

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

**A ABORDAGEM RAMADGE-WONHAM NO CONTROLE DE
SISTEMAS A EVENTOS DISCRETOS: CONTRIBUIÇÕES À TEORIA**

ROBERTO M. ZILLER

**DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA
CATARINA PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA
ELÉTRICA**

FLORIANÓPOLIS, 21/10/1993.

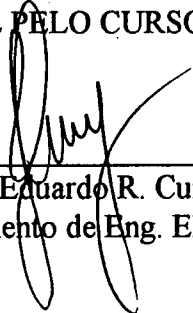
A ABORDAGEM RAMADGE-WONHAM NO CONTROLE DE SISTEMAS A EVENTOS DISCRETOS: CONTRIBUIÇÕES À TEORIA

ROBERTO M. ZILLER

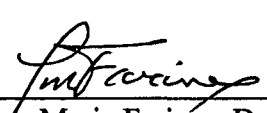
ESTA DISSERTAÇÃO FOI JULGADA PARA A OBTENÇÃO DO TÍTULO DE

MESTRE EM ENGENHARIA

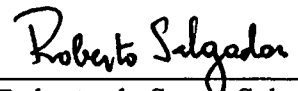
ESPECIALIDADE ENGENHARIA ELÉTRICA, ÁREA DE CONCENTRAÇÃO SISTEMAS DE CONTROLE E AUTOMAÇÃO INDUSTRIAL, E APROVADA EM SUA FORMA FINAL PELO CURSO DE PÓS-GRADUAÇÃO.



Prof. José Eduardo R. Cury, Dr. D'Etat
Departamento de Eng. Elétrica, UFSC
Orientador

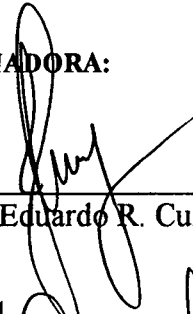


Prof. Jean-Marie Farines, Dr. Ing.
Departamento de Eng. Elétrica, UFSC
Co-orientador



Prof. Roberto de Souza Salgado, Ph. D.
Coordenador do Curso de Pós-Graduação em Engenharia Elétrica

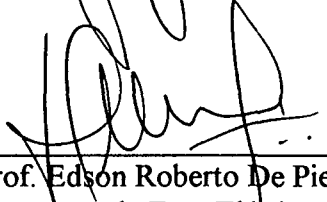
BANCA EXAMINADORA:




Prof. José Eduardo R. Cury, Dr. D'Etat
Orientador



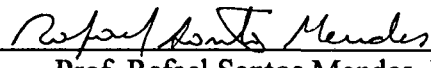
Prof. Jean-Marie Farines, Dr. Ing.
Co-orientador



Prof. Edson Roberto De Pieri, Dr.
Departamento de Eng. Elétrica, UFSC



Prof.ª Janette Cardoso, Dr.ª
Departamento de Eng. Elétrica, UFSC



Prof. Rafael Santos Mendes, Dr.
Departamento de Eng. Elétrica, UNICAMP

"O artista vê que a arte não tem limite; verifica ele quanto está sempre longe do seu ideal e, enquanto é admirado por todos, chora não poder alcançar o mito ambicionado, que sempre lhe foge e fulge como um sol longínquo".

Ludwig van Beethoven

Este trabalho é dedicado a meus tios, Anneliese e Rolf Moecke e à minha prima Cristine. Mais do que a casa e a comida que me deram, foram seu amor e seu carinho que tornaram inesquecíveis estes anos de convívio.

AGRADECIMENTOS

Ao término deste trabalho, quero manifestar minha gratidão ao professor José Eduardo Ribeiro Cury, pela dedicação e pela amizade que marcaram seu trabalho como orientador durante toda a elaboração desta dissertação. O sucesso alcançado se deve sobretudo à sua iniciativa de visitar a Universidade de Toronto, em setembro de 1992, com o que iniciou um intercâmbio de informações entre o LCMI (Laboratório de Controle e Microinformática do Departamento de Engenharia Elétrica da UFSC) e o Systems Control Group daquela universidade, dirigido pelo professor W. M. Wonham. Esta visita foi retribuída em fevereiro deste ano com a vinda do professor Wonham ao LCMI. As suas palestras, o material bibliográfico e o software recebidos por ocasião desses dois contatos foram de valor inestimável para o progresso do trabalho.

Agradeço ainda ao professor Wonham a sugestão de trabalhar no esclarecimento do papel das linguagens geradas e marcadas no modelo RW, e à Dr.^a Karen G. Rudie, da Queen's University, Kingston, Canadá, pela ajuda na realização desta tarefa.

Ao professor Jean-Marie Farines agradeço pelo acompanhamento do trabalho como co-orientador e pelas valiosas sugestões na busca de aplicações da teoria à Informática.

Agradeço também aos demais professores e colegas do LCMI pela amizade e pelas palavras de ânimo que nunca faltaram; à família do meu tio Rolf Moecke, da qual fiz parte nestes quase três anos de trabalho; à minha mãe, por ter feito da educação de seus filhos a meta maior em sua vida, e a todos os amigos que, com palavras, gestos ou ações, expressaram sua aprovação pelo meu trabalho e me incentivaram.

Ao contribuinte brasileiro e ao CNPq meu muito obrigado pelo apoio financeiro.

RESUMO

O estudo da Teoria de Sistemas a Eventos Discretos tem importância crescente na tecnologia moderna, devido à sua aplicação em áreas tais como a automação da manufatura, a robótica, a supervisão de tráfego, os sistemas de comunicação de computadores e a verificação de software, dentre outras. Em razão disso, têm surgido diversos modelos matemáticos para representar e estudar esses sistemas. O presente trabalho apresenta uma visão geral das abordagens existentes, aprofundando-se em seguida naquela proposta por Ramadge e Wonham, baseada em Teoria de Linguagens e de Autômatos. Uma possibilidade de formulação de problemas, não explorada no modelo original, é analisada e desenvolvida, dando origem a um novo problema abstrato de síntese de supervisores. Os resultados obtidos são então estendidos ao controle supervisorio descentralizado.

ABSTRACT

The study of Discrete Event Systems Theory has an increasing importance in current technology, due to its application in many different areas, like manufacturing automation, robotics, traffic control, computer communication systems and software verification. Hence, several abstract models to represent this class of systems have been suggested. The present work gives an overview of the existing models, treating in detail the one proposed by Ramadge and Wonham, which is based on Language and Automata Theory. A problem formulation not explored in the original theory is analyzed and developed, giving origin to a new abstract supervisor synthesis problem. The results obtained are extended to the decentralized control case.

SUMÁRIO

Agradecimentos	v
Resumo.....	vi
Abstract	vii
Lista de Figuras.....	x
Lista de Tabelas	xii
Notação	xiii
Abreviaturas e Siglas.....	xvi
1 - Introdução	1-1
1.1 - Apresentação e Abrangência	1-1
1.2 - Justificativa para a Escolha da Abordagem Ramadge-Wonham.....	1-2
1.3 - Objetivos e Organização do Trabalho.....	1-3
2 - Sistemas a Eventos Discretos.....	2-1
2.1 - Características e Definição	2-1
2.2 - Sistemas Discretos e Sistemas Contínuos	2-3
2.3 - Áreas Relacionadas à Teoria de Sistemas a Eventos Discretos.....	2-4
2.4 - Inexistência de um Modelo Universal para SED's	2-5
2.5 - Modelos para SED's.....	2-6
2.5.1 - Redes de Petri	2-7
2.5.2 - Redes de Petri Controladas.....	2-8
2.5.3 - Cadeias de Markov.....	2-9
2.5.4 - Teoria das Filas	2-10
2.5.5 - Processos Semi-Markovianos Generalizados e Simulação.....	2-10
2.5.6 - Álgebra de Processos	2-11
2.5.7 - Álgebra Max-Plus.....	2-11
2.5.8 - Lógica Temporal	2-12
2.5.9 - Lógica Temporal de Tempo Real Generalizada (GRTTL).....	2-14
2.5.10 - Teoria de Autômatos e de Linguagens.....	2-14
2.6 - Comparação e Conclusões	2-15
3 - Teoria de Linguagens e de Autômatos	3-1
3.1 - SED's e Teoria de Linguagens.....	3-1
3.2 - Autômatos	3-5
3.3 - Geradores	3-9
3.4 - Projeções	3-12
3.5 - Combinação de Geradores.....	3-14
4 - O Modelo RW	4-1
4.1 - Representação de SED's no Modelo RW.....	4-1
4.2 - Geradores com Entrada de Controle.....	4-3

4.3 - Supervisores	4-5
4.3.1 - Definição.....	4-5
4.3.2 - Linguagens.....	4-10
4.3.3 - Supervisores próprios.....	4-12
4.4 - Formulação de Problemas no Modelo RW.....	4-14
4.4.1 - Formulação em Termos de Linguagens Marcadas.....	4-14
4.4.2 - Formulação em Termos de Linguagens Geradas.....	4-14
4.5 - L-fechamento.....	4-15
4.6 - Controlabilidade.....	4-16
4.7 - Existência de Supervisores.....	4-18
4.8 - Síntese de Supervisores.....	4-23
4.9 - Conclusão.....	4-27
5 - Contribuições ao Modelo RW.....	5-1
5.1 - PCSII.....	5-1
5.2 - L-compatibilidade.....	5-2
5.3 - Existência de Supervisores - Um Novo Teorema.....	5-2
5.4 - Solução para PCSII.....	5-3
5.5 - Equivalência entre as formulações apresentadas.....	5-4
5.6 - Conclusão.....	5-10
6 - Contribuições ao Modelo RW Descentralizado.....	6-1
6.1 - Noções Preliminares.....	6-2
6.1.1 - Supervisores Globais e Supervisores Locais.....	6-2
6.1.2 - Normalidade e Decomponibilidade.....	6-3
6.1.3 - Conjunção de Supervisores.....	6-4
6.1.4 - Extensão Global de um Supervisor Local.....	6-5
6.2 - Tipos de Problemas de Controle Descentralizado.....	6-6
6.3 - A Formulação Encontrada na Literatura.....	6-7
6.3.1 - Problema Local (LP).....	6-8
6.3.2 - Problema Global (GP).....	6-9
6.4 - As Formulações Propostas.....	6-11
6.5 - O Problema da Transmissão Sequencial de Dados.....	6-12
6.5.1 - Caso 1: Transmissor Solução e Receptor Solução.....	6-19
6.5.2 - Caso 2: Transmissor Solução e Receptor Genérico.....	6-21
6.5.3 - Caso 3: Transmissor Genérico e Receptor Solução.....	6-27
6.5.4 - Caso 4: Transmissor Genérico e Receptor Genérico.....	6-27
6.6 - Caminhos Opcionais.....	6-28
6.7 - Conclusão do Capítulo.....	6-36
7 - Conclusão.....	7-1
8 - Referências Bibliográficas.....	8-1

LISTA DE FIGURAS

Fig. 2.1 - Trajetória típica de um sistema a eventos discretos.....	2-2
Fig. 2.2 - Trajetória típica de um sistema dinâmico com variáveis contínuas.....	2-4
Fig. 2.3 - Áreas relacionadas à Teoria de Sistemas a Eventos Discretos.....	2-5
Fig. 2.4 - Disparo de transição numa rede de Petri.....	2-8
Fig. 2.5 - Rede de Petri controlada.....	2-9
Fig. 2.6 - Semântica dos operadores da lógica temporal.....	2-13
Fig. 3.1 - Modelo ilustrativo de um autômato.....	3-5
Fig. 3.2 - Diagrama de transição representando o autômato do exemplo 3.6.....	3-7
Fig. 3.3 - Gerador coaccessível x gerador não coaccessível.....	3-12
Fig. 3.4 - Geradores para o exemplo 3.11.....	3-13
Fig. 3.5 - Gerador para o exemplo 3.13.....	3-13
Fig. 3.6 - Geradores para o exemplo 3.14.....	3-14
Fig. 3.7 - Resultado do produto síncrono.....	3-15
Fig. 4.1 - Gerador não finito.....	4-2
Fig. 4.2 - Gerador representando uma máquina com três estados.....	4-2
Fig. 4.3 - Acoplamento de planta e supervisor no modelo RW.....	4-5
Fig. 4.4 - O almoço dos filósofos orientais.....	4-6
Fig. 4.5 - Planta composta pelos filósofos F4 e F5.....	4-7
Fig. 4.6 - Comportamento desejado para F4 e F5.....	4-8
Fig. 4.7 - Filósofos F4 e F5 - planta e supervisor acoplados.....	4-9
Fig. 4.8 - Gerador para o exemplo 4.5.....	4-13
Fig. 4.9 - Supervisor com bloqueio e comportamento resultante.....	4-13
Fig. 4.10 - Supervisor com rejeição e comportamento resultante.....	4-14
Fig. 4.11 - Gerador para ilustrar L-fechamento.....	4-15
Fig. 4.12 - Planta composta pelos filósofos F1 e F3.....	4-17
Fig. 4.13 - Comportamento desejado para os filósofos F1 e F3.....	4-17
Fig. 6.1 - Controle supervisorio descentralizado.....	6-6
Fig. 6.2 - Gerador para ilustrar as limitações de GP.....	6-10
Fig. 6.3 - Topologia do Sistema de Comunicação.....	6-13
Fig. 6.4 - Gerador "CANAL".....	6-15
Fig. 6.5 - Gerador STX - Transmissor solução.....	6-16
Fig. 6.6 - Gerador SRX - Receptor solução.....	6-16
Fig. 6.7 - Gerador GTX - Transmissor genérico.....	6-16
Fig. 6.8 - Gerador GRX - Receptor genérico.....	6-16
Fig. 6.9 - Gerador "LEGAL".....	6-17
Fig. 6.10 - Gerador G1.....	6-19
Fig. 6.11 - Gerador E11.....	6-20

Fig. 6.12 - Gerador E12	6-20
Fig. 6.13 - Gerador G2	6-22
Fig. 6.14 - Gerador E2	6-22
Fig. 6.15 - Gerador E21	6-23
Fig. 6.16 - Gerador E22	6-23
Fig. 6.17 - Gerador TEST2	6-24
Fig. 6.18 - Gerador SUPE22	6-25
Fig. 6.19 - Gerador RES2	6-25
Fig. 6.20 - Gerador com caminhos opcionais	6-29
Fig. 6.21 - Gerador com duas componentes irredutíveis	6-30
Fig. 6.22 - Gerador G4	6-32
Fig. 6.23 - Gerador E4	6-32
Fig. 6.24 - Gerador X4	6-33
Fig. 6.25 - Gerador Y4	6-34

LISTA DE TABELAS

Tab. 2.1 - Modelos para Sistemas a Eventos Discretos	2-15
Tab. 3.1 - Tabela de transição para o autômato do exemplo 3.6.....	3-7
Tab. 4.1 - Mapa de controle para os filósofos F4 e F5	4-8
Tab. 4.2 - Mapa de controle para F4 e F5, forma matricial.....	4-8
Tab. 4.3 - Linguagens utilizadas no modelo RW	4-11
Tab. 6.1 - Algumas funções do programa TCT	6-18
Tab. 6.2 - Mapa de controle	6-35
Tab. 6.3 - Mapa de controle	6-35

NOTAÇÃO

\in	Pertencente a
\notin	Não pertencente a
:	Tal que
\exists	Existe
\forall	Para todo
\emptyset	Conjunto vazio
\cup	Operador união de conjuntos
\cap	Operador interseção de conjuntos
\wedge	Operador "e"
\vee	Operador "ou"
◆	Indica final de demonstrações e de exemplos
\mathbb{N}	Conjunto dos números naturais
$\alpha, \beta, \lambda, \mu, \dots$	Eventos
Σ	Alfabeto ou conjunto de eventos
r, s, u, w, \dots	Palavras
$ s $	Comprimento da palavra s
ϵ	Palavra nula
$\text{Pre}(s)$	Conjunto de todos os prefixos da palavra s
Σ^+	O conjunto de todas as palavras que podem ser formadas com letras de Σ
Σ^*	Idem, incluindo a palavra nula ϵ
Σ^k	O conjunto das palavras $\{s: s \in \Sigma^* \wedge s = k\}$
$\delta(\sigma, q)!$	A função δ é definida para o par (σ, q)
$\delta(s, q)!$	A função δ é definida para o par (s, q)
$\delta \Sigma^* \times Q - Q_1$	A função δ restrita ao domínio $\Sigma^* \times Q - Q_1$
Q, X	Conjuntos de estados
$ Q $	Cardinalidade do conjunto Q
A	Autômato

G, S, T	Geradores
K, L	Linguagens
\bar{K}	Prefixo-fechamento de K
KL	A linguagem $\{st: s \in K \wedge t \in L\}$
$K - L$	A linguagem $\{s: s \in K \wedge s \notin L\}$
$L(G)$	Linguagem gerada de G
$L_m(G)$	Linguagem marcada de G
Σ_c	Conjunto de eventos controláveis
Σ_u	Conjunto de eventos não controláveis
$\Sigma(q)$	O conjunto de eventos tais que $\sigma \in \Sigma(q) \Rightarrow \delta(\sigma, q)!$, isto é, o conjunto de eventos que podem ocorrer a partir do estado q
$\langle G, \Gamma \rangle$	Planta formada pelo gerador G com conjunto de entradas de controle válidas Γ
$\langle S, \Phi \rangle$	Supervisor formado pelo gerador S e pelo mapa de controle Φ
$L(S/G)$	Linguagem gerada de $\langle G, \Gamma \rangle$ sob supervisão de $\langle S, \Phi \rangle$
$L_c(S/G)$	Linguagem controlada de $\langle G, \Gamma \rangle$ sob supervisão de $\langle S, \Phi \rangle$
$L_m(S/G)$	Linguagem marcada de $\langle G, \Gamma \rangle$ sob supervisão de $\langle S, \Phi \rangle$
$C(K)$	Conjunto das sublinguagens $L(G)$ -controláveis de K
$F(K)$	Conjunto das sublinguagens $L_m(G)$ -fechadas de K
$P(K)$	Conjunto das sublinguagens prefixo-fechadas de K
$T(K)$	Conjunto das sublinguagens $L_m(G)$ -compatíveis de K
$\sup C(K)$	Máxima sublinguagem $L(G)$ -controlável de K
$\sup F(K)$	Máxima sublinguagem $L_m(G)$ -fechada e K
$\sup P(K)$	Máxima sublinguagem prefixo-fechada de K
$\sup T(K)$	Máxima sublinguagem $L_m(G)$ -compatível de K
$\sup CF(K)$	Máxima sublinguagem $L(G)$ -controlável e $L_m(G)$ -fechada de K
$\sup CT(K)$	Máxima sublinguagem $L(G)$ -controlável e $L_m(G)$ -compatível de K
Σ_l	Alfabeto local
Σ_{lc}	Conjunto de eventos controláveis em nível local

- $P: \Sigma^* \rightarrow \Sigma_i^*$ Projeção mapeando palavras de Σ^* em Σ_i^*
 $\langle S \times T, \Phi * \Psi \rangle$ Conjunção dos supervisores $\langle S, \Phi \rangle$ e $\langle T, \Psi \rangle$
 $\langle \tilde{S}, \tilde{\Phi} \rangle$ Extensão global do supervisor $\langle S, \Phi \rangle$
 $\bigwedge_{i=1}^n \tilde{S}_i$ Conjunção dos supervisores $\langle \tilde{S}_i, \tilde{\Phi}_i \rangle, i = 1, \dots, n$
 $L\left(\bigwedge_{i=1}^n \tilde{S}_i / G\right)$ Linguagem gerada da planta $\langle G, \Gamma \rangle$ sob ação da conjunção dos supervisores $\langle S_i, \Phi_i \rangle, i = 1, \dots, n$

ABREVIATURAS E SIGLAS

Def.	Definição
Dem.	Demonstração
Ex.	Exemplo
Fig.	Figura
DSP1	Decentralized Supervisory Control Problem 1
DSP2	Decentralized Supervisory Control Problem 2
GP	Problema Global
GPG	Problema Global formulado em termos de linguagens geradas
GPM	Problema Global formulado em termos de linguagens marcadas
LP	Problema Local
PCSI	Problema de Controle Supervisório I
PCSIG	PCSI para Linguagens Geradas
PCSII	Problema de Controle Supervisório II
Prop.	Proposição
RW	Ramadge-Wonham
SCP	Supervisory Control Problem
SED	Sistema a eventos discretos
SMP	Supervisory Marking Problem
sse	se e somente se
Tab.	Tabela
Teo.	Teorema

CAPÍTULO 1

INTRODUÇÃO

1.1 APRESENTAÇÃO E ABRANGÊNCIA

A tecnologia moderna tem produzido, em escala crescente, sistemas com a finalidade de executar tarefas que, seja pela importância que adquirem em seu contexto, seja por sua complexidade e seu custo, justificam o esforço despendido na sua otimização e automação. Tais sistemas estão presentes em uma série de aplicações, incluindo por exemplo a automação da manufatura, a robótica, a supervisão de tráfego, a logística (canalização e armazenamento de produtos, organização e prestação de serviços), sistemas operacionais, redes de comunicação de computadores, concepção de software, gerenciamento de bases de dados e otimização de processos distribuídos ([RW89a], [CR90]).

Tais sistemas têm em comum a maneira pela qual percebem as ocorrências no ambiente à sua volta, o que se dá pela recepção de estímulos, denominados *eventos*. São exemplos de eventos o início e o término de uma tarefa e a percepção de uma mudança de estado em um sensor. Estes eventos são, por sua natureza, instantâneos, o que lhes confere um caráter discreto no tempo. Sistemas com estas características são denominados *sistemas a eventos discretos* (SED's), em oposição aos sistemas de variáveis contínuas, tratados pela Teoria de Controle clássica.

A natureza discreta dos SED's faz com que os modelos matemáticos convencionais, baseados em equações diferenciais, não sejam adequados para tratá-los. Por outro lado, a sua importância faz com que seja altamente desejável encontrar soluções para problemas relacionados ao seu controle.

Em razão disso, existe uma intensa atividade de pesquisa voltada à busca de modelos matemáticos adequados à sua representação, sem que se tenha conseguido até agora encontrar um modelo que seja "matematicamente tão conciso e computacionalmente tão adequado como o são as equações diferenciais para os sistemas dinâmicos de variáveis contínuas. Não existe, por isso, consenso sobre qual seja o melhor modelo." (citação de [Flem88] em [CH90]). Em [Ho87] e [Ho89], o autor afirma ser precisamente a inexistência de um tal paradigma a razão do estado relativamente primitivo em que se encontram a análise e a síntese dos sistemas citados.

Dentre os modelos existentes, destaca-se o proposto por Ramadge e Wonham em [RW87], baseado em Teoria de Linguagens e de Autômatos e denominado "modelo RW". Este faz uma distinção clara entre o sistema a ser controlado, denominado *planta*, e a entidade que o controla, que recebe o nome de *supervisor*. A planta é um modelo que reflete o comportamento fisicamente possível do sistema, isto é, todas as ações que este é capaz de

executar na ausência de qualquer ação de controle. Em geral, este comportamento inclui a capacidade de realizar determinadas atividades que produzam um resultado útil, sem contudo se limitar a esse comportamento desejado. Por exemplo, dois robôs trabalhando em uma célula de manufatura podem ter acesso a um depósito de uso comum, o que pode ser útil para passar peças de um ao outro. No entanto, cria-se com isso a possibilidade física de ocorrer um choque entre ambos, o que é, em geral, indesejável.

O papel do supervisor no modelo RW é, então, o de exercer uma ação de controle restritiva sobre a planta, de modo a confinar seu comportamento àquele que corresponde a uma dada especificação.

Uma vantagem desse modelo é a de permitir a síntese de supervisores, sendo estes obtidos de forma a restringir o comportamento da planta apenas o necessário para evitar que esta realize ações proibidas. Desta forma, pode-se verificar se uma dada especificação de comportamento pode ou não ser cumprida e, caso não possa, identificar a parte dessa especificação que pode ser implementada de forma minimamente restritiva. Um critério de aceitação pode então ser utilizado para determinar se, com a parte implementável da especificação, o sistema trabalha de maneira satisfatória.

