#_r)$. Defina-se $K_i = m_i(K)$, com i em $\{1, 2\}$. Enumeram-se, a seguir, propriedades de K_i:

1. K_i é não vazia, uma vez que K é não vazia e a projeção m_i apenas apaga eventos em $\Sigma_j - \Sigma_{rj}$ das palavras de K, com j em $\{1, 2\}$ e j diferente de i.

2. $K_1 \| K_2 = m_1^{-1}(K_1) \cap m_2^{-1}(K_2) = K$, e ainda, para toda palavra u de K existe um único par correspondente (u_1, u_2) em $K_1 \times K_2$.

3. Sejam $u_i \in K_i$ e qualquer $u \in K$ correspondente, e considere o caso não-trivial que u_i (e u) não terminam por um evento relevante (veja explicação no ítem 3 da enumeração da primeira parte da prova). Pelos lemas 6.1 e 6.3, todo controle (γ, M) em $\Gamma_{S(s)}(u)$ é tal que $\gamma = \gamma_1' \dot\cup \gamma_2' \dot\cup \gamma_r$, onde $\gamma_i' \subseteq \Sigma_i - \Sigma_{ir}$, $\gamma_r \subseteq \Sigma_r$ e $\dot\cup$ denota a união disjunta. Também pelos lemas 6.1 e 6.3, afirma-se que há (γ_i, M) em $\Gamma_{S_i'(s_i')}(u_i)$ tal que $\gamma_i = \gamma_i' \dot\cup \Sigma_r$. Assim, escreve-se que há um controle (γ_i, M) em $\Gamma_{S_i'(s_i')}(u_i)$ tal que $\gamma_i \cap \Sigma_{L_{S_i'(s_i')}}(u_i) = (\gamma_i' \dot\cup \gamma_r) \cap ((\Sigma_{L_{S(s)}}(u) \cap (\Sigma_i - \Sigma_{ri})) \dot\cup (\Sigma_{L_{S(s)}}(u) \cap \Sigma_r)) = (\gamma_i' \cap \Sigma_{L_{S(s)}}(u) \cap (\Sigma_i - \Sigma_{ir})) \dot\cup (\gamma_r \cap \Sigma_{L_{S(s)}}(u) \cap \Sigma_r) = (\Sigma_K(u) \cap (\Sigma_i - \Sigma_{ir}) \dot\cup (\Sigma_K(u) \cap \Sigma_r) = \Sigma_{K_i}(u_i)$. Assim, para todo u_i em K_i há um controle (γ_i, M) em $\Gamma_{S_i'(s_i')}(u_i)$ tal que $\gamma_i \cap \Sigma_{L_{S_i'(s_i')}}(u_i) = \Sigma_{K_i}(u_i)$.

4. Sejam agora $u_i \in \overline{K_i} - K_i$ e qualquer $u \in K$ correspondente. Novamente, pelos lemas 6.1 e 6.3, todo controle (γ, N) em $\Gamma_{S(s)}(u)$ é tal que $\gamma = \gamma_1' \dot\cup \gamma_2' \dot\cup \gamma_r$, onde $\gamma_i' \subseteq \Sigma_i - \Sigma_{ir}$ e $\gamma_r \subseteq \Sigma_r$. Ainda, há um controle (γ_i, N) em $\Gamma_{S_i'(s_i')}(u_i)$ tal que $\gamma_i = \gamma_i' \dot\cup \Sigma_r$. Conforme o desenvolvimento do ítem 3 acima, conclui-se que para todo u_i em $\overline{K_i} - K_i$ há um controle (γ_i, N) em $\Gamma_{S_i'(s_i')}(u_i)$ tal que $\gamma_i \cap \Sigma_{L_{S_i'(s_i')}}(u_i) = \Sigma_{K_i}(u_i)$.

5. A partir dos ítens 3 e 4 acima, conclui-se que K_i é $(L_{S_i'(s_i')}, \Gamma_{S_i'(s_i')})$-compatível.

6. Sejam u_i em $L_{S_i',voc}(s_i')$ em $u \in L_{S,voc}(s)$ uma palavra correspondente a u_i, pelo lema 6.2. Se $w_{ie}(s_i' u_i) \in \gamma_r$, então u_i termina com um evento relevante. Então, u termina com o mesmo evento relevante que u_i. Então, por hipótese, $w(su) \in \gamma_r$ e $u \in K$. Como $K = K_1 \| K_2$ e u e u_i terminam com o mesmo evento relevante, conclui-se que $u_i \in K_i$.

7. Se $\#_r = N$, para as palavras s em S correspondentes a s_i', lema 6.1, tem-se $K - L_{S,voc}(s) = \emptyset$. Como a projeção m_i apenas apaga eventos em $\Sigma_j - \Sigma_{jr}$ de palavras em Σ^*, tem-se $K_i - L_{S_i',voc}(s_i') = m_i(K - L_{S,voc}(s)) = \emptyset$.

8. Se $\#_r = M$, para as palavras correspondentes a s'_i em S, diga-se s, lema 6.1, tem-se $K - L_{S,voc}(s) \neq \emptyset$. E obtém-se $K_i - L_{S'_i,voc}(s'_i) \neq \emptyset$ da mesma forma que acima.

Pela enumeração de propriedades de K_i acima, conclui-se que $(\gamma_r, \#_r)$ pertence a $\Gamma_Q(t)$, o que completa a prova. □

O teorema 6.1 permite estabelecer um método baseado no raciocínio supor-garantir para a construção de abstrações consistentes por composição. Dados os SEDs S_1 e S_2 ambos sobre os alfabetos Σ_1 e Σ_2, e um conjunto de eventos relevantes $\Sigma_r \subseteq \Sigma_1 \cup \Sigma_2$, com conjuntos de eventos relevantes locais $\Sigma_{ri} = \Sigma_r \cap \Sigma_i$, para i igual a 1 ou 2, enumeram-se os passos do método a seguir:

1. Construir as máximas abstrações consistentes $P_i = \langle S_i, \Sigma_{ri} \rangle$.
2. Construir os ambientes P'_i: $L_{P'_i} = L_{P_i}$ e,

$$\Gamma_{P'_i}(t) \;=\; 2^{\Sigma_{L_{P'_i}}(t)} \times \{M, N\}, \tag{6.8}$$

   para todo $t \in L_{P'_i}$.
3. Construir os sistemas expandidos $S'_i = S_i \| P'_j$, com $j \in \{1, 2\}$ e $j \neq i$.
4. Construir as abstrações $Q_i = \langle S'_i, \Sigma_r \rangle$.
5. Construir a composição $Q = Q_1 \| Q_2$.

Pelo teorema 6.1 garante-se que $Q = \langle S, \Sigma_r \rangle$, onde $S = S_1 \| S_2$. A vantagem sobre o método que usa abstrações confiáveis é a obtenção da máxima abstração consistente do sistema composto, que é a abstração com informação de controle mais refinada o possível para síntese de alto nível, dados o sistema composto e o conjunto de eventos relevantes. Uma desvantagem é que possivelmente os sistemas envolvidos nos passos intermediários possuam espaço de estados maiores que os sistemas nas abstrações confiáveis.

Note-se que nesta abordagem usando o raciocínio supor-garantir, a condição adicional para quebrar a circularidade do raciocínio, $S_i \subseteq P_i$ na equação (6.6), é implícita na construção dos ambientes P_i' a partir das máximas abstrações consistentes P_i, com i igual a 1 ou 2.

Exemplo 6.5 (Supor-garantir) *Para ilustrar os resultados desta seção, considere o sistema trivial do exemplo 6.3. Na figura 6.6, P_i' é obtido a partir de P_i, $i \in \{1,2\}$, marcando-se todos os estados e fazendo-se todos os eventos controláveis. Q_i é obtido pelo cálculo da máxima abstração consistente de $S_i' = S_i \| P_j'$ e.r.a. Σ_r, $j \in \{1,2\}$ e $j \neq i$. Verifica-se que o produto $Q_1 \| Q_2$ é igual à máxima abstração consistente de S e.r.a. Σ_r, P na figura 6.2. Como observação final, os conjuntos de controle para os estados de $Q_1 \| Q_2$ são obtidos simplesmente fazendo a interseção dos conjuntos de controle dos estados correspondentes de Q_1 e Q_2, teorema 6.1.*

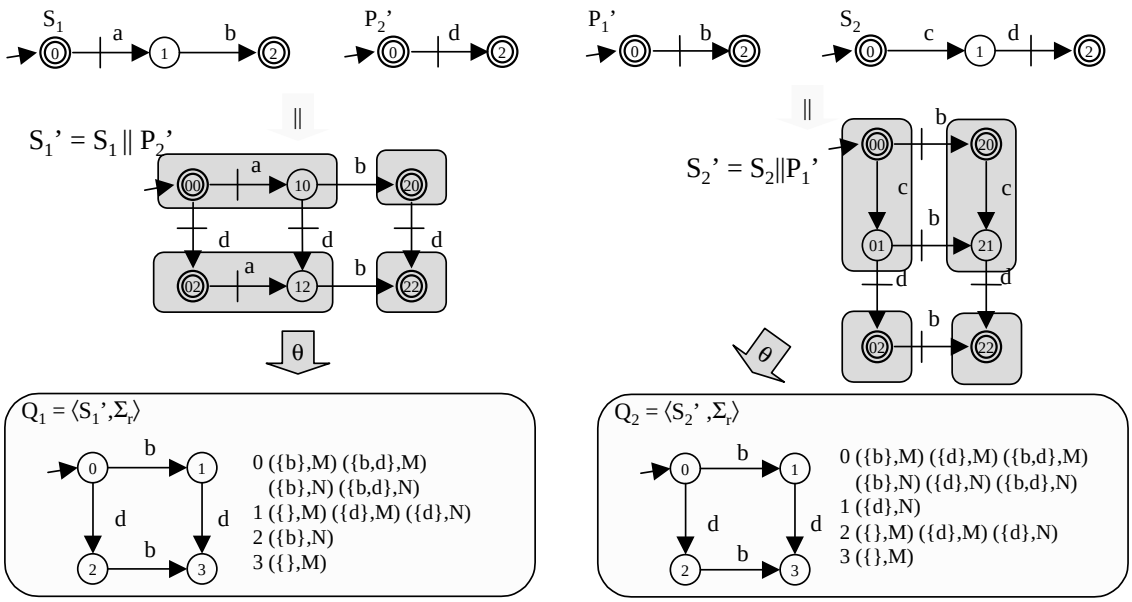


Figura 6.6: Raciocínio supor-garantir para o sistema trivial.

6.5 O projeto de um piloto automático inteligente para um automóvel

Nesta seção apresenta-se um exemplo baseado numa aplicação real, para ilustrar o método supor-garantir proposto.

Considere o projeto de um piloto automático adaptativo cooperativo (PAAC)[2] para um automóvel (Girard et al. 2001). Na plataforma experimental, ilustrada pelo diagrama na figura 6.7, os carros *líder* e *seguidor* são equipados com radar e rádio. No sistema PAAC, o carro seguidor inicialmente navega automaticamente mantendo uma velocidade determinada pelo motorista; se o carro líder for detectado pelo radar e não houver disponibilidade de comunicação via radio, o carro seguidor ou mantém a velocidade determinada pelo motorista, ou mantém um certo intervalo de tempo para o carro líder, onde o intervalo de tempo corresponde ao tempo medido, por um observador fixo na estrada, entre a passagem da traseira do carro líder e a dianteira do carro seguidor (Girard et al. 2001). Ainda, se a comunicação entre os carros estiver disponível, os carros passam a coordenar entre si as mudanças de velocidade, e o carro seguidor passa a seguir o carro líder mantendo um intervalo de tempo menor que para o caso sem comunicação. No diagrama da figura 6.7 também são mostrados os componentes do sistema PAAC. A interface discreta do piloto automático é um componente que encapsula a dinâmica do carro, informação sensorial do radar e do rádio e o piloto automático com as suas leis de controle. O protocolo modela a comunicação entre os carros, compreendendo a troca de mensagens e o status da comunicação. O propósito do coordenador é sincronizar a operação dos dois componentes segundo especificações de coordenação.

Mostra-se o SED S_1 para a interface discreta do piloto automático na figura 6.8. S_1 possui seis estados correspondendo a cinco leis de controle para o piloto automático, descritas a seguir. Os estados 1 e 2 representam as leis de controle *manter velocidade* e *manter distância*, respectivamente, onde o carro seguidor ou mantém a velocidade

[2]Originalmente em inglês *cooperative adaptive cruise controller* (CACC).

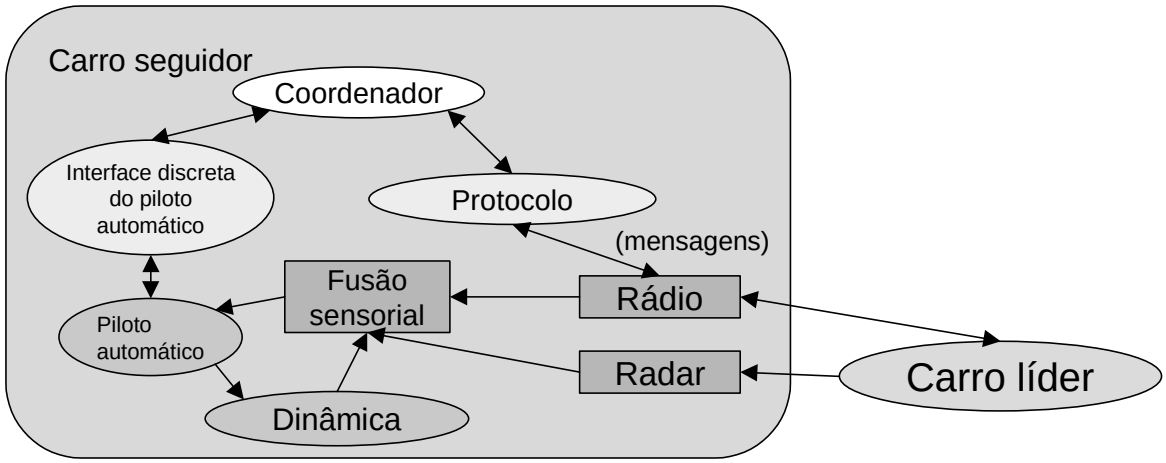


Figura 6.7: Componentes do PAAC.

desejada pelo motorista, ou mantém uma distância do carro líder, função da velocidade do carro e do intervalo de tempo desejado. No estado 0, o sistema decide qual das leis anteriores vai aplicar. O estado 3 corresponde à lei de controle *aproximar* (original *join*) onde, seguindo um perfil de velocidade pré-calculado, o carro seguidor se aproxima do carro líder até uma distância adequada para segui-lo. O estado 4 corresponde a lei de controle *seguir* (original *follow*), onde o carro seguidor segue o carro líder a uma pequena distância e os carros cooperam entre si trocando informações detalhadas das mudanças de velocidade realizadas. Por fim, o estado 5 corresponde a lei de controle *afastar* (original *split*) onde, também seguindo um perfil de velocidade, o carro seguidor se afasta do carro líder até uma distância correspondente ao modo não cooperativo. Detalhes de descrição e implementação destas leis de controle não estão disponíveis em (Girard et al. 2001), e a descrição acima baseia-se nas leis de controle longitudinais em (Godbole e Lygeros 1994). S_1 é dotado com a estrutura de controle RW padrão, e as informações de controlabilidade de eventos e marcação de estados estão indicadas no diagrama da figura 6.8. Da figura, os eventos com prefixo $inicio\_$ são controláveis e os eventos com prefixo $fim\_$ e $limite\_$ são incontroláveis, ao passo que os estados 0, 1, 2 e 4 são considerados tarefas completas do sistema. Neste modelo não se consideram as

ações em caso de falha numa manobra ou ações em caso de falha nas comunicações.

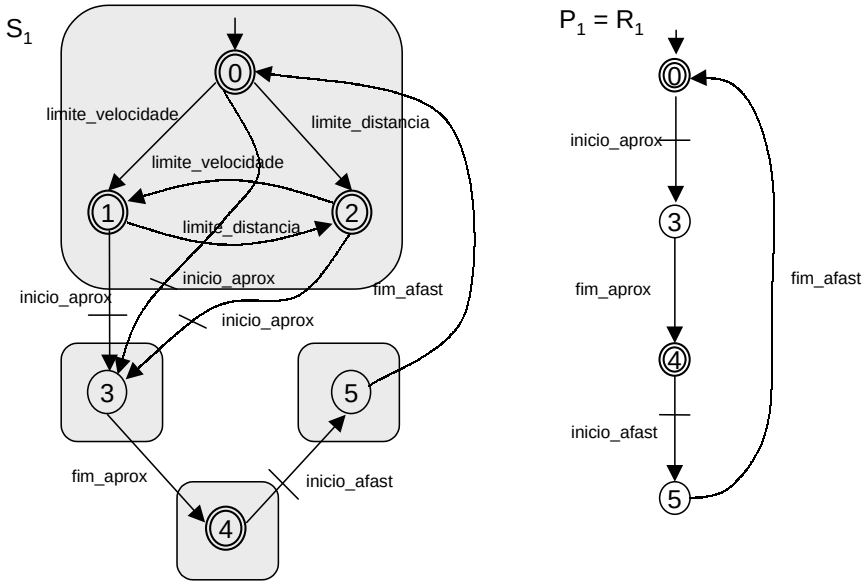


Figura 6.8: Interface discreta do piloto automático.