O presente trabalho apresenta de forma resumida os principais modelos descritos na literatura, aprofundando-se em seguida na abordagem Ramadge-Wonham.

1.2 JUSTIFICATIVA PARA A ESCOLHA DA ABORDAGEM RAMADGE-WONHAM

Como se pode ver nas seções 2.5 e 2.6, onde as principais abordagens em uso são descritas e comparadas, a maioria dos modelos têm recursos formais apenas para a análise, mas não para a síntese de SED's. Ao se empregarem tais ferramentas em um projeto, é necessário executar os passos de síntese com base em outros métodos, muitas vezes fortemente dependentes da experiência do projetista, e então utilizar os modelos formais para análise e verificação, modificando o projeto até que o resultado da análise seja satisfatório.

O desejo de se ter ferramentas mais eficientes na síntese de controladores para SED's torna, naturalmente, mais interessantes as abordagens voltadas para este problema. A escolha da abordagem RW para tema da presente dissertação se justifica pela formulação e solução de uma série de problemas de síntese. Na pesquisa sobre os diversos modelos em uso (seção 2.5), foram ainda encontradas características semelhantes nas redes de Petri controladas (v. subseção 2.5.2). No entanto, o modelo Ramadge-Wonham, com suas variações e intensa atividade de pesquisa, é o mais desenvolvido dos dois.

A formalização do processo de especificação e síntese na abordagem escolhida justifica também esperanças de se conseguirem métodos de desenvolvimento de sistemas que sejam menos dependentes da experiência de quem os emprega e, portanto, acessíveis a um maior número de projetistas.

Além disso, considerou-se a experiência anterior do LCMI em Teoria de Controle Geométrica, também desenvolvida por W. M. Wonham e que permite fazer analogias com o caso discreto. Finalmente, a presença da Teoria de Controle e da Informática dentro do mesmo laboratório formam um ambiente ideal para se estudar uma teoria que, como descrito na seção 2.3, utiliza conhecimentos destas duas áreas.

1.3 OBJETIVOS E ORGANIZAÇÃO DO TRABALHO

A presente dissertação de mestrado foi proposta e realizada com os seguintes objetivos:

- ♦ adquirir conhecimentos sobre a teoria RW dentro do LCMI, da UFSC e do Brasil;
- ♦ gerar uma obra de referência para trabalhos futuros;
- ♦ estudar as diferentes técnicas para a obtenção de supervisores na teoria RW;
- ♦ investigar a aplicabilidade da teoria a problemas de Informática;
- ♦ contribuir, eventualmente, com resultados novos para a teoria.

O restante deste trabalho está organizado como segue: o capítulo 2 identifica as características dos sistemas a eventos discretos e suas diferenças fundamentais em relação aos sistemas contínuos. Apresenta ainda as áreas de estudo envolvidas, as razões para a inexistência de um modelo universal para SED's e os principais modelos existentes, comparando-os entre si.

O capítulo 3 apresenta alguns conceitos da Teoria de Linguagens e de Autômatos, necessários ao entendimento do modelo RW.

O capítulo 4 apresenta este modelo em sua forma básica, tal como encontrado na literatura. Trata da representação de SED's por geradores dotados de um mecanismo de controle, o que permite introduzir as noções de planta e de supervisor. Pode-se então apresentar a formulação de problemas no modelo RW e definir *L-fechamento* e *L-controlabilidade* de linguagens, conceitos fundamentais para estabelecer condições de existência de supervisores. Estas permitem, finalmente, formular e resolver um problema abstrato de síntese.

O capítulo 5 apresenta a principal contribuição teórica do presente trabalho, que consiste em introduzir a definição de *L-compatibilidade*. Em consequência, é possível estabelecer condições de existência de supervisores para problemas utilizando uma formulação diferente daquela encontrada na literatura. Um novo problema abstrato de síntese é então formulado, e os resultados decorrentes daquela definição são utilizados na sua solução. Um estudo da equivalência entre a formulação da literatura e a nova formulação introduzida permite demonstrar um teorema que estabelece a dualidade entre ambas.

O capítulo 6 estende os resultados obtidos no capítulo 5 ao modelo RW de controle supervisorio descentralizado, aplicando as contribuições feitas a um problema clássico de comunicação de dados, cuja solução é conhecida como *protocolo do bit alternante*. Esta aplicação não tem somente caráter ilustrativo, mas é também uma crítica do material contido em [RW89b] e [Rudi92] no que diz respeito ao emprego de linguagens geradas e marcadas na formulação e na solução de problemas. Como contribuição à teoria, um dos problemas de controle descentralizado é reformulado, com base nos resultados obtidos no capítulo 5. Faz-se também uma sugestão para melhorar a especificação do comportamento desejado sob supervisão. Finalmente, introdução da definição de *caminho opcional* no grafo de um gerador permite propor uma solução para o problema do protocolo do bit alternante num caso deixado sem solução na literatura pesquisada.

A conclusão analisa o conteúdo do trabalho em termos dos objetivos propostos e resume as contribuições teóricas feitas, apresentando ainda sugestões para trabalhos futuros.

CAPÍTULO 2

SISTEMAS A EVENTOS DISCRETOS

Este capítulo define uma classe de sistemas denominados *sistemas a eventos discretos* (SED's), apresentando suas características e comparando-os aos sistemas dinâmicos de variáveis contínuas. Discutem-se ainda as áreas de estudo envolvidas, as razões para a inexistência de um modelo universal para representar SED's e os principais modelos existentes, comparando-os entre si.

2.1 CARACTERÍSTICAS E DEFINIÇÃO

De um modo geral, um sistema é uma parte limitada do Universo que interage com o mundo externo através das fronteiras que o delimitam. Este conceito se aplica também aos sistemas tratados no presente trabalho, que apresentam ainda as características descritas a seguir.

Os sistemas de interesse percebem as ocorrências no mundo externo através da recepção de estímulos, denominados *eventos*. São exemplos de eventos o início e o término de uma tarefa (mas não sua execução), a chegada de um cliente a uma fila ou a recepção de uma mensagem em um sistema de comunicação. A ocorrência de um evento causa, em geral, uma mudança interna no sistema, a qual pode ou não se manifestar a um observador externo. Além disso, uma mudança pode ser causada pela ocorrência de um evento interno ao próprio sistema, tal como o término de uma atividade ou o fim de uma temporização. Em qualquer caso, essas mudanças se caracterizam por serem abruptas e instantâneas: ao perceber um evento, o sistema reage imediatamente, acomodando-se em tempo nulo numa nova situação, onde permanece até que ocorra um novo evento.

Desta forma, a simples passagem do tempo não é suficiente para garantir que o sistema evolua; para tanto, é necessário que ocorram eventos, sejam estes internos ou externos. Note ainda que a ocorrência desses eventos pode depender de fatores alheios ao sistema, de modo que este não tem, em geral, como prevê-los.

O que se disse acima permite apresentar agora a seguinte definição:

Def. 2.1: *Sistema a eventos discretos (SED)* é um sistema dinâmico que evolui de acordo com a ocorrência abrupta de eventos físicos, em intervalos de tempo em geral irregulares e desconhecidos.

Diz-se ainda que, entre a ocorrência de dois eventos consecutivos, o sistema permanece num determinado *estado*. A ocorrência de um evento causa então uma *transição* ou

mudança de estado no sistema, de forma que sua evolução no tempo pode ser representada pela trajetória percorrida no seu espaço de estados, conforme ilustrado na figura 2.1.

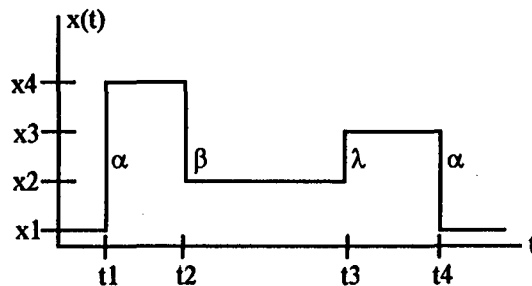


Fig. 2.1 - Trajetória típica de um sistema a eventos discretos

Nesta trajetória ocorrem eventos representados pelas letras α , β e λ . Vê-se que um mesmo evento pode ter efeitos diferentes, dependendo do estado em que ocorre. Por exemplo, se o sistema está no estado $x1$ e ocorre o evento α , o próximo estado será $x4$; se α ocorre em $x3$, volta-se para $x1$. A trajetória pode continuar indefinidamente, inclusive com a ocorrência de outros eventos, não representados na figura. Para todos os sistemas tratados, porém, assume-se que o número total de eventos diferentes que podem ocorrer é finito. O número de estados pode ser ilimitado no caso geral, embora a classe de sistemas com um número finito de estados seja um caso particular importante.

Costuma-se distinguir um dos estados do sistema dos demais, o qual recebe o nome de *estado inicial*. Este é o estado em que o sistema se encontra antes de ocorrer o primeiro evento. Na prática, em geral é possível forçar um sistema a voltar a esse estado, antes de iniciar sua utilização para um determinado fim, processo denominado de *reinicialização*.

Cabe distinguir agora duas grandes classes de sistemas.

Nos primeiros, denominados sistemas *determinísticos*, o estado representa a informação necessária e suficiente para se determinar a reação do sistema a um dado evento. A cada par (estado, evento) está associado um único estado, ao qual o sistema passa caso ocorra o evento em questão. Um sistema determinístico, quando reinicializado e submetido a uma mesma seqüência de eventos, descreverá sempre a mesma trajetória no espaço de estados.

Nos outros sistemas, denominados *não determinísticos*, o conhecimento do estado não é suficiente para determinar sua reação a um evento. A cada par (estado, evento) está associado um conjunto de estados, sendo que, na ocorrência do evento em questão, o sistema escolhe qual dos estados desse conjunto será o próximo, sem que se possa prevê-lo antecipadamente. Um sistema não determinístico, quando reinicializado e submetido a uma mesma seqüência de eventos pode, portanto, apresentar trajetórias diferentes no espaço de estados a cada ensaio realizado, dependendo das escolhas feitas internamente.

É importante salientar ainda que o termo "não determinístico" não tem um uso uniforme na literatura, sendo empregado também para fazer referência a uma outra característica que os SED's podem apresentar e descrita a seguir.

Conforme mencionado anteriormente, a ocorrência de eventos depende, em geral, de fatores externos ao sistema, o que não significa que este não tenha qualquer influência sobre aqueles. De fato, existem casos em que a escolha do evento que deve ocorrer cabe ao próprio sistema (por exemplo, se este determina, de forma aleatória, a seqüência em que um dado conjunto de tarefas deve ser realizado). Desta forma, mesmo mantidas constantes as condições externas, a seqüência de eventos produzida será, no caso geral, diferente para cada ensaio realizado.

Note, porém, que isto, por si só, não implica em que o sistema seja não determinístico: conforme a descrição anterior, um sistema é não determinístico quando, submetido a uma *mesma* seqüência de eventos, atinge estados diferentes em sua trajetória a cada ensaio; aqui, é a seqüência de estados visitados que não pode ser repetida.

Não obstante, o termo "não determinístico" tem sido usado para descrever ambos os tipos de sistemas. Para evitar confusão, diz-se neste trabalho que, na primeira acepção, os sistemas são *determinísticos (ou não) com relação a eventos* e que, na segunda, são *determinísticos (ou não) com relação a estados*.

Os sistemas de interesse deste trabalho são determinísticos com relação a estados, podendo ser determinísticos ou não com relação a eventos.

2.2 SISTEMAS DISCRETOS E SISTEMAS CONTÍNUOS

Os sistemas a eventos discretos, entendidos segundo a definição 2.1, contrastam com os sistemas dinâmicos a variáveis contínuas, descritos por equações diferenciais¹.

É instrutivo comparar a trajetória típica de um SED, apresentada na figura 2.1, com a de um sistema dinâmico de variáveis contínuas, apresentada na figura 2.2.

O espaço de estados de um SED é limitado a um conjunto enumerável, ao passo que é contínuo nos sistemas contínuos. Estes, em geral, mudam de estado a cada instante, tendo o seu comportamento descrito por uma função que relaciona o estado (variável dependente) ao tempo (variável independente). Já os sistemas a eventos discretos, conforme observado na seção 2.1, podem permanecer um tempo arbitrariamente longo em um mesmo estado, sendo que sua trajetória pode ser dada por uma lista de eventos na forma $\{\sigma_1, \sigma_2, \dots\}$, incluindo-se eventualmente os instantes de tempo em que tais eventos ocorrem, na forma

¹Como observado em [Ho89], embora os sistemas governados por equações a diferenças finitas sejam também discretos no tempo, eles estão mais próximos dos sistemas contínuos do que dos sistemas a eventos discretos aqui tratados, apesar desta semelhança.

$\{(\sigma_1, t_1), (\sigma_2, t_2), \dots\}$. A quantidade de informação necessária depende dos objetivos da aplicação. Em particular, como notado em [CR90], a presença ou não da informação temporal distingue as duas principais direções de pesquisa atualmente em atividade.

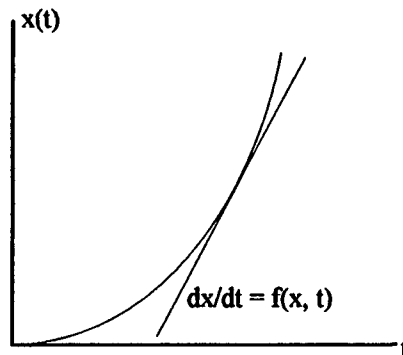


Fig. 2.2 - Trajetória típica de um sistema dinâmico com variáveis contínuas

Para ilustrar, a trajetória da figura 2.1 poderia aplicar-se a um robô numa célula de manufatura, cujos estados são o repouso, a movimentação até uma peça a ser trabalhada, a execução de uma tarefa sobre esta peça e a movimentação de volta à posição inicial e que transiciona entre estes estados de acordo com informações recebidas através de sensores, representadas pelos eventos. Já a trajetória da figura 2.2 poderia dizer respeito a um dos movimentos realizados pelo robô.

De um modo geral, a teoria de SED's trata da utilização e do controle das atividades que um sistema pode executar, sem se preocupar com sua realização física. Já a teoria de sistemas contínuos se preocupa com a realizabilidade física dessas atividades, sem que seja necessário especificar com que finalidade serão utilizadas.

2.3 ÁREAS RELACIONADAS À TEORIA DE SISTEMAS A EVENTOS DISCRETOS

O acima exposto permite ver a tarefa de especificar o comportamento de um sistema a eventos discretos como sendo a de estabelecer seqüências ordenadas de eventos que levem à realização de determinados objetivos. Com uma formulação tão abrangente, não é surpreendente que tenha havido esforços em mais de uma área para abordar o problema. Em [Ho89] a Teoria de Sistemas a Eventos Discretos é apresentada como pertencendo à área da Pesquisa Operacional, valendo-se ainda de resultados da Ciência da Computação (em particular da Inteligência Artificial e do Processamento de Linguagens), bem como da Teoria de Controle, como ilustra a figura 2.3.

Cabe citar ainda que os sistemas a eventos discretos não são uma invenção recente, embora a difusão deste nome entre os cientistas da área de controle o seja. A Pesquisa Operacional e a Ciência da Computação têm tratado tais sistemas com sucesso e, sob este

ponto de vista, a participação de cientistas da área de controle pode ser vista como tardia. Não obstante, suas contribuições são já substanciais e "há razões para crer que a Teoria de Controle possa ter um impacto significativo nesta área" ([CR90]).

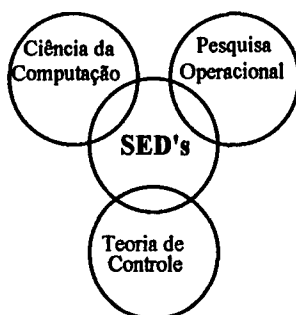


Fig. 2.3 - Áreas relacionadas à Teoria de Sistemas a Eventos Discretos

2.4 INEXISTÊNCIA DE UM MODELO UNIVERSAL PARA SED'S

Conforme mencionado no capítulo introdutório, foram desenvolvidos vários modelos para SED's sem que, no entanto, um deles tivesse conseguido se afirmar como universal. Em [Ho87] e [Ho89] o autor argumenta que isto se deve ao fato de nenhum dos modelos existentes considerar simultaneamente todos os seguintes aspectos:

- a) *a natureza descontínua dos eventos discretos*: a influência do caráter discreto dos eventos físicos sobre o modelo matemático adotado não pode ser desconsiderada, ainda que, em alguns casos, aproximações contínuas possam ser úteis;
- b) *a natureza contínua das medidas de desempenho*: estas medidas se estendem, em geral, sobre intervalos de tempo e são expressas em termos de valores médios para o sistema como um todo. Muitas dessas medidas são contínuas por convenção e definição, mas nem por isso deixam de ser aplicáveis aos SED's;
- c) *a importância da formulação probabilística*: conforme descrito na seção 2.1, os SED's podem ou não ser determinísticos, sendo portanto desejável que se possa tratar ambos os tipos de sistemas;
- d) *a necessidade de análise hierárquica*: em muitos casos, a complexidade dos SED's e a quantidade de informação necessária em diferentes etapas do seu planejamento e da sua operação, assim como a explosão combinatória do número de estados (v. abaixo), exigem uma representação do sistema que permita escolher o nível de detalhamento de acordo com os aspectos analisados, sem que tenham que estar presentes todos os estados e seqüências de eventos possíveis;

- e) *a presença de dinâmica*: um modelo comparável aos existentes para sistemas contínuos deve poder expressar dinâmica e comportamento transitório de SED's;
- f) *a viabilidade do uso de soluções computacionais*: o modelo de um SED é muitas vezes obtido a partir da composição de modelos menores, que representam partes do sistema. O número de estados resultante cresce de forma exponencial com o número de componentes, fenômeno conhecido como explosão combinatória. Em sistemas grandes, a enumeração de todos os estados possíveis se torna inviável, mesmo para os computadores mais potentes. Um modelo que não leve em conta este fato não é aplicável senão a exemplos didáticos e sistemas relativamente pequenos.

Podem-se citar ainda as restrições de tempo real e o alto grau de paralelismo encontrados em alguns sistemas, que impõem exigências adicionais ao modelo escolhido.

Embora não seja, em geral, necessário considerar todos estes itens quando se trabalha com um problema específico, um modelo que pretenda ser visto como universal deve oferecer esta possibilidade.

2.5 MODELOS PARA SED'S

Os principais modelos utilizados para sistemas a eventos discretos são os seguintes:

- Redes de Petri com e sem temporização;
- Redes de Petri Controladas com e sem temporização;
- Cadeias de Markov;
- Teoria das Filas;
- Processos Semi-Markovianos Generalizados (GSMP) e Simulação;
- Álgebra de Processos;
- Álgebra Max-Plus;
- Lógica Temporal e Lógica Temporal de Tempo Real;
- Lógica Temporal de Tempo Real Generalizada (GRTTL);
- Teoria de Linguagens e Autômatos (Ramadge-Wonham), incluindo modelos concentrados, distribuídos, hierárquicos e vetoriais, temporizados ou não.

Segue-se uma descrição simplificada de cada modelo, incluindo referências para o leitor que desejar aprofundar-se mais.

2.5.1 REDES DE PETRI

As redes de Petri são um modelo matemático com representação gráfica para sistemas a eventos discretos que se apresentem como concorrentes, assíncronos, distribuídos, paralelos, não determinísticos e/ou estocásticos, com ou sem temporização. Para atender a tal variedade de sistemas, existe um número considerável de tipos diferentes de redes. Está fora do escopo deste trabalho fazer um levantamento dos tipos existentes. O leitor interessado é convidado a consultar [Mura89] como fonte de referências. Para uma introdução à teoria com exemplos de aplicações ver [Pete81].

Em sua forma mais simples, uma rede de Petri é uma quádrupla $R = \langle P, T, \text{Pre}, \text{Post} \rangle$, onde:

- P é um conjunto finito de *lugares*, representados graficamente por círculos;
- T é um conjunto finito de *transições*, representadas graficamente por retângulos;
- $\text{Pre}: P \times T \rightarrow \mathbb{N}$ é a *função de entrada*, que relaciona lugares a transições; tais lugares são chamados *lugares de entrada* das transições. Esta relação é representada graficamente por um arco direcionado do lugar em P para a transição em T ; a função atribui ao arco um peso (daí o mapeamento no conjunto dos números naturais) que, quando diferente de 1, é escrito ao lado do arco;
- $\text{Post}: T \times P \rightarrow \mathbb{N}$ é a *função de saída*, que relaciona transições aos *lugares de saída*, de forma análoga a Pre.

Uma rede marcada é um par $RM = \langle R, M \rangle$, onde R é uma rede de Petri e

- $M: P \rightarrow \mathbb{N}$ é uma função, denominada *marcação* e associando a cada lugar de P um número de *fichas*,

determinando assim o estado em que o sistema representado se encontra. Graficamente, as fichas aparecem sob a forma de pontos desenhados no interior dos círculos que representam os lugares.

Num modelo de sistema, pode-se associar os lugares de entrada a recursos necessários, as transições a eventos e os lugares de saída a produtos finais, respectivamente. A presença de uma ficha em um lugar de entrada é interpretada como indicando a disponibilidade do recurso associado ao lugar. [Mura89] dá ainda outras interpretações úteis, tais como dados ou sinais de entrada e "buffers" para lugares de entrada e dados ou sinais de saída, "buffers" e conclusões para lugares de saída.

O comportamento do sistema modelado é descrito em termos das mudanças de estado que sofre. Para modelar esta evolução, a marcação da rede de Petri, correspondente ao estado do sistema, é modificada de acordo com a seguinte *regra de disparo de transições* ([Mura89]):

- i. uma transição t é dita *habilitada* se cada lugar de entrada p de t tem no mínimo $w(p, t)$ fichas, onde $w(p, t) = \text{Pre}(p, t)$ para $\text{Pre}(p, t) > 0$;
- ii. uma transição habilitada pode ou não disparar, representando a ocorrência ou não do evento a ela associado;
- iii. o disparo de uma transição habilitada remove $w(p, t)$ fichas de cada lugar de entrada p de t e acrescenta $w(t, p)$ fichas a cada lugar de saída p de t , onde $w(t, p) = \text{Post}(t, p)$ para $\text{Post}(t, p) > 0$.

A regra acima é ilustrada com o seguinte exemplo, reproduzido de [Mura89] e representando a reação química de formação da água a partir dos elementos que a constituem: $2\text{H}_2 + \text{O}_2 \rightarrow 2\text{H}_2\text{O}$. Duas fichas em cada lugar de entrada na figura 2.4a mostram que duas unidades de H_2 e de O_2 estão disponíveis, e que a transição t está habilitada. A marcação após o disparo de t é a mostrada na figura 2.4b, onde a transição não mais está habilitada.

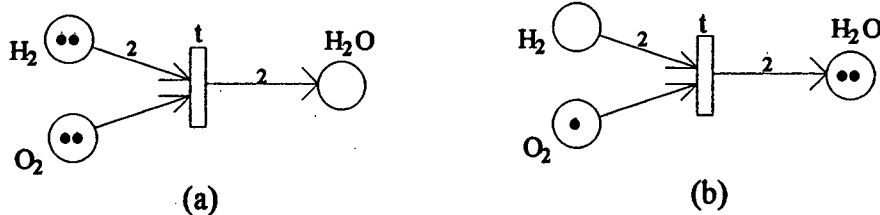


Fig. 2.4 - Disparo de transição numa rede de Petri

Como ferramenta gráfica, as redes de Petri são úteis pela representação visual que proporcionam, de forma semelhante aos fluxogramas ou aos diagramas em blocos. Como ferramenta matemática, permitem escrever equações que governam o comportamento do sistema representado. Cabe observar que não se faz menção a qualquer mecanismo de síntese, mas apenas a técnicas de análise de redes.

2.5.2 REDES DE PETRI CONTROLADAS

Tipicamente, a lógica de controle faz parte do modelo quando se emprega uma rede de Petri convencional. Desta forma, modificações na estratégia de controle requerem modificações na própria rede.

Em [HK90] e [KH91], Holloway e Krogh empregam as redes de Petri controladas, introduzidas pelo segundo autor em [Krog87], para separar a lógica de controle do modelo, através do uso de transições controladas. Representa-se o sistema livre de qualquer ação de controle por uma rede de Petri convencional que permanece inalterada durante as etapas seguintes do projeto. A esta rede são agregados os chamados lugares de controle, que permitem inibir as transições (eventos) às quais estão associados. Modificações na estratégia

de controle implicam apenas em habilitar ou inibir tais transições. Por exemplo, para representar o mecanismo de disparo da reação química mostrada no item anterior, seria necessário acrescentar mais um lugar à rede, modelando a ação externa que torna disponível a energia para iniciar a reação. No caso das redes controladas, o exemplo ficaria como ilustra a figura 2.5, onde o lugar de controle C1 é acrescentado sem modificar a rede original.

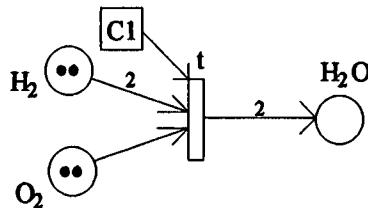


Fig. 2.5 - Rede de Petri controlada

Em [HK90] e [KH91], os autores mostram que é possível utilizar este modelo não apenas na análise do sistema, como no caso das redes convencionais, mas também para sintetizar a lógica de controle necessária. Em particular, as propriedades estruturais das redes de Petri permitem reduzir o espaço de estados, propiciando vantagens do ponto de vista computacional.

As redes de Petri controladas permitem explorar as propriedades da representação da rede para reduzir o problema da explosão combinatória e encontram paralelo dentro da abordagem de Ramadge e Wonham, nos chamados sistemas a eventos discretos vetoriais.

2.5.3 CADEIAS DE MARKOV

As cadeias de Markov são objeto de estudo da Teoria de Processos Markovianos, "o mais amplo e mais importante capítulo da Teoria de Processos Estocásticos" ([Çinl75]).

Dado um sistema caracterizado por um conjunto enumerável de estados S (que pode ser, por exemplo, o conjunto dos números inteiros ou um subconjunto deste) e observado nos instantes $n = 0, 1, 2, \dots$, seja X_n o estado do sistema no instante n . No caso mais simples, aplicável aos sistemas discutidos no presente trabalho, o comportamento do sistema a partir de um instante qualquer independe dos estados passados; apenas o estado atual influencia o comportamento futuro. Esta propriedade recebe o nome de *propriedade Markov* ([HPS72]), e sistemas apresentando tal propriedade são denominados *cadeias de Markov*. A referida propriedade é formalmente dada por:

$$P(X_{n+1} = x_{n+1} | X_0 = x_0, \dots, X_n = x_n) = P(X_{n+1} = x_{n+1} | X_n = x_n),$$

para todo $n \in \mathbb{N}$ e $x_0, \dots, x_n \in S$.

As probabilidades $P(X_{n+1} = y | X_n = x)$ são chamadas *probabilidades de transição*. Conhecidas estas probabilidades para um dado sistema, pode-se obter sua representação em forma de cadeia de Markov.

Este modelo é adequado à representação de sistemas quando se deseja considerar a probabilidade de ocorrência de eventos, como acontece na análise de desempenho. Um exemplo de utilização, incluindo comparações a outros modelos, é dado em [CH90]; para um aprofundamento na teoria v. por exemplo [HPS72] ou [Çin75].

2.5.4 TEORIA DAS FILAS

Sistemas de filas são um caso particular da classe dos *sistemas de fluxo* ([Klei75]), mais abrangentes do que aqueles. Num sistema de fluxo, objetos e/ou informações fluem, movem-se ou são transportados através de um ou mais *canais* de capacidade limitada, de um ponto a outro do sistema. Conseqüência deste limite é a formação de filas de espera pela disponibilidade de um ou mais canais.

Para se descrever um sistema deste tipo, é necessário conhecer o processo de chegada das solicitações de transporte e a maneira como estas solicitações são atendidas. Diferentes representações para estes dados dão origem a diversos tipos de filas, cada qual com características e aplicações próprias.

Os sistemas de filas encontram aplicação na análise de desempenho de sistemas a eventos discretos. O artigo [CH90], citado no item anterior, inclui um exemplo de modelagem de sistema a eventos discretos utilizando esta teoria; o leitor interessado em aprofundar-se no assunto é convidado a consultar [Klei75].

2.5.5 PROCESSOS SEMI-MARKOVIANOS GENERALIZADOS E SIMULAÇÃO

Como visto na seção 2.5.3, as cadeias de Markov podem ser utilizadas para descrever sistemas em que a reação a um dado evento depende unicamente do estado em que o sistema se encontra quando aquele ocorre. Uma das medidas importantes que se pode obter a partir de uma cadeia de Markov é o tempo médio de permanência do sistema em um dado estado. Por outro lado, a *propriedade Markov* exige que a probabilidade de ocorrência de qualquer evento seja constante no tempo. Além disso, o tempo de permanência num dado estado não pode ser considerado na definição da função de transição. Isto significa que, neste modelo, não é possível tornar a reação do sistema a um certo evento função do tempo de permanência em um dado estado (ou, equivalentemente, do instante em que o evento ocorre). Os processos semi-markovianos contêm a classe dos processos markovianos ([Klei75]) e permitem incluir esta informação.

A abordagem utilizando processos semi-markovianos generalizados procura formalizar programas e linguagens para simular sistemas a eventos discretos.

As partes discretas do sistema, tais como quantidade de recursos disponíveis ou de produtos, são representados por estados; às partes contínuas, como por exemplo o tempo restante para a conclusão de uma atividade, associam-se os tempos de permanência num determinado estado, ou seja, os intervalos entre eventos. Atribui-se a cada evento possível em cada estado um limite de tempo para que ocorra. Simula-se, então, em cada estado, a ocorrência do evento com o menor limite de tempo.

Para um exemplo de aplicação ver [CH90]; maiores detalhes sobre a teoria podem ser encontrados em [Glyn89].

2.5.6 ÁLGEBRA DE PROCESSOS

A um conjunto de eventos ocorridos em uma dada seqüência dá-se o nome de *processo*. Dentro desta acepção, os eventos gerados por um SED podem ser vistos como um processo.

Freqüentemente, o modelo para um dado SED pode ser obtido através da composição de modelos menores, que representam partes do sistema. Isto é útil porque, em geral, é mais fácil modelar o sistema passo a passo do que como um todo. Conseqüentemente, torna-se necessário compor os modelos das partes para chegar ao modelo global. Em se desejando utilizar a representação por processos, isto equivale a dizer que é necessário definir um conjunto de operações sobre processos que permitam representar sua composição.

O emprego destas operações permite escrever expressões algébricas envolvendo processos, bem como demonstrar identidades entre estas expressões. Um conjunto de operadores, axiomas e identidades entre tais expressões forma uma *álgebra de processos*.

Várias álgebras de processos têm sido desenvolvidas para o tratamento de SED's, inspiradas sobretudo nos trabalhos de Milner ([Miln80]) e Hoare ([Hoar85]), indicados para consulta sobre a teoria. Uma destas abordagens, ilustrada com um exemplo em [CH90], é a dos *processos recursivos finitos*, definidos em [IV88], onde os autores desenvolvem um modelo para SED's utilizando a teoria de Hoare.

Cabe ainda citar o trabalho de Heymann em [Heym90], onde o autor define um novo formalismo para tratar processos concorrentes e o compara à abordagem Ramadge-Wonham. O mesmo artigo é ainda recomendado como fonte de referências para outras abordagens algébricas desenvolvidas dentro da Ciência da Computação.

2.5.7 ÁLGEBRA MAX-PLUS

Este formalismo foi proposto em [CDQV85], e se baseia no uso de um dióide, estrutura algébrica constituída de um conjunto e de duas operações sobre este definidas. No caso da álgebra max-plus, o conjunto utilizado é $\mathbb{N} \cup \{-\infty\}$, onde \mathbb{N} é o conjunto dos números naturais e as operações são a adição (\oplus) e a multiplicação (\otimes), definidas como segue:

$$a \oplus b = \max(a, b)$$

$$a \otimes b = a + b.$$

As propriedades algébricas desta estrutura mostram-se úteis na descrição de sistemas a eventos discretos, permitindo uma representação que lembra em muito a da Teoria de Controle para Sistemas Dinâmicos com Variáveis Contínuas. Em particular, o comportamento de SED's que envolvem um conjunto de atividades repetitivas pode ser caracterizado pela solução de uma equação matricial escrita na álgebra proposta. Detalhes da teoria e um exemplo detalhado encontram-se no artigo acima citado; um resumo dos principais resultados, também acompanhado de exemplo, é dado em [CH90].

2.5.8 LÓGICA TEMPORAL

Como toda lógica, este formalismo compreende sistemas de axiomas e regras com os quais se pode formular e tentar provar proposições. Além dos operadores "não" (símbolo \sim), "ou" (\vee) e "e" (\wedge), definidos na lógica clássica, a lógica temporal utiliza operadores tais como necessidade (\Box) e possibilidade (\Diamond), oriundos da lógica modal. A interpretação destes conceitos de forma que sejam úteis à representação de fenômenos levando em conta a passagem do tempo dá origem à lógica temporal.

Cabe notar aqui que, apesar do nome, a lógica temporal não permite considerar instantes de tempo determinados (ou seja, o tempo real), mas apenas noções como "antes", "depois", "doravante" ou "eventualmente"².

Dentro deste contexto, a aplicação do conceito de necessidade a uma proposição p qualquer, representada por $\Box p$, forma uma asserção interpretada como "p é verdadeira agora e em todos os instantes subseqüentes"; o conceito de possibilidade é substituído pelo de eventualidade, interpretando-se $\Diamond p$ como "é necessário que p seja verdadeira em algum instante subseqüente". Pode-se utilizar ainda o operador "no instante seguinte", representado por O , interpretando-se $O p$ como "a proposição p será verdadeira no próximo instante". A figura 2.6 ilustra a semântica destes operadores, em termos da árvore de transições característica de um SED. Nesta figura, os nós representam o estado do sistema a cada instante; as asserções $\Box p$, $\Diamond p$ e $O p$ se aplicam a partir do nó raiz. Um nó circundado significa que a proposição p é verdadeira no instante correspondente.

A lógica temporal pode ser utilizada de duas maneiras distintas no tratamento de SED's: no primeiro caso, o sistema, condicionado à ação de um controlador que tem por

²É importante explicitar aqui a semântica atribuída ao termo "eventualmente". Ao passo que, na língua portuguesa, ele equivale a "possivelmente", na língua inglesa o termo mais próximo é "certamente". É este último o sentido empregado para o termo dentro da lógica temporal. Por exemplo, ao se afirmar que "eventualmente a proposição p se tornará verdadeira", o que se quer dizer é que ela certamente o será, embora não se saiba quando.

objetivo assegurar seu funcionamento de acordo com uma especificação, é modelado empregando-se uma técnica que não a lógica temporal, como por exemplo um autômato ou uma rede de Petri; a lógica temporal é utilizada para formular asserções que refletem o funcionamento desejado do sistema. É necessário que exista um mapeamento entre as duas representações, através do qual se procura então verificar que as asserções não são violadas pelo modelo. Conseguindo isto, assume-se que este é correto. Esta abordagem é denominada *dual*, por empregar dois formalismos distintos. Um exemplo deste tipo de abordagem, utilizando máquinas de estados finitos na modelagem do sistema, é a desenvolvida por Knight e Passino em [KP90].

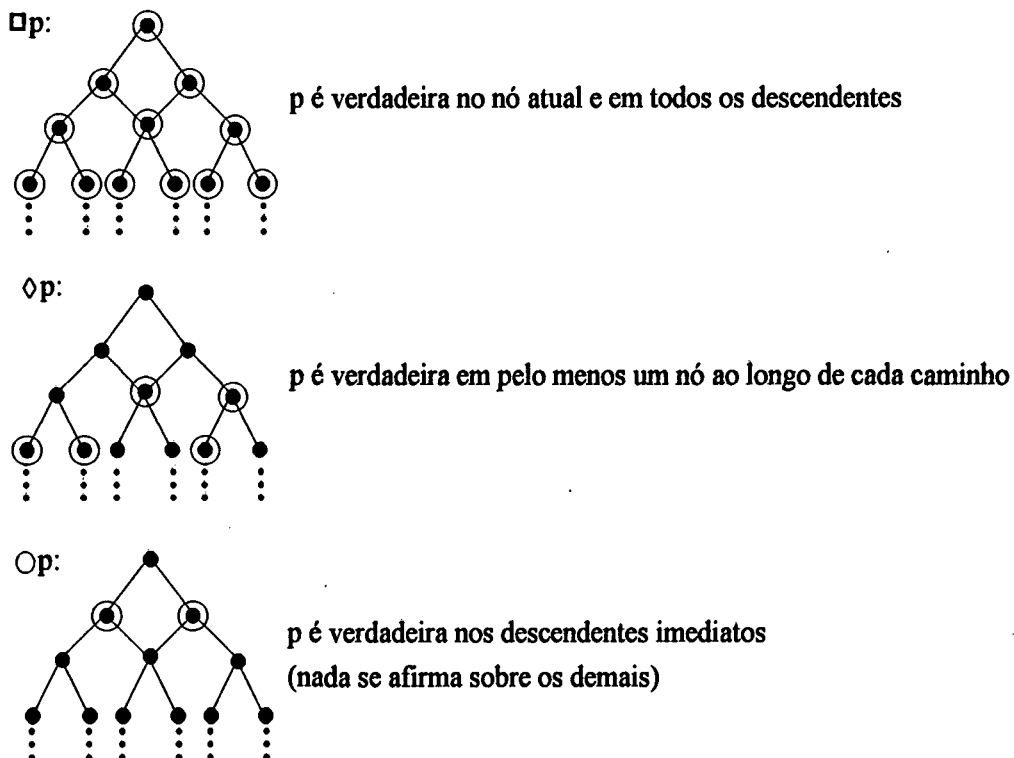


Fig. 2.6 - Semântica dos operadores da lógica temporal

Por outro lado, é possível empregar a própria lógica temporal para a modelagem do sistema, de maneira que tanto o modelo quanto as asserções que garantem sua correção sejam dadas pelo mesmo formalismo. O que se faz neste caso é tentar provar as asserções como se fossem teoremas, a partir da representação do modelo. Este é considerado correto se todas as asserções puderem ser provadas. Esta abordagem é denominada *primal*, por se utilizar um único formalismo. Um exemplo de sistema primal utilizando autômatos é o apresentado por Thistle e Wonham em [TW86]. Lin e Ionescu estenderam esta abordagem em [LI90] para tratar sistemas não determinísticos, também de forma primal.

A lógica temporal pode ser dotada de meios que possibilitem explicitar os instantes de tempo em que os eventos ocorrem. Tem-se então a chamada *lógica temporal de tempo real*.

Uma abordagem desta classe é a de Ostroff e Wonham, apresentada em [OW90], onde se introduzem limites superiores e inferiores de tempo para a ocorrência de eventos.

Além das referências específicas para cada uma das abordagens mencionadas acima, recomendam-se [HC77], [Krög87] e [Ostr89] para consulta sobre lógica modal, lógica temporal e lógica temporal de tempo real, respectivamente.

2.5.9 LÓGICA TEMPORAL DE TEMPO REAL GENERALIZADA (GRTTL)

A lógica temporal de tempo real generalizada foi introduzida em [Silv92] e utiliza elementos de outras lógicas de tempo real, mencionadas na subseção 2.5.8. Da abordagem Thistle-Wonham, herda o arcabouço axiomático e o sistema de prova; da abordagem Ostroff-Wonham, tomam-se os conceitos de limites inferiores e superiores do tempo de permanência num estado, os elementos do sistema de prova referentes às restrições de tempo real e a introdução dos quantificadores existencial e universal. A abordagem Lin-Ionescu contribui com os operadores "certeza" e "possibilidade", fornecendo à GRTTL ferramentas para tratar SED's não determinísticos. "O formalismo emergente da fusão destes elementos traz características interessantes na análise de SED's que apresentam não determinismo e restrições temporais na sua dinâmica" ([Silv92]).

2.5.10 TEORIA DE AUTÔMATOS E DE LINGUAGENS

Os autômatos (ou máquinas de estados finitos) e as linguagens a eles associadas têm sido extensamente empregados na modelagem de sistemas a eventos discretos, existindo ampla literatura e intensa atividade de pesquisa nesta área. Exemplos de trabalhos são [Rama89], [This92] e o próprio modelo proposto por Ramadge e Wonham, objeto do presente trabalho.

Este último, denominado "modelo RW", difere das demais abordagens apresentadas ao longo do capítulo pelos resultados obtidos sobre a existência de um supervisor para impor à planta um dado padrão de comportamento minimamente restritivo (único) e os meios para sintetizá-los.

Os primeiros estudos que levaram a este modelo foram feitos no início da década de 80, como objeto da tese de doutoramento de Peter G. Ramadge, na Universidade de Toronto, Canadá, sob a orientação do professor W. Murray Wonham. Os primeiros resultados foram publicados em 1983, num documento do Systems Control Group da referida universidade, aparecendo mais tarde como [RW87]. O artigo [WR87] complementa o anterior, discutindo a implementação do processo de síntese. A partir deste ponto, vários autores passaram a contribuir para o desenvolvimento da teoria, que hoje compreende modelos considerando observabilidade parcial de eventos ([LW88a]), [CDFV88], [LW90], [Rudi92]), controle descentralizado ([LW88b], [LW90], [RW89b], [CDFV88], [Rudi92]), controle hierárquico ([ZW90], [WW92]), SED's vetoriais ([LW91]) e sistemas temporizados ([BW92], [BWB92]). A síntese é ainda discutida em [Rudi88], [LVW88], [BGK+90] e [KGM91].

Não é objetivo deste trabalho abordar cada uma destas variantes em profundidade. Os capítulos 4 e 5 tratam em detalhe o modelo básico, e o capítulo 6 trata o controle descentralizado. O leitor interessado nas demais variantes é convidado a consultar as referências citadas acima.

2.6 COMPARAÇÃO E CONCLUSÕES

O material apresentado permite compreender as razões para a inexistência de um modelo único para SED's, dadas as características dos modelos existentes. Ho classifica os modelos para SED's em temporizados e não temporizados, conforme permitam ou não considerar os instantes de tempo em que ocorrem os eventos, e ainda em algébricos, lógicos e de análise de desempenho, conforme sejam orientados ao uso de equações, ao uso de asserções lógicas ou à formulação probabilística, respectivamente. A tabela 2.1, baseada em [Ho89] e modificada para incluir modelos mais recentes, segue este critério.

MODELOS	Temporizados	Não temporizados
Lógicos	<ul style="list-style-type: none"> • Ramadge-Wonham temporizado • Lógica Temporal de Tempo Real • GRTTL • Redes de Petri temporizadas • Redes de Petri Controladas temporizadas 	<ul style="list-style-type: none"> • Ramadge-Wonham • Lógica Temporal • Redes de Petri • Redes de Petri Controladas
Algébricos	<ul style="list-style-type: none"> • Álgebra Max-Plus 	<ul style="list-style-type: none"> • Álgebra de Processos
de Análise de Desempenho	<ul style="list-style-type: none"> • Cadeias de Markov • Teoria das Filas • GSMP / Simulação • Redes de Petri com temporização estocástica 	

Tab. 2.1 - Modelos para Sistemas a Eventos Discretos

A tabela mostra as duas linhas de pesquisa (com e sem temporização) citadas no capítulo introdutório, agrupando ainda as abordagens com características semelhantes. Em termos das idéias expostas na seção 2.4, pode-se dizer que todos os modelos consideram a natureza discreta dos SED's; características de análise de desempenho aparecem apenas na álgebra max-plus e nos modelos estocásticos; estes últimos são também os únicos em que se pode considerar probabilidades de ocorrência de eventos. A decomposição hierárquica encontra melhor suporte no modelo RW; a dinâmica, em termos comparáveis aos sistemas contínuos, aparece na álgebra max-plus e em algumas extensões do modelo RW sobre estabilidade de sistemas a eventos discretos ([BH90], [KGM93]), não discutidas aqui.

Finalmente, as dificuldades computacionais atingem, de uma forma ou de outra, todas as abordagens e parece pouco provável que se consiga algum dia um modelo que não seja limitado neste aspecto.

Os modelos apresentados servem sobretudo à análise de SED's. São exceções as redes de Petri controladas e o modelo Ramadge-Wonham, dotados de procedimentos de síntese, o que lhes dá vantagens indiscutíveis em se tratando de projeto de sistemas.

Nenhum dos modelos tem a pretensão de servir como paradigma em sua forma atual. A intensa atividade de pesquisa na área certamente fará com que alguns deles passem a incluir características hoje existentes em outros, e mesmo características novas. Ao mesmo tempo, toda a teoria existente ainda está longe de ter o prestígio da teoria correspondente para sistemas contínuos. As aplicações e exemplos encontrados na literatura são, em sua maioria, didáticas ou de pequeno porte. O estado atual da Teoria de Controle de Sistemas a Eventos Discretos e as exigências colocadas por aplicações de grande porte permitem antever ainda um longo caminho a ser percorrido no seu desenvolvimento.

CAPÍTULO 3

TEORIA DE LINGUAGENS E DE AUTÔMATOS

Este capítulo apresenta alguns conceitos da Teoria de Linguagens e de Autômatos, necessários ao entendimento do modelo RW. Tais conceitos são, na maioria, independentes da Teoria de Controle de Sistemas a Eventos Discretos. Não obstante, sua apresentação é feita de modo a evidenciar as propriedades que os tornam úteis ao estudo desses sistemas.

A primeira seção traz conceitos básicos da Teoria de Linguagens, mostrando como um SED pode ser representado por uma linguagem. As duas seções restantes apresentam autômatos e geradores, bem como as linguagens a eles associadas e as propriedades que tornam os geradores adequados à representação de SED's.

Cumpramos esclarecer ainda que autômatos e geradores não admitem a idéia de simultaneidade, de modo que não se pode levar em conta a ocorrência simultânea de dois ou mais eventos distintos. Assume-se portanto que os sistemas aqui tratados apresentam evolução seqüencial. Além disso, a representação por autômatos e geradores não considera os instantes de tempo em que os eventos ocorrem, mas apenas a ordem em que acontecem.

O material apresentado aqui tem origem em [HU79], [CL89] e [RW87]. Uma referência bastante acessível sobre Teoria de Conjuntos é [Moor66].

3.1 SED'S E TEORIA DE LINGUAGENS

Considere novamente a trajetória típica de um SED mostrada na figura 2.1. A representação do conjunto de eventos gerados pelo sistema por um conjunto de símbolos torna interessante a seguinte definição:

Def. 3.1: *Alfabeto* é um conjunto (não vazio) de símbolos.

Alfabetos são representados por letras gregas maiúsculas, em geral pela letra Σ . Associa-se aqui a noção de alfabeto ao conjunto de eventos, e permite-se o uso de expressões tais como "alfabeto de eventos", ou "um evento do alfabeto Σ ".

Ex. 3.1: O alfabeto associado ao sistema da figura 2.1 (supondo-se que não haja eventos adicionais, não representados na figura) é o conjunto $\Sigma = \{\alpha, \beta, \lambda\}$. ♦

De acordo com a seção 2.2, numa descrição não temporizada do sistema, a trajetória representando a evolução no tempo de um SED pode ser registrada simplesmente anotando-se a seqüência de eventos - ou seja, de símbolos - na forma $\alpha\beta\lambda\alpha$. De forma análoga à escrita habitual, a justaposição de símbolos de um alfabeto é denominada *palavra*:

Def. 3.2: *Palavra sobre o alfabeto Σ* é qualquer justaposição de um número finito de símbolos (com ou sem repetição) de Σ , na forma $\sigma_1\sigma_2\dots\sigma_k$, com $\sigma_i \in \Sigma$ para $i = 1, 2, \dots, k$.

Diz-se, portanto, que um SED produz - ou *gera* - palavras de comprimento crescente, à medida que evolui. A noção de comprimento de uma palavra é formalizada como segue:

Def. 3.3: O *comprimento* de uma palavra s , representado por $|s|$, é igual ao número de símbolos que a compõem.