O SED S_2 para o protocolo de comunicações está na figura 6.9. S_2 modela os protocolos para as manobras de aproximação e afastamento dos carros. Para as duas manobras o protocolo prevê os seguintes passos: requerimento de uma manobra (eventos com prefixo $req\_$), espera para aceitação e não aceitação de manobra pelo carro líder (eventos com prefixo $ac\_$ e $nac\_$) e espera para conclusão da manobra (eventos com prefixo $conf\_$). A manobra de afastamento pode ser solicitada tanto pelo carro seguidor quanto pelo carro líder, neste caso evento $conv\_afast$. Este modelo para protocolo está baseado no projeto apresentado em (Hsu et al. 1993). O teste de comunicação e a possibilidade de falha na comunicação são omitidas no modelo. A estrutura de controle para S_2, conforme a abordagem RW, está indicada na figura 6.9.

O sistema composto resultante do produto síncrono da interface discreta do piloto automático e do protocolo de comunicações $S = S_1 \| S_2$ possui 42 estados, 130 transições envolvendo 15 eventos. Na transcrição de S para o modelo apresentado no capítulo

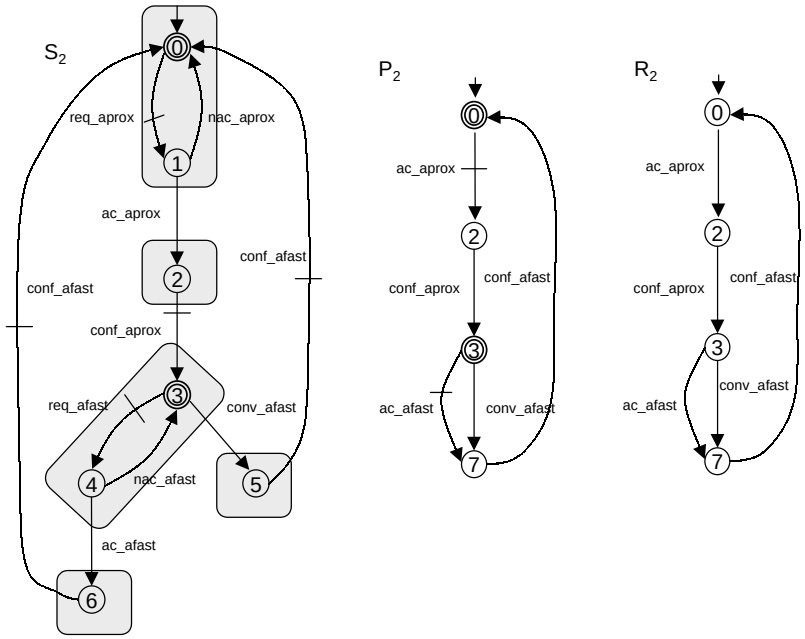


Figura 6.9: Protocolo de comunicações.

4, o número máximo de controles por estado de S é 8. Denotam-se estas informações sobre o tamanho de um SED com marcação flexível por (42, 130, 15, 8). No lugar de se optar em fazer uma síntese monolítica para as especificações de coordenação, aplica-se a síntese hierárquica, conforme o capítulo 5, usando-se apenas um conjunto de eventos relevantes para a coordenação. Consideram-se eventos relevantes para coordenação os seguintes (Σ_r): para a interface discreta do piloto automático (Σ_{r1}), os eventos $inicio\_aprox$, fim_aprox, $inicio\_afast$ e fim_afast, e para o protocolo (Σ_{r2}), os eventos ac_aprox, $conf\_aprox$, ac_afast, $conv\_afast$ e $conf\_afast$. Um exemplo de conjunto de especificações de coordenação está disposto na figura 6.10. A máxima abstração consistente de S e.r.a. Σ_r, $P = \langle S, \Sigma_r \rangle$, possui tamanho (20, 40, 9, 6). P não é mostrado pelo seu número de estados, mas afirma-se que P não possui estrutura de controle padrão RW. Para a especificação $E = E_1 \| E_2 \| E_3 \| E_4$, oriunda das composição das especificações da figura 6.10, o supervisor de alto nível calculado a partir de P possui 8 estados e 9 transições.

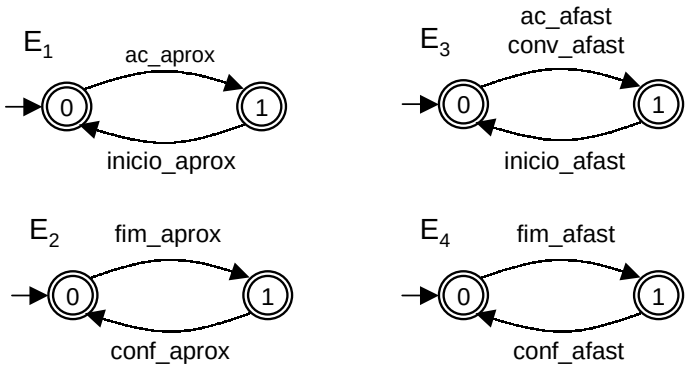


Figura 6.10: Algumas especificações de coordenação.

Nas figuras 6.8 e 6.9 mostram-se as máximas abstrações consistentes de S_1 e S_2 e.r.a. os conjuntos individuais de eventos relevantes, quais sejam respectivamente $P_1 = \langle S_1, \Sigma_{1r} \rangle$ e $P_2 = \langle S_2, \Sigma_{2r} \rangle$. Verifica-se que P e $P_1 \| P_2$ não são comparáveis, com isto, a composição $P_1 \| P_2$ não é uma abstração consistente de S e.r.a. Σ_r. De forma alternativa, pode-se calcular as máximas abstrações confiáveis, $R_1 = [S_1, \Sigma_{1r}]$ e $R_2 = [S_2, \Sigma_{2r}]$, também indicadas respectivamente nas figuras 6.8 e 6.9. Verifica-se que $R_1 \| R_2 \leq P$, isto é, o produto $R_1 \| R_2$ é uma abstração consistente de S. Entretanto, conforme antes citado, $R_1 \| R_2$ é uma abstração consistente grosseira para o sistema, no sentido que possui fraca informação de controle para síntese de supervisores no alto nível. De fato, a especificação E citada anteriormente não é $(L_{R_1 \| R_2}, \Gamma_{R_1 \| R_2})$-compatível e a máxima linguagem $(L_{R_1 \| R_2}, \Gamma_{R_1 \| R_2})$-compatível contida em E é vazia.

Agora, considere a aplicação do método baseado no raciocínio supor-garantir proposto na seção 6.4. Construam-se os sistemas P_1' e P_2', marcando-se todos os estados de P_1 e P_2, respectivamente, e fazendo todos os eventos controláveis. Construam-se os sistemas $S_1' = S_1 \| P_2'$ (24,70,11,16) e $S_2' = S_2 \| P_1'$ (28,68,13,8), e as máximas abstrações consistentes $Q_1 = \langle S_1 \| E_2, \Sigma_r \rangle$ (10,36,9,16) e $Q_2 = \langle S_2 \| E_1, \Sigma_r \rangle$ (24,56,9,4). Calculando-se o produto síncrono $Q_1 \| Q_2$, tem-se que $Q_1 \| Q_2 = P$, isto é, $Q_1 \| Q_2$ é a máxima abstração consistente de $S_1 \| S_2$ e.r.a. Σ_r.

Note-se que todos os sistemas envolvidos no método proposto possuem espaço de estados menor que S e o resultado do método é a máxima abstração consistente de S e.r.a. Σ_r, diferentemente da abordagem usando abstrações confiáveis. De forma geral, os sistemas compostos tratados no método supor-garantir possuem menor espaço de estados que o sistema composto completo. Isso se torna vantajoso numa extensão dos resultados para sistemas com mais de dois componentes, onde o número de estados do sistema composto cresce exponencialmente com o número de componentes.

Os cálculos do exemplo aqui apresentado foram feitos utilizando-se a ferramenta computacional para controle hierárquico de SEDs baseada na ferramenta **Grail**, desenvolvida neste trabalho, vide apêndice A.

6.6 Conclusões

Neste capítulo apresenta-se um método baseado no raciocínio supor-garantir para a construção de abstrações consistentes por composição. Tal método se aplica para construção de abstrações consistentes para o controle hierárquico de sistemas compostos, quando não for interessante construir o sistema composto completo pelo tamanho do seu espaço de estados.

Sistemas compostos no controle hierárquico são também tratados em (Wonham e Zhong 1990) e (Pu 2000). Em (Wonham e Zhong 1990), afirma-se que os mapas repórteres dos sistemas componentes do baixo nível devem atender a certa condição chamada *livre de atraso*, no original *delay free*. Entretanto nenhum método construtivo é sugerido para transformar um mapa repórter em livre de atraso, e a condição só é válida para sistemas modelados por linguagens prefixo fechadas. Em (Pu 2000), introduz-se o conceito de abstrações confiáveis. No modelo de Pu (2000), a condição de existência de uma abstração confiável é que o mapa repórter seja um observador. Na tradução das abstrações confiáveis para o modelo usado neste trabalho, provou-se que a condição de mapa repórter observador não é necessária para garantir a existência das abstrações confiáveis, e mais, que sempre existe uma abstração confiável dado um

sistema e um alfabeto de eventos relevantes. Comparando-se o método baseado no raciocínio supor-garantir com as abordagens anteriores, a vantagem é que a abstração obtida é a máxima abstração consistente, isto é, leva-se ao alto nível o máximo de informação de controle para síntese de supervisores. Uma potencial desvantagem é o tamanho maior dos espaços de estados dos sistemas envolvidos nos passos intermediários.

Neste capítulo trabalha-se com dois sistemas componentes apenas, vide seção 6.3. Como ponto aberto para o desenvolvimento encontra-se a generalização do resultado para mais de dois sistemas componentes. Com base em resultados semelhantes na literatura para o raciocínio supor-garantir, como em (Stark 1985), (Alur e Henzinger 1999) e (Maier 2001), acredita-se que a extensão seja direta. Possivelmente na generalização do método, ao invés do ambiente de um sistema ser a modificação da abstração consistente de um outro sistema, o ambiente será a modificação do produto das abstrações consistentes dos outros sistemas componentes.

Tem-se por hipótese que o sistema composto é um sistema produto, isto é, os componentes não compartilham eventos, vide seção 6.3. Um ponto de extensão é verificar a validade dos resultados para sistemas compostos onde os componentes compartilham eventos. Acredita-se que o resultado do método, sob esta nova hipótese, ainda seja uma abstração consistente do sistema composto. Entretanto o resultado ótimo, isto é, a obtenção da máxima abstração consistente, pode não acontecer em função da sincronização dos eventos comuns ao se compor os sistemas componentes.

Neste capítulo trabalhou-se apenas com o conceito de abstrações consistentes, que estão relacionadas à consistência hierárquica. Outro ponto em aberto é a derivação dos resultados para o caso da consistência hierárquica forte. É necessário criar um esquema para combinação das diferentes renomeações de eventos feitas durante a construção das abstrações para os sistemas componentes expandidos. Entretanto, não é óbvio que se obtenha, por composição, um sistema de alto nível com consistência hierárquica forte.

Muito embora os resultados deste capítulo estejam instanciados para o SED com marcação flexível do capítulo 4, pode-se provar que os resultados são válidos para

o modelo RW padrão. A condição é que se lance mão de um método para construir hierarquias consistentes usando exclusivamente estruturas de controle RW padrão. Como foi visto nos capítulos anteriores, ou verificando as outras abordagens de controle hierárquico, veja Zhong e Wonham (1990), Wong e Wonham (1996a), Pu (2000), Hubbard e Caines (2002) e a construção de hierarquias de controle com consistência hierárquica envolve diferentes conjuntos de condições, a maioria sem indicação de um método construtivo para que as mesmas sejam impostas sobre um dado sistema, havendo fundamentalmente condições *ad hoc*. A proposta de controle hierárquico desta tese, além de desconsiderar a necessidade das condições impostas em (Zhong e Wonham 1990), (Wong e Wonham 1996a), (Pu 2000) e (Hubbard e Caines 2002), aponta de forma clara um método construtivo para a construção de hierarquias de controle de SEDs com consistência hierárquica.

Capítulo 7

Conclusões

Divide-se a exposição deste capítulo em quatro seções. Na seção 7.1 descrevem-se, de forma sumária, os pontos abordados e as principais contribuições desta tese. Na seção 7.2 comentam-se as limitações e pontos não abordados. Finalmente, na seção 7.3 propõem-se alguns pontos de pesquisa que são considerados como possíveis extensões desta tese.

7.1 Pontos abordados e principais contribuições

De forma sumária, os pontos abordados e as principais contribuições deste trabalho são os seguintes.

Fez-se uma revisão do controle hierárquico de SEDs dentro da abordagem RW, levantando-se quais são os principais problemas tratados e comparando-se os resultados das principais abordagens da literatura.

Desenvolveu-se uma nova classe de SEDs dotados de controle, chamados SEDs com marcação flexível. Os SEDs com marcação flexível objetivam-se a representar adequadamente os aspectos que surgem quando da abstração de um SED para o controle hierárquico.

Propôs-se uma nova abordagem para o controle hierárquico de SEDs, baseado nos SEDs com marcação flexível, onde garante-se consistência hierárquica, em qualquer de

suas gradações, de forma direta, por construção, e sem levar em conta nenhuma das condições impostas pelas outras abordagens de controle hierárquico.

Considerando a abordagem de controle hierárquico de SEDs proposta, apresentou-se um método fundamentado no raciocínio supor-garantir para construção da hierarquia de controle de SEDs quando se consideram sistemas compostos.

Desenvolveu-se uma ferramenta para implementação computacional dos modelos e algoritmos da abordagem para o controle hierárquico de SEDs proposta.

7.2 Limitações e pontos não abordados

Nesta seção, são feitos comentários sobre algumas limitações deste trabalho.

Como crítica à abordagem de controle hierárquico em geral, comenta-se que, conforme exposto na seção 5.6, existe a possibilidade de que o sistema do gerente possua número de estados maior que o sistema do operador. Este problema é comum a todas as abordagens para o controle hierárquico. É necessária uma forma para se determinar a escolha de eventos relevantes para que esta situação possa ser controlada. Entretanto, foi observada de forma geral nos exemplos tratados uma redução substancial no número de estados do sistema do operador para o sistema do gerente.

Uma crítica para a abordagem de controle hierárquico apresentada neste trabalho é a utilização do SED com marcação flexível. Tal modelo é mais abstrato e complexo, em termos de estruturas para sua representação, que o modelo da abordagem RW padrão. Um argumento em favor do SED com marcação flexível é que este se mostra mais adequado para o controle hierárquico que o modelo RW padrão. Isso porque modela naturalmente a abstração dos sistemas e permite obter consistência hierárquica de forma mais direta. É pela utilização do SED com marcação flexível que se contornam as condições impostas quando se força que os modelos nos diversos níveis hierárquicos sejam o RW padrão (Zhong e Wonham 1990, Wong e Wonham 1996a, Pu 2000, Hubbard e Caines 2002). Outro argumento favorável é que o supervisor resultante da síntese SED com marcação flexível é representado por uma linguagem marcada por um autômato,

tal como o resultado da síntese na abordagem RW padrão. Assim, as duas abordagens fornecem o mesmo tipo de controlador.

A ferramenta de controle hierárquico desenvolvida tem a característica de ser acadêmica. Isto quer dizer que o porte dos sistemas tratados é limitado. As limitações começam na biblioteca de funções usada para construir a ferramenta. Citam-se três limitações da biblioteca **Grail** (Raymond e Wood 1996a). A primeira é a não utilização de estruturas de dados ótimas para minimizar o uso de memória durante os cálculos. Um exemplo de tal estrutura é o Diagrama de Decisão Binário Ordenado (no inglês *Ordered Binary Decision Diagram*, referenciado OBDD), usado para representar estruturas de transição (Buchholz e Kemper 2002). A segunda, é a forma de armazenamento que a biblioteca adota para os dados no disco, feito em arquivos texto. Isso gera arquivos grandes quando comparados com arquivos binários ou com compressão. E a terceira é a limitação do número de estados e transições das máquinas de estado tratadas pelo **Grail**. No **Grail**, os estados são numerados pelo tipo `int`, assim, numa máquina onde o tipo `int` seja representado por 16 bits, o número de estados limita-se a $2^{15} = 32.768$ estados.

A própria ferramenta para controle hierárquico insere limitações adicionais para o porte dos sistemas a serem tratados. A forma de representação do SED com marcação flexível na ferramenta corresponde a duas máquinas de estado do **Grail**, uma para o autômato da linguagem outro para a estrutura de controle, vide apêndice A. Para um dado estado do SED, um controle válido é armazenado na forma de um conjunto de transições na máquina de estados da estrutura de controle. Assim, o tamanho da estrutura de controle do SED é limitado pelo número de transições de uma máquina de estados no **Grail**.