A *palavra nula*, definida abaixo, é útil para representar a situação em que o sistema ainda não iniciou a geração de símbolos:

Def. 3.4: A *palavra nula* é a (única) palavra de comprimento nulo.

Representa-se a palavra nula por ϵ . Cabe notar aqui que $\epsilon \notin \Sigma$, por ser ϵ uma palavra e não um símbolo de Σ .

Conjuntos de palavras são estruturas fundamentais na Teoria de Linguagens. Um exemplo de conjunto é o definido a seguir:

Def. 3.5: Dado $k \in \mathbb{N}$, denota-se por Σ^k o conjunto de todas as palavras sobre Σ cujo comprimento é igual a k .

Ex. 3.2: Dado $\Sigma = \{\alpha, \beta\}$,

$$\Sigma^0 = \{\epsilon\}$$

$$\Sigma^1 = \{\alpha, \beta\}$$

$$\Sigma^2 = \{\alpha\alpha, \alpha\beta, \beta\alpha, \beta\beta\}$$

$$\Sigma^3 = \{\alpha\alpha\alpha, \alpha\alpha\beta, \alpha\beta\alpha, \alpha\beta\beta, \beta\alpha\alpha, \beta\alpha\beta, \beta\beta\alpha, \beta\beta\beta\}. \quad \blacklozenge$$

Note que não há inconsistência em permitir que $\epsilon \in \Sigma^0$, uma vez que Σ^0 é um conjunto de palavras e ϵ é uma palavra.

Def. 3.6: Dado um alfabeto Σ , definem-se os conjuntos Σ^+ e Σ^* por:

$$\Sigma^+ = \bigcup_{k=1}^{\infty} \Sigma^k = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \dots$$

$$\Sigma^* = \bigcup_{k=0}^{\infty} \Sigma^k = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \dots$$

Interpreta-se Σ^+ como o conjunto de todas as palavras que podem ser formadas com os símbolos do alfabeto Σ . Σ^* difere de Σ^+ apenas por incluir a palavra nula: $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$.

Um conjunto de palavras formadas com símbolos de um alfabeto é denominado *linguagem*:

Def. 3.7: Dado um alfabeto Σ , L é uma *linguagem sobre Σ* sse $L \subseteq \Sigma^*$.

Esta definição implica que tanto \emptyset quanto Σ^* são linguagens. Note que a linguagem vazia, $\emptyset = \{\}$, é diferente da linguagem formada apenas pela palavra nula, $\Sigma^0 = \{\epsilon\}$.

O conjunto de todas as palavras que um sistema de alfabeto Σ é capaz de gerar é, portanto, uma linguagem $L \subseteq \Sigma^*$. Em geral, um sistema não gera qualquer seqüência de eventos de Σ^* que se possa imaginar: escolhendo símbolos de Σ arbitrariamente, é possível construir palavras que não correspondam a seqüências de eventos fisicamente possíveis no sistema. Isto significa que, em geral, a linguagem gerada pelo sistema é um subconjunto próprio de Σ^* .

A seguinte definição é útil para compor palavras:

Def. 3.8: Dadas duas palavras s_1 e s_2 sobre um alfabeto Σ , com $s_1 = \sigma_1\sigma_2\dots\sigma_k$ e $s_2 = \sigma_{k+1}\sigma_{k+2}\dots\sigma_n$, a *concatenação* de s_1 e s_2 , denotada por s_1s_2 , é dada por $\sigma_1\sigma_2\dots\sigma_k\sigma_{k+1}\dots\sigma_n$.

Quando se deseja fazer referência a uma parte inicial de uma palavra, de comprimento arbitrário, utiliza-se a noção de *prefixo*:

Def. 3.9: *Prefixo* de uma palavra w sobre um alfabeto Σ é qualquer palavra $u \in \Sigma^*$ que possa ser completada com outra palavra $v \in \Sigma^*$ para formar a palavra w .

Ex. 3.3: Dado $\Sigma = \{\alpha, \lambda, \mu\}$, $u = \alpha\beta$ é um prefixo de $w = \alpha\beta\lambda\alpha$, porque $\exists v \in \Sigma^*$ tal que $uv = w$ (no caso, $v = \lambda\alpha$). ♦

Def. 3.10: Dada uma palavra s , denota-se por $\text{Pre}(s)$ o conjunto de todos os prefixos de s (inclusive a palavra nula ϵ).

Ex. 3.4: No sistema da figura 2.1, $\text{Pre}(\alpha\beta\lambda\alpha) = \{\epsilon, \alpha, \alpha\beta, \alpha\beta\lambda, \alpha\beta\lambda\alpha\}$. Note que $\alpha\beta\lambda\alpha$ é prefixo de si mesma, pois $\alpha\beta\lambda\alpha\epsilon = \alpha\beta\lambda\alpha$, satisfazendo a condição da definição 3.9. ♦

Dada uma linguagem $L \subseteq \Sigma^*$, existe uma linguagem a ela associada, formada pelas palavras de L e por todos os seus prefixos. Este fato é motivação para as seguintes definições:

Def. 3.11: O *prefixo-fechamento*, ou simplesmente *fechamento* de L , é dado por:

$$\bar{L} = \{u: \exists v \in \Sigma^* \wedge uv \in L\}.$$

Conseqüência desta definição é que:

$$L \subseteq \bar{L}. \quad (3.1).$$

Def. 3.12: Uma linguagem L é *prefixo-fechada* ou simplesmente *fechada* sse $L = \bar{L}$.

Conseqüentemente, se L é prefixo-fechada, então, para cada palavra r pertencente a L , vale $\text{Pre}(r) \subseteq L$.

Considerando-se a evolução seqüencial dos SED's, pode-se afirmar que, se um dado sistema produziu uma palavra qualquer r , então produziu anteriormente todas as palavras do conjunto $\text{Pre}(r)$. Em outras palavras, a linguagem gerada por um SED incluirá, para cada palavra, também todos os seus prefixos.

Em vista do apresentado, tem-se a seguinte proposição:

Prop. 3.1: O comportamento lógico de qualquer sistema a eventos discretos em que não ocorram eventos simultâneos pode ser representado por uma linguagem prefixo-fechada.

Dem.: Pelas definições 3.1 a 3.12. ♦

Def. 3.13: A linguagem prefixo-fechada que representa o comportamento lógico de um sistema a eventos discretos é denominada *linguagem gerada* do sistema.

Após partir do estado inicial e percorrer uma determinada trajetória em seu espaço de estados, um SED acabará, via de regra, completando uma ou mais tarefas. As seqüências de eventos (palavras) que levam a tarefas completadas formam também uma linguagem. Note que linguagens deste tipo não são necessariamente prefixo-fechadas.

Def. 3.14: A linguagem que representa o conjunto de tarefas que um sistema a eventos discretos é capaz de executar é denominada *linguagem marcada* do sistema.

Cabe observar que, para gerar qualquer palavra desta linguagem, o SED em questão tem que gerar todos os seus prefixos. Em outras palavras, um SED que produza as palavras contidas numa linguagem (não necessariamente fechada) L_m também produzirá as palavras contidas em \bar{L}_m . Se L é a linguagem gerada de um sistema e L_m sua linguagem marcada, então

$$L_m \subseteq \bar{L}_m \subseteq L = \bar{L} \quad (3.2).$$

Existe uma classe importante de linguagens, que podem ser expressas pelas assim chamadas *expressões regulares* ([HU79], [CL89]). Tais linguagens são denominadas *linguagens regulares*. As expressões regulares são seqüências de símbolos obtidas pela aplicação repetitiva de um conjunto de regras de formação. Estas regras não serão discutidas em detalhe aqui; são dadas apenas as informações seguintes, suficientes para possibilitar a compreensão das expressões utilizadas ao longo do texto:

- utilizam-se as notações σ^n e s^n para representar uma repetição do evento σ e da palavra s por n vezes, respectivamente;

- utilizam-se as notações σ^* e s^* para representar uma repetição do evento σ e da palavra s um número arbitrário de vezes (inclusive zero), respectivamente;
- o símbolo $+$ é empregado como operador "ou", indicando uma opção entre duas ou mais possibilidades.

Esta notação permite escrever representações finitas para linguagens formadas por um número infinito de palavras.

Ex. 3.5: Um SED opera de modo que dois eventos, α e β , ocorram alternadamente, sendo α o primeiro deles. A linguagem (prefixo-fechada) gerada por este sistema é $L = \{\epsilon, \alpha, \alpha\beta, \alpha\beta\alpha, \alpha\beta\alpha\beta, \dots\} = \{(\alpha\beta)^*(\alpha + \epsilon)\}$, que se lê " $(\alpha\beta)$ pode ocorrer um número arbitrário de vezes (inclusive zero); após isso, pode ocorrer mais um evento α , ou então a palavra nula (ou seja, nada)". Note que L é um subconjunto próprio da linguagem $\Sigma^* = \{\alpha^*\beta^*\}^*$, a qual inclui por exemplo as palavras $\beta\alpha\beta\alpha$ e $\alpha\alpha\alpha$, que não podem ser geradas.

Além disso, a seqüência $\alpha\beta$ pode representar uma tarefa completada no sistema, de modo que as palavras da linguagem marcada $L_m = \{\epsilon, \alpha\beta, \alpha\beta\alpha\beta, \dots\} = \{(\alpha\beta)^*\}$ caracterizam o término de zero ou mais tarefas. ♦

A classe de linguagens regulares pode ainda ser representada de forma interessante por *autômatos determinísticos finitos* (ou simplesmente autômatos ou ainda máquinas de estados finitos). É este o assunto da próxima seção.

3.2 AUTÔMATOS

Autômatos são modelos matemáticos de máquinas que reconhecem um conjunto de palavras sobre um dado alfabeto. A figura 3.1 ilustra o conceito de autômato em termos de uma entidade de controle que lê seqüencialmente uma fita de símbolos deste alfabeto e acende uma lâmpada sempre que for lida uma palavra do conjunto de palavras reconhecidas.

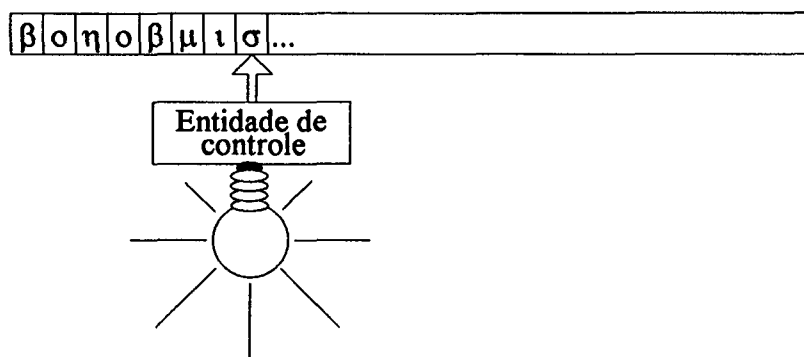


Fig. 3.1 - Modelo ilustrativo de um autômato

Internamente, a entidade de controle tem uma variável que representa o estado do autômato. Cada símbolo lido causa uma atualização desta variável, conforme determinado por uma função de transição, a qual associa um e somente um novo estado a cada par (evento, estado).¹ O conjunto de valores que esta variável pode assumir é o *conjunto de estados* do autômato, sendo o estado em que este se encontra antes de ler o primeiro símbolo da fita denominado *estado inicial*. Alguns dos estados são *estados marcados*, e o autômato reconhece exatamente as palavras da fita que, quando processadas, o levam a um estado marcado.

Formalmente, um autômato é definido como segue:

Def. 3.15: Um *autômato determinístico finito* ou simplesmente *autômato* é uma quintupla $A = \langle \Sigma, Q, \delta, q_0, Q_m \rangle$, onde:

Σ é um alfabeto;

Q é um conjunto finito de estados;

$\delta: \Sigma \times Q \rightarrow Q$ é uma função de transição de estados;

$q_0 \in Q$ é o estado inicial e

$Q_m \subseteq Q$ é o conjunto de estados marcados.

Entende-se aqui que a função de transição é completa, ou seja, definida para todo par $(\sigma, q) \in \Sigma \times Q$.

Definir um autômato significa definir cada um dos elementos da quintupla apresentada na definição 3.15. O alfabeto, os conjuntos de estados e o estado inicial são simples listas de símbolos; já a função de transição pode ser dada em forma de uma lista relacionando os pares de $\Sigma \times Q$ a elementos de Q (v. o exemplo 3.6), ou pelos *diagramas de transição de estados*, ou ainda por uma *tabela de transição de estados*.

Um diagrama de transição de estados é um grafo² orientado, cujos vértices representam os estados e cujos arcos representam a função de transição:

Def. 3.16: Dado o autômato $A = \langle \Sigma, Q, \delta, q_0, Q_m \rangle$, o *diagrama de transição de estados* associado a A é o grafo orientado $G = \{V, W\}$, onde:

$V = Q$ e

$W = \{ \langle p, q, \sigma \rangle : p, q \in Q \wedge \sigma \in \Sigma \wedge \delta(\sigma, p) = q \}$.

Além disso, estados marcados são caracterizados por vértices desenhados com linhas duplas e o estado inicial é identificado por uma seta. Com isso, obtém-se uma representação gráfica completa para o autômato.

¹Este fato caracteriza os autômatos assim definidos como determinísticos.

²Para detalhes sobre Teoria de Grafos v. [Chri75].

A *tabela de transição de estados* é uma matriz cujas linhas são enumeradas pelo conjunto de estados do autômato e cujas colunas são enumeradas pelo seu conjunto de transições. O elemento t_{ij} da matriz é o estado para o qual o autômato irá se, no estado i , ocorrer o evento j .

Ex. 3.6: Seja um autômato com

$$\Sigma = \{\alpha, \beta\},$$

$$Q = \{0, 1, 2\},$$

$$\delta(\alpha, 0) = 2, \delta(\beta, 0) = 1, \delta(\alpha, 1) = 2, \delta(\beta, 1) = 0, \delta(\alpha, 2) = 2, \delta(\beta, 2) = 1,$$

$$q_0 = 0 \text{ e}$$

$$Q_m = \{2\}.$$

De acordo com a definição 3.16, o grafo correspondente é $G = \{V, W\}$, com

$$V = \{0, 1, 2\} \text{ e}$$

$$W = \{\langle 0, 2, \alpha \rangle, \langle 0, 1, \beta \rangle, \langle 1, 2, \alpha \rangle, \langle 1, 0, \beta \rangle, \langle 2, 2, \alpha \rangle, \langle 2, 1, \beta \rangle\}.$$

O diagrama de transição correspondente a este autômato é dado na figura 3.2; a tabela de transição de estados é a da tabela 3.1. ♦

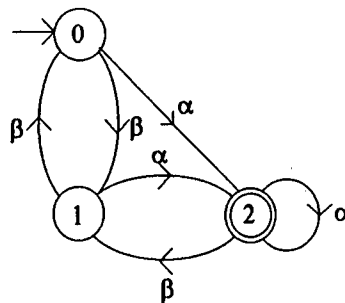


Fig. 3.2 - Diagrama de transição representando o autômato do exemplo 3.6

	α	β
0	2	1
1	2	0
2	2	1

Tab. 3.1 - Tabela de transição para o autômato do exemplo 3.6

Conhecidas a função de transição e o estado atual do autômato, é possível determinar seu estado após processar um dado símbolo. Em se desejando processar uma cadeia de símbolos (isto é, uma palavra), pode-se simplesmente repetir este procedimento para cada um

dos símbolos que a compõem. No entanto, é conveniente estender a definição da função de transição para processar palavras:

Def. 3.17: Dado um autômato $A = \langle \Sigma, Q, \delta, q_0, Q_m \rangle$, a *função de transição estendida*, denotada por $\bar{\delta}$, é a função $\bar{\delta}: \Sigma^* \times Q \rightarrow Q$ tal que:

$$\bar{\delta}(\epsilon, q) = q \text{ e}$$

$$\bar{\delta}(s\sigma, q) = \delta(\sigma, \bar{\delta}(s, q)), \text{ para } q \in Q \text{ e } s \in \Sigma^*.$$

No restante deste trabalho dispensa-se a notação $\bar{\delta}$, representando a função de transição estendida simplesmente por δ .

Definida esta função, pode-se determinar qual será o estado do autômato após processar uma palavra sobre seu alfabeto, a partir de um dado estado. Dependendo da palavra escolhida, o estado final a que se chega poderá ou não pertencer ao conjunto de estados marcados. Isto significa que existem dois conjuntos de palavras com respeito a um dado autômato: as que levam do estado inicial a um estado marcado e as que levam do estado inicial a um estado não marcado. Isto é motivo para a seguinte definição:

Def. 3.18: Dado um autômato $A = \langle \Sigma, Q, \delta, q_0, Q_m \rangle$, diz-se que A *reconhece* a palavra $s \in \Sigma^*$ sse $\delta(s, q_0) \in Q_m$.

Ex. 3.7: O autômato do exemplo 3.6 reconhece todas as palavras sobre o alfabeto $\Sigma = \{\alpha, \beta\}$ que terminam com o símbolo α . ♦

Conforme visto na seção 3.1, uma coleção de palavras forma uma linguagem:

Def. 3.19: Dado um autômato $A = \langle \Sigma, Q, \delta, q_0, Q_m \rangle$, a *linguagem marcada* (ou *linguagem aceita* ou ainda *linguagem reconhecida*) de A , denotada por $L_m(A)$, é:

$$L_m(A) = \{s: s \in \Sigma^* \wedge \delta(s, q_0) \in Q_m\}.$$

A linguagem marcada de um autômato é a coleção de todas as palavras por ele reconhecidas. Note que a linguagem marcada é, no caso geral, uma linguagem não prefixo-fechada.

Ex. 3.8: O autômato do exemplo 3.6 reconhece a linguagem $\{(\alpha^*\beta^*)^*\alpha\}$. ♦

Viu-se na seção 3.1 que as tarefas que um dado SED é capaz de completar podem ser representadas por uma linguagem, denominada linguagem marcada do sistema. Conclui-se daí que é possível representar um SED de linguagem marcada L_m por um autômato A tal que $L_m(A) = L_m$.

É também desejável ter uma representação por máquinas de estados finitos para a linguagem gerada do sistema. No entanto, os autômatos definidos acima não possuem um

mecanismo que permita fazê-lo. A solução para este problema está em se permitir que a função de transição δ seja definida apenas para alguns pares (evento, estado) do conjunto $\Sigma \times Q$. Diz-se então que δ é uma *função parcial*. Embora se possa falar de "autômatos com função de transição parcial", utiliza-se aqui o termo "gerador" para denominar esta classe de máquinas, discutida a seguir.

3.3 GERADORES

A seguinte definição formaliza o que se disse no final da seção 3.2:

Def. 3.20: Um *gerador* é uma quintupla $G = \langle \Sigma, Q, \delta, q_0, Q_m \rangle$, onde:

Σ é um alfabeto;

Q é um conjunto finito de estados;

$\delta: \Sigma \times Q \rightarrow Q$ é uma função (em geral parcial) de transição de estados;

$q_0 \in Q$ é o estado inicial e

$Q_m \subseteq Q$ é o conjunto de estados marcados.

A única diferença entre autômatos e geradores é o fato de que nestes a função de transição pode ser parcial, ou seja, definida apenas para um subconjunto de eventos para cada estado do gerador. Assim como os autômatos, os geradores podem também ser representados por diagramas de transição, com a diferença de que, neste caso, estão presentes apenas os arcos para os quais a função de transição está definida.

Def. 3.21: Dado um gerador $G = \langle \Sigma, Q, \delta, q_0, Q_m \rangle$, associa-se a cada estado $q \in Q$ o conjunto de eventos definidos $\Sigma(q)$, dado por

$$\Sigma(q) = \{\sigma: \sigma \in \Sigma \wedge \delta(\sigma, q)!\},$$

onde $\delta(\sigma, q)!$ significa "δ está definida para o par (σ, q) ".

Este fato tem várias conseqüências, a começar pela função de transição estendida, que só pode ser definida para aquelas palavras que, quando processadas pelo gerador, se mantiverem dentro dos limites de definição da função de transição:

Def. 3.22: Dado um gerador $G = \langle \Sigma, Q, \delta, q_0, Q_m \rangle$, a *função de transição estendida*, denotada por $\bar{\delta}$, é uma função $\bar{\delta}: \Sigma^+ \times Q \rightarrow Q$ tal que:

$$\bar{\delta}(\epsilon, q) = q \text{ e}$$

$$\bar{\delta}(s\sigma, q) = \delta(\sigma, \bar{\delta}(s, q)), \text{ para } q \in Q \text{ e } s \in \Sigma^+ \text{ sempre que}$$

$$q' = \bar{\delta}(s, q) \text{ e } \delta(\sigma, q') \text{ estiverem ambos definidos.}$$

Como antes, dispensa-se de agora em diante a notação $\bar{\delta}$, representando-se a função de transição estendida simplesmente por δ .

Esta limitação da função de transição estendida se reflete no conjunto de palavras que o gerador pode processar. Tal conjunto forma, novamente, uma linguagem, definida como segue:

Def. 3.23: Dado um gerador $G = \langle \Sigma, Q, \delta, q_0, Q_m \rangle$, a *linguagem gerada de G* , denotada por $L(G)$, é:

$$L(G) = \{s : s \in \Sigma^* \wedge \delta(s, q_0) \in Q_m\}.$$

Enquanto que os autômatos, com sua função de transição completa, podem processar qualquer palavra sobre seu alfabeto, o caráter parcial da função de transição dos geradores faz com que a linguagem gerada seja, em geral, um subconjunto próprio de Σ^* . Além disso, esta linguagem é prefixo-fechada, como mostrado a seguir.

Prop. 3.2: Dado um gerador $G = \langle \Sigma, Q, \delta, q_0, Q_m \rangle$, a linguagem gerada $L(G)$ é prefixo-fechada.

Dem.: Seja uma palavra qualquer $s : s \in L(G)$. Então $\exists q \in Q : \delta(s, q_0) = q$. Considerando que q é necessariamente atingido através de outro estado $q' \in Q$, pode-se afirmar que s pode ser escrita na forma $s = s'\sigma$, com $\sigma \in \Sigma$ e $s' \in \text{Pre}(s)$ e que valem as expressões $\delta(s', q_0) = q'$ e $\delta(\sigma, q') = q$. Procedendo recursivamente desta forma, chega-se ao estado inicial q_0 , mostrando que $\text{Pre}(s) \in L(G)$. ♦

Portanto, para todo gerador G vale:

$$L(G) = \overline{\overline{L(G)}} \quad (3.3).$$

De forma análoga ao caso dos autômatos, o conjunto de estados marcados dos geradores pode ser utilizado para distinguir dois conjuntos de palavras geradas:

Def. 3.24: Dado um gerador $G = \langle \Sigma, Q, \delta, q_0, Q_m \rangle$, a *linguagem marcada de G* (ou *linguagem aceita* ou ainda *linguagem reconhecida por G*), denotada por $L_m(G)$, é:

$$L_m(G) = \{s : s \in \Sigma^* \wedge \delta(s, q_0) \in Q_m\}.$$

Portanto, dado um SED de linguagem gerada L e de linguagem marcada L_m , pode-se representá-lo por um gerador G tal que $L(G) = L$ e $L_m(G) = L_m$. Assim, os geradores permitem representar ambas as linguagens associadas a um SED, o que não era possível no caso dos autômatos.

Pelas definições 3.23 e 3.24 é claro que

$$L_m(G) \subseteq L(G) \quad (3.4).$$

Também é fácil ver que, se duas linguagens quaisquer K e L são tais que $K \subseteq L$, então $\overline{K} \subseteq \overline{L}$. Portanto, tomando o prefixo-fechamento de ambos os membros de 3.4, vem:

$$\overline{L_m(G)} \subseteq \overline{L(G)} \quad (3.5)$$

e, pela equação 3.3,

$$\overline{L_m(G)} \subseteq L(G) \quad (3.6).$$

Considerando-se as equações 3.1 a 3.6, é possível ordenar as linguagens apresentadas da seguinte forma:

$$L_m(G) \subseteq \overline{L_m(G)} \subseteq L(G) = \overline{L(G)} \quad (3.7).$$

É interessante comparar esta equação com a equação 3.2.

Note agora que a definição 3.20 não impõe qualquer restrição à estrutura do gerador. Em particular, é possível definir um gerador com estados inacessíveis, isto é, estados que jamais podem ser alcançados a partir do estado inicial. Tais estados formam "ilhas" que não são ligadas ao estado inicial por qualquer caminho. Isto é motivo para a seguinte definição:

Def. 3.25: A *componente acessível* do gerador $G = \langle \Sigma, Q, \delta, q_0, Q_m \rangle$ é

$Ac(G) = \langle \Sigma, Q_{ac}, \delta_{ac}, q_0, Q_{ac,m} \rangle$, com

$Q_{ac} = \{q: \exists w \in \Sigma^* \wedge \delta(w, q_0) = q\}$,

$Q_{ac,m} = Q_{ac} \cap Q_m$ e

$\delta_{ac} = \delta|_{(\Sigma \times Q_{ac})}$,

onde $\delta_{ac} = \delta|_{(\Sigma \times Q_{ac})}$ é a função δ restrita ao domínio $\Sigma \times Q_{ac}$.

Um gerador G é dito *acessível* sse $G = Ac(G)$.

Uma outra propriedade que um gerador pode ou não ter é a coacessibilidade:

Def. 3.26: Um gerador G é dito *coacessível* sse toda palavra em $L(G)$ for um prefixo de uma palavra em $L_m(G)$, ou seja, sse $L(G) \subseteq \overline{L_m(G)}$.

Esta definição diz que um gerador é coacessível sse, a partir de qualquer um de seus estados, existir ao menos um caminho que leve a um estado marcado. Considerando-se que a equação 3.6 vale para todo gerador G , é possível substituir a inclusão na definição 3.26 pela igualdade. Portanto, um gerador é coacessível sse

$$L(G) = \overline{L_m(G)} \quad (3.8).$$

Ex. 3.9: A figura 3.3 mostra dois geradores, o primeiro coacessível e o segundo não coacessível. A linguagem gerada por ambos os geradores é $\{\alpha, \alpha\beta, \alpha\beta\gamma\}$. Qualquer palavra

desta linguagem é um prefixo de uma palavra marcada no primeiro gerador, cuja linguagem marcada é $\{\alpha, \alpha\beta\gamma\}$, mas não no segundo: as palavras $\alpha\beta$ e $\alpha\beta\gamma$ não são prefixos de $\{\alpha\}$. ♦

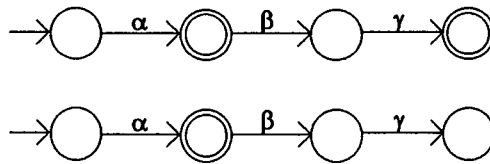


Fig. 3.3 - Gerador coaccessível x gerador não coaccessível

Def. 3.27: Um gerador que é ao mesmo tempo acessível e coaccessível é dito *trim*.

3.4 PROJEÇÕES

Existem casos em que é necessário trabalhar com uma visão do sistema tratado que exclua um subconjunto dos seus eventos, os quais são considerados *não observáveis*. Diz-se que um sistema com esta característica possui *observação parcial de eventos* ([LW88a]). O conceito de *projeção* é fundamental para tratar formalmente esse tipo de sistema.

Def. 3.28: Dados dois alfabetos Σ e Σ_1 , com $\emptyset \neq \Sigma_1 \subseteq \Sigma$, define-se a *projeção* $P: \Sigma^* \rightarrow \Sigma_1^*$ como:

$$P\varepsilon = \varepsilon$$

$$P\sigma = \varepsilon, \quad \sigma \in \Sigma - \Sigma_1$$

$$P\sigma = \sigma, \quad \sigma \in \Sigma_1$$

$$P(s\sigma) = (Ps)(P\sigma), \quad s \in \Sigma^* \wedge \sigma \in \Sigma.$$

O conjunto $\Sigma - \Sigma_1$ é denominado *núcleo* da projeção P .

O efeito da projeção P sobre uma palavra s é o de apagar de s todos os símbolos que não pertencem a Σ_1 .

Ex. 3.10: Sejam $\Sigma = \{\alpha, \beta, \lambda\}$ e $\Sigma_1 = \{\alpha, \lambda\}$. Então $P(\alpha\beta\lambda\alpha\beta) = \alpha\lambda\alpha$ e $P(\beta) = \varepsilon$. ♦

Esta definição pode ser estendida a linguagens, resultando ser a projeção da linguagem L dada por:

$$PL = \{s' : s' = Ps \wedge s \in L\} \tag{3.9}$$

Ex. 3.11: Sejam $\Sigma = \{\alpha, \beta, \lambda\}$ e $\Sigma_1 = \{\alpha, \lambda\}$. Então, para um observador limitado a ver os eventos de Σ_1 , a planta representada pelo gerador da figura 3.4a tem o aspecto mostrado na figura 3.4b. ♦

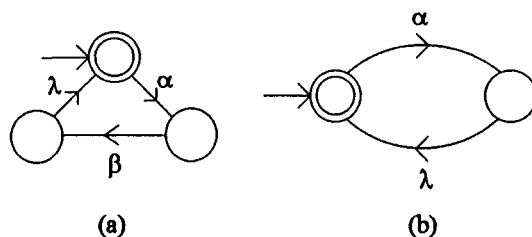


Fig. 3.4 - Geradores para o exemplo 3.11

Igualmente importante é o conceito de *projeção inversa*, definida como segue:

Def. 3.29: Dada uma projeção $P: \Sigma^* \rightarrow \Sigma_1^*$, a *projeção inversa de P*, denotada por P^{-1} , é definida como:

$$P^{-1}s = \{s' : s' \in \Sigma^* \wedge Ps' = s\}, \quad s \in \Sigma_1^*.$$

Note que, em geral, a projeção inversa de uma palavra s é um conjunto de palavras. Este conjunto é formado por todas as palavras s' que, quando vistas através da projeção, são iguais a s .

Ex. 3.12: Sejam $\Sigma = \{\alpha, \beta, \lambda\}$, $\Sigma_1 = \{\alpha, \lambda\}$ e $s = \alpha\lambda$. Então $P^{-1}s = \beta^*\alpha\beta^*\lambda\beta^*$. ♦

A projeção inversa também pode ser estendida a linguagens, sendo a projeção inversa da linguagem $L \subseteq \Sigma^*$ dada por:

$$P^{-1}L = \{s' : s' \in \Sigma^* \wedge Ps' \in L\} \quad (3.10)$$

Dado um gerador G , um gerador para $P^{-1}L(G)$ pode ser obtido do primeiro adicionando-se "selfloops" dos eventos de $\Sigma - \Sigma_1$ a todos os seus estados.

Ex. 3.13: Considere novamente o gerador da figura 3.4b, obtido no exemplo 3.11. A projeção inversa da linguagem gerada por aquele gerador é a linguagem gerada pelo gerador da figura 3.5. Compare também este exemplo com o exemplo 3.12.

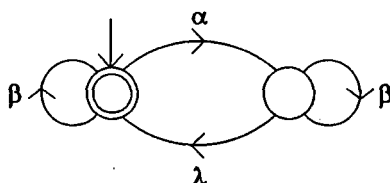


Fig. 3.5 - Gerador para o exemplo 3.13

As definições 3.28 e 3.29 permitem concluir que, para uma linguagem qualquer K ,

$$K \subseteq P^{-1}PK \quad (3.11).$$

3.5 COMBINAÇÃO DE GERADORES

Muitos sistemas de interesse prático apresentam características modulares e/ou distribuídas, de modo que podem ser vistos como constituídos de diversos subsistemas, cada qual gerando certos eventos. O comportamento do sistema como um todo é determinado então pelos eventos produzidos nesses subsistemas.

Seja um sistema constituído por n subsistemas. Numa representação por geradores, cada um desses subsistemas é representado por um gerador G_i , de subalfabeto $\Sigma_i \subseteq \Sigma$, onde

$\Sigma = \bigcup_{i=1}^n \Sigma_i$ é o alfabeto do sistema visto como um todo. Os subalfabetos Σ_i não são, em geral, disjuntos. Neste caso, assume-se que, se um evento comum a dois ou mais geradores chega a ocorrer, então ele ocorre em todos esses geradores simultaneamente, fato que determina um certo grau de sincronismo entre os subsistemas. O sistema como um todo pode então ser representado pelo autômato G obtido pelo *produto síncrono* dos geradores constituintes, definido como segue:

Def. 3.30: Dados n geradores G_i , cada qual com alfabeto Σ_i , o *produto síncrono* desses geradores é um gerador G de alfabeto $\Sigma = \bigcup_{i=1}^n \Sigma_i$, tal que:

$$L(G) = \{s: \forall i, 1 \leq i \leq n, P_i s \in L(G_i)\} \quad \text{e}$$

$$L_m(G) = \{s: \forall i, 1 \leq i \leq n, P_i s \in L_m(G_i)\},$$

sendo P_i é a projeção $P_i: \Sigma^* \rightarrow \Sigma_i^*$.

Ex. 3.14: Sejam os geradores G_1 e G_2 mostrados na figura 3.6. O gerador resultante do produto síncrono desses geradores, denotado por $G_1 \parallel G_2$, é dado na figura 3.7.

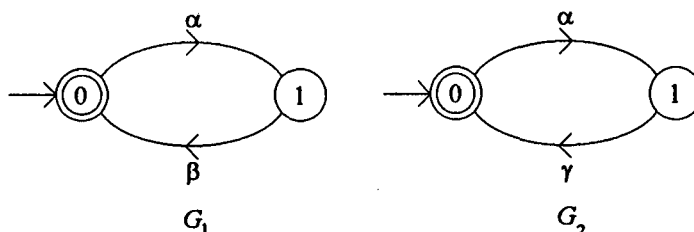


Fig. 3.6 - Geradores para o exemplo 3.14

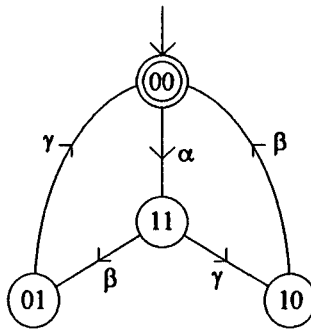


Fig. 3.7 - Resultado do produto síncrono

Os estados do gerador resultante são numerados com dois dígitos, correspondentes aos estados de G_1 e G_2 , respectivamente. Note que o evento α só é possível no estado 00, por ser este o único em que tal evento pode ocorrer em ambos os geradores componentes. Os eventos β e γ não necessitam qualquer sincronização, uma vez que não são eventos comuns a G_1 e G_2 . ♦

As definições e propriedades apresentadas neste capítulo constituem a base teórica necessária para iniciar o estudo do modelo Ramadge-Wonham para SED's, o assunto dos capítulos seguintes.

CAPÍTULO 4

O MODELO RW

Este capítulo apresenta o modelo Ramadge-Wonham em sua forma básica, classificado na tabela 2.1 como um modelo lógico não temporizado.

A apresentação se fundamenta em [RW87], contando ainda com algumas idéias de [RW89a] e com exemplos ilustrativos, criados a partir de um problema clássico da Ciência da Computação. Inicia-se pela representação de SED's por geradores. Em seguida, estes são dotados de um mecanismo de controle, que permite introduzir as noções de planta e de supervisor. Pode-se então apresentar a formulação de problemas no modelo RW. A seguir, definem-se o L_m -fechamento e a L -controlabilidade de linguagens, conceitos fundamentais para estabelecer as condições de existência de supervisores. Estas permitem, finalmente, formular e resolver um problema abstrato de síntese de supervisores e analisá-lo em um caso particular.¹

4.1 REPRESENTAÇÃO DE SED'S NO MODELO RW

Mostrou-se na seção 3.3 que um SED de linguagem gerada L e linguagem marcada L_m pode ser representado por um gerador G tal que $L(G) = L$ e $L_m(G) = L_m$.

A questão que surge neste ponto é: "dada uma linguagem $L \subseteq \Sigma^*$, existe sempre um gerador finito G tal que $L(G) = L$?". A resposta a esta pergunta é não. Sabe-se da Teoria de Linguagens que existem linguagens não representáveis por autômatos ou geradores finitos, como mostra o exemplo a seguir.

Ex. 4.1: Seja $\Sigma = \{\alpha, \beta\}$. A linguagem $L = \overline{\{\alpha^n \beta^{n+1} : n \geq 0\}}$ não é representável por um gerador finito: para cada n eventos α gerados, seria necessário prever um ramo do gerador com $n+1$ eventos β . Dado que n é ilimitado, o número de estados do gerador também o é. A figura 4.1 ilustra a tentativa de se construir este gerador.

Uma vez que a definição 3.20 exige que o conjunto de estados seja limitado, não se trata de um gerador finito. ♦

Formalmente, o teorema de Nerode mostra que uma linguagem L sobre o alfabeto Σ pode ser representada por um autômato (gerador) finito sse a *relação induzida por L em Σ^** tiver posto finito ([HU79], [CL89]). A Teoria de Linguagens prova ainda que a classe de linguagens representáveis por autômatos (geradores) finitos é exatamente a classe de linguagens regulares, citada na seção 3.1.

¹O leitor encontrará em [RW87] dois problemas de síntese, denominados "Supervisory Marking Problem" (SMP) e "Supervisory Control Problem" (SCP). No entanto, devido ao reduzido interesse prático do primeiro, apenas o segundo é tratado no presente trabalho.

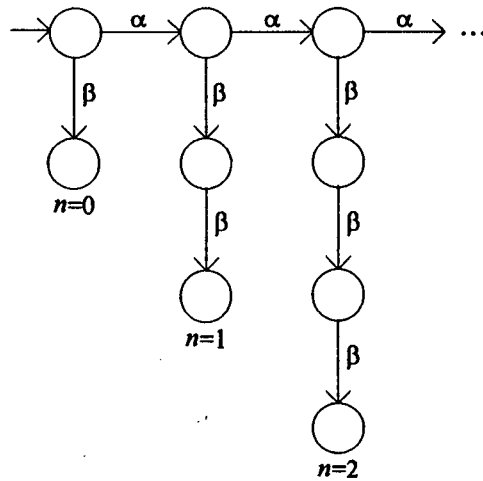


Fig. 4.1 - Gerador não finito

Os resultados estabelecidos na Teoria RW são formulados em termos da Teoria de Linguagens. Portanto, o modelo RW não é limitado à classe de SED's representáveis por geradores finitos, embora seus resultados computacionalmente úteis geralmente o sejam. Em razão disso, na maioria dos casos, tratam-se sistemas representáveis por geradores finitos. O exemplo a seguir ilustra um SED simples modelado por um gerador finito.

Ex. 4.2: A figura 4.2 mostra um gerador G que modela uma máquina com três estados, a saber R (em repouso), A (em atividade) e M (em manutenção). Há quatro transições possíveis, cada uma identificada por um evento do alfabeto $\Sigma = \{\alpha, \beta, \lambda, \mu\}$. O modelo permite concluir que a máquina parte do estado de repouso, de onde pode passar ao estado ativo (α) e deste voltar ao repouso (β) ou então sofrer uma pane (λ) e entrar em manutenção, de onde voltará eventualmente à condição de repouso (μ).

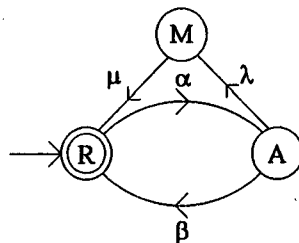


Fig. 4.2 - Gerador representando uma máquina com três estados

A linguagem gerada, $L(G)$, é o conjunto de todas as palavras obtidas partindo-se do estado inicial R e seguindo o grafo. A expressão regular que representa esta linguagem pode ser escrita como:

$$L(G) = (\alpha\beta + \alpha\lambda\mu)^*(\epsilon + \alpha + \alpha\lambda).$$

Note que o único estado marcado de G é o estado inicial. Isto significa que a linguagem marcada $L_m(G)$ compreende as palavras que representam um ciclo completo no grafo, seja pelo caminho $\alpha\beta$ ou pelo caminho $\alpha\lambda\mu$:

$$L_m(G) = (\alpha\beta + \alpha\lambda\mu)^*. \quad \blacklozenge$$

$L(G)$ é interpretada como uma representação do comportamento fisicamente possível do sistema e $L_m(G)$ como uma representação do conjunto de tarefas que este é capaz de executar. Isto permite utilizar a coacessibilidade de um gerador como critério para a ausência de bloqueio: num gerador coacessível, $L(G) = \overline{L_m(G)}$ (eq. 3.8), significando que toda palavra gerada é prefixo de alguma palavra marcada. Isto significa que toda seqüência de eventos fisicamente possível tem pelo menos uma continuação que leva a uma tarefa completa. Note que o gerador do exemplo 4.2 é coacessível.

Dentro desta interpretação, um gerador coacessível e o sistema por ele representado são ditos *não bloqueantes*.

4.2 GERADORES COM ENTRADA DE CONTROLE

Um SED modelado por um gerador como o do exemplo 4.2 está limitado a produzir seqüências de eventos, sem qualquer interferência. Para introduzir um mecanismo de controle no gerador $G = \langle \Sigma, Q, \delta, q_0, Q_m \rangle$, postula-se que existe um subconjunto de eventos $\Sigma_c \subseteq \Sigma$, denominados *eventos controláveis* e que podem ser inibidos, isto é, impedidos de ocorrer por um agente externo. Os demais eventos são ditos *não controláveis* e considerados permanentemente habilitados. Denota-se este conjunto por Σ_u , de maneira que valem as relações:

$$\Sigma = \Sigma_u \cup \Sigma_c \quad (4.1)$$

e

$$\Sigma_u \cap \Sigma_c = \emptyset \quad (4.2).$$

Esta partição do alfabeto de eventos reflete características de sistemas físicos. Por exemplo, o início de operação de uma máquina e o envio de uma mensagem num sistema de comunicação são, em geral, controláveis, ao contrário de uma pane ou da perda de uma mensagem, geralmente modeladas como eventos não controláveis.

Para que seja possível interferir no funcionamento do gerador, este precisa ser dotado de uma interface através da qual se possa informar quais eventos devem ser habilitados e quais devem ser inibidos. Considerando-se as eqs. 4.1 e 4.2, basta especificar um destes conjuntos para se ter o outro. Convenciona-se especificar o conjunto de eventos que se deseja habilitar, o qual recebe o nome de *entrada de controle*. Naturalmente, esta especificação não deve tentar inibir eventos não controláveis. Por esta razão, uma entrada de controle é considerada *válida* somente se contiver o conjunto de eventos não controláveis. A definição seguinte formaliza estas idéias.

Def. 4.1: Dados um gerador $G = \langle \Sigma, Q, \delta, q_0, Q_m \rangle$ e uma partição $\Sigma = \Sigma_u \cup \Sigma_c$, o conjunto de entradas de controle válidas associado a G é dado por $\Gamma = \{\gamma: \Sigma_u \subseteq \gamma \subseteq \Sigma\}$.

Numa abordagem rigorosa, caberia agora modificar ligeiramente a função de transição δ do gerador, para completar a interface de controle (v. [RW87]). Isto, no entanto, exige a utilização de uma notação mais complexa. Para evitá-la, adota-se aqui um tratamento equivalente, menos formal e baseado em [RW89a]. Considera-se então simplesmente que tudo se passa como se a função de transição de um gerador cujo alfabeto foi particionado em eventos controláveis e não controláveis deixasse de ser definida para os eventos inibidos por uma entrada de controle aplicada, enquanto esta estiver presente, sem explicitar este fato na notação. Sob este ponto de vista, tem-se a seguinte definição:

Def. 4.2: Um gerador com entrada de controle ou ainda gerador controlado é um par $\langle G, \Gamma \rangle$, onde G é um gerador com alfabeto particionado em eventos controláveis e não controláveis e Γ é o conjunto de entradas de controle determinado por essa partição.

Por analogia com a Teoria de Controle clássica, utiliza-se o termo *planta* para designar o sistema a ser controlado. A linguagem gerada pelo gerador que o representa é denominada *linguagem da planta* e interpretada como representando o comportamento do sistema na ausência de qualquer ação de controle que possa restringir sua operação. Um gerador controlado $\langle G, \Gamma \rangle$ representa corretamente um sistema quando as palavras de $L(G)$ correspondem exatamente às seqüências de eventos fisicamente possíveis neste sistema, estando todos os eventos controláveis habilitados (isto é, na ausência de qualquer ação de controle).

Quando se aplica uma entrada de controle γ a uma planta, esta se comporta como se os eventos inibidos fossem momentaneamente apagados da sua estrutura de transição, afetando com isso a linguagem gerada. É este o princípio de funcionamento do mecanismo de controle adotado no modelo RW, que consiste em chavear as entradas de controle em resposta ao comportamento observado do sistema, de modo a conformar a linguagem gerada a uma dada especificação.

Ex. 4.3: Considere novamente o sistema representado pelo gerador da figura 4.2. Supondo-se que o início de cada ciclo de operação e o término da manutenção são controláveis, vem: $\Sigma_c = \{\alpha, \mu\}$, $\Sigma_u = \{\beta, \lambda\}$. O conjunto de entradas de controle válidas é então $\Gamma = \{\{\beta, \lambda\}, \{\alpha, \beta, \lambda\}, \{\beta, \lambda, \mu\}, \{\alpha, \beta, \lambda, \mu\}\}$. Aplicar a entrada de controle $\gamma = \{\beta, \lambda, \mu\}$ significaria deter o sistema no estado de repouso. ♦

Cabe ressaltar aqui que, embora simples e intuitivamente aceitável, a utilidade prática deste mecanismo depende de se encontrarem condições necessárias e suficientes para a existência do controlador que faça o chaveamento das entradas de controle da planta de modo

a satisfazer uma dada especificação de comportamento, bem como um procedimento de síntese para obter tal controlador. O mérito do modelo RW está justamente nas soluções propostas para estes problemas, das quais decorrem as demais conquistas desta teoria.

4.3 SUPERVISORES

A figura 4.3 ilustra o conceito de supervisor como um agente que tem a capacidade de observar os eventos gerados e, em resposta, determinar entradas de controle $\gamma, \gamma', \gamma'', \dots \in \Gamma$, as quais são realimentadas à planta, determinando o comportamento do sistema sob supervisão. Novamente por analogia com a Teoria de Controle clássica, este comportamento é também denominado de *comportamento em malha fechada*.

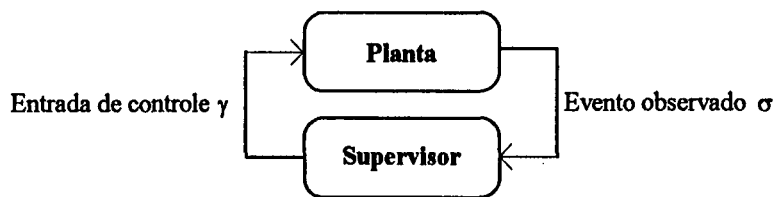


Fig. 4.3 - Acoplamento de planta e supervisor no modelo RW

Enquanto prática comum na Teoria de Controle clássica, a separação entre sistema a controlar (planta) e controlador (supervisor) para SED's pode parecer artificial em algumas aplicações, principalmente na área de sistemas computacionais, onde muitas vezes é difícil determinar as fronteiras entre um e outro. No entanto, esta separação tende a simplificar o problema quando se especifica o comportamento desejado para um sistema capaz de realizar diversas ações, permitindo ainda a formulação de problemas abstratos de síntese de supervisores.

4.3.1 DEFINIÇÃO

Def. 4.3: Um *supervisor* para o gerador controlado $\langle G, \Gamma \rangle$ é um par $\langle S, \Phi \rangle$, composto de um gerador $S = \langle \Sigma, X, \xi, x_0, X_m \rangle$ e de um mapa de controle Φ , onde:

Σ é o mesmo alfabeto do gerador $G = \langle \Sigma, Q, \delta, q_0, Q_m \rangle$;

X é um conjunto de estados;

$\xi: \Sigma^* \times X \rightarrow X$ é uma função de transição (parcial) estendida;

$x_0 \in X$ é o estado inicial;

$X_m \subseteq X$ é o conjunto de estados marcados e

$\Phi: X \rightarrow \Gamma$ é uma função que associa a cada $x \in X$ uma entrada de controle $\gamma \in \Gamma$.