7.3 Perspectivas de extensão

Nesta seção, descrevem-se alguns dos possíveis pontos para extensão desta tese.

Consideram-se duas extensões para os SEDs com marcação flexível. A primeira

é a expressão do problema de controle supervisório que considere o comportamento infinito dos sistemas, como em (Thistle e Wonham 1994a) e (Thistle e Wonham 1994b). Acredita-se que usando-se os SEDs com marcação flexível consiga-se expressar de forma compacta o comportamento infinito de um sistema em malha fechada, sem recorrer às estruturas de transição usadas na literatura.

A segunda possível extensão para os SEDs com marcação flexível é motivada pelo problema do gato e do rato, apresentado na seção 4.1. Neste problema, o supervisor deve garantir que o sistema complete dois tipos distintos de tarefa, correspondendo ou ao gato ou ao rato se alimentarem. Na abordagem padrão RW, considera-se que as tarefas em malha fechada correspondem a tarefas completas por todos os sistemas componentes. Entretanto, o supervisor para o gato e o rato garante que os componentes completem suas tarefas separadamente. Assim, uma extensão possível para o SED com marcação flexível é considerar o atributo de marcação *colorido* para os componentes do sistema. Está se desenvolvendo no grupo uma extensão da teoria de controle supervisório para sistemas com múltiplas tarefas, estas definidas pelas tarefas individuais dos componentes (de Queiroz 2002).

Consideram-se também duas possíveis extensões para a abordagem de controle hierárquico de SEDs. A primeira possibilidade é considerar um esquema de supervisão hierárquico de dois níveis onde se deseja aplicar controle modular no nível do gerente. Uma questão a ser respondida é qual é a condição que expressa a modularidade no SEDs com marcação flexível. Outra questão é, considerando que duas linguagens no nível do gerente atendem à condição correspondente à modularidade e sejam compatíveis, será que se pode garantir que a conjunção dos supervisores correspondentes não causa bloqueio no operador? Pode-se provar que as condições acima não são suficientes para garantir o não bloqueio, e condições adicionais estão sendo consideradas em (Torrico e Cury 2002b).

Uma outra possibilidade de extensão da abordagem de controle hierárquico é definir um esquema de supervisão hierárquica de dois níveis o o sistema do operador seja o equivalente discreto de um sistema dinâmico híbrido. O problema é que, em geral, no

lugar de tal equivalente discreto usa-se uma aproximação conservadora de tal equivalente discreto (González et al. 2001). Assim, cabe investigar se as condições para a consistência hierárquica ainda são válidas.

Um problema do método de construção do esquema de supervisão hierárquica proposto é a construção da estrutura de controle do nível do gerente, vide seção 5.3. Observa-se que os conjuntos de controles para os estados de um SED com marcação flexível possuem uma ordem parcial, pois considera-se que tais conjuntos sejam fechados em relação à união, proposição 5.4. Assim, a exploração dessa ordem parcial para construção da estrutura de controle é uma possível forma de redução da complexidade do cálculo. Por exemplo, se no teste de validade de um controle vocal para um certo estado, o controle é considerado como inválido, esta informação pode ser usada para invalidar uma classe de controles relacionados ao primeiro controle, reduzindo assim, o número de testes necessários para o estado.

Na abordagem usando o raciocínio supor-garantir para sistemas compostos, tratam-se sistemas produto com apenas dois componentes, vide seção 6.4. Dois passos apontam-se como sendo extensões desta abordagem. Num primeiro passo, devem-se estender os resultados para sistemas produto com mais de dois componentes. Acredita-se que esta extensão seja trivial em função do visto para abordagens similares, como, por exemplo, em (Stark 1985), (Alur e Henzinger 1999) ou (Maier 2001). Um segundo passo é estender os resultados para sistemas compostos cujos componentes possam compartilhar eventos. Caso os eventos compartilhados pelos componentes sejam todos relevantes, acredita-se que o resultado ótimo do teorema 6.1 seja mantido. Caso haja eventos não relevantes entre os eventos compartilhados pelos componentes, é possível mostrar que o resultado ótimo do teorema 6.1 não é válido. Entretanto, acredita-se que esse resultado ainda seja uma abstração consistente do sistema composto.

O método de construção do esquema de supervisão hierárquica atende de forma ótima às condições de consistência hierárquica ou consistência hierárquica forte. Esse método construtivo possui uma certa complexidade computacional, como visto na seção 5.5. Uma alternativa para redução dessa complexidade é abrir mão do detalhamento

da informação de controle para os níveis mais altos, ao passo que atendem-se condições mais relaxadas para a consistência hierárquica. Acredita-se que, seja possível exprimir a construção dos sistemas dos níveis superiores usando um conjunto de regras simplificadas que não acarretem cálculos exaustivos para a estrutura de controle. Acredita-se também que seja possível, inclusive, usar o modelo da abordagem RW para os sistemas, no espírito de se perder informação de controle no gerente para ganhar redução do tamanho da representação das estruturas de controle.

Como necessidade de exploração de aplicações, verifica-se a necessidade de uma experiência de implementação prática do esquema de supervisão hierárquica em um sistema de controle real.

Apêndice A

Manual da ferramenta para controle hierárquico de SEDs

Neste apêndice apresenta-se um manual do usuário para a ferramenta para controle hierárquico de SEDs desenvolvida neste trabalho.

A descrição das seções neste apêndice é como segue. Na seção A.1, apresenta-se a ferramenta **Grail**, sobre a qual se baseia a ferramenta para controle hierárquico de SEDs. Na seção A.2, apresenta-se a ferramenta para controle supervisório de SEDs desenvolvida no âmbito do Departamento de Automação e Sistemas da Universidade Federal de Santa Catarina, da qual a ferramenta para controle hierárquico de SEDs utiliza algumas funções. Na seção A.3, apresenta-se a solução de um problema de controle supervisório para ilustrar o uso da ferramenta para controle supervisório de SEDs. Na seção A.4, apresenta-se a ferramenta para controle hierárquico de SEDs, com seus objetos de trabalho e funções. Por fim, na seção A.5, ilustra-se a utilização da ferramenta para controle hierárquico por intermédio de dois exemplos.

A.1 O Grail

Esta seção apresenta a ferramenta **Grail** usada como base para o desenvolvimento da ferramenta para controle hierárquico de SEDs. A exposição se baseia em (Raymond e

Wood 1996a).

O **Grail** é uma coleção de programas para o processamento de máquinas de estados finitos[1] e expressões regulares. Para um usuário, o **Grail** consiste num conjunto de funções que manipulam máquinas de estados finitos e expressões regulares. As máquinas de estados finitos podem ser minimizadas, tornadas deterministas, renumeradas, revertidas, executadas (para alguma palavra), enumeradas, completadas, complementadas, reduzidas a conjuntos alcançáveis de estados, e convertidas em expressões regulares. Expressões regulares podem ser convertidas em máquinas de estados finitos e ter sua forma minimizada. Também há um conjunto de funções predicado que testam condições do tipo determinismo, completude, isomorfismo e universalidade. O uso dessas funções é descrito em detalhes no guia do usuário para o **Grail** e nas páginas de manual associadas às funções (Raymond e Wood 1996d). As máquinas de estados finitos e expressões regulares do **Grail** são parametrizáveis, a significar que é possível compilar o **Grail** com um alfabeto para as máquinas de estado e expressões regulares definido por qualquer tipo ou classe válida do C++. Como distribuído, o **Grail** inclui alfabetos de caracteres ASCII, pares ordenados de inteiros, e expressões regulares (Raymond e Wood 1996a). O **Grail** baseia-se numa biblioteca de classes em C++ que pode ser chamada diretamente a partir de qualquer programa C++. O uso desta classe é descrito no guia do programador para o **Grail** (Raymond e Wood 1996b, Raymond e Wood 1996c).

Os três objetos principais do **Grail** são as máquinas de estados finitos (chamadas *finite machines* ou FMs), as expressões regulares (*regular expressions* ou REs) e as ditas linguagens finitas (*finite languages* ou FLs), estas últimas a significar conjuntos finitos de palavras com comprimento finito. Nas ferramentas para controle supervisório e controle hierárquico, apenas as FMs são utilizadas.

Uma máquina de estados finitos G, designada FM, é definida por uma quíntupla $(\Sigma, Q, \delta, Q_0, Q_m)$, onde Σ é conjunto finito de etiquetas de transição, Q é um conjunto

[1]No inglês original, *finite state machines*, a denominar um autômato não determinista de estados finitos no contexto do **Grail**.

finito de estados, $\delta : Q \times \Sigma \rightarrow 2^Q$ é uma função de transição, parcial, $Q_0 \subseteq Q$ é um conjunto de estados iniciais e $Q_m \subseteq Q$ é um conjunto de estados marcados (Raymond e Wood 1996a). Observe que a FM do **Grail** admite múltiplos estados iniciais. No **Grail** especifica-se uma FM simplesmente listando suas transições, como abaixo:

```
1 a 3
2 b 2
3 b 3
2 c 4
```

na forma de uma lista de transições na forma *estado-de-entrada etiqueta-de-transição estado-de-saída*.

O *Grail* usa uma lista especial de pseudo-transições para indicar os estados iniciais e marcados. Na figura A.1 há a descrição completa de uma FM com a sua especificação correspondente.

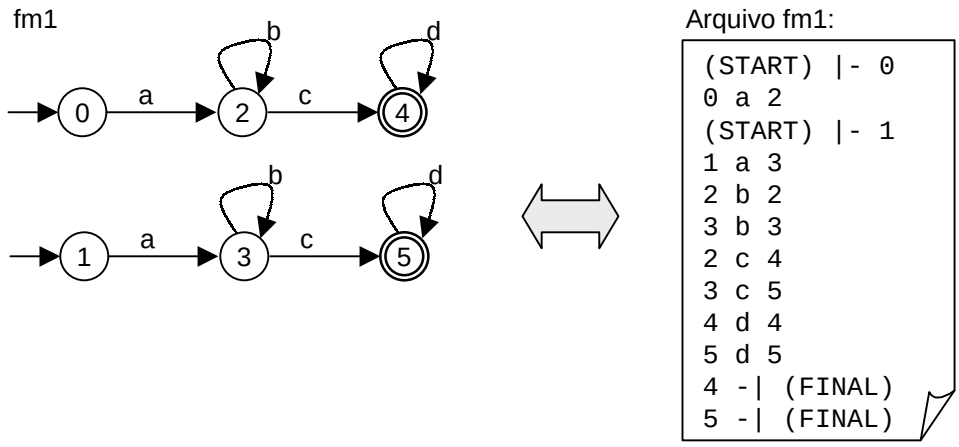


Figura A.1: Especificação de uma máquina de estados finitos no **Grail**.

No **Grail** todos os estados são designados por números naturais, e as etiquetas de transição por seqüências de caracteres, dependendo do tipo em C++ usadas para designar as etiquetas de evento. As pseudo-etiquetas de transição `|-` e `-|` são reservadas para indicar as pseudo-transições que designam que um dado estado é inicial

ou marcado. A palavra `(START)` deve ser usada apenas para indicar um estado inicial, e a palavra `(FINAL)` aparece apenas para indicar um estado marcado. Ambas as palavras reservadas anteriormente somente devem aparecer junto das respectivas pseudo-etiquetas de transição. Não é necessário ordenar as transições na lista que define uma FM. Se o alfabeto de uma FM não for um caracter ASCII, então a etiqueta da transição conterá a representação textual do objeto que parametriza o alfabeto.

Os objetos manipuláveis pelo **Grail** são armazenados em arquivos-texto com sua descrição. Qualquer editor de texto pode ser usado para gerar os arquivos. Usa-se indistintamente o nome do objeto e o nome do arquivo que o armazena.

A versão do **Grail** usada tanto na ferramenta para controle supervisório quanto na ferramenta para controle hierárquico é a 2.5, disponível na página indicada na referência (Raymond e Wood 1996a).

A.2 A ferramenta para controle supervisório de SEDs

Nesta seção descreve-se a ferramenta para controle supervisório de SEDs desenvolvida pelo Departamento de Automação e Sistemas da Universidade Federal de Santa Catarina (Cury 2001).

A ferramenta para controle supervisório baseia-se na biblioteca de classes em C++ para máquinas de estados finitos e expressões regulares fornecida pela ferramenta **Grail**. A ferramenta para controle supervisório pode ser baixada a partir da página `http://www.das.ufsc.br/~cury/ensino-seds.html`, onde também se encontram instruções de instalação da ferramenta.

Dos objetos fornecidos pela biblioteca, a ferramenta para controle supervisório utiliza-se das FMs, e as etiquetas das FMs podem ser quaisquer palavras formadas por caracteres alfanuméricos, excetuando as palavras e símbolos reservados do **Grail** (Raymond e Wood 1996a). Na sua versão atual, os objetos RE e FL, juntamente com

suas funções estão desativados. Algumas das funções básicas da ferramenta **Grail** para FMs estão incluídas no pacote de distribuição da ferramenta para controle supervisório.

Seguindo a filosofia de interfaceamento do **Grail**, na ferramenta para controle supervisório as funções estão disponíveis como comandos no *prompt* do sistema, uma vez a ferramenta instalada. Não há um executável específico para invocar a ferramenta para controle supervisório, e usa-se qualquer janela de terminal do sistema para acessar suas funções, incluindo janelas *telnet* ou *ssh*.

Adicionalmente existem funções que permitem gerar arquivos para visualização das FMs em ferramentas de visualização gráfica. Como será visto a seguir, a ferramenta para controle supervisório gera arquivos que podem ser abertos para visualização nas ferramentas **Graphviz**, disponível em `http://www.research.att.com/sw/tools/graphviz/` e **VCG**, disponível em `http://rw4.cs.uni-sb.de/users/sander/html/gsvcg1.html`.

A ferramenta para controle supervisório manipula sistemas a eventos discretos (SEDs). Como visto anteriormente, um SED é um sistema dinâmico com espaço de estados discreto, que evolui de acordo com a ocorrência abrupta de eventos físicos, em intervalos de tempo em geral irregulares e desconhecidos. Representa-se um SED por um autômato G, correspondendo a uma quíntupla $(\Sigma, Q, q_0, \delta, Q_m)$, onde Σ é um conjunto discreto e finito de eventos físicos possíveis de acontecer no sistema, Q é o espaço de estados discreto, q_0 é o estado inicial do sistema, Q_m é o conjunto de estados marcados, correspondendo a tarefas completas do sistema, e δ é a função de transição, parcial. Na teoria de controle supervisório (Ramadge e Wonham 1989), o controle é exercido sob o sistema por inibição da ocorrência dos eventos. O conjunto de eventos Σ é particionado em um subconjunto Σ_c, de eventos ditos controláveis, que podem ser inibidos, e um subconjunto Σ_{nc}, de eventos ditos não-controláveis, que não podem ser inibidos. No **Grail**, um SED é representado por uma FM onde as etiquetas de transição são os eventos que provocam a evolução do sistema. Para as funções que fazem uso da informação de quais eventos são não controláveis Σ_{nc}, deve-se fornecê-los por uma FM que reconhece Σ_{nc}^*. Esta FM possui um único estado, sendo este inicial e final,

com transições para si mesmo (ditos *selfloops*) com as etiquetas de eventos em Σ_{nc}^*. De forma geral, sempre que se representa um conjunto X nesta ferramenta, utiliza-se a FM que representa X^*. Para ilustrar esta representação, considere o exemplo da máquina de três estados (vide exemplo 2.3), reproduzida na figura A.2. Representa-se o autômato G pela FM `g` e o conjunto de eventos não controláveis Σ_{nc} pela FM `ncont`, indicados na figura A.2.

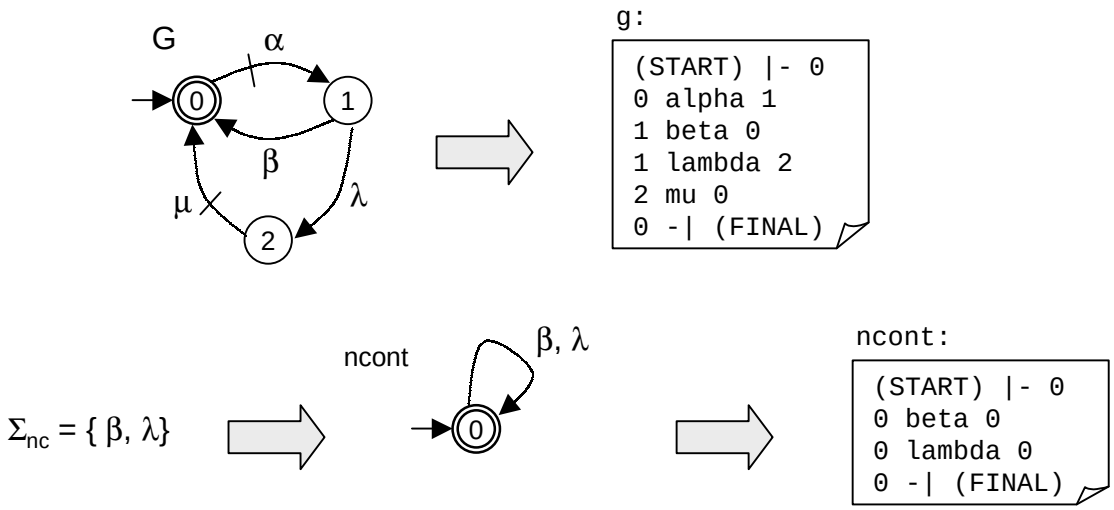


Figura A.2: Máquina de três estados representada no **Grail**.