A especificação de um alfabeto idêntico para S e G é necessária para que se possa interconectá-los como na figura 4.3. Com isso, o supervisor pode ser construído de modo a rastrear as palavras geradas pela planta, evoluindo no seu espaço de estados de acordo com a função ξ . A função Φ , denominada *mapa de controle*, associa a cada estado $x \in X$ uma entrada de controle $\gamma \in \Gamma$ para aplicar em G .

Ex. 4.4: Para ilustrar o funcionamento de um supervisor, considere o problema do almoço dos filósofos orientais. O problema clássico da área de computação (v. por exemplo [PETE81]) compreende cinco filósofos que podem estar, cada um, em um de dois estados: comendo (C) ou meditando (M). Para comer, um filósofo utiliza dois palitos, um encontrado à sua esquerda e outro à sua direita. Desta forma, dois filósofos adjacentes não podem comer ao mesmo tempo, pois o palito entre eles não pode ser utilizado simultaneamente por ambos. O problema clássico foi aumentado aqui com a inclusão de um copo de vinho de arroz, que também só pode ser utilizado por um dos filósofos de cada vez. Além disso, imagina-se que somente os filósofos F1 e F3 têm direito ao vinho; desta forma, estes possuem um terceiro estado, no qual estão bebendo (B). A figura 4.4 mostra os filósofos à mesa e os geradores que os representam.

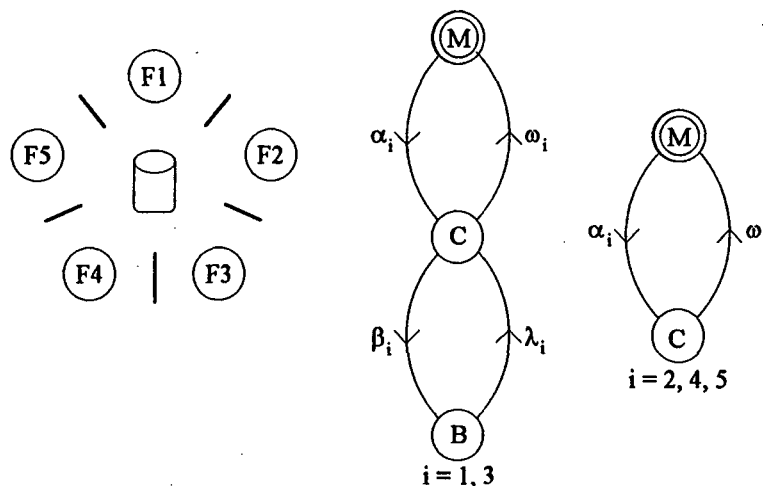


Fig. 4.4 - O almoço dos filósofos orientais

Os eventos são interpretados da seguinte forma:

$\alpha_i, i = 1, 2, \dots, 5$: o filósofo F_i pára de meditar e tenta apanhar os palitos com que come;

$\omega_i, i = 1, 2, \dots, 5$: o filósofo F_i pára de comer, deposita os palitos e volta a meditar;

$\beta_i, i = 1, 3$: o filósofo F_i pára de comer e tenta apanhar o copo;

$\lambda_i, i = 1, 3$: o filósofo F_i pára de beber e tenta apanhar os palitos para voltar a comer.

Considera-se que apenas os eventos α_i são controláveis. Os filósofos são entendidos como dotados de vontade própria, sem uma convenção que regulamente a ordem em que podem comer. A única maneira de influenciar seu comportamento é inibir ou habilitar os eventos α_i . O objetivo do problema é controlar os filósofos de modo que não surjam conflitos durante a refeição, isto é, para que dois deles não tentem acessar um recurso comum (copo, palito) ao mesmo tempo. Além disso, é possível imaginar diversos sub-problemas interessantes, considerando apenas alguns dos filósofos de cada vez. Com isso é possível trabalhar com geradores menores do que aqueles que resultariam da análise do problema completo, sem contudo afetar o poder ilustrativo dos exemplos.

Considere os filósofos F4 e F5. Com o produto síncrono, definido na seção 3.5, obtém-se um gerador que representa a planta formada pelos dois filósofos. Este gerador está representado na figura 4.5 e tem alfabeto $\Sigma = \{\alpha_4, \alpha_5, \omega_4, \omega_5\}$. Sendo $\Sigma_c = \{\alpha_4, \alpha_5\}$, deduz-se que o conjunto de entradas de controle válidas é dado por: $\Gamma = \{\{\omega_4, \omega_5\}, \{\alpha_4, \omega_4, \omega_5\}, \{\alpha_5, \omega_4, \omega_5\}, \{\alpha_4, \alpha_5, \omega_4, \omega_5\}\}$. Aplicando-se uma entrada de controle que não contenha o evento α_i , é possível deter o filósofo F_i no estado de meditação, embora não se possa impedir-lo de voltar a este estado, uma vez que esteja comendo.

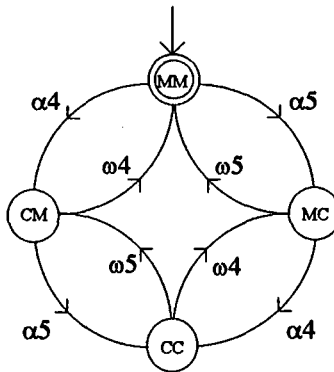


Fig. 4.5 - Planta composta pelos filósofos F4 e F5

O gerador para a planta representa o comportamento fisicamente possível em termos dos eventos envolvidos. Os estados deste gerador correspondem às situações em que F4 e F5 estão ambos meditando (MM), um está meditando e o outro comendo (MC e CM) e ainda à situação em que (teoricamente) ambos estariam comendo (CC). Dado que F4 e F5 utilizam um palito comum, a planta deve ser impedida de executar qualquer transição que pudesse levá-la a este último estado.

Note que uma solução trivial seria simplesmente manter os eventos α_4 e α_5 permanentemente inibidos. Embora esta solução certamente garantisse que os filósofos não tentariam comer simultaneamente, implicaria também na sua morte por inanição. No entanto, basta manter um dos filósofos meditando enquanto o outro come. Esta idéia de "melhor

solução possível" está presente também nos problemas de síntese de supervisores apresentados neste capítulo e no capítulo 5.

Dito isto, é compreensível que o comportamento desejado para o sistema possa ser representado pelo gerador da figura 4.6, obtido da planta por eliminação do estado proibido e dos eventos a ele ligados. O supervisor $\langle S, \Phi \rangle$ para garantir que este comportamento seja obedecido pode então ser construído adotando-se para S exatamente a estrutura da figura 4.6 e para Φ o mapa da tabela 4.1.

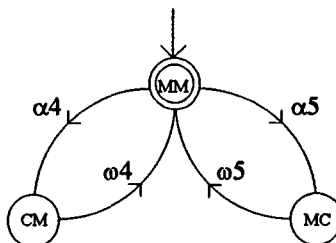


Fig. 4.6 - Comportamento desejado para F4 e F5

Estado do gerador ($x \in X$)	Entrada de controle ($\gamma \in \Gamma$)
MM	$\{\alpha_4, \alpha_5, \omega_4, \omega_5\}$
CM	$\{\alpha_4, \omega_4, \omega_5\}$ ou $\{\omega_4, \omega_5\}$
MC	$\{\alpha_5, \omega_4, \omega_5\}$ ou $\{\omega_4, \omega_5\}$

Tab. 4.1 - Mapa de controle para os filósofos F4 e F5

Note que nos estados CM e MC existe mais de uma entrada de controle adequada: é indiferente habilitar ou não eventos controláveis cuja ocorrência num dado estado é fisicamente impossível. O mapa de controle Φ pode ainda ser dado sob forma matricial, com os estados enumerando as linhas e os eventos as colunas da matriz. O elemento Φ_{ij} é definido como:

- $\Phi_{ij} = '0'$ se o evento j está inibido no estado i ;
- $\Phi_{ij} = '1'$ se o evento j está habilitado no estado i ;
- $\Phi_{ij} = '-'$ se é indiferente habilitar ou não o evento j no estado i .

A tabela 4.2 apresenta o mapa Φ do exemplo sob forma matricial. Esta será a representação utilizada deste ponto em diante.

Φ	α_4	α_5	ω_4	ω_5
MM	1	1	1	1
CM	-	0	1	1
MC	0	-	1	1

Tab. 4.2 - Mapa de controle para F4 e F5, forma matricial

O funcionamento do sistema em malha fechada é ilustrado na figura 4.7. É importante frisar que, apesar de se utilizarem os mesmos nomes para os estados de S e de G , estes são geradores distintos. A G cabe gerar eventos, que S usa para rastrear o comportamento da planta. Em resposta, S habilita ou inibe eventos controláveis de G , impedindo que o estado proibido seja atingido. Cabe aqui chamar a atenção para a forte analogia existente entre esta solução e a observação aliada à realimentação de estados, da Teoria de Controle clássica. ♦

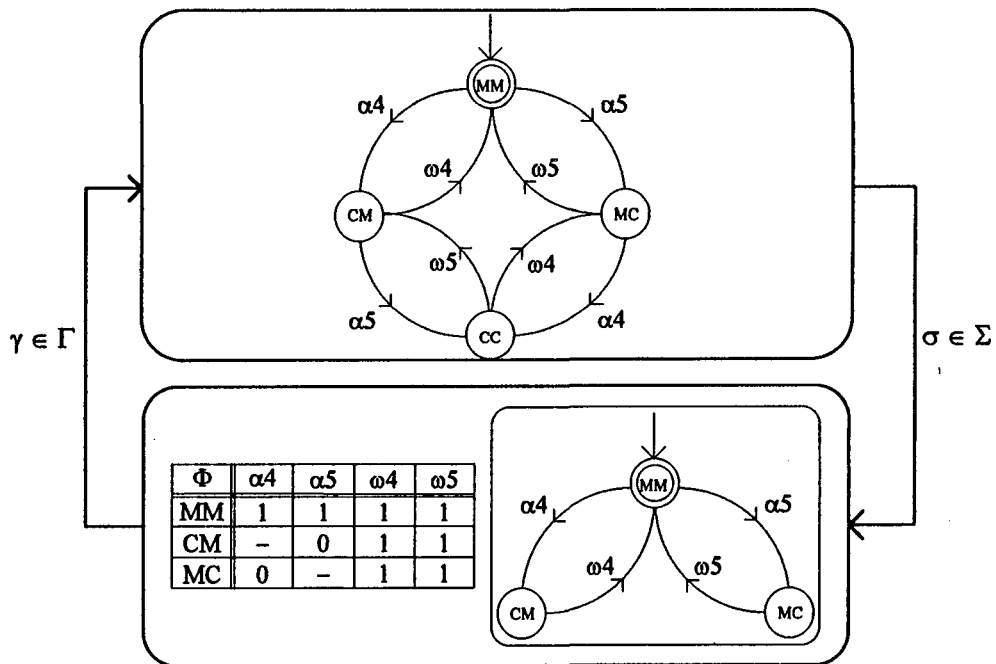


Fig. 4.7 - Filósofos F4 e F5 - planta e supervisor acoplados

O exemplo acima é bastante simples. Em particular, a obtenção do supervisor foi feita de forma intuitiva, sem um procedimento formal de síntese. A teoria a ser desenvolvida permitirá construir supervisores para sistemas mais complexos, onde obtê-los como aqui seria uma tarefa no mínimo fatigante.

Cabe citar ainda que os problemas que podem ser resolvidos desta forma não se limitam àqueles formulados em termos de estados proibidos. Problemas mais gerais incluem seqüências de eventos proibidas, caso em que o gerador do supervisor pode ter mais estados do que aquele da planta, a fim de "memorizar" todas as situações possíveis. No exemplo acima, poder-se-ia modificar o supervisor de maneira a permitir que cada filósofo não pudesse comer mais do que um certo número de vezes seguidas, implementando uma política de justiça. A planta ficaria inalterada. Esta idéia ilustra a possibilidade interessante oferecida pelo modelo RW de, com uma mesma planta, obter vários padrões de comportamento, simplesmente trocando-se o supervisor que a controla.

4.3.2 LINGUAGENS

A ação de controle modifica as linguagens associadas ao gerador. O comportamento do sistema sob supervisão é formalizado pelas seguintes definições:

Def. 4.4: Dados uma planta $\langle G, \Gamma \rangle$ e um supervisor $\langle S, \Phi \rangle$, a *linguagem gerada* pelo sistema sob supervisão, denotada por $L(S/G)$, é definida pelas seguintes asserções:

$$\varepsilon \in L(S/G) \text{ e}$$

$$s\sigma \in L(S/G) \text{ sse } s \in L(S/G) \wedge s\sigma \in L(G) \wedge \sigma \in \Phi(\xi(s, x_0)).$$

A definição 4.4 diz que os eventos que podem ocorrer sob supervisão são os eventos fisicamente possíveis e habilitados. Dado que uma palavra s pertence a $L(S/G)$ somente se pertencer a $L(G)$, é claro que:

$$L(S/G) \subseteq L(G) \quad (4.3).$$

Ainda, dado que uma palavra $s\sigma \in L(S/G)$ somente se $s \in L(S/G)$, pode-se afirmar que $L(S/G)$ é prefixo-fechada, isto é, que

$$L(S/G) = \overline{\overline{L(S/G)}}. \quad (4.4)$$

Naturalmente, a supervisão também tem efeito sobre a linguagem marcada do gerador:

Def. 4.5: Dados uma planta $\langle G, \Gamma \rangle$ e um supervisor $\langle S, \Phi \rangle$, a *linguagem controlada* do sistema sob supervisão, denotada por $L_c(S/G)$, é definida como:

$$L_c(S/G) = L(S/G) \cap L_m(G).$$

A linguagem controlada é simplesmente a parte da linguagem marcada original que "sobrevive" sob a ação de controle. Se $L_m(G)$ representa as tarefas que podem ser completadas pela planta, então $L_c(S/G)$ representa as tarefas que podem ser completadas sob supervisão. Em consequência desta definição,

$$L_c(S/G) \subseteq L(S/G) \quad (4.5)$$

e

$$L_c(S/G) \subseteq L_m(G) \quad (4.6).$$

Além disso, o conjunto de estados marcados do supervisor pode ser utilizado para alterar a linguagem marcada do sistema sob supervisão. Convenciona-se que uma palavra gerada no sistema em malha fechada é reconhecida sse tanto G quanto S a reconhecerem. Isto dá origem à terceira linguagem associada à planta supervisionada:

Def. 4.6: Dados uma planta $\langle G, \Gamma \rangle$ e um supervisor $\langle S, \Phi \rangle$, a *linguagem marcada* do sistema sob supervisão, denotada por $L_m(S/G)$, é definida como:

$$L_m(S/G) = L_c(S/G) \cap \bar{L}_m(S).$$

Esta definição implica que

$$L_m(S/G) \subseteq L_c(S/G) \quad (4.7),$$

e, lembrando que, pela definição 3.24, $L_m(S) = \{w: w \in \Sigma^* \wedge \xi(w, x_0) \in X_m\}$, conclui-se que a igualdade é atingida quando $X_m = X$. Na maioria das aplicações, o supervisor não desempenha papel quanto à marcação, de maneira que, em geral, todos os seus estados são marcados.

As equações 4.3, 4.5 e 4.7 permitem ainda ordenar as linguagens apresentadas da seguinte forma:

$$L_m(S/G) \subseteq L_c(S/G) \subseteq L(S/G) \subseteq L(G) \quad (4.8).$$

Cabe aqui um resumo das principais informações apresentadas, antes de prosseguir com a exposição da teoria.

Notação	Descrição	Interpretação	Fechamento	Def.
$L(G)$	Linguagem gerada de G	Comportamento fisicamente possível da planta $\langle G, \Gamma \rangle$ sem ação de controle	$L(G) = \overline{L(G)}$	3.23
$L_m(G)$	Linguagem marcada de G	Tarefas que podem ser completadas pela planta $\langle G, \Gamma \rangle$ sem ação de controle	$L_m(G) \subseteq \overline{L_m(G)}$	3.24
$L(S/G)$	Ling. gerada de $\langle G, \Gamma \rangle$ sob ação de $\langle S, \Phi \rangle$	Comportamento fisicamente possível da planta $\langle G, \Gamma \rangle$ sob ação de $\langle S, \Phi \rangle$	$L(S/G) = \overline{L(S/G)}$	4.4
$L_c(S/G)$	Ling. controlada de $\langle G, \Gamma \rangle$ sob ação de $\langle S, \Phi \rangle$	Tarefas que a planta $\langle G, \Gamma \rangle$ pode completar sob ação de $\langle S, \Phi \rangle$	$L_c(S/G) \subseteq \overline{L_c(S/G)}$	4.5
$L_m(S/G)$	Ling. marcada de $\langle G, \Gamma \rangle$ sob ação de $\langle S, \Phi \rangle$	Tarefas marcadas como completadas pela planta $\langle G, \Gamma \rangle$ sob ação de $\langle S, \Phi \rangle$	$L_m(S/G) \subseteq \overline{L_m(S/G)}$	4.6

Tab. 4.3 - Linguagens utilizadas no modelo RW

A tabela 4.3 apresenta as linguagens definidas até aqui e suas principais características. Além de permitir ao leitor revisar os diferentes conceitos e compará-los entre si, pode ser utilizada como referência a partir de outros pontos no trabalho.

4.3.3 SUPERVISORES PRÓPRIOS

Quando se diz que o supervisor é capaz de rastrear o funcionamento da planta, o que se faz implicitamente é exigir que sua função de transição seja definida para todo evento habilitado fisicamente possível. Embora se possa, teoricamente, conceber supervisores em que isto não aconteça sem contrariar a definição 4.3, só terão sentido prático aqueles concebidos para acompanhar a evolução da planta. Este conceito é formalizado na definição seguinte:

Def. 4.7: Um supervisor $\langle S, \Phi \rangle$ para uma planta $\langle G, \Gamma \rangle$ é dito *completo* quando as três condições seguintes:

- (i) $s \in L(S/G)$, (ii) $s\sigma \in L(G)$ e (iii) $\sigma \in \Phi(\xi(s, x_0))$ juntas implicarem em
- (iv) $\xi(s\sigma, x_0)!$.

A definição 4.7 diz que se (i) s é uma palavra que pode ocorrer no sistema supervisionado, (ii) o evento σ é uma continuação fisicamente possível da palavra s e (iii) este evento está habilitado, então (iv) a palavra $s\sigma$ deve estar definida na função de transição do supervisor. Assume-se ao longo deste trabalho que os supervisores são sempre completos.

A fim de controlar um SED de forma satisfatória, as linguagens $L_m(S/G)$, $L_c(S/G)$ e $L(S/G)$ devem ainda satisfazer duas restrições adicionais:

Def. 4.8: Um supervisor $\langle S, \Phi \rangle$ para a planta $\langle G, \Gamma \rangle$ é dito *não bloqueante* sse

$$\overline{L_c(S/G)} = L(S/G).$$

Def. 4.9: Um supervisor $\langle S, \Phi \rangle$ para a planta $\langle G, \Gamma \rangle$ é dito *não rejeitante* sse

$$\overline{L_m(S/G)} = \overline{L_c(S/G)}.$$

Pelas equações 4.4 e 4.5, vale:

$$\overline{L_c(S/G)} \subseteq L(S/G) \tag{4.9}.$$

Se a condição de ausência de bloqueio é violada, então esta inclusão é própria e existe ao menos uma palavra fisicamente possível em $L(S/G)$ que não é prefixo de qualquer palavra em $L_c(S/G)$ e portanto nunca leva a uma tarefa completada.

Por outro lado, a equação 4.7 implica:

$$\overline{L_m(S/G)} \subseteq \overline{L_c(S/G)} \tag{4.10}.$$

Se a condição de não rejeição deixa de ser satisfeita, isto é, se a inclusão é própria, então existe ao menos uma palavra em $\overline{L_c(S/G)}$ que representa uma tarefa completada, mas que não pertence a $\overline{L_m(S/G)}$. Isto significa que o sistema pode atingir um estado a partir do

qual nenhuma tarefa completada é reconhecida como tal. Note que a rejeição não é caracterizada pela não marcação de uma tarefa completada (este é, de fato, o papel da marcação pelo supervisor) mas sim pela impossibilidade de se marcar qualquer tarefa a partir de um certo estado.

Estas duas situações indesejáveis são ilustradas no exemplo a seguir.

Ex. 4.5: Seja o gerador $G = \langle \Sigma, Q, \delta, q_0, Q_m \rangle$, com $\Sigma = \{\alpha, \beta, \lambda\}$ mostrado na figura 4.8, cuja linguagem marcada é $L_m(G) = (\alpha\lambda^*\beta)^*$. Seja ainda $\Sigma_c = \{\beta\}$.

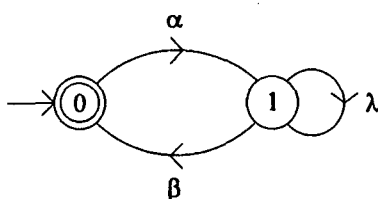


Fig. 4.8 - Gerador para o exemplo 4.5

Considere primeiramente o supervisor da figura 4.9a. Quando acoplado ao gerador acima, o comportamento do sistema fica restrito ao representado pelo gerador da figura 4.9b.

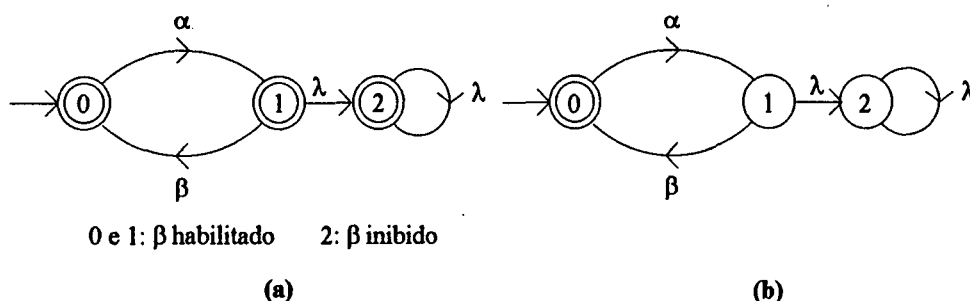


Fig. 4.9 - Supervisor com bloqueio e comportamento resultante

Vê-se que $L(S/G) = (\alpha\beta)^*(\epsilon + \alpha\lambda^*)$ e que $L_c(S/G) = (\alpha\beta)^*$, de modo que a condição de ausência de bloqueio não é satisfeita. Nenhuma seqüência incluindo o evento λ leva a uma tarefa completada sob supervisão, pois o evento β é definitivamente inibido logo após a primeira ocorrência de λ .

Considere agora o supervisor da figura 4.10a e o comportamento resultante de sua aplicação à planta, dado na figura 4.10b.

Pode-se ver que $L_c(S/G) = L_m(G) = (\alpha\lambda^*\beta)^*$. Isto significa que todas as tarefas fisicamente possíveis podem ser completadas sob supervisão. No entanto, $L_m(S/G) = (\alpha\beta)^*$, violando a condição de ausência de rejeição. A partir da primeira ocorrência do evento λ todas as tarefas completadas deixam de ser marcadas. ♦

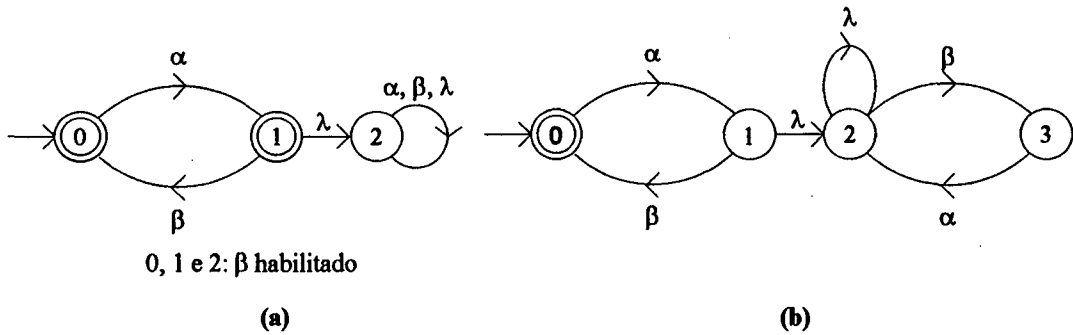


Fig. 4.10 - Supervisor com rejeição e comportamento resultante

Naturalmente, é sempre desejável construir supervisores não bloqueantes e não rejeitantes. Estes conceitos são aliados ao de supervisor completo na seguinte definição:

Def. 4.10: Um supervisor completo, não bloqueante e não rejeitante é dito *supervisor próprio*. Como consequência, um supervisor próprio é um supervisor completo tal que

$$\overline{L_m(S/G)} = \overline{L_c(S/G)} = L(S/G).$$

4.4 FORMULAÇÃO DE PROBLEMAS NO MODELO RW

As linguagens apresentadas na subseção 4.3.2 permitem a formulação de problemas abstratos de síntese de supervisores. De um modo geral, um problema desse tipo supõe que se represente o comportamento fisicamente possível do sistema e o comportamento desejado sob supervisão por linguagens, sendo o objetivo construir um supervisor para a planta tal que o comportamento do sistema em malha fechada se limite ao comportamento desejado.

4.4.1 FORMULAÇÃO EM TERMOS DE LINGUAGENS MARCADAS

Na formulação adotada em [RW87], aqui denominada *formulação em termos de linguagens marcadas*, o sistema a ser controlado é representado por uma planta $\langle G, \Gamma \rangle$ e o comportamento desejado é especificado na forma de uma *linguagem-alvo* $K \subseteq L_m(G)$, que representa as tarefas que se deseja sejam completáveis sob supervisão. O objetivo do problema é encontrar (se possível) um supervisor próprio $\langle S, \Phi \rangle$ para $\langle G, \Gamma \rangle$ tal que o comportamento em malha fechada satisfaça a igualdade $L_c(S/G) = K$.

4.4.2 FORMULAÇÃO EM TERMOS DE LINGUAGENS GERADAS

De forma alternativa, pode-se especificar o comportamento desejado na forma de uma linguagem-alvo prefixo-fechada $K \subseteq L(G)$, que representa o comportamento fisicamente possível desejado sob supervisão. O objetivo do problema é então encontrar (se possível) um supervisor $\langle S, \Phi \rangle$ para $\langle G, \Gamma \rangle$ tal que o comportamento em malha fechada satisfaça a igualdade $L(S/G) = K$.

Cabe citar aqui que, além dos dois tipos de problemas apresentados, aparece ainda na literatura um terceiro ([RW87]), onde o objetivo é construir um supervisor de modo que a linguagem $L_m(S/G)$ seja igual a uma dada linguagem-alvo marcada $K \subseteq L_m(G)$. O papel do supervisor $\langle S, \Phi \rangle$ é, então, o de marcar um subconjunto das tarefas completadas, o que se consegue com uma escolha adequada do conjunto de estados marcados de S . No entanto, devido ao seu reduzido interesse prático, este problema não será tratado em mais detalhes no presente trabalho.

O restante deste capítulo trata das condições de existência de supervisores e de soluções para problemas abstratos de síntese formulados em termos de linguagens marcadas.

4.5 L-FECHAMENTO

Esta seção introduz o conceito de *fechamento em relação a uma linguagem*, que se mostrará fundamental para o estabelecimento das condições de existência de supervisores na seção 4.7. Considere para tanto a seguinte definição:

Def. 4.11: Dadas duas linguagens quaisquer $K, L \subseteq \Sigma^*$, K é dita *fechada em relação a L* ou simplesmente *L -fechada* sse

$$K = \bar{K} \cap L.$$

Embora a definição 4.11 seja aplicável a quaisquer linguagens K, L , ela aparece na teoria RW envolvendo a linguagem-alvo $K \subseteq L_m(G)$ e a linguagem marcada da planta, $L_m(G)$. Se K é $L_m(G)$ -fechada, isto é, se $K = \bar{K} \cap L_m(G)$, então todo prefixo de qualquer palavra de K que pertença a $L_m(G)$ é também uma palavra de K .

Ex. 4.6: Considere o gerador da figura 4.11, cuja linguagem marcada é $\{\alpha\beta, \alpha\beta\lambda\mu\}$. Suponha que se desejasse especificar a linguagem-alvo $K \subseteq L_m(G)$ de forma que $\alpha\beta\lambda\mu \in K$. Uma vez que a geração da palavra $\alpha\beta\lambda\mu$ exige a geração prévia de seus prefixos, é inevitável que $\alpha\beta$ seja gerada cada vez que se gerar $\alpha\beta\lambda\mu$. Assim, é visível que um supervisor que permita a ocorrência de $\alpha\beta\lambda\mu$ é obrigado a permitir também a ocorrência de $\alpha\beta$, indicando que um supervisor para K só pode existir se K for $L_m(G)$ -fechada.

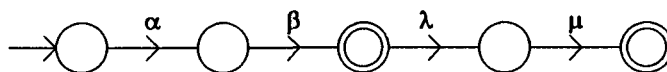


Fig. 4.11 - Gerador para ilustrar L -fechamento

Prop. 4.1: Seja $\langle S, \Phi \rangle$ um supervisor para a planta $\langle G, \Gamma \rangle$. Então $L_c(S/G)$ é $L_m(G)$ -fechada.

Dem.: Partindo da definição 4.5:

$$L_c(S/G) = L(S/G) \cap L_m(G)$$

$$\overline{L_c(S/G)} = \overline{L(S/G) \cap L_m(G)}$$

$$\overline{L_c(S/G)} \subseteq \overline{L(S/G)} \cap \overline{L_m(G)}.$$

Pela equação 4.4,

$$\overline{L_c(S/G)} \subseteq L(S/G) \cap \overline{L_m(G)}$$

e, por interseção com $L_m(G)$,

$$\overline{L_c(S/G)} \cap L_m(G) \subseteq L(S/G) \cap L_m(G).$$

Pela definição 4.5,

$$\overline{L_c(S/G)} \cap L_m(G) \subseteq L_c(S/G).$$

Por outro lado, pela equação 3.1,

$$L_c(S/G) \subseteq \overline{L_c(S/G)}$$

e, pela definição 4.5,

$$L_c(S/G) \subseteq L_m(G)$$

donde

$$L_c(S/G) \subseteq \overline{L_c(S/G)} \cap L_m(G).$$

Logo,

$$L_c(S/G) = \overline{L_c(S/G)} \cap L_m(G). \quad \blacklozenge$$

4.6 CONTROLABILIDADE

O objetivo desta seção é introduzir a definição de *controlabilidade* de uma linguagem, conceito que, tal qual o $L_m(G)$ -fechamento apresentado na seção 4.5, é fundamental para estabelecer as condições de existência de supervisores.

Considere a seguinte definição:

Def. 4.12: Dados um alfabeto Σ , particionado em eventos controláveis e não controláveis de acordo com as equações 4.1 e 4.2 e linguagens K e L sobre Σ tais que $K \subseteq L$, a linguagem K é dita *L-controlável* sse

$$\overline{K} \Sigma_u \cap L \subseteq \overline{K}.$$

Aplicando-se a definição 4.12 à linguagem-alvo $K \subseteq L_m(G)$ e tomando-se $L = L(G)$, resulta que K é $L(G)$ -controlável sse

$$\overline{K} \Sigma_u \cap L(G) \subseteq \overline{K} \tag{4.11},$$

ou seja, se toda palavra $s\sigma: s \in \overline{K} \wedge \sigma \in \Sigma_u \wedge s\sigma \in L(G)$ pertencer também a \overline{K} .

Ex. 4.7: Considere novamente o almoço dos filósofos orientais, introduzido no exemplo 4.4. Considere agora os filósofos F1 e F3. Novamente, a planta que os representa é obtida pelo produto síncrono dos geradores, como ilustrado na figura 4.12.

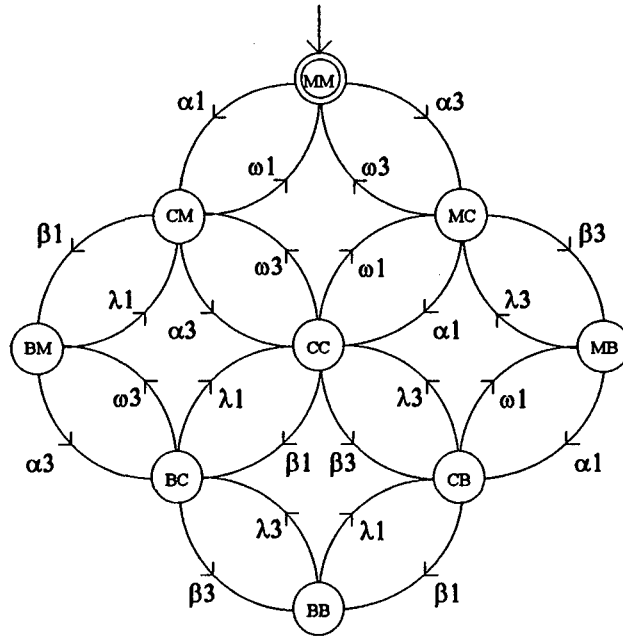


Fig. 4.12 - Planta composta pelos filósofos F1 e F3

Os estados da planta correspondem às situações em que os filósofos estão ambos meditando (MM), um está meditando e o outro comendo (MC e CM), ambos estão comendo (CC), um está meditando e o outro bebendo (MB e BM), um está comendo e o outro bebendo (BC e CB) e (teoricamente) ambos estão bebendo (BB). Dado que o copo é um recurso único, este último estado é proibido. Deste modo, o comportamento desejado para o sistema pode ser representado pelo gerador da figura 4.13, obtido da planta por eliminação do estado proibido BB e dos eventos a ele ligados.

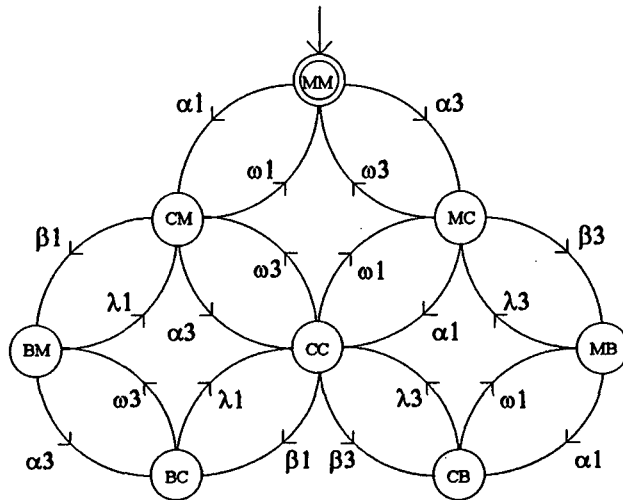


Fig. 4.13 - Comportamento desejado para os filósofos F1 e F3

Até aqui, o procedimento adotado é idêntico ao do exemplo 4.4. O próximo passo seria determinar o mapa de controle para o supervisor que limitasse o comportamento da planta ao comportamento desejado. Mas tal supervisor teria que desabilitar os eventos β_3 (no

estado BC) e β_1 (no estado CB), ambos não controláveis, o que é impossível por hipótese. Conclui-se daí que o comportamento desejado só é realizável se o gerador que o representa for tal que não seja necessário inibir, em qualquer estado, eventos não controláveis.

De um modo geral, pode-se ver que é condição necessária à existência de um supervisor que, dados uma palavra $s \in \bar{K}$ e um evento $\sigma \in \Sigma_u$ tais que $s\sigma \in L(G)$, não se proíba o evento σ de ocorrer após a palavra s , ou seja, é necessário que $s\sigma \in \bar{K}$, conforme exigido pela equação 4.11. No exemplo anterior, esta condição é violada por palavras como $s = \alpha_1\beta_1\alpha_3 \in \bar{K}$, que pode ser continuada com $\sigma = \beta_3 \in \Sigma_u$, sendo que $s\sigma = \alpha_1\beta_1\alpha_3\beta_3 \notin \bar{K}$.

4.7 EXISTÊNCIA DE SUPERVISORES

Conforme dito no final da seção 4.2, a utilidade do mecanismo de controle proposto no modelo RW depende de se determinar sob que condições existe um supervisor que realize uma dada especificação e, caso exista, como encontrá-lo. O objetivo desta seção é o de apresentar os teoremas de existência de supervisores para problemas formulados em termos de linguagens geradas e em termos de linguagens marcadas. A apresentação difere daquela encontrada em [RW87], tendo sido as demonstrações refeitas com o intuito de destacar alguns resultados que serão referenciados a partir do capítulo 5. Além disso, o autor acredita que a apresentação que segue é menos complexa do que a original.

A proposição seguinte permite chamar a atenção para a controlabilidade da linguagem gerada de qualquer sistema sob supervisão.

Prop. 4.2: Seja $\langle S, \Phi \rangle$ um supervisor completo para a planta $\langle G, \Gamma \rangle$. Então $L(S/G)$ é prefixo-fechada e $L(G)$ -controlável.

Dem.: Seja $\langle S, \Phi \rangle$, com $S = \langle \Sigma, X, \xi, x_0, X_m \rangle$, um supervisor completo para a planta $\langle G, \Gamma \rangle$. O prefixo-fechamento de $L(S/G)$ decorre da sua definição (v. equação 4.4).

Para verificar que $L(S/G)$ é $L(G)$ -controlável, seja $s\sigma$ uma palavra tal que:

$s\sigma \in L(S/G)\Sigma_u \cap L(G)$,	isto é,
$s\sigma: s \in L(S/G) \wedge \sigma \in \Sigma_u \wedge s\sigma \in L(G)$.	Por ser σ não controlável,
$s\sigma: s \in L(S/G) \wedge \sigma \in \Phi(\xi(s, x_0)) \wedge s\sigma \in L(G)$	e, sendo $\langle S, \Phi \rangle$ completo,
$s\sigma \in L(S/G)$,	para toda palavra $s\sigma$, ou seja,
$L(S/G)\Sigma_u \cap L(G) \subseteq L(S/G)$. ♦	

O teorema seguinte estabelece as condições de existência de supervisores para problemas formulados em termos de linguagens geradas.

Teo. 4.1: Dados um gerador G tal que $L(G)$ represente o comportamento fisicamente possível da planta $\langle G, \Gamma \rangle$ e uma linguagem-alvo $K \subseteq L(G)$, existe um supervisor completo $\langle S, \Phi \rangle$ tal que $L(S/G) = K$ sse K for prefixo-fechada e $L(G)$ -controlável.

Dem.:

Necessidade (somente se):

Seja $\langle S, \Phi \rangle$, onde $S = \langle \Sigma, X, \xi, x_0, X_m \rangle$, um supervisor completo para $\langle G, \Gamma \rangle$ tal que $L(S/G) = K$. Então, pela proposição 4.2, K é prefixo-fechada e $L(G)$ -controlável.

Suficiência (se):

Seja K prefixo-fechada e $L(G)$ -controlável. Então, por ser K prefixo-fechada, é possível construir um gerador $S = \langle \Sigma, X, \xi, x_0, X_m \rangle$ tal que

$$L(S) = K \quad (4.12).$$

Cabe mostrar em primeiro lugar que é de fato possível definir o mapa de controle $\Phi: X \rightarrow \Gamma$. A definição 4.3 exige que o mapa Φ seja definido para todo $x \in X$; isto significa que, para todo estado $x \in X$, deve ser possível determinar univocamente, para todo evento σ fisicamente possível neste estado, se σ deve ser inibido ou habilitado. Considere para tanto os conjuntos:

$$\Sigma_x^0 = \{\sigma: \exists s \in K \wedge \xi(s, x_0) = x \wedge s\sigma \in L(G) \wedge s\sigma \notin K\} \quad (4.13)$$

e

$$\Sigma_x^1 = \{\sigma: \exists s \in K \wedge \xi(s, x_0) = x \wedge s\sigma \in K\} \quad (4.14),$$

definidos para todo $x \in X$. Σ_x^0 é o conjunto dos eventos que, por serem fisicamente possíveis após a ocorrência de s e tais que $s\sigma \notin K$, precisam ser inibidos quando S se encontrar no estado x , de modo que não podem pertencer a $\Phi(x)$. Por outro lado, Σ_x^1 é o conjunto dos eventos que, por formarem continuações da palavra s tais que $s\sigma \in K$, precisam estar habilitados quando S se encontrar no estado x , de modo que devem pertencer a $\Phi(x)$. Os eventos pertencentes a $\Sigma - \{\Sigma_x^0 \cup \Sigma_x^1\}$ são aqueles para os quais é indiferente se estão habilitados ou inibidos, pois não são fisicamente possíveis quando S se encontra no estado x .

O que se disse acima exige que

$$\Sigma_x^0 \cap \Sigma_x^1 = \emptyset \quad (4.15).$$

Para verificar esta igualdade, suponha que $\exists \sigma: \sigma \in \Sigma_x^0 \cap \Sigma_x^1$, com

$$s^0: s^0 \in K \wedge \xi(s^0, x_0) = x \wedge s^0\sigma \notin K \text{ e}$$

$$s^1: s^1 \in K \wedge \xi(s^1, x_0) = x \wedge s^1\sigma \in K.$$

Então, dado que $L(S) = K$,

$$\xi(s^0\sigma, x_0) = \xi(\sigma, \xi(s^0, x_0)) = \xi(\sigma, x) \text{ não pode estar definida, pois } s^0\sigma \notin K, \text{ mas}$$

$\xi(s^1\sigma, x_0) = \xi(\sigma, \xi(s^1, x_0)) = \xi(\sigma, x)$ tem que estar definida, pois $s^1\sigma \in K$.

A contradição obtida prova a equação 4.15.

Além disso, a definição 4.1 exige que $\Sigma_x^0 \subseteq \Sigma_c$, o que é garantido pela controlabilidade. Desta forma, é possível construir o mapa Φ tomando

$$\Phi(x) = \gamma : \gamma \in \Gamma \wedge \gamma \supseteq \Sigma_x^1 \wedge \gamma \cap \Sigma_x^0 = \emptyset \quad (4.16).$$

A equação acima não define, em geral, um valor único para γ , permitindo escolher se os eventos não fisicamente possíveis devem ser habilitados ou não.

Em segundo lugar, cabe mostrar que $L(S/G) = K$. A igualdade 4.12 garante que

$$L(S/G) \subseteq K \quad (4.17).$$

Não obstante, é necessário verificar que a ação inibidora do mapa Φ não destrói a inclusão reversa. O seguinte raciocínio indutivo, utilizando o comprimento das palavras $s \in L(G)$, mostra que a igualdade continua válida sob supervisão: se $|s|=1$, isto é, se $s = \sigma$ para algum $\sigma \in \Sigma$, então

$\sigma \in K \Rightarrow \sigma \in \Sigma_x^1 \Rightarrow \sigma \in \Phi(x_0) \Rightarrow \sigma \in L(S/G)$, pelas equações 4.14 e 4.16 e pela definição 4.4, respectivamente.

Para uma linguagem L qualquer, seja $L^{(j)} = \{s : s \in L \wedge |s|=j\}$, $j = 0, 1, \dots, \infty$. Note que

$\bigcup_{j=0}^{\infty} L^{(j)} = L$. Nestes termos, a linha acima pode ser reescrita como

$$L^{(1)}(S/G) \supseteq K^{(1)} \quad \text{e, considerando a equação 4.17,}$$

$$L^{(1)}(S/G) = K^{(1)}.$$

Seja $L^{(i)}(S/G) = K^{(i)}$, $i = 0, 1, \dots, j$, e seja $s \in L^{(j)}(S/G)$, de modo que $x = \xi(s, x_0)$ é definido. Considere a palavra $s\sigma \in L(G)$. Então, como antes,

$$s\sigma \in K \Rightarrow \sigma \in \Sigma_x^1 \Rightarrow \sigma \in \Phi(x) \Rightarrow s\sigma \in L(S/G). \quad \text{Logo,}$$

$$L^{(j+1)}(S/G) \supseteq K^{(j+1)} \quad \text{e, novamente pela equação 4.17,}$$

$$L^{(j+1)}(S/G) = K^{(j+1)}, \quad \text{donde, por indução,}$$

$$L(S/G) = K.$$

Resta mostrar que $\langle S, \Phi \rangle$ é um supervisor completo. Para tanto, sejam $s \in L(S/G)$, $s\sigma \in L(G)$ e $\sigma \in \Phi(x)$, com $x = \xi(s, x_0)$. Então, para que $\langle S, \Phi \rangle$ não fosse completo, seria necessário que $s\sigma \notin K$ e, com isso, $\sigma \in \Sigma_x^0$, implicando $\sigma \notin \Phi(x)$, o que é contraditório. Logo, $s\sigma \in K$ e $\langle S, \Phi \rangle$ é completo. ♦

O teorema seguinte estabelece as condições de existência de supervisores para problemas formulados em termos de linguagens marcadas. O item (i) é inédito na literatura pesquisada, enquanto que os itens (ii) e (iii) correspondem ao teorema 6.1 em [RW87].

Teo. 4.2:

(i) Dados um gerador G tal que $L_m(G)$ represente as tarefas que podem ser completadas pela planta $\langle G, \Gamma \rangle$ na ausência de qualquer ação de controle e uma linguagem-alvo $K \subseteq L_m(G)$, existe um supervisor completo $\langle S, \Phi \rangle$ tal que $L_c(S/G) = K$ sse K for $L_m(G)$ -fechada e existir uma linguagem prefixo-fechada e $L(G)$ -controlável K' tal que $K' \cap L_m(G) = K$.

(ii) Existe um supervisor não bloqueante $\langle S, \Phi \rangle$ tal que $L_c(S/G) = K$ sse K for $L_m(G)$ -fechada e $L(G)$ -controlável.

(iii) Nas condições do item (ii), o supervisor $\langle S, \Phi \rangle$ será próprio se o gerador $S = \langle \Sigma, X, \xi, x_0, X_m \rangle$ for tal que $X_m = X$.

Dem.:**Item (i):****Necessidade (somente se):**

Seja $\langle S, \Phi \rangle$, com $S = \langle \Sigma, X, \xi, x_0, X_m \rangle$, um supervisor completo para $\langle G, \Gamma \rangle$ tal que $L_c(S/G) = K$. Então, pela proposição 4.1, K é $L_m(G)$ -fechada.

Para demonstrar a existência de K' suponha que $K' = L(S/G)$. Então, pelo teorema 4.1, K' é prefixo-fechada e $L(G)$ -controlável. Além disso, pela definição 4.5,

$$L(S/G) \cap L_m(G) = L_c(S/G). \text{ Pondo } L(S/G) = K' \text{ e } L_c(S/G) = K, \text{ vem:}$$

$$K' \cap L_m(G) = K, \text{ provando que } L(S/G) \text{ é uma escolha válida para } K'.$$

Suficiência (se):

Considerando-se que K' é prefixo-fechada e $L(G)$ -controlável, o teorema 4.1 permite afirmar que existe um supervisor $\langle S, \Phi \rangle$ tal que

$$L(S/G) = K' \tag{4.18}.$$

Nestas condições, a linguagem controlada do sistema sob supervisão será:

$$L_c(S/G) = L(S/G) \cap L_m(G) = K' \cap L_m(G) = K.$$

Item (ii):**Necessidade(somente se):**

Suponha que $\langle S, \Phi \rangle$ é não bloqueante, de forma que

$$L(S/G) = \overline{L_c(S/G)} = \overline{K}.$$

Então, pelo teorema 4.1, \overline{K} é $L(G)$ -controlável e, pela definição 4.12, K é $L(G)$ -controlável.

O fechamento de K em relação a $L_m(G)$ decorre diretamente da proposição 4.1.