A seguir descrevem-se as funções disponíveis na ferramenta para controle supervisório de SEDs. O conjunto inicial de funções corresponde às funções originais do **Grail**, disponíveis para auxiliar na manipulação dos autômatos que representam os SEDs.

1. `fmcat` – concatena duas FMs.

   - **Descrição:** Calcula a concatenação das FMs `fm1` e `fm2`. Conectam-se os estados marcados de `fm1` aos estados de `fm2` alcançáveis, por apenas uma transição, de algum dos estados iniciais de `fm2`.
   - **Sintaxe:** `fmcat fm1 fm2`

- **Saída:** FM representando a concatenação de `fm1` e `fm2` na saída padrão[2].

2. `fmcment` – calcula o complemento de uma FM.

   - **Descrição:** Calcula o complemento da FM `fm1`. O complemento de `fm1` é uma outra FM que aceita qualquer outra palavra não aceita por `fm1`. Considera-se como alfabeto de referência o alfabeto de `fm1`.
   - **Sintaxe:** `fmcment fm1`
   - **Saída:** Complemento de `fm1` na saída padrão.

3. `fmcomp` – completa uma FM.

   - **Descrição:** Calcula a FM completa correspondente à FM `fm1`. Uma FM completa é aquela em que todo estado é estado de origem de transições para todos os símbolos do seu alfabeto.
   - **Sintaxe:** `fmcomp fm1`
   - **Saída:** FM completa correspondente a `fm1` na saída padrão.

4. `fmcross` – calcula o produto cartesiano de duas FMs

   - **Descrição:** Calcula o produto cartesiano das FMs `fm1` e `fm2`. Ambas as FMs podem ser especificadas na linha de comando, ou uma delas pode ser lida da entrada padrão. O produto cartesiano contém transições do tipo $((x_1, x_2), \alpha, (y_1, y_2))$, para cada par de transições (x_1, α, y_1) em `fm1` e (x_2, α, y_2) em `fm2` sobre a mesma etiqueta α. A numeração de estados na FM resultante é como segue: $s_0 = s_1 + (max + 1) \cdot s_2$, onde s_0 é o número do estado da FM resultante correspondente ao par (s_1, s_2), s_1 é um estado de `fm1`, s_2 é um estado de `fm2`, e max é o máximo número de estado encontrado em `fm1`. Uma vez que a numeração dos estados do resultado possui uma única fatoração em termos dos números dos estados de `fm1` e

[2]Em inglês, *standard output*.

`fm2`, pode-se determinar que par de estados de `fm1` e `fm2` correspondem a um determinado estado do resultado (Raymond e Wood 1996a). O cálculo do produto cartesiano de duas FMs gera a interseção de suas linguagens (Raymond e Wood 1996a).

- **Sintaxe:** `fmcross fm1 fm2`
- **Saída:** O produto cartesiano de `fm1` e `fm2` na saída padrão.

5. `fmdeterm` – torna uma FM determinista.

   - **Descrição:** Calcula uma FM determinista equivalente à FM `fm1`, usando o método de construção de um equivalente determinista, vide capítulo 2.
   - **Sintaxe:** `fmdeterm fm1`
   - **Saída:** Equivalente determinista de `fm1` na saída padrão.

6. `fmenum` – enumera a linguagem de uma FM.

   - **Descrição:** Enumera a linguagem de uma FM `fm1`. Produz 100 palavras, ou `num` palavras caso o parâmetro `-n num` estiver especificado na chamada da função. Produz palavras segundo o seu tamanho, partindo da palavra de menor tamanho. Para um mesmo tamanho, as palavras são ordenadas lexicograficamente.
   - **Sintaxe:** `fmenum [-n num] fm1`
   - **Saída:** Enumeração de palavras de `fm1` na saída padrão.

7. `fmexec` – executa uma FM para uma palavra de entrada.

   - **Descrição:** Testa se a FM `fm1` reconhece a palavra `s` em sua linguagem marcada. A opção `-d` faz com que `fmexec` escreva a lista de transições que corresponde ao acompanhamento da palavra. Dependendo do tipo de *shell* do sistema, deve-se pôr a palavra `s` entre aspas (Raymond e Wood 1996a).

- **Sintaxe:** `fmexec [-d] fm s`
- **Saída:** Se a palavra `s` for aceita, `fmexec` retorna 1 e escreve `accepted` na saída padrão, senão `fmexec` retorna 0 e escreve `not accepted` na saída padrão.

8. `fmmin` – calcula o equivalente mínimo de uma FM.

   - **Descrição:** Calcula o equivalente mínimo da FM `fm1`, vide capítulo 2. Utiliza o algoritmo de partição de Hopcroft (Raymond e Wood 1996a) e não processa FMs não deterministas, devendo-se aplicar, neste caso, a função `fmdeterm` antes.
   - **Sintaxe:** `fmmin fm1`
   - **Saída:** Equivalente mínimo de `fm1` na saída padrão.

9. `fmminrev` – calcula o equivalente mínimo de uma FM.

   - **Descrição:** Calcula o equivalente mínimo de uma FM `fm1` por método diferente do usado em `fmmin`, aceitando FMs deterministas e não deterministas. Calcula o equivalente mínimo executando a seguinte seqüência de operações: inverte-se o sentido das transições `fm1`, calcula-se o equivalente deteminista, inverte-se novamente o sentido das transições do resultado e calcula-se novamente o equivalente determinista. Garante-se que o resultado é determinista e obtém-se resultados isomorfos aos resultados de `fmmin` (Raymond e Wood 1996a).
   - **Sintaxe:** `fmminrev fm1`
   - **Saída:** Equivalente mínimo de `fm1` na saída padrão.

10. `fmplus` – calcula o "+"de uma FM.

    - **Descrição:** Calcula o "+"de uma FM `fm1`, isto é, uma FM que reconhece uma ou mais ocorrências de palavras reconhecidas por `fm1`. O "+"de uma

FM é construído ligando-se todas as transições que vão a estados marcados de `fm1` também a estados iniciais de `fm1` (Raymond e Wood 1996a).

- **Sintaxe:** `fmplus fm1`
- **Saída:** O "+"da `fm1` na saída padrão.

11. `fmreach` – calcula a componente acessível de uma FM.

    - **Descrição:** Calcula o componente acessível da FM `fm1`, vide capítulo 2.
    - **Sintaxe:** `fmreach fm1`
    - **Saída:** Componente acessível de `fm1` na saída padrão.

12. `fmrenum` – renumera uma FM.

    - **Descrição:** Renumera os estados de uma FM `fm1` de acordo com um esquema de numeração canônico, *breadth-first* e lexicográfico sobre as etiquetas de transição (Raymond e Wood 1996a). Não processa FMs não deterministas.
    - **Sintaxe:** `fmrenum fm1`
    - **Saída:** FM com renumeração canônica para `fm1` na saída padrão.

13. `fmrevers` – reverte uma FM.

    - **Descrição:** Inverte o sentido de todas as transições da FM `fm1`, torna marcados os estados iniciais, e iniciais os estados marcados.
    - **Sintaxe:** `fmrevers fm1`
    - **Saída:** Reversa de `fm1` na saída padrão.

14. `fmstar` – calcula o "*"de uma FM.

    - **Descrição:** Calcula o "*"da FM `fm1`, fechamento Kleene. Primeiramente calcula o "+"de `fm1`, então replica o estado inicial, fazendo deste um estado marcado.

- **Sintaxe:** `fmstar fm1`
- **Saída:** A FM "*"para a `fm1` na saída padrão.

15. `fmstats` – apresenta algumas estatísticas sobre uma FM.

    - **Descrição:** Produz alguns dados estatísticos para a FM `fm1`. Função modificada da original para a ferramenta para controle supervisório. Considere o exemplo para a FM que representa o autômato G da figura 2.5.

```
type g
(START) |- 0
0 a 1
1 b 2
1 c 5
1 d 7
1 a 3
2 c 0
3 b 4
4 c 4
4 a 3
6 a 3
6 b 2
7 -| (FINAL)
2 -| (FINAL)
fmstats g
number of start states:       1
number of final states:       2
number of states:             8
number of reachable states:   7
number of coreachable states: 5
```

```
number of nonblocking states: 6
number of transitions:        11
number of symbols:            4
```

- **Sintaxe:** `fmstats fm1`
- **Saída:** Estatísticas de `fm1` na saída padrão.

16. `fmunion` – calcula a união de duas FMs.

    - **Descrição:** Calcula a união das FMs `fm1` e `fm2`, renumerando-se os estados de `fm2` e então juntando suas transições às transições de `fm1` (Raymond e Wood 1996a).
    - **Sintaxe:** `fmunion fm1 fm2`
    - **Saída:** FM que corresponde à união de `fm1` e `fm2` na saída padrão.

17. `iscomp` – Testa se uma FM é completa.

    - **Descrição:** Testa se a FM `fm1` é completa, isto é, se em `fm1` todo estado possui uma transição para cada evento de seu alfabeto. Considera-se que o alfabeto de referência é o conjunto de todas as etiquetas de transição presentes em `fm1`.
    - **Sintaxe:** `iscomp fm1`
    - **Saída:** Retorna 1 e escreve `complete` na saída padrão se `fm1` for completa, caso contrário, retorna 0 e escreve `incomplete`.

18. `isdeterm` – testa se uma FM é determinista.

    - **Descrição:** Testa se a FM `fm1` é determinista.
    - **Sintaxe:** `isdeterm fm1`
    - **Saída:** Retorna 1 e escreve `deterministic` na saída padrão se `fm1` for determinista, caso contrário, retorna 0 e escreve `nondeterministic`.

19. `isomorph` – testa se duas FMs são isomorfas.

    - **Descrição:** Testa se as FMs `fm1` e `fm2` são isomorfas. Duas FMs são isomorfas se elas são iguais a menos da numeração de estados.
    - **Sintaxe:** `isomorph fm1 fm2`
    - **Saída:** Retorna 1 e escreve `isomorphic` na saída padrão se `fm1` e `fm2` são isomorfas, caso contrário retorna 0 e escreve `nonisomorphic`.

20. `isuniv` – testa se FM é universal.

    - **Descrição:** Testa se a FM `fm1` é universal, isto é, se é completa e se todos os estados acessíveis são também finais (Raymond e Wood 1996a).
    - **Sintaxe:** `isuniv fm1`
    - **Saída:** Retorna 1 e escreve `universal` na saída padrão se `fm1` for universal, caso contrário retorna 0 e escreve `nonuniversal`.

Deste ponto em diante, listam-se as funções escritas especificamente para a ferramenta para controle supervisório:

1. `fmalpha` – obtém o alfabeto de uma FM.

    - **Descrição:** Obtém o alfabeto Σ de uma FM `fm1`.
    - **Sintaxe:** `fmalpha fm1`
    - **Saída:** FM que representa Σ^* na saída padrão.

2. `fmloop` – faz o *selfloop* de eventos numa FM.

    - **Descrição:** Dados uma FM `fm1` e um conjunto de eventos Σ, representado por uma FM `fm2`, faz o *selfloop* dos eventos em Σ na `fm1`, isto é, para cada estado de `fm1` cria transições para si mesmo, *selfloops*, envolvendo todos os eventos em Σ.

- **Sintaxe:** `fmloop fm1 fm2`
- **Saída:** FM resultante do *selfloop* na saída padrão.

3. `fmmark` – marca todos os estados da FM.

   - **Descrição:** Marca todos os estados da FM `fm1`.
   - **Sintaxe:** `fmmark fm1`
   - **Saída:** FM correspondente a `fm1` com todos os seus estados marcados na saída padrão.

4. `fmproj` – faz a projeção de uma FM.

   - **Descrição:** Dados um autômato G, representado pela FM `fm1`, e um conjunto de eventos Σ_r, representado pela FM `fm2` e contido no alfabeto de G, calcula $proj(G, \Sigma_r)$, usando um método distinto do apresentado na seção 2.4.6. O método de `fmproj` utiliza-se de autômatos de estados finitos com ϵ-transições.[3] (Hopcroft e Ullmann 1979).
   - **Sintaxe:** `fmproj fm1 fm2`
   - **Saída:** $proj(G, \Sigma_r)$ na saída padrão.

5. `fmremove` – elimina estados de uma FM.

   - **Descrição:** Dada uma FM `fm1` e um conjunto de estados X, representado pela FM `fm2` e contido nos estados de `fm1`, remove-se os estados em X de `fm1` com suas transições correspondentes.
   - **Sintaxe:** `fmremove fm1 fm2`
   - **Saída:** FM correspondente a `fm1` com os estados em X removidos na saída padrão.

6. `fmsort` – ordena as transições de uma FM.

[3] Em inglês, *finite automata with ϵ-moves*.

- **Descrição:** Ordena as transições da FM `fm1`, primeiro por número de estado de origem das transições, depois por etiqueta de transição.
- **Sintaxe:** `fmsort fm1`
- **Saída:** `fm1` ordenada na saída padrão.

7. `fmtrim` – encontra a componente *trim* de uma FM.

   - **Descrição:** Calcula a componente *trim* da FM `fm1`.
   - **Sintaxe:** `fmtrim fm1`
   - **Saída:** Componente *trim* de `fm1` na saída padrão.

8. `fmsync` – Calcula a composição síncrona de duas FMs.

   - **Descrição:** Calcula a composição síncrona das FMs `fm1` e `fm2`, ver capítulo 2.
   - **Sintaxe:** `fmsync fm1 fm2`
   - **Saída:** Composição síncrona de `fm1` e `fm2` na saída padrão.

9. `fmsupc` – calcula a máxima linguagem controlável.

   - **Descrição:** Dados um SED representado por duas FMs, `fm1` (autômato G) e `fm2` (eventos não controláveis Σ_{nc}), e uma linguagem-alvo K, representada pela FM `fm3`, calcula a máxima linguagem controlável e.r.a. G e Σ_{nc} contida em K, $\sup C_{(G,\Sigma_{nc})}(K)$.
   - **Sintaxe:** `fmsupc fm1 fm2 fm3`
   - **Saída:** $\sup C_{(G,\Sigma_{nc})}(K)$ na saída padrão.

10. `fmcondat` – informa dados de controle em supervisão.

    - **Descrição:** Dados uma planta representada por duas FMs, `fm1` (autômato G) e `fm2` (eventos não controláveis Σ_{nc}), e um supervisor, representado pela FM `fm3`, calcula os eventos desabilitados pelo supervisor.

- **Sintaxe:** `fmcondat fm1 fm2 fm3`
- **Saída:** Para cada estado de `fm3` informa que eventos são desabilitados na planta na saída padrão.

11. `fmtodot` – converte uma FM para o formato reconhecido pela ferramenta **Graphviz**.

    - **Descrição:** Converte a FM `fm1` para um formato reconhecido pela ferramenta **Graphviz**.
    - **Sintaxe:** `fmtodot fm1`
    - **Saída:** Representação de `fm1` para o **Graphviz** na saída padrão.

12. `fmtovcg` – converte uma FM para o formato reconhecido pela ferramenta **VCG**.

    - **Descrição:** Converte a FM `fm1` para um formato reconhecido pela ferramenta **VCG**.
    - **Sintaxe:** `fmtovcg fm1`
    - **Saída:** representação de `fm1` para o **VCG** na saída padrão.

Na próxima seção apresenta-se um exemplo de solução de um PCS utilizando-se ferramenta para controle supervisório de SEDs.

A.3 Exemplo de solução de um PCS

Nesta seção é apresentado um problema de controle supervisório e sua correspondente solução pelo uso da ferramenta para controle supervisório de SEDs.

Considere o exemplo de uma pequena fábrica composta por duas máquinas, M_1 e M_2, e um armazém intermediário com capacidade para uma peça, vide figura A.3, (Cury et al. 2001). Considera-se que o comportamento básico de cada máquina, referenciada M_i, é o seguinte: ao receber o sinal de comando, a máquina M_i carrega uma peça

(evento a_i) e realiza uma operação sobre a mesma, e ao finalizar a operação (evento b_i), a máquina descarrega automaticamente a peça. O modelo do comportamento da máquina M_i é representado pelo autômato indicado na figura A.3. Considera-se que os eventos controláveis são os inícios de operação, $\Sigma_c = \{a_1, a_2\}$, enquanto que os eventos não controláveis são os fins de operação das máquinas $\Sigma_{nc} = \{b_1, b_2\}$. Considere também uma restrição de coordenação que impeça que haja *overflow* (M_1 descarregar peça no armazém cheio) ou *underflow* (M_2 inicia processo de carga com o armazém vazio). Esta restrição é modelada pelo autômato E também indicado na figura A.3. O fato de se marcar apenas o estado inicial de E, vide figura A.3, indica que se deseja considerar como tarefa completa no sistema controlado apenas as palavras que correspondem a situações em que o armazém está vazio. O objetivo é sintetizar um supervisor para imprimir na planta o comportamento desejado.

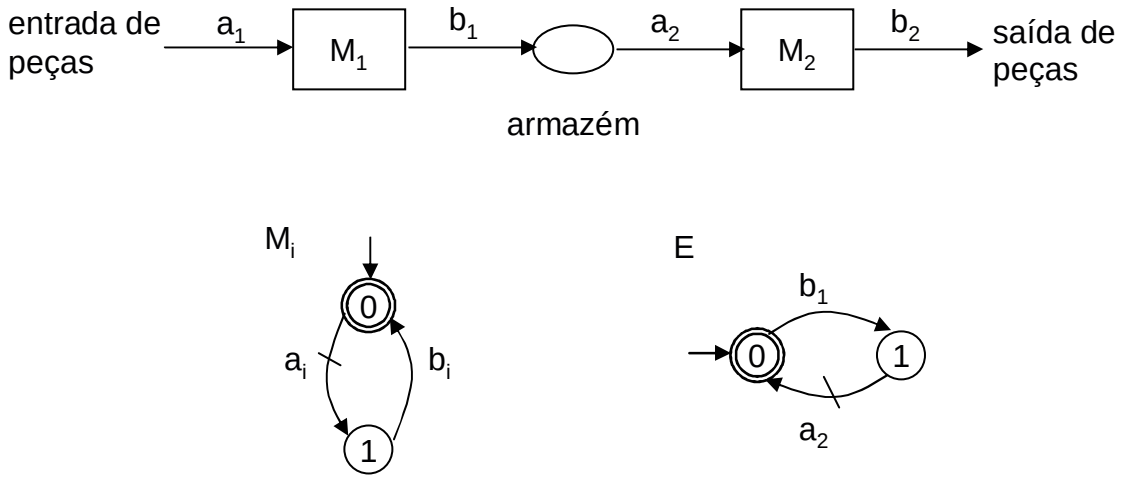


Figura A.3: Exemplo de uma pequena fábrica.

O primeiro passo é inserir os modelos dos componentes da planta e das especificações em arquivos-texto a serem manipulados pelas funções do **Grail**. A FM `m1`, que representa o comportamento da máquina M_1, é armazenada num arquivo-texto de

mesmo nome como a seguir:

```
c:\>type m1
(START) |- 0
0 a1 1
1 b1 0
0 -| (FINAL)
```

Observe-se que, neste apêndice, as listagens de resultados de comandos são feitas no *prompt* de comando do **Windows**. A FM `m2`, que representa a máquina M_2, também é armazenada num arquivo-texto:

```
c:\>type m2
(START) |- 0
0 a2 1
1 b2 0
0 -| (FINAL)
```

Bem como a restrição de coordenação E:

```
c:\>type e
(START) |- 0
0 b1 1
1 a2 0
0 -| (FINAL)
```

A parte de modelagem do exemplo está concluída com cada modelo armazenado em um arquivo, agora podem-se aplicar as funções do **Grail** para encontrar o supervisor ótimo.

1. Constrói-se a planta livre, através da composição síncrona de `m1` e `m2`:

```
c:\>fmsync m1 m2 > planta
c:\>type planta
(START) |- 0
0 a1 1
0 a2 2
1 a2 3
1 b1 0
2 a1 3
2 b2 0
3 b1 2
3 b2 1
0 -| (FINAL)
```

O símbolo `>` acima representa o redirecionamento da saída de um programa, originalmente direcionado para a saída padrão. No caso acima, redireciona-se a saída do comando `fmsync m1 m2` para o arquivo planta.

2. Calcula-se a linguagem-alvo, resultante da composição síncrona da planta livre com a restrição:

```
c:\>fmsync planta e > alvo
c:\>type alvo
(START) |- 0
0 a1 1
1 b1 4
2 a1 3
2 b2 0
3 b1 6
3 b2 1
4 a1 5
```

```
4 a2 2
5 a2 3
6 a1 7
6 b2 4
7 b2 5
0 -| (FINAL)
```

3. Cria-se um arquivo de um único estado com *selfloop* dos eventos não controláveis (arquivo `ncont`, por exemplo):

```
c:\>type ncont
(START) |- 0
0 b1 0
0 b2 0
0 -| (FINAL)
```

   Por exemplo, pode ser feito a partir do alfabeto da planta, obtido por `fmalpha planta > ncont`, e editando `ncont` depois para se apagar os eventos controláveis.

4. Encontra-se o supervisor ótimo:

```
c:\>fmsupc planta alvo ncont > sup
c:\>type sup
(START) |- 0
3 b2 0
0 a1 1
4 b2 1
2 a2 3
3 a1 4
1 b1 2
5 b2 2
```

```
4 b1 5
0 -| (FINAL)
```

5. Eventualmente, pode-se verificar os dados de controle para o supervisor usando-se a função `fmcondat`, onde para cada estado do supervisor são indicados os eventos que são desabilitados na planta.

```
c:\>fmcondat planta sup
0 a2 0
1 a2 1
2 a1 2
5 a1 5
```

Para visualizar os resultados é possível utilizar a ferramenta **Graphviz**, já que o **Grail** possui a função `fmtodot` que converte o arquivo ASCII com a FM para o formato da ferramenta, aqui representado por um arquivo com extensão .do. Para converter um arquivo basta usar a seguinte seqüência de comandos:

```
c:\>fmtodot sup > sup.do
c:\>dotty sup.do
```

O procedimento de visualização no **Graphviz** está automatizado com o *script* `show`, também no diretório de binários do **Grail**. O resultado da visualização do supervisor acima calculado, está na figura A.4.

A.4 A ferramenta para controle hierárquico de SEDs

Esta seção apresenta a ferramenta para controle hierárquico de SEDs.

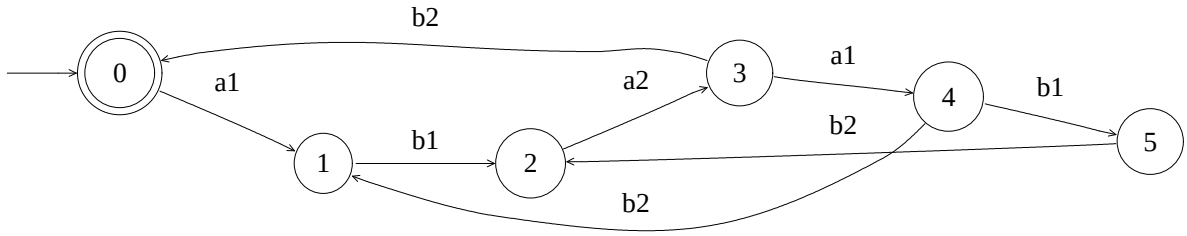


Figura A.4: Supervisor visualizado no **Graphviz**.

A ferramenta de controle hierárquico está disponível no endereço `http://www.das.ufsc.be/~aecc`, onde também estão presentes instruções de instalação. Funciona basicamente como um *toolbox* para a ferramenta para controle supervisório de SEDs, extendendo as funções disponíveis. Consiste num conjunto de funções para manipulação de SEDs com marcação flexível, sendo estas disponíveis em linha de comando para qualquer terminal do sistema, seja um *shell*, *prompt*, *telnet* ou *ssh*. Não há um executável que invoque a ferramenta, ficando as funções disponíveis após a instalação. Os arquivos que armazenam os dados dos modelos são arquivos-texto comuns ASCII, contendo as informações em dados reconhecíveis pela ferramenta, e que podem ser criados e modificados usando qualquer editor de texto.

O objeto manipulado pela ferramenta, um SED com marcação flexível é representado por um autômato G e uma estrutura de controle Γ, vide seção 4.1. Na ferramenta, representa-se esse par por duas FMs, digam-se `fm1` e `fm2`. A FM `fm1` representa o autômato G como já feito anteriormente. A FM `fm2` representa a estrutura de controle Γ, segundo a convenção:

1. `fm2` não possui transições representando estados iniciais ou finais, consiste apenas numa lista de transições.

2. Os estados de saída das transições de `fm2` são os estados de `fm1`, e conseqüentemente de G.

3. Os estados de entrada de transições em `fm2` possuem numeração diferente da numeração de qualquer estado em `fm1`.

4. Para um conjunto de transições em `fm2` com o mesmo estado de origem x, existem tantos estados de entrada correspondentes quanto controles para x em Γ.

5. Para um estado x de G e um controle $(\gamma, \#)$ em $\Gamma(x)$, onde $\gamma = \{a1, a2, \ldots, an\}$ e $\#$ é igual a M ou N, existe um conjunto de transições em `fm2` da seguinte forma:

   ```
   x a1 y
   x a2 y
   ...
   x an y
   x M y
   ```

   para $\#$ sendo igual a M, ou

   ```
   x a1 y
   x a2 y
   ...
   x an y
   x N y
   ```

   para $\#$ sendo igual a N. O estado y acima é um estado com numeração distinta de qualquer estado de `fm1`.

6. Para o total de transições de `fm2` cujo estado de origem é x, o estado y só deve aparecer como estado de destino para as transições correspondendo ao controle $(\gamma, \#)$ em $\Gamma(x)$.

7. Quando um estado de `fm1` não aparece como estado de origem de transições de `fm2`, assume-se que o seu conjunto de controles é vazio.

Assim, para um estado x de G e um controle $(\gamma, \#)$ em $\Gamma(x)$ existe um estado y em `fm2`, único para as transições em `fm2` cujo estado de origem é x, que é estado de

destino de um conjunto de transições em `fm2` cujas etiquetas representam o controle $(\gamma, \#)$.

Pode-se provar que esta forma de representação de um SED com marcação flexível permite fazer a transcrição de um autômato e uma estrutura de controle para um par de FMs do **Grail** e vice e versa, sem ambigüidade. A figura A.5 apresenta um exemplo de representação de um SED com marcação flexível na ferramenta para controle hierárquico de SEDs .

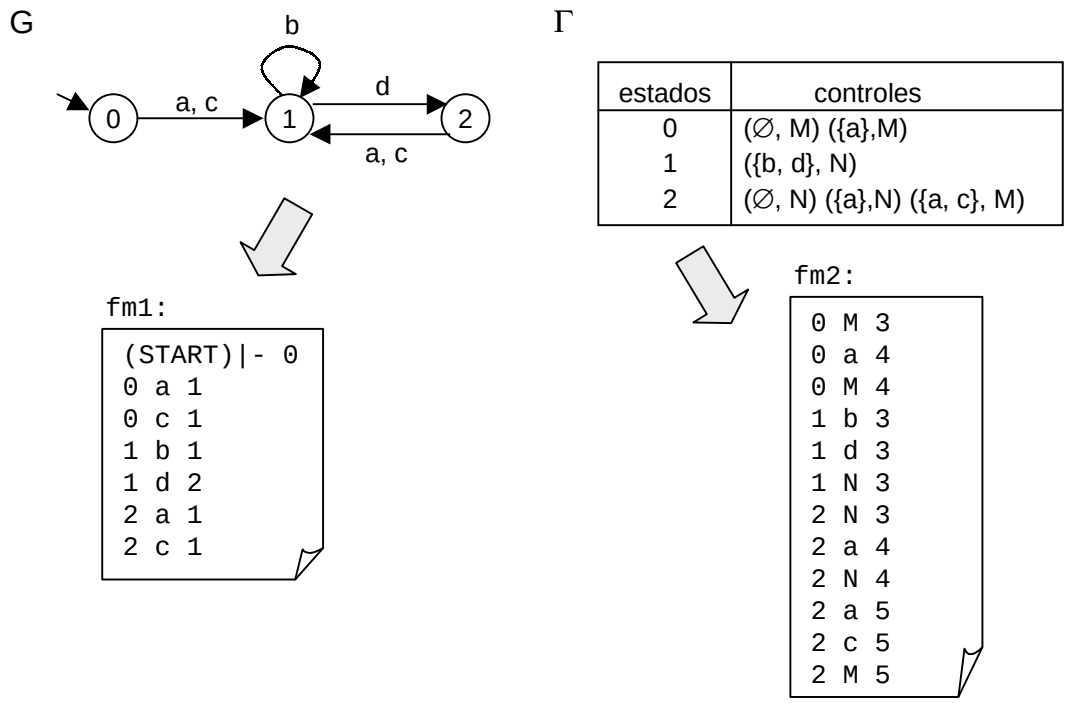


Figura A.5: Representação de um SED com marcação flexível.

O esquema de representação de um SED com marcação flexível apresentado possui algumas limitações, comentadas a seguir. Uma limitação refere-se ao uso das etiquetas de transição "M"e "N"para representar o atributo de marcação do controle. Para a ferramenta para controle hierárquico, as etiquetas de transição "M"e "N"tornam-se reservadas, não podendo ser usadas para representar eventos de um sistema. Uma outra limitação refere-se a capacidade computacional da ferramenta. A princípio, pode-se considerar como um limite computacional para o **Grail** o número de transições

de uma FM (Raymond e Wood 1996a). Por outro lado, o número de eventos do sistema determina o número máximo de controles que o sistema pode possuir. Para um sistema com e eventos, cujo comportamento é representado por um autômato de n estados, o número total de controles na estrutura de controle fica limitado a $n \cdot 2^{e+1}$, correspondendo a cada estado com o número máximo de controles possível. Como cada evento num controle está correspondendo a uma transição na FM que representa a estrutura de controle, conclui-se que o número de transições desta FM depende exponencialmente do número de eventos do sistema.

A seguir, faz-se uma descrição dos grupos de funções disponibilizadas no pacote da ferramenta para controle hierárquico de SEDs. Uma nota sobre o prefixo das funções: seguindo a notação do **Grail** o prefixo da função representa o objeto principal manipulado pela mesma (Raymond e Wood 1996a). Assim, são definidas funções com prefixo `fm` para FMs, prefixo `cntrl` para conjuntos de controles, e `cdes` para SEDs com marcação flexível. Uma particularidade para as funções de controle hierárquico é que quando estas são invocadas sem argumento ou de forma errada, apresenta-se na saída padrão um texto de ajuda de utilização.

O primeiro grupo de funções corresponde a funções desenvolvidas para manipular SEDs com marcação flexível, descrito a seguir:

1. `fmtocdes` – converte um SED com modelo RW para um SED com marcação flexível.

   - **Descrição:** Dados um SED dotado de estrutura de controle Ramadge-Wonham, representado pelas FMs `fm1` (autômato G) e `fm2` (eventos não controláveis Σ_{nc}), encontra a estrutura de controle dependente do estado para a representação por SED com marcação flexível Γ, vide exemplo 4.6. Supõe-se o caso do controle supervisório com marcação.
   - **Sintaxe:** `fmtocdes fm1 fm2`
   - **Saída:** A estrutura de controle Γ do SED, representada por uma FM, na saída padrão.

2. `cdesmin` – Calcula o equivalente mínimo de um SED com marcação flexível.

   - **Descrição:** Dado o SED com marcação flexível S, representado pelas FMs `fm1` (autômato) e `fm2` (estrutura de controle), calcula um SED S_{min}, com número de estados mínimo possível equivalente a S.
   - **Nota:** A equivalência de SEDs com marcação flexível é definida a partir da comparação $\leq$, definição 6.2. Dois SEDs S_1 e S_2 são equivalentes se e somente se $S_1 \leq S_2$ e $S_2 \leq S_1$. O procedimento de obtenção de um equivalente mínimo para um SED com marcação flexível, além de considerar o critério usual para obtenção do equivalente mínimo de um autômato, vide capítulo 2, adicionalmente considera que dois estados são equivalentes, isto é, estados que podem ser substituídos por um único estado, quando estes possuírem os mesmos conjuntos de controles.
   - **Sintaxe:** `cdesmin fm1 fm2 fm3 fm4`
   - **Saída:** O SED minimizado S_{min}, salvo nos arquivos `fm3` (autômato) e `fm4` (estrutura de controle).

3. `cdedeterm` – calcula o equivalente determinista de um SED com marcação flexível.

   - **Descrição:** Dado o SED com marcação flexível S, representado pelas FMs `fm1` (autômato G) e `fm2` (estrutura de controle Γ), com G não determinista, calcula SED com marcação flexível S^D equivalente a S.
   - **Nota:** Representa-se S^D pelo autômato G^D e estrutura de controle Γ^D. O autômato G^D é o equivalente determinista de G, vide capítulo 2. A estrutura de controle Γ^D é tal que, para o estado q^D de G^D, define-se $\Gamma^D(q^D)$ por

$$\begin{aligned}\Gamma^D(q^D) \;=\; \{(\gamma,\#) \;&\in 2^{\Sigma_{G^D}(q^D)} \times \{M,N\} : (\forall q \in q^D)\ (\exists(\gamma',\#') \in \Gamma(q)) \\ &\text{tais que } ((\gamma' = \gamma \cap \Sigma_G(q)) \text{ e } (\#' = \#))\}.\end{aligned} \tag{A.1}$$

Observe que Γ^D é definido conforme a estrutura de controle do sistema do gerente, equação (5.6).

- **Sintaxe:** `cdesdeterm fm1 fm2 fm3 fm4`
- **Saída:** O SED S^D, salvo nos arquivos `fm3` (autômato) e `fm4` (estrutura de controle).

4. **cdescontain** – compara dois SEDs com marcação flexível.

   - **Descrição:** Dados dois SEDs com marcação flexível S_1 e S_2, representados pelas FMs `fmi1` (autômato) e `fmi2` (estrutura de controle), onde `i` é igual a 1 ou 2 respectivamente a indicar S_1 e S_2, compara S_1 e S_2.
   - **Sintaxe:** `cdescontain fm11 fm12 fm21 fm22`
   - **Saída:** Texto na saída padrão fazendo uma das quatro afirmações: $S_1 \leq S_2$, $S_2 \leq S_1$, $S_1 = S_2$ ou que S_1 e S_2 não são comparáveis.

5. **cdeswrite** – escreve a estrutura de controle de um SED com marcação flexível.

   - **Descrição:** Dado um SED com marcação flexível S, representado pelas FMs `fm1` (autômato G) e `fm2` (estrutura de controle Γ), escreve Γ de forma legível na saída padrão.
   - **Sintaxe:** `cdeswrite fm1 fm2`
   - **Saída:** Estrutura de controle de S escrita de forma legível na saída padrão.

6. **cdessync** – calcula o produto síncrono de dois SEDs com marcação flexível.

   - **Descrição:** Dados dois SEDs com marcação flexível S_1 e S_2, representados pelas FMs `fmi1` (autômato) e `fmi2` (estrutura de controle), onde `i` é igual a 1 ou 2 respectivamente a indicar S_1 e S_2, calcula o produto síncrono $S_1 \| S_2$, vide definição 6.1.
   - **Sintaxe:** `cdessync fm11 fm12 fm21 fm22 fm3 fm4`

- **Saída:** O SED com marcação flexível $S_1 \| S_2$, representado pelas FMs salvas nos arquivos `fm3` (autômato) e `fm4` (estrutura de controle).

7. **cdessupc** – calcula a máxima linguagem compatível.

   - **Descrição:** Dados um SED com marcação flexível S, representado pelas FMs `fm1` (autômato G) e `fm2` (estrutura de controle Γ) e a linguagem K, marcada por um autômato representado pela FM `fm3`, encontra a máxima linguagem (G, Γ)-compatível contida em K, $\sup C_{(G,\Gamma)}(K)$, vide algoritmo 4.1.
   - **Sintaxe:** `cdessupc fm1 fm2 fm3`
   - **Saída:** Autômato que marca $\sup C_{(G,\Gamma)}(K)$ na saída padrão.

8. `cdescondat` – calcula dados de supervisão.

   - **Descrição:** Dados um SED com marcação flexível S, representado pelas FMs `fm1` (autômato G) e `fm2` (estrutura de controle Γ), e um supervisor representado pela FM `fm3` (autômato F), escreve os dados de controle.
   - **Sintaxe:** `cdescondat fm1 fm2 fm3`
   - **Saída:** Lista na saída padrão na seguinte forma:

     ```
     (x,y) -> info
     ```

     onde:

     - `x` é um estado do supervisor,
     - `y` é um estado do sistema S,
     - `info` assume os seguintes valores:
       - $(\gamma, \#)$ – o controle em $\Gamma(y)$ selecionado no estado x do supervisor.
       - `bad state` – se na busca do controle desejado no estado x do supervisor, $\Gamma(y)$ retorne a condição de *mau estado*,

  - $(\gamma, \#)$ `bad marked` – se na busca do controle desejado no estado x do supervisor, $\Gamma(y)$ retorne um controle $(\gamma, \#)$ com a condição de *mau estado marcado*, e
  - $(\gamma, \#)$ `bad transition` – se na busca do controle desejado no estado x do supervisor, $\Gamma(y)$ retorne um controle $(\gamma, \#)$ com a condição de *má transição*.

- **Nota:** Para entender às condições de *mau estado*, *mau estado marcado* e *má transição*, veja a explicação que segue o algoritmo 4.1 na seção 4.3.

O segundo grupo corresponde às funções para controle hierárquico de SEDs com marcação flexível.

1. **fmvocals** – calcula os estados vocais.

   - **Descrição:** Dados um autômato G, representado pela FM `fm1`, e o conjunto de eventos relevantes Σ_r, representado pela FM `fm2`, encontra os estados vocais de G (estados que são de entrada de transições envolvendo os eventos relevantes, mais o estado inicial, seção 5.2).
   - **Sintaxe:** `fmvocals fm1 fm2`
   - **Saída:** Uma FM na saída padrão, contendo os estados vocais de G postos nas etiquetas de transição.

2. **cdessubsystem** – calcula o subsistema de um estado.

   - **Descrição:** Dados um SED com marcação flexível S, representado por duas FMs `fm1` (autômato G) e `fm2` (estrutura de controle Γ), o conjunto de eventos relevantes Σ_r, contido no alfabeto de S e representado pela FM `fm3`, e o estado `q` de S, representado por um número natural, calcula o subsistema para o estado `q`, $S(q)$, algoritmo 5.1.
   - **Sintaxe:** `cdessubsystem fm1 fm2 fm3 q`

- **Saída:** O subsistema $S(q)$, salvo nos arquivos de nomes `[fm1]s[q]` (autômato) e `[fm1]s[q]cont`, onde `[fm1]` e `[q]` indicam os nomes dados aos parâmetros `fm1` e `q` na chamada da função `cdessubsystem`.

3. **cdesvocalcontrols** – calcula os controles vocais de um estado.

   - **Descrição:** Dados um SED com marcação flexível S, representado por duas FMs `fm1` (autômato G) e `fm2` (estrutura de controle Γ), o conjunto de eventos relevantes Σ_r, contido no alfabeto de S e representado pela FM `fm3`, e o estado `q` de S, representado por um número natural, calcula o conjunto de controles vocais para o estado `q`, $\Gamma_{S,voc}(q)$, algoritmo 5.2.
   - **Sintaxe:** `cdesvocalcontrols fm1 fm2 fm3 q`
   - **Saída:** O conjunto de controles vocais $\Gamma_{S,voc}(q)$, salvo no arquivo de nome `[fm1]s[q]`, onde `[fm1]` e `[q]` indicam os nomes dados aos parâmetros `fm1` e `q` na chamada da função `cdesvocalcontrols`. Adicionalmente, gera-se informação sobre transições envolvendo eventos relevantes que podem ser definidas após `q`, função ψ no algoritmo 5.3, salva no arquivo de nome `[fm1]s[q]slice`.

4. **cdesconsistent** – calcula a máxima abstração consistente.

   - **Descrição:** Dados um SED com marcação flexível S, representado por duas FMs `fm1` (autômato G) e `fm2` (estrutura de controle Γ), e o conjunto de eventos relevantes Σ_r, contido no alfabeto de S e representado pela FM `fm3`, calcula a máxima abstração consistente de S e.r.a. Σ_r, $\langle S, \Sigma_r \rangle$, algoritmo 5.3.
   - **Sintaxe:** `cdesconsistent fm1 fm2 fm3 fm4 fm5`
   - **Saída:** A máxima abstração consistente de S e.r.a. Σ_r, salva nos arquivos de nome `fm4` (autômato) e `fm5` (estrutura de controle).

5. **cdesrelabel** – renomeia SED e conjunto de eventos relevantes para que o mapa repórter seja determinista.

   - **Descrição:** Dados um SED com marcação flexível S, representado por duas FMs `fm1` (autômato G) e `fm2` (estrutura de controle Γ), e o conjunto de eventos relevantes Σ_r, contido no alfabeto de S e representado pela FM `fm3`, modifica as etiquetas de eventos em Σ_r, criando um novo SED S' com novo conjunto de eventos relevantes Σ'_r de tal forma que o mapa repórter associado é determinista, algoritmo 5.4.
   - **Sintaxe:** `cdesrelabel fm1 fm2 fm3 fm4 fm5 fm6`
   - **Saída:** O novo sistema S', salvo nos arquivos `fm4` (autômato) e `fm5` (estrutura de controle), e o novo conjunto de eventos relevantes Σ'_r, salvo no arquivo `fm6`.
   - **Nota:** As instâncias de um evento de nome `a` em Σ_r são criadas da seguinte forma: a primeira instância é o próprio `a`, a segunda instância é `a_1`, a terceira é `a_2` e assim por diante.

6. **fmrelabel** – renomeia as etiquetas de um autômato de acordo com as etiquetas criadas por `cdesrelabel`.

   - **Descrição:** Dados uma linguagem K sobre um alfabeto Σ_r, sendo a linguagem marcada por um autômato representado pela FM `fm1`, e um conjunto de eventos Σ'_r, representado por uma FM `fm2`, construído a partir de Σ_r por criação de instâncias de eventos, *conforme o método implementado na função cdesrelabel*, encontra a linguagem K'_r por substituição das transições com etiquetas de evento em Σ_r no autômato que marca K por transições com etiquetas de evento em Σ'_r correspondentes (vide ilustração deste método no exemplo 5.9).
   - **Sintaxe:** `fmrelabel fm1 fm2`

- **Saída:** A linguagem K'_r representada por uma FM na saída padrão.

A seguir são apresentadas algumas funções auxiliares desenvolvidas para FMs e conjuntos de controles. Essas funções surgiram em passos intermediários para a construção das funções para SEDs com marcação flexível.

Uma observação preliminar, para o alfabeto Σ, um conjunto de controles C é um subconjunto qualquer de $2^{\Sigma} \times \{M, N\}$, vide capítulo 4. De forma análoga a estrutura de controle de um SED com marcação flexível, representa-se um conjunto de controles C por uma FM fm1 onde cada conjunto de transições com o mesmo estado de destino representa um controle válido em C. Escolhe-se o estado de origem das transições em fm1, único neste caso, arbitrariamente.

1. **fmproj1** – calcula o autômato projeção.

   - **Descrição:** Dados um autômato G, representado pela FM `fm1`, e o conjunto de eventos relevantes Σ_r, representado pela FM `fm2`, encontra autômato projeção $proj(G, \Sigma_r)$, algoritmo 2.1.
   - **Sintaxe:** `fmproj1 fm1 fm2`
   - **Saída:** O autômato projeção $proj(G, \Sigma_r)$ na saída padrão.

2. **fmcontain** – compara dois autômatos.

   - **Descrição:** Dados dois autômatos, digam-se G_1 e G_2, representados pelas FMs `fm1` e `fm2` respectivamente, determina a relação entre as linguagens marcadas de G_1 e G_2.
   - **Sintaxe:** `fmcontain fm1 fm2`
   - **Saída:** Texto na saída padrão especificando uma de quatro opções, a saber, $L_{G_1,m} \subset L_{G_2,m}$, $L_{G_1,m} \supset L_{G_2,m}$, $L_{G_1,m} = L_{G_2,m}$ ou que $L_{G_1,m}$ e $L_{G_2,m}$ não são comparáveis.

3. **cntrlunion** – calcula a união de dois conjuntos de controles.

- **Descrição:** Dados dois conjuntos de controles C_1 e C_2, representados pelas FMs `fm1` e `fm2`, calcula a união $C_1 \cup C_2$.
- **Sintaxe:** `cntrlunion fm1 fm2`
- **Saída:** $C_1 \cup C_2$ na saída padrão.

4. **cntrlintersect** – calcula a interseção de dois conjuntos de controles.

   - **Descrição:** Dados dois conjuntos de controles C_1 e C_2, representados pelas FMs `fm1` e `fm2`, calcula a interseção $C_1 \cap C_2$.
   - **Sintaxe:** `cntrlintersect fm1 fm2`
   - **Saída:** $C_1 \cap C_2$ na saída padrão.

5. **cntrlcontain** – compara dois conjuntos de controles.

   - **Descrição:** Dados dois conjuntos de controle C_1 e C_2, representados pelas FMs `fm1` e `fm2`, compara C_1 e C_2.
   - **Sintaxe:** `cntrlcontain fm1 fm2`
   - **Saída:** texto na saída padrão, com uma das seguintes afirmações: $C_1 \subset C_2$, $C_1 \supset C_2$, $C_1 = C_2$ ou que C_1 e C_2 não são comparáveis.

6. **cntrlwrite** – escreve um conjunto de controles.

   - **Descrição:** Dado o conjunto de controle C, representado pela FM `fm1`, escreve os controles em C de forma ordenada.
   - **Sintaxe:** `cntrlwrite fm1`
   - **Saída:** Texto com o controle C escrito de forma legível na saída padrão.

A.5 Exemplos de utilização da ferramenta

Nesta seção apresentam-se dois exemplos de utilização da ferramenta para controle hierárquico de SEDs para o **Grail**.

A.5.1 Exemplo de síntese

O primeiro exemplo consiste na manipulação de um SED com marcação flexível e cálculo da máxima linguagem compatível. Considere o SED com marcação flexível S representado pelo autômato G e estrutura de controle Γ, representados na figura A.6 (este exemplo é equivalente ao exemplo 4.9 a menos da nomenclatura dos eventos).

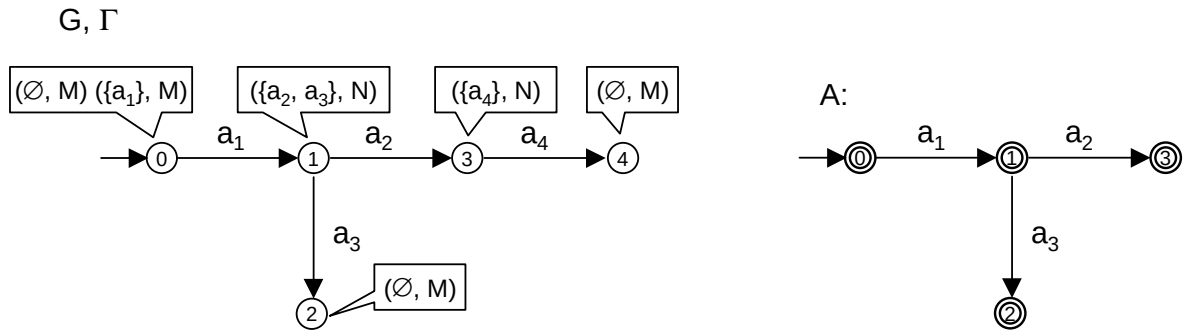


Figura A.6: Exemplo de cálculo da máxima linguagem compatível no **Grail**.

Representa-se o autômato G e a estrutura de controle Γ pelas FMs `s` e `scont` abaixo:

```
c:\>type s
(START) |- 0
0 a1 1
1 a2 2
1 a3 3
2 a4 4
c:\>type scont
0 M 5
0 M 6
0 a1 6
1 a2 7
1 a3 7
1 N 7
```

```
2 a4 8
2 N 8
3 M 9
4 M 10
```

As listagens acima foram feitas no *prompt* do sistema operacional **Windows**.

Os conjuntos de controles podem ser apresentados de forma mais clara usando-se a função `cdeswrite`, como abaixo:

```
c:\> cdeswrite s scont
0
({}, M) ({ a1}, M)
1
({ a2 a3}, N)
2
({ a4}, N)
3
({}, M)
4
({}, M)
```

Como visto acima, a função `cdeswrite` lista na tela o estados seguidos de seus controles correspondentes.

Considere então a linguagem-alvo A na figura A.6, representada pela FM `a` abaixo:

```
c:\>type a
(START) |- 0
0 a1 1
1 a2 2
1 a3 3
0 -| (FINAL)
```

```
1 -| (FINAL)
2 -| (FINAL)
3 -| (FINAL)
```

Para mostrar que A é não (G,Γ)-compatível usa-se a função `cdescondat` como a seguir:

```
c:\>cdescondat s scont a
(plant state, sup state) -> control
(0, 0) -> ({ a1}, 1)
(1, 1) -> ({ a2 a3}, 0) bad marked
(2, 2) -> bad state
(3, 3) -> ({}, 1)
```

Seja $K = \sup C_{(G,\Gamma)}(A)$, a máxima linguagem (G,Γ)-compatível contida em A. Calcula-se K usando a função `cdessupc`. Na linha de comando abaixo, a FM `k` representa um autômato que marca K.

```
c:\>cdessupc s scont a > k
c:\>type k
(START) |- 0
0 -| (FINAL)
```

Repare que $K = \{\epsilon\}$. Por fim, pode-se verificar a (G,Γ)-compatibilidade de K usando-se novamente a função `cdescondat` como abaixo:

```
c:\>cdescondat s scont k
(plant state, sup state) -> control
(0, 0) -> ({}, 1)
```

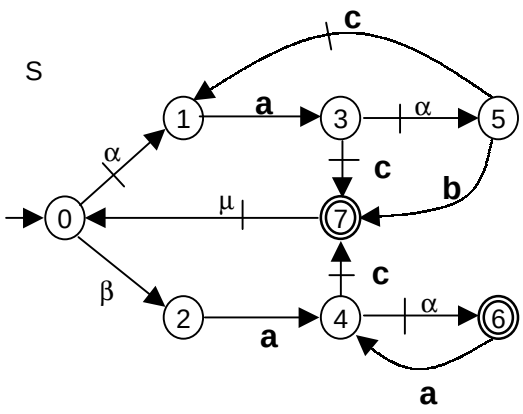


Figura A.7: Exemplo de SED para o controle hierárquico no **Grail**.

A.5.2 Exemplo de controle hierárquico

Considere o SED S com estrutura de controle RW, representado na figura A.7 (equivalente ao SED do exemplo 5.1.

Representa o autômato G correspondente ao comportamento do SED S pela FM `s` e o conjunto de eventos não controláveis Σ_{nc} pela FM `ncont`, abaixo representadas.

```
c:\>type s
(START) |- 0
0 alpha 1
0 beta 2
1 a 3
2 a 4
3 alpha 5
3 c 7
4 alpha 6
4 c 7
5 c 1
5 b 7
6 a 4
```

```
7 mu 0
6 -| (FINAL)
7 -| (FINAL)
c:\>type ncont
(START) |- 0
0 beta 0
0 a 0
0 b 0
0 -| (FINAL)
```

O primeiro passo para trabalhar com a ferramenta para controle hierárquico é traduzir os sistemas para SEDs com marcação flexível. Para traduzir o sistema S basta construir a estrutura de controle Γ, pois o autômato G já fornece o comportamento gerado do sistema. Constrói-se a estrutura de controle usando-se a função `fmtocdes` como abaixo:

```
c:\>fmtocdes s ncont > scont
c:\>cdeswrite s scont
0
({ beta}, N) ({ beta alpha}, N)
1
({ a}, N)
2
({ a}, N)
3
({}, N) ({ alpha}, N) ({ c}, N) ({ c alpha}, N)
4
({}, N) ({ alpha}, N) ({ c}, N) ({ c alpha}, N)
5
({ b}, N) ({ b c}, N)
```

```
6
({ a}, M) ({ a}, N)
7
({}, M) ({ mu}, M) ({}, N) ({ mu}, N)
```

Considera-se que S seja o sistema para um operador num esquema de supervisão hierárquica de dois níveis, conforme descrito na seção 3.1. O objetivo é construir um sistema para o gerente com consistência hierárquica, e após, consistência hierárquica forte usando a ferramenta. O conjunto de eventos relevantes para o gerente Σ_r é representado pela FM `sigmar` representada abaixo:

```
c:\>type sigmar
(START) |- 0
0 a 0
0 c 0
0 b 0
0 -| (FINAL)
```

Primeiramente constrói-se um sistema para o gerente com consistência hierárquica passo a passo, usando as funções disponíveis na ferramenta.

Inicialmente os estados vocais de S podem ser obtidos pela função `fmvocals` como a seguir. Verifica-se que a saída é uma FM cujas etiquetas de eventos são os estados vocais.

```
c:\>fmvocals s sigmar
(START) |- 0
0 1 0
0 3 0
0 4 0
0 7 0
0 0 0
0 -| (FINAL)
```

O autômato não determinista que corresponde ao comportamento visto pelo gerente, a observar apenas os eventos relevantes, é obtido pela função `fmproj1`, vide autômato G' na figura 5.5. Repare que os estados do autômato são os estados vocais de S e os estados marcados estão removidos. A FM `p-nm-nd` abaixo representa o autômato em questão.

```
c:\>fmproj1 s sigmar > p-nm-nd
c:\>type p-nm-nd
(START) |- 0
1 a 3
3 c 7
3 b 7
3 c 1
4 c 7
4 a 4
7 a 3
7 a 4
0 a 3
0 a 4
```

Os controles vocais para os estados vocais são calculados usando-se a função `cdesvocalcontrols`. A seguinte seqüência de comandos calcula os controles vocais e concatena todos num único arquivo `p-nm-ndcont` (são usados comandos do *prompt* do **Windows**).

```
c:\>for %i in (0 1 3 4 7) do cdesvocalcontrols s scont sigmar %i
c:\>copy ss0vcont+ss1vcont+ss3vcont+ss4vcont+ss7vcont p-nm-ndcont
```

Eventualmente pode-se estar interessado em ver o subsistema para um estado. Isso pode ser obtido pela função `cdessubsystem`. Compare a saída da seqüência de comandos abaixo com o subsistema para o estado 4 de S, representado na figura 5.3.

```
c:\>cdessubsystem s scont sigmar 4
c:\>type ss4
(START) |- 4
4 c 8
4 alpha 6
6 a 8
c:\>cdeswrite ss4 ss4cont
4
({}, N) ({ alpha}, N) ({ c}, N) ({ c alpha}, N)
6
({ a}, M) ({ a}, N)
8
({}, M)
```

Assim, o sistema para o gerente corresponde ao SED com marcação flexível representado pelas FMs `p-nm-nd` e `p-nm-ndcont`. Compare a lista de controles abaixo com a lista na figura 5.5.

```
c:\>cdeswrite p-nm-nd p-nm-ndcont
0
({ a}, N)
1
({ a}, N)
3
({ b}, N) ({ b c}, N)
4
({ a}, M) ({ a c}, M) ({ c}, N) ({ a}, N) ({ a c}, N)
7
({}, M) ({ a}, M) ({ a}, N)
```

Encontra-se o equivalente determinista do sistema usando-se a função `cdesdeterm`,

como visto abaixo. Compare o resultado da listagem abaixo, as FMs `p-nm` e `p-nmcont`, com o sistema do gerente na figura 5.6.

```
c:\>cdesdeterm p-nm-nd p-nm-ndcont p-nm p-nmcont
c:\>type p-nm
(START) |- 0
0 a 1
1 a 2
1 b 3
1 c 4
2 a 2
2 c 3
3 a 1
4 a 1
c:\>cdeswrite p-nm p-nmcont
0
({ a}, N)
1
({ b c}, N) ({ a b}, N) ({ a b c}, N)
2
({ a}, M) ({ a c}, M) ({ c}, N) ({ a}, N) ({ a c}, N)
3
({}, M) ({ a}, M) ({ a}, N)
4
({ a}, N)
```

Eventualmente pode-se desejar minimizar o número de estados do resultado. Para isto usa-se a função `cdesmin`. Considera-se então que o sistema do gerente, diga-se P, é representado pelo par de FMs `p` e `pcont`, obtidos abaixo.

```
c:\>cdesmin p-nm p-nmcont p pcont
```

```
c:\>type p
(START) |- 0
0 a 1
1 a 2
1 b 3
1 c 0
2 a 2
2 c 3
3 a 1
c:\>cdeswrite p pcont
0
({ a}, N)
1
({ b c}, N) ({ a b}, N) ({ a b c}, N)
2
({ a}, M) ({ a c}, M) ({ c}, N) ({ a}, N) ({ a c}, N)
3
({}, M) ({ a}, M) ({ a}, N)
```

Todos os passos acima resumem-se apenas a um comando ao se usar a função `cdesconsistent`, como ilustrado abaixo.

```
c:\>cdesconsistent s scont sigmar p pcont
```

Pode-se provar que há consistência hierárquica entre S e P acima, entretanto esta consistência não é forte, como pode ser verificado no desenvolvimento a seguir. Considere a especificação E envolvendo os eventos relevantes para o gerente, marcada pelo autômato na figura A.8.

Representa-se E por uma FM no **Grail** como a seguir:

```
c:\>type e
```

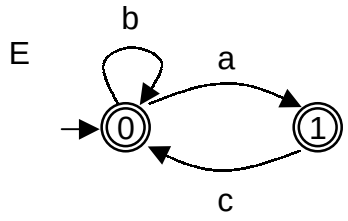


Figura A.8: Exemplo de especificação para o exemplo de controle hierárquico usando o **Grail**.

```
(START) |- 0
0 b 0
0 a 1
1 c 0
0 -| (FINAL)
1 -| (FINAL)
```

Para obter a linguagem-alvo para gerente, diga-se E_{ge}, basta fazer a interseção $E_{ge} = E \cap L_P$. Para fazer este produto é necessário obter um autômato que marque L_P. Isso é facilmente obtido pelo uso da função `fmmark`, como visto abaixo. A FM `pm` representa um autômato que marca L_P e a FM `ege` representa um autômato que marca E_{ge}.

```
c:\>fmmark p > pm
c:\>fmcross pm e | fmreach > ege
c:\>type ege
(START) |- 0
0 a 5
5 c 0
0 -| (FINAL)
5 -| (FINAL)
```

Pode-se provar que $\sup C_{(L_P,\Gamma_P)}(E_{ge}) = K_{ge} = \emptyset$, como visto abaixo. Assim não há comportamento para o gerente que realize uma linguagem que siga à especificação E.

```
c:\>cdessupc p pcont ege > kge
c:\>type kge
c:\>
```

Como visto no capítulo 5, dado um comportamento E_{ge} desejado para o gerente, a linguagem-alvo correspondente para o operador é $E_{op} = \theta^{-1}(E_{ge})$, onde θ é o mapa repórter associado a S e Σ_r. O comportamento realizável para o operador para esta linguagem-alvo é $K_{op} = \sup C_{(L_S,\Gamma_S)}(E_{op}) = \sup C_{(L_S,\Gamma_S)}(\theta^{-1}(E_{ge}))$. E a imagem de K_{op} para o sistema do gerente pelo canal de informação é $K_{opge} = \theta(K_{op}) = \theta(\sup C_{(L_S,\Gamma_S)}(\theta^{-1}(E_{ge})))$. Quando há consistência hierárquica forte, para todo comportamento desejado para o gerente, as linguagens $K_{ge} = \sup C_{(L_P,\Gamma_P)}(E_{ge})$ e K_{opge} são iguais, como mostra o teorema 5.2.

Dada o comportamento desejado para o gerente E_{ge}, deseja-se calcular K_{opge} como explicado acima. Primeiramente calcula-se $E_{op} = \theta^{-1}(E_{ge})$. Pela particularidade do mapa repórter ser definido sobre eventos relevantes para o operador, tem-se $\theta^{-1}(E_{ge}) = p_r^{-1}(E_{ge}) \cap L_S$, onde p_r é a projeção de palavras sobre Σ em Σ_r. A imagem inversa $p_r^{-1}(E_{ge})$ é obtida a partir do autômato que marca E_{ge} inserindo-se *selfloops*, transições de um estado para si mesmo, em todos os estados com os eventos em $\Sigma - \Sigma_r$. Usa-se a função `fmloop` para inserir *selfloops* em um autômato. Assim, na lista de comandos abaixo, calcula-se eop, uma FM que representa um autômato que marca a linguagem E_{op}.

```
c:\>fmmark s > sm
c:\>fmloop ege sigmanr | fmcross sm | fmreach > eop
c:\>type eop
(START) |- 0
0 beta 2
0 alpha 1
1 a 43
2 a 44
```

```
7 mu 0
43 c 7
43 alpha 45
44 c 7
44 alpha 46
45 c 1
0 -| (FINAL)
1 -| (FINAL)
2 -| (FINAL)
7 -| (FINAL)
43 -| (FINAL)
44 -| (FINAL)
45 -| (FINAL)
46 -| (FINAL)
```

Em seguida calcula-se $K_{op} = \sup C_{(L_S,\Gamma_S)}(E_{op})$, como a seguir:

```
c:\>cdessupc s scont eop > kop
c:\>type kop
(START) |- 0
5 mu 0
0 alpha 2
0 beta 1
4 c 5
3 c 5
2 a 4
1 a 3
5 -| (FINAL)
```

Por fim, calcula-se $K_{opge} = \theta(K_{op})$. Veja que $K_{opge} \neq \emptyset = K_{ge}$. Assim não há consistência hierárquica forte entre S e P.

```
c:\>fmproj kop sigmar | fmminrev > kopge
c:\>type kopge
(START) |- 0
0 a 1
1 c 2
2 a 1
2 -| (FINAL)
c:\>fmcontain kopge kge
fmcontain: Lm(g1) >= Lm(g2)
```

A seguir, executa-se uma modificação em S e Σ_r para que a nova hierarquia possua consistência hierárquica forte. Para isso, modificam-se as etiquetas de eventos em Σ_r e correspondentes transições em S para que o mapa repórter associado ao novo sistema seja determinista. Na ferramenta de controle hierárquico usa-se a função `cdesrelabel` para fazer esta modificação. No resultado do comando abaixo, as FMs `snew` e `snewcont` representam o novo sistema S_{new} e a FM `sigmarnew` representa o novo alfabeto $\Sigma_{r,new}$.

```
c:\>cdesrelabel s scont sigmar snew snewcont sigmarnew
c:\>type snew
(START) |- 0
0 beta 2
0 alpha 1
1 a 3
3 c 7
3 alpha 5
4 c 7
4 alpha 6
5 b 7
2 a_1 4
5 c_1 1
```

```
7 mu 0
6 a_2 4
6 -| (FINAL)
7 -| (FINAL)
c:\>cdeswrite snew snewcont
0
({ beta}, N) ({ beta alpha}, N)
1
({ a}, N)
2
({ a_1}, N)
3
({}, N) ({ alpha}, N) ({ c}, N) ({ c alpha}, N)
4
({}, N) ({ alpha}, N) ({ c}, N) ({ c alpha}, N)
5
({ b}, N) ({ b c_1}, N)
6
({ a_2}, M) ({ a_2}, N)
7
({}, M) ({ mu}, M) ({}, N) ({ mu}, N)
c:\>type sigmarnew
0 a 0
0 b 0
0 c 0
0 a_1 0
0 c_1 0
0 a_2 0
```

O sistema do gerente correspondente P_{new} é representado pelas FMs `pnew` e

`pnewcont` obtidas da aplicação da função `cdesconsistent` conforme mostrado abaixo.

```
c:\>cdesconsistent snew snewcont sigmarnew pnew pnewcont
c:\>type pnew
(START) |- 0
0 a 3
0 a_1 4
1 a 3
3 c 7
3 b 7
3 c_1 1
4 c 7
4 a_2 4
7 a 3
7 a_1 4
c:\>cdeswrite pnew pnewcont
0
({ a_1}, N) ({ a a_1}, N)
1
({ a}, N)
3
({ c}, N) ({ b}, N) ({ b c_1}, N) ({ b c}, N) ({ b c c_1}, N)
4
({ a_2}, M) ({ c a_2}, M) ({ a_2}, N) ({ c}, N) ({ c a_2}, N)
7
({}, M) ({ a_1}, M) ({ a a_1}, M) ({ a_1}, N) ({ a a_1}, N)
```

Sabe-se, pelo teorema 5.2 que existe consistência hierárquica forte entre S_{new} e P_{new} acima. Como uma forma de verificação, será refeito o procedimento de síntese anterior, agora usando os novos sistemas.

Para isso é necessário exprimir a especificação E no novo alfabeto $\Sigma_{r,new}$. Para isso, usa-se a função `fmrelabel`. O resultado é a especificação E_{new} representada pela FM `enew` abaixo.

```
c:\>fmrelabel e sigmarnew enew
c:\>type enew
(START) |- 0
0 a 1
0 a_1 1
0 a_2 1
0 b 0
1 c 0
1 c_1 0
0 -| (FINAL)
1 -| (FINAL)
```

A lista de comandos a seguir executa o procedimento de síntese para o nível do gerente. O comportamento realizável resultante corresponde à linguagem $K_{ge,new}$ representada pela FM `kgenew`.

```
c:\>fmmark pnew > pnewm
c:\>fmcross pnewm enew | fmreach > egenew
c:\>cdessupc pnew pnewcont egenew > kgenew
c:\>type kgenew
(START) |- 0
1 c 3
2 c 3
0 a 1
3 a 1
0 a_1 2
```

```
3 a_1 2
3 -| (FINAL)
```

Por outro lado, a lista de comandos a seguir calcula $K_{opge,new}$ a linguagem vista pelo gerente do resultado da síntese no nível do operador, representada pela FM `kopgenew`.

```
c:\>fmmark snew > snewm
c:\>fmloop egenew sigmanr | fmcross snewm | fmreach > topnew
c:\>cdessupc snew snewcont topnew > kopnew
c:\>fmproj kopnew sigmarnew | fmminrev > kopgenew
c:\>type kopgenew
(START) |- 0
0 a 1
0 a_1 1
1 c 2
2 a 1
2 a_1 1
2 -| (FINAL)
```

Por fim, comparam-se as linguagens $K_{opge,new}$ e $K_{ge,new}$, usando-se a função `fmcontain`. O resultado abaixo confirma que $K_{opge,new} = K_{ge,new}$, resultado esperado em função da consistência hierárquica forte.

```
c:\>fmcontain kopgenew kgenew
fmcontain: Lm(g1) == Lm(g2)
```

A distribuição da ferramenta para controle hierárquico contém um diretório de nome `exemplos` onde existem arquivos e *scripts* de comandos para o exemplo apresentado nesta seção e para os outros exemplos apresentados nesta tese, a saber o gato e o rato no labirinto do capítulo 4, a célula de manufatura do capítulo 5 e o sistema de piloto automático no capítulo 6.

Referências Bibliográficas

Alur, R. e Henzinger, T. A. (1999). Reactive modules, *Formal Methods in System Design* **15**(1): 7–48.

Boel, R., Cao, X.-R., Cohen, G., Guia, A., Wonham, W. M. e van Schuppen, J. H. (2002). Unity in diversity, diversity in unity: Retrospective and prospective views on control of discrete event systems, *Discrete Event Dynamic Systems: Theory and Applications* **12**: 253 – 264.

Brandin, B. A. (1996). The real-time supervisory control of an experimental manufacturing cell, *IEEE Transactions on Robotics and Automation* **12**(1): 1–14.

Brandin, B. A. e Wonham, W. M. (1994). Supervisory control of timed discrete-event systems, *IEEE Transactions on Automatic Control* **39**(2): 329–342.

Brave, Y. e Heymann, M. (1993). Control of discrete event systems modeled as hierarchical machines, *IEEE Transactions on Automatic Control* **38**(12): 1803–1819.

Buchholz, P. e Kemper, P. (2002). Efficient computation and representation of large reachability sets for composed automata, *Discrete Event Dynamic Systems* **12**(3): 265–286.

Caines, P. E. e Wei, Y. J. (1995). The hierarchical lattices of a finite machine, *System and Control Letters* **30**(3): 257–263.

Cassandras, C. G. e Lafortune, S. (1999). *Introduction to Discrete Event Systems*, 2 ed., Kluwer Academic Publishers, Massachussets.

CDES Group (2002). Desco – discrete event systems controller, *Technical report*, Universidade de Chelmers. `http://www.s2.chalmers.se/graduate/courses/cdes/` acessada pela última vez em 27 de janeiro de 2003.

Cury, J. E. R. (2001). *Teoria de Controle Supervisório de Sistemas a Eventos Discretos*, V Simpósio Brasileiro de Automação Inteligente, Gramado, RS.

Cury, J. E. R., Krogh, B. H. e Niinomi, T. (1998). Synthesis of supervisory controllers for hybrid systems based on approximating automata, *IEEE Transactions on Automatic Control* **43**(4): 564–568.

Cury, J. E. R., Torrico, C. R. C. e da Cunha, A. E. C. (2001). A new approach for supervisory control of discrete event systems, *Anais da European Control Conference*, Porto, Portugal.

Cury, J. E. R., Torrico, C. R. C. e da Cunha, A. E. C. (2002). Supervisory control of discrete event systems with dynamical marking attribution. Submetido ao European Journal of Control, disponível em `http://www.das.ufsc.br/~aecc`.

da Cunha, A. E. C. e Cury, J. E. R. (2000). Uma metodologia de redução de modelos para sistemas a eventos discretos para síntese de supervisores, *Anais do XIII Congresso Brasileiro de Automática*, Florianópolis, SC, p. 2257–2262.

da Cunha, A. E. C. e Cury, J. E. R. (2002a). Building consistent hierarchies of discrete event systems. Submetido à revista Journal of Discrete Event Dynamic Systems: Modelling and Control, disponível em `http://www.das.ufsc.br/~aecc`.

da Cunha, A. E. C. e Cury, J. E. R. (2002b). Hierarchically consistent controlled discrete event systems, *Anais do IFAC World Congress*, Barcelona, Espanha.

da Cunha, A. E. C., Cury, J. E. R. e Krogh, B. H. (2002). An assume-guarantee reasoning for hierarchical coordination of discrete event systems, *Anais do Workshop on Discrete Event Systems (WODES)*, Zaragoza, Espanha, p. 75–80.

de Queiroz, M. H. (2000). *Controle Modular Local para Sistemas de Grande Porte*, Dissertação (mestrado), Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, SC.

de Queiroz, M. H. (2002). *Contribuições ao controle supervisório de sistemas compostos*, Proposta de tese, Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis.

de Queiroz, M. H. e Cury, J. E. R. (2002a). Controle supervisório modular de sistemas de manufatura, *Revista Controle & Automação* **13**(2): 115–125.

de Queiroz, M. H. e Cury, J. E. R. (2002b). Synthesis and implementation of local modular supervisory control for a manufacturing cell, *Anais do Workshop on Discrete Event Systems (WODES)*, Zaragoza, Espanha, p. 377–382.

Eyzell, J. M. e Cury, J. E. R. (2001). Exploiting symmetry in the synthesis of supervisors for discrete event systems, *IEEE Transactions on Automatic Control* **46**(9): 1500–1505.

Fabian, M. e Hellgren, A. (1998). PLC-based implementation of supervisory control for discrete-event systems, *Anais da* 37^a *IEEE Conference on Decision and Control*, Tampa, Florida, USA.

Girard, A. R., de Souza, J. B., Misener, J. A. e Hedrick, J. K. (2001). A control architecture for integrated cooperative cruise control and collision warning systems, *Anais da Control and Decision Conference*, Florida.

Godbole, D. N. e Lygeros, J. (1994). Longitudinal control of the lead car of a platoon, *IEEE Transactions on Vehicular Technology* **43**(4): 1125–1135.

Gohari-Moghadam, P. e Wonham, W. M. (1998). A linguistic framework for controlled hierarchical DES, *Anais do Workshop on Discrete Event Systems*, Cagliari, Itália, p. 207–212.

Gohari, P. e Wonham, W. M. (2000). Reduced supervisors for timed discrete-event systems, *In:* R. Boel e G. Stremersch (Eds.), *Discrete Event Systems: Analysis and Control*, Kluwer Academic Publishers, Ghent, Bélgica, p. 119–130.

Golaszewski, C. e Ramadge, P. (1987). Control of discrete event processes with forced events, *Anais da* 26^a *Conference on Decision and Control*, Los Angeles, CA, USA, p. 247–252.

González, J. M. E. (2000). *Aspectos de Síntese de Supervisores para Sistemas a Eventos Discretos e Sistemas Híbridos*, Tese (doutorado), Programa de Pós Graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, Brasil.

González, J. M. E., da Cunha, A. E. C., Cury, J. E. R. e Krogh, B. H. (2001). Supervision of event driven hybrid systems: Modeling and synthesis, *In:* M. D. Benedetto e A. Sangiovanni-Vincentelli (Eds.), *Hybrid Systems: Computation and Control, 4th International Workshop, HSCC 2001*, v. 2034 de *LNCS*, Rome, Italy, p. 0247–0261.

Guan, Y.-C. (1997). *Implementation of Hierarchical Observer Theory*, Dissertação (mestrado), Systems Control Group, Department of Electrical & Computer Engineering, University of Toronto, Toronto, Canada.

Harel, D. (1987). Statecharts: A visual formalism for complex systems, *Science of Computer Programming* **3**(8): 231–274.

Hopcroft, J. E. e Ullmann, J. D. (1979). *Introduction to Automata Theory, Languages and Computation*, 1 ed., Addison Wesley Publishing Company, Reading.

Hsu, A., Eskafi, F., Sachs, S. e Varaiya, P. (1993). Protocol design for an automated highway system, *Discrete Event Dynamic Systems: Theory and Applications* **2**(1): 183–206.

Hubbard, P. e Caines, P. E. (2002). Dynamical consistency in hierarchical superviosry control, *IEEE Transactions on Automatic Control* **47**(1): 37–52.

Kumar, R. e Garg, V. K. (1995). *Modeling and Control of Logical Discrete Event Systems*, 1 ed., Kluwer Academic Publishers, Boston.

Lauzon, S. C., Ma, A. K. L., Mills, J. K. e Benhabib, B. (1996). Application of discrete-event-system theory to flexible manufacturing, *IEEE Control Systems Magazine* p. 41–48.

Leduc, R. J. (1996). *PLC Implementation of DES Supervisor for a Manufacturing Testbed: an Implementation Perpective*, Dissertação (mestrado), Systems Control Group, Department of Electrical & Computer Engineering, University of Toronto, Toronto, Canada.

Leduc, R. J., Brandin, B. A., Wonham, W. M. e Lawford, M. (2001). Hierarchical interface-based supervisory control: Serial case, *Anais da 40ª IEEE Conference on Decision and Control*, v. 5, p. 4116–4121.

Li, Y., Lin, F. e Lin, Z. H. (1998). A generalized framework for supervisory control of discrete event systems, *International Journal of Intelligent Control and Systems* **2**(1): 139–159.

Li, Y., Lin, F. e Lin, Z. H. (1999). Supervisory control of probabilistic discrete event systems with recovery, *IEEE Transactions on Automatic Control* **44**(10): 1971–1974.

Lima, E. L. (1989). *Curso de Análise*, v. 1 de *Projeto Euclides*, 6 ed., Instituto de Matemática Pura e Aplicada – CNPq, Rio de Janeiro.

Ma, C. (1999). *A Computational Approach to Top-Down Hierarchical Supervisory Control of DES*, Dissertação (mestrado), Systems Control Group, Department of Electrical & Computer Engineering, University of Toronto, Toronto, Canada.

Maier, P. (2001). A set-theoretic framework for assume-guarantee reasoning, *Relatório Técnico MPI-I-2001-2-002*, Max-Planck-Institut für Informatik.

Martins, E. D. (1999). *Uma Arquitetura Física para a Implementação do Controle Supervisório de Sistemas a Eventos Discretos*, Dissertação (mestrado), Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, SC.

Martins, E. D. e Cury, J. E. R. (1997). Uma arquitetura para implementação de controle supervisório de sistemas a eventos discretos, *Anais do 3º Simpósio Brasileiro de Automação Inteligente*, Vitória, ES, p. 184–189.

Menezes, P. B. (2001). *Linguagens Formais e Autômatos*, n. 3 de *Série Livros Didáticos*, 4 ed., Editora Sagra Luzzato, Instituto de Informática da UFRGS.

Mesarovic, M. D., Macko, D. e Takahara, Y. (1970). *Theory of Hierarchical Multilevel Systems*, 1 ed., Academic Press, New York.

Park, S.-J. e Lim, J.-T. (2001). Hierarchical supervisory control of discrete event systems with model uncertainty, *International Journal of Systems Science* **32**(6): 739 – 744.

Pu, K. Q. (2000). *Modeling and Control of Discrete-Event Systems with Hierarchical Abstraction*, Dissertação (mestrado), Systems Control Group, Department of Electrical & Computer Engineering, University of Toronto, Toronto, Canada.

Ramadge, P. J. G. e Wonham, W. M. (1989). The control of discrete event systems, *Proceedings of the IEEE* **77**(1): 81–98.

Ramadge, P. J. e Wonham, W. M. (1987a). Modular feedback logic for discrete event systems, *SIAM Journal of Control and Optimization* **25**(5): 1202–1218.

Ramadge, P. J. e Wonham, W. M. (1987b). Supervisory control of a class of discrete event processes, *SIAM Journal of Control and Optimization* **25**(1): 206–230.

Raymond, D. e Wood, D. (1996a). Grail: Engineering automata in C++: Version 2.5, `http://www.csd.uwo.ca/research/grail/` verificada em 31 de janeiro de 2003.

Raymond, D. e Wood, D. (1996b). Programmer's guide to grail: Version 2.5, `http://www.csd.uwo.ca/research/grail/` verificada em 31 de janeiro de 2003.

Raymond, D. e Wood, D. (1996c). Release notes for grail: Version 2.5, `http://www.csd.uwo.ca/research/grail/` verificada em 31 de janeiro de 2003.

Raymond, D. e Wood, D. (1996d). User's guide to grail: Version 2.5, `http://www.csd.uwo.ca/research/grail/` verificada em 31 de janeiro de 2003.

Rudie, K. G. (1988). *SOFTWARE FOR THE CONTROL OF DISCRETE EVENT SYSTEMS: A Complexity Study*, Dissertação (mestrado), Systems Control Group, Department of Electrical & Computer Engineering, University of Toronto, Toronto, Canada.

Rudie, K. e Wonham, W. M. (1992). Think globally, act locally: Decentralized supervisory control, *IEEE Transactions on Automatic Control* **37**(11): 1692–1708.

Sathaye, A. S. e Krogh, B. H. (1998). Supervisor synthesis for real-time discrete event systems, *Discrete Event Dynamic Systems: Theory and Applications* **8**(1): 5–35.

Stark, E. W. (1985). A proof technique for rely-guarantee properties, *FST & TCS 85: Foundations of Software Technology and Theoretical Computer Science*, v. 206 de *Lecture Notes in Computer Science*, Springer-Verlag, p. 369–391.

Thistle, J. G. e Wonham, W. M. (1994a). Control of infinite behavior of finite automata, *SIAM J. Control and Optimization* **32**(4): 1075–1097.

Thistle, J. G. e Wonham, W. M. (1994b). Supervision of infinite behavior of discrete-event systems, *SIAM J. Control and Optimization* **32**(4): 1098–1113.

Torrico, C. C. e Cury, J. E. R. (2002a). Hierarchical supervisory control of discrete event systems based on state aggregation, *Anais do IFAC World Congress*, Barcelona, Espanha.

Torrico, C. R. C. (1999). *Implementação de Controle Supervisório de Sistemas a Eventos Discretos Aplicado a Processos de Manufatura*, Dissertação (mestrado), Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, SC.

Torrico, C. R. C. (2003). *Controle Supervisório Hierárquico de Sistemas a Eventos Discretos: Uma Abordagem Baseada na Agregação de Estados*, Tese (doutorado), Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis.

Torrico, C. R. C. e Cury, J. E. R. (2002b). Controle supervisório hierárquico modular por agregação de estados, *Anais do XIV Congresso Brasileiro de Automática*, Natal, RN, Brasil, p. 1936–1941.

UMDES Group (1998). User's guide for UMDES-LIB software, *Technical report*, The Electrical Engineering and Computer Science Department, University of Michigan. `http://www.eecs.umich.edu/umdes` acessada pela última vez em 27 de janeiro de 2003.

Vaz, A. F. e Wonham, W. M. (1986). On supervisor reduction in discrete-event systems, *International Journal of Control* **44**(2): 475–491.

Wang, B. (1995). *Top-Down Design for RW Supervisory Control Theory*, Dissertação (mestrado), Systems Control Group, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada.

Wong, K. C. (1994). *Discrete-Event Control Architecture: an Algebraic Approach*, Tese (doutorado), Systems Control Group, Department of Electrical Engineering, University of Toronto, Toronto, Canada.

Wong, K. C. (1998). On the complexity of projections of discrete-event systems, *Anais do Workshop on Discrete Event Systems*, Cagliari, Itália, p. 201–206.

Wong, K. C., Thistle, J. G., Malhame, R. P. e Hoang, H.-H. (1995). Conflict resolution in modular control with feature interaction, *Proceedings of The 34th Conference of Decision and Control*, New Orleans, LA, p. 416–421.

Wong, K. C., Thistle, J. G., Malhame, R. P. e Hoang, H.-H. (1998). Supervisory control of distributed systems: Conflict resolution, *Anais da 37ª IEEE Conference on Decision and Control*, Tampa, Florida, p. 3275–3280.

Wong, K. C. e Wonham, W. M. (1996a). Hierarchical control of discrete-event systems, *Discrete Event Dynamic Systems* **6**(3): 241–273.

Wong, K. C. e Wonham, W. M. (1996b). Hierarchical control of timed discrete-event systems, *Discrete Event Dynamic Systems* **6**(3): 275–306.

Wong, K. C. e Wonham, W. M. (1998). Modular control and coordination of discrete-event systems, *Discrete Event Dynamic Systems* **8**(3): 247–297.

Wong, K. C. e Wonham, W. M. (2000). On the computation of observers in discrete-event systems, *Anais da 2000 Conference on Information Sciences and Systems*, Princenton University.

Wong, K., Thistle, J., Malhamé, R. e Hoang, H.-H. (2000). Supervisory control of distributed systems: Conflict resolution, *Discrete Event Dynamic Systems: Theory and Applications* **10**(1–2): 131–186.

Wonham, W. M. (2002a). CTCT software for control synthesis, Systems Control Group, Dept. of Electrical & Computer Engineering, U. of Toronto, em `http://www.control.utoronto.ca` sob o título *Research*.

Wonham, W. M. (2002b). *Notes on Control of Discrete-Event Systems*, Systems Control Group, Department of Electrical & Computer Engineering, University of Toronto, Toronto, Canada.

Wonham, W. M. e Ramadge, P. J. (1987). On the supremal controllable sublanguage of a given language, *SIAM Journal of Control and Optimization* **25**(3): 637–659.

Wonham, W. e Zhong, H. (1990). Hierarchical coordination, *Anais do 5º IEEE International Symposium on Intelligent Control*, v. 1, p. 8–14.

Zad, S. H. (1999). *Fault Diagnosis in Discrete-Event and Hybrid Systems*, Tese (doutorado), Systems Control Group, Department of Electrical & Computer Engineering, University of Toronto, Toronto, Canada.

Zad, S. H., Kwong, R. H. e Wonham, W. M. (1998). Fault diagnosis in discrete-event systems: Framework and model reduction, *Anais da 37ª IEEE Conference on Decision and Control*, Tampa, Florida, USA, p. 3769–3774.

Zhong, H. e Wonham, W. M. (1990). On the consistency of hierarchical supervision in discrete-event systems, *IEEE Transactions on Automatic Control* **35**(10): 1125–1134.

Ziller, R. M. (1993). *A Abordagem Ramadge-Wonham no Controle de Sistemas a Eventos Discretos: Contribuições à Teoria*, Dissertação (mestrado), Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, SC.