Suficiência (se):

Suponha que K é $L_m(G)$ -fechada e $L(G)$ -controlável. Então, pela definição 4.12, \bar{K} é $L(G)$ -controlável e, pelo teorema 4.1, existe um supervisor completo $\langle S, \Phi \rangle$ tal que $L(S/G) = \bar{K}$. Este supervisor satisfaz as exigências do teorema pois, sendo K $L_m(G)$ -fechada,

$$\bar{K} \cap L_m(G) = K$$

$$L(S/G) \cap L_m(G) = K$$

$$L_c(S/G) = K. \quad \text{Além disso, o supervisor é não bloqueante, pois}$$

$$\overline{L_c(S/G)} = \bar{K} \quad \text{e assim}$$

$$\overline{L(S/G)} = L(S/G).$$

Item (iii):

Seja $\langle S, \Phi \rangle$ um supervisor não bloqueante para $\langle G, \Gamma \rangle$. Então, se $X_m = X$,

$$L_m(S/G) = L_c(S/G), \quad \text{o que permite escrever}$$

$$\overline{L_m(S/G)} = \overline{L_c(S/G)} = L(S/G), \quad \text{de modo que } \langle S, \Phi \rangle \text{ é próprio.} \quad \blacklozenge$$

Cumpra observar que o item (i) do teorema 4.2 mostra que um problema formulado em termos da linguagem marcada $K \subseteq L_m(G)$ pode ter solução mesmo se K não for $L(G)$ -controlável. Neste caso, porém, como mostra o item (ii) do mesmo teorema, o supervisor será bloqueante.

Os teoremas 4.1 e 4.2 permitem determinar procedimentos para a obtenção de supervisores.

Para problemas formulados em termos de linguagens geradas, o teorema 4.1 mostra que, satisfeitas as condições necessárias e suficientes para a existência do supervisor $\langle S, \Phi \rangle$, o gerador S e o mapa Φ podem ser obtidos de acordo com as equações 4.12 e 4.16, respectivamente.

Para problemas formulados em termos de linguagens marcadas, existem dois casos a considerar:

- i. se K não é $L(G)$ -controlável e estão satisfeitas as condições do item (i) do teorema 4.2, então o problema tem como solução um supervisor bloqueante que pode ser obtido resolvendo-se o problema em termos de linguagens geradas correspondente à equação 4.18, isto é, encontrando-se um supervisor tal que $L(S/G) = K'$. Note que os teoremas apresentados não fornecem um meio de obter K' ;

- ii. se K satisfaz as condições do item (ii) do teorema 4.2, então o supervisor pode ser obtido encontrando-se um supervisor tal que $L(S/G) = \bar{K}$ pois, neste caso, $L_c(S/G) = \bar{K} \cap L_m(G) = K$.

Embora de utilidade indiscutível, este resultado não pode ser empregado nos casos em que a linguagem-alvo K deixar de satisfazer as condições estabelecidas para a existência de um supervisor. Esta situação não é incomum na prática, uma vez que a definição do que se deseja não tem necessariamente uma relação com o que pode ser realizado. A seção seguinte mostra que, nesses casos, existe uma linguagem contida em K que é a melhor aproximação implementável da linguagem-alvo.

4.8 SÍNTESE DE SUPERVISORES

Esta seção apresenta um dos resultados mais marcantes da Teoria RW, segundo o qual toda linguagem-alvo K que não satisfaça as condições de $L_m(G)$ -fechamento e de $L(G)$ -controlabilidade contém uma sublinguagem K' , a qual satisfaz as referidas condições de maneira minimamente restritiva. Desta forma, se K' for adequada aos propósitos do problema, isto é, se $K' \supseteq A$, onde A é uma linguagem que representa o comportamento mais restrito que pode ser tolerado, K' pode ser utilizada para substituir a linguagem-alvo original. O problema tem então como solução um supervisor que implementa K' .

Pode-se então enunciar o seguinte problema abstrato de síntese de supervisores:²

Problema de Controle Supervisório I (PCSI): Dadas uma planta $\langle G, \Gamma \rangle$, uma linguagem-alvo marcada $E \subseteq L_m(G)$ e uma mínima linguagem admissível $A \subseteq E$, encontrar um supervisor próprio $\langle S, \Phi \rangle$ tal que

$$A \subseteq L_c(S/G) \subseteq E.$$

Nos casos em que E é $L_m(G)$ -fechada e $L(G)$ -controlável, existe um supervisor tal que $L_c(S/G) = E$, o que significa que o problema tem uma solução não restritiva. O restante desta seção trata da existência de uma solução minimamente restritiva para os casos em que E deixe de satisfazer essas condições.

Considere o conjunto de todas as sublinguagens $L(G)$ -controláveis de E , dado por:

$$C(E) = \{K' : K' \subseteq E \wedge K' \Sigma_u \cap L(G) \subseteq K'\} \quad (4.19)$$

e o conjunto de todas as sublinguagens $L_m(G)$ -fechadas de E , dado por:

$$F(E) = \{K' : K' \subseteq E \wedge \overline{K'} \cap L_m(G) = K'\} \quad (4.20).$$

Para esclarecer a terminologia empregada a seguir, considere a seguinte definição:

²Este problema corresponde ao "Supervisory Control Problem" apresentado em [RW87], ali designado por SCP.

Def. 4.13: Dados um conjunto W qualquer e uma operação Z sobre seus elementos, W é dito *fechado sob a operação Z* sse, para todo par (w_1, w_2) ,

$$w_1, w_2 \in W \Rightarrow w_1 Z w_2 \in W.$$

Teo. 4.3: Seja $H(K) = \{K' : K' \subseteq K \wedge [\forall K'_1, K'_2 \in H(K)] K'_1 \cup K'_2 \in H(K)\}$ um conjunto não vazio de subconjuntos de K , fechado sob a operação de união. Então existe em $H(K)$ o elemento supremo

$$\sup H(K) = \bigcup \{K : K \in H(K)\}.$$

Dem.: Para demonstrar o teorema é necessário mostrar que $\sup H(K)$ contém qualquer elemento de $H(K)$ e que $\sup H(K) \in H(K)$. O primeiro fato decorre diretamente da definição de união, e o segundo da hipótese de ser $H(K)$ fechado sob essa operação. Além disso, $\sup H(K)$ está sempre definido, pois $H(K)$ é, por hipótese, não vazio. ♦

Considere ainda que, dadas duas linguagens quaisquer $L, M \subseteq \Sigma^*$, as operações de união de conjuntos e de prefixo-fechamento comutam, isto é, que $\overline{L \cup M} = \overline{L} \cup \overline{M}$, e que este raciocínio pode ser estendido a um número arbitrário de linguagens sobre Σ^* .

Com isto, pode-se apresentar a seguinte proposição:

Prop. 4.3: $C(E)$ e $F(E)$ são não vazios e fechados sob a operação de união.

Dem.: Dado que a linguagem vazia \emptyset é $L(G)$ -controlável e $L_m(G)$ -fechada, pode-se afirmar que $\emptyset \in C(E)$ e $\emptyset \in F(E)$, de modo que estes são conjuntos não vazios.

Para verificar o fechamento de $C(E)$ sob união, considere um conjunto de índices A para enumerar os membros desse conjunto, de modo que

$$K_\alpha \in C(E), \quad \alpha \in A.$$

Sendo os K_α controláveis, vale:

$$K_\alpha \Sigma_u \cap L(G) \subseteq K_\alpha, \quad \alpha \in A.$$

Então

$$\begin{aligned} \overline{\left(\bigcup_\alpha K_\alpha \right) \Sigma_u \cap L(G)} &= \overline{\left(\bigcup_\alpha \overline{K_\alpha} \right) \Sigma_u \cap L(G)} \\ &= \bigcup_\alpha \overline{K_\alpha \Sigma_u \cap L(G)} \\ &= \bigcup_\alpha [\overline{K_\alpha \Sigma_u \cap L(G)}] \\ &\subseteq \bigcup_\alpha \overline{K_\alpha}. \end{aligned}$$

Logo, a união de quaisquer elementos de $C(E)$ é $L(G)$ -controlável e pertence a $C(E)$.

Para verificar o fechamento de $F(E)$ sob união, considere um conjunto de índices B para enumerar os membros desse conjunto, de modo que

$$K_\beta \in F(E), \quad \beta \in B.$$

Sendo os K_β $L_m(G)$ -fechados, vale:

$$K_\beta = \overline{K_\beta} \cap L_m(G).$$

Então

$$\overline{\left(\bigcup_\beta K_\beta\right)} \cap L_m(G) = \left(\bigcup_\beta \overline{K_\beta}\right) \cap L_m(G) = \bigcup_\beta [\overline{K_\beta} \cap L_m(G)] = \bigcup_\beta K_\beta.$$

Logo, a união de quaisquer elementos de $F(E)$ é $L_m(G)$ -fechada e pertence a $F(E)$. \blacklozenge

Corolário 4.1: Existem em $C(E)$ e $F(E)$, respectivamente, os elementos supremos

$$\sup C(E) = \bigcup \{K : K \in C(E)\} \quad e$$

$$\sup F(E) = \bigcup \{K : K \in F(E)\}.$$

Dem.: Pelo teorema 4.3. \blacklozenge

Os elementos $\sup C(E)$ e $\sup F(E)$ são denominados respectivamente de *máxima sublinguagem $L(G)$ -controlável* e de *máxima sublinguagem $L_m(G)$ -fechada de E* .

Além disso, vale a seguinte proposição:

Prop. 4.4: O conjunto $C(E) \cap F(E)$ é não vazio e fechado sob a operação de união.

Dem.: $C(E) \cap F(E)$ é não vazio, pois $\emptyset \in C(E) \cap F(E)$. Para ver o fechamento sob união, considere duas linguagens quaisquer $K_1, K_2 \in C(E) \cap F(E)$. Então

$$K_1 \in C(E) \wedge K_1 \in F(E) \wedge K_2 \in C(E) \wedge K_2 \in F(E).$$

Sendo $C(E)$ e $F(E)$ fechados sob união, vem:

$$K_1 \cup K_2 \in C(E) \wedge K_1 \cup K_2 \in F(E) \text{ e assim}$$

$$K_1 \cup K_2 \in C(E) \cap F(E). \quad \blacklozenge$$

Corolário 4.2: Existe em $C(E) \cap F(E)$ o elemento supremo

$$\sup CF(E) = \bigcup \{K : K \in C(E) \cap F(E)\}.$$

Dem.: Pelo teorema 4.3. \blacklozenge

O elemento $\sup CF(E)$ é denominado de *máxima sublinguagem $L(G)$ -controlável e $L_m(G)$ -fechada de E* .

Em vista do corolário 4.2, pode-se considerar demonstrado o seguinte teorema, principal resultado do presente capítulo:

Teo. 4.4: PCSI tem solução se e somente se $\sup CF(E) \supseteq A$, sendo $\sup CF(E)$ a solução minimamente restritiva.

Considere ainda o caso particular em que todos os estados do gerador da planta são marcados. Tem-se então:

$$L_m(G) = \overline{L_m(G)} = L(G) \quad (4.21)$$

e, conseqüentemente, pela equação 4.3 e pela definição 4.5,

$$L_c(S/G) = L(S/G) \quad (4.22).$$

Além disso, toda sublinguagem prefixo-fechada $E \subseteq L_m(G)$ é $L_m(G)$ -fechada, de modo que

$$[\forall E: E \subseteq L_m(G)] \quad E = \sup F(E) \quad (4.23)$$

e

$$\sup CF(E) = \sup \hat{C}(E) \quad (4.24).$$

Com isso, o problema PCSI tem a seguinte versão para linguagens geradas:

PCSI para Linguagens Geradas (PCSIG): Dadas uma planta $\langle G, \Gamma \rangle$ tal que $L_m(G) = L(G)$, uma linguagem-alvo prefixo-fechada $E \subseteq L(G)$ e uma mínima linguagem admissível $A \subseteq E$, encontrar um supervisor próprio $\langle S, \Phi \rangle$ tal que

$$A \subseteq L(S/G) \subseteq E.$$

Sua solução é conseqüência imediata do teorema 4.2 e das equações 4.14 a 4.17:

Teo. 4.3: PCSIG tem solução sse $\sup C(E) \supseteq A$, sendo $\sup C(E)$ a solução minimamente restritiva.

Cabe observar ainda que o fato de todos os estados do gerador da planta serem marcados significa, em princípio, que qualquer supervisor obtido será não bloqueante. No entanto, o problema PCSIG tem sido utilizado na literatura em casos onde nem todos os estados da planta são marcados a priori. Neste caso, o que se faz é marcar todos os estados para poder resolver o problema. Perde-se com isso a informação representada pela marcação e, como conseqüência, o supervisor encontrado pode bloquear. Este problema é resolvido no capítulo 5, com a introdução da definição de *L-compatibilidade*, a qual permite utilizar essa informação.

4.9 CONCLUSÃO

O capítulo contém uma apresentação do modelo RW básico. Os principais resultados são os que dizem respeito à existência de supervisores e à existência da máxima sublinguagem $L_m(G)$ -fechada e $L(G)$ -controlável de uma dada linguagem, que permitem resolver PCSI.

Não se aborda neste trabalho o cálculo de soluções para PCSI. A obtenção da máxima sublinguagem $L(G)$ -controlável de uma dada linguagem é discutida em [WR87], [Rudi88], [LVW88], [BGK⁺90] e [KGM91]. O algoritmo apresentado em [WR87] inclui a restrição adicional de que o gerador resultante seja trim (isto é, acessível e coacessível), o que é interessante para a síntese de supervisores não bloqueantes.

Quanto ao cálculo da máxima sublinguagem $L_m(G)$ -fechada de uma dada linguagem, não foram encontrados na literatura pesquisada algoritmos ou fórmulas para $\text{sup} F(\cdot)$ ou $\text{sup} CF(\cdot)$. Em razão disso, os problemas que podem ser resolvidos atualmente se limitam aos casos em que a linguagem-alvo é $L_m(G)$ -fechada.

CAPÍTULO 5

CONTRIBUIÇÕES AO MODELO RW

O capítulo 4 apresentou o problema abstrato de síntese PCSI, formulado em termos de linguagens marcadas. O teorema 4.2 estabelece as condições de existência de supervisores próprios para problemas empregando esta formulação, de modo que se pode garantir a ausência de bloqueio no sistema sob supervisão.

O mesmo não acontece para problemas formulados em termos de linguagens geradas. O teorema 4.1 estabelece apenas as condições de existência para supervisores completos, e não existe um critério para avaliar a ausência de bloqueio a partir da linguagem-alvo $K \subseteq L(G)$. O presente capítulo apresenta um novo problema abstrato de síntese de supervisores, formulado em termos de linguagens geradas. Este difere do caso particular estudado no capítulo 4 por não exigir que $L_m(G) = L(G)$, isto é, por não exigir que todos os estados da planta sejam marcados, ao mesmo tempo que seu objetivo é encontrar um supervisor próprio.

Introduz-se uma nova definição na teoria, a de *L-compatibilidade*. Esta permite estabelecer um critério para a ausência de bloqueio, que utiliza a linguagem gerada pelo sistema sob supervisão e a linguagem marcada da planta. Resulta daí que a formulação em termos de linguagens geradas passa a trabalhar com a informação representada pelos estados marcados, o que a eleva ao mesmo nível da formulação em termos de linguagens marcadas. Além disso, estabelecem-se as condições de existência de supervisores próprios para problemas formulados em termos de linguagens geradas, as quais permitem resolver o novo problema de síntese proposto. Estuda-se ainda a equivalência entre as formulações em termos de linguagens marcadas e em termos de linguagens geradas.

5.1 PCSII

O novo problema de síntese proposto é o seguinte:

Problema de Controle Supervisório II (PCSII): Dadas uma planta $\langle G, \Gamma \rangle$, uma linguagem-alvo prefixo-fechada $E \subseteq L(G)$ e uma mínima linguagem admissível $A \subseteq E$, encontrar um supervisor próprio $\langle S, \Phi \rangle$ tal que

$$A \subseteq L(S/G) \subseteq E.$$

As próximas seções tratam das condições de existência de supervisores próprios para problemas formulados em termos de linguagens geradas e da solução para o problema acima.

5.2 L-COMPATIBILIDADE

Considere a seguinte definição:

Def. 5.1: Dadas duas linguagens quaisquer $K, L \subseteq \Sigma^*$, K é dita *L-compatível* sse $K = \overline{K \cap L}$.

Embora a definição 5.1 seja aplicável a quaisquer linguagens K, L , ela é de interesse quando envolve a linguagem-alvo $K \subseteq L(G)$ e a linguagem marcada da planta, $L_m(G)$. Se K é $L_m(G)$ -compatível, isto é, se $K = \overline{K \cap L_m(G)}$, então (i) K é prefixo-fechada e (ii) $K \subseteq \overline{L_m(G)}$, o que garante que toda palavra de K é um prefixo de alguma palavra em $L_m(G)$. Este fato sugere o seguinte critério para a ausência de bloqueio:

Prop. 5.1: Um supervisor $\langle S, \Phi \rangle$ para uma planta $\langle G, \Gamma \rangle$ é não bloqueante sse $L(S/G)$ for $L_m(G)$ -compatível.

Dem.:

Necessidade (somente se):

Seja $\langle S, \Phi \rangle$ não bloqueante, de modo que

$$\overline{L_c(S/G)} = L(S/G). \quad \text{Pela definição 4.5,}$$

$$\overline{L_c(S/G)} = \overline{L(S/G) \cap L_m(G)}, \quad \text{donde}$$

$$L(S/G) = \overline{L(S/G) \cap L_m(G)}.$$

Suficiência (se):

Seja $L(S/G)$ $L_m(G)$ -compatível. Então

$$L(S/G) = \overline{\overline{L(S/G) \cap L_m(G)}}. \quad \text{Pela definição 4.5 é imediato que}$$

$$L(S/G) = \overline{L_c(S/G)}. \quad \blacklozenge$$

5.3 EXISTÊNCIA DE SUPERVISORES - UM NOVO TEOREMA

O seguinte teorema estabelece as condições de existência de supervisores próprios para problemas formulados em termos de linguagens geradas:

Teo. 5.1:

(i) Dados um gerador G tal que $L(G)$ represente o comportamento fisicamente possível da planta $\langle G, \Gamma \rangle$ e uma linguagem-alvo $K \subseteq L(G)$, existe um supervisor não bloqueante $\langle S, \Phi \rangle$ tal que $L(S/G) = K$ sse K for $L_m(G)$ -compatível e $L(G)$ -controlável.

(ii) Nas condições do item (i), o supervisor $\langle S, \Phi \rangle$ será próprio se o gerador $S = \langle \Sigma, X, \xi, x_0, X_m \rangle$ for tal que $X_m = X$.

Dem.:

Item (i):

Pelo teorema 4.1, é necessário e suficiente que K seja prefixo-fechada e $L(G)$ -controlável para que se possa construir um supervisor $\langle S, \Phi \rangle$ tal que $L(S/G) = K$. Portanto, basta demonstrar que $\langle S, \Phi \rangle$ é não bloqueante sse K for $L_m(G)$ -compatível, resultado que decorre da proposição 5.1.

Item (ii):

Seja $\langle S, \Phi \rangle$ um supervisor não bloqueante para $\langle G, \Gamma \rangle$. Então, se $X_m = X$,

$$L_m(S/G) = L_c(S/G), \quad \text{o que permite escrever}$$

$$\overline{L_m(S/G)} = \overline{L_c(S/G)} = L(S/G), \quad \text{de modo que } \langle S, \Phi \rangle \text{ é próprio.} \quad \blacklozenge$$

5.4 SOLUÇÃO PARA PCSII

Nos casos em que a linguagem-alvo E satisfaz as condições do teorema 5.1, PCSII tem uma solução não restritiva e a linguagem do sistema sob supervisão corresponde ao comportamento desejado. O restante desta seção trata da existência de soluções minimamente restritivas para os casos em que a linguagem-alvo não seja $L(G)$ -controlável e $L_m(G)$ -compatível, o que permite apresentar, por fim, a solução abstrata para o problema.

Considere o conjunto de todas as sublinguagens $L(G)$ -controláveis de E , dado por:

$$C(E) = \{K' : K' \subseteq E \wedge K' \Sigma_u \cap L(G) \subseteq K'\} \quad (5.1),$$

e o conjunto de todas as sublinguagens $L_m(G)$ -compatíveis de E , dado por:

$$T(E) = \{K' : K' \subseteq E \wedge K' = \overline{K' \cap L_m(G)}\} \quad (5.2).$$

Prop. 5.2: $T(E)$ é um conjunto não vazio e fechado sob a operação de união.

Dem.: Dado que a linguagem vazia \emptyset é $L_m(G)$ -compatível, pode-se afirmar que $\emptyset \in T(E)$, de modo que este conjunto é não vazio.

Para verificar o fechamento de $T(E)$ sob união, considere um conjunto de índices Λ para enumerar os membros de $T(E)$, de modo que

$$K_\lambda \in T(E), \quad \lambda \in \Lambda. \quad \text{Sendo os } K_\lambda \text{ } L_m(G)\text{-compatíveis, vale:}$$

$$\overline{K_\lambda \cap L_m(G)} = K_\lambda, \quad \lambda \in \Lambda. \quad \text{Então}$$

$$\overline{\left(\bigcup_\lambda K_\lambda \right) \cap L_m(G)} = \bigcup_\lambda \overline{[K_\lambda \cap L_m(G)]} = \bigcup_\lambda \overline{[K_\lambda \cap L_m(G)]} = \bigcup_\lambda K_\lambda.$$

Logo, a união de quaisquer elementos de $T(E)$ é $L_m(G)$ -compatível e pertence a $T(E)$. \blacklozenge

Corolário 5.1: Existe em $T(E)$ o elemento supremo $\sup T(E) = \bigcup \{K: K \in T(E)\}$.

Dem.: Pelo teorema 4.3. ♦

O elemento $\sup T(E)$ é denominado *máxima sublinguagem $L_m(G)$ -compatível de E* .

Prop. 5.3: O conjunto $C(E) \cap T(E)$ é não vazio e fechado sob a operação de união.

Dem.: A demonstração é análoga à da proposição 4.4. ♦

Corolário 5.2: Existe em $C(E) \cap T(E)$ o elemento supremo $\sup CT(E) = \bigcup \{K: K \in C(E) \cap T(E)\}$.

Dem.: Pelo teorema 4.3. ♦

O elemento $\sup CT(E)$ é denominado de *máxima sublinguagem $L(G)$ -controlável e $L_m(G)$ -compatível de E* .

Em vista do corolário 5.2, pode-se considerar demonstrado o seguinte teorema:

Teo. 5.2: PCSII tem solução sse $\sup CT(E) \supseteq A$, sendo $\sup CT(E)$ a solução minimamente restritiva.

5.5 EQUIVALÊNCIA ENTRE AS FORMULAÇÕES APRESENTADAS

Os resultados estabelecidos acima permitem estabelecer uma dualidade entre as formulações de problemas em termos de linguagens marcadas e em termos de linguagens geradas. O seguinte teorema estabelece a equivalência entre as duas formulações no caso em que as condições de existência dos supervisores estão satisfeitas:

Teo. 5.3:

(i) Dados um gerador G tal que $L_m(G)$ represente as tarefas que podem ser completadas pela planta $\langle G, \Gamma \rangle$ e uma linguagem-alvo $L_m(G)$ -fechada e $L(G)$ -controlável $K \subseteq L_m(G)$, $\langle S, \Phi \rangle$ é um supervisor tal que $L_c(S/G) = K$ sse $L(S/G) = \bar{K}$.

(ii) Dados um gerador G tal que $L(G)$ represente o comportamento fisicamente possível da planta $\langle G, \Gamma \rangle$ e uma linguagem-alvo $L_m(G)$ -compatível e $L(G)$ -controlável $K \subseteq L(G)$, $\langle S, \Phi \rangle$ é um supervisor tal que $L(S/G) = K$ sse $L_c(S/G) = K \cap L_m(G)$.

Dem.:

Item (i):

Necessidade (somente se):

Seja $\langle \hat{S}, \Phi \rangle$ um supervisor tal que $L_c(\hat{S}/G) = K$.

Então, pelo item (ii) do teorema 4.2, $\langle S, \Phi \rangle$ é não bloqueante, de modo que

$$L(S/G) = \overline{L_c(S/G)} = \overline{K}.$$

Suficiência (se):

Seja $\langle S, \Phi \rangle$ um supervisor tal que $L(S/G) = \overline{K}$. Então:

$$L(S/G) \cap L_m(G) = \overline{K} \cap L_m(G). \quad \text{Pelas definições 4.5 e 4.11,}$$

$$L_c(S/G) = K.$$

Item (ii):

Necessidade (somente se):

Seja $\langle S, \Phi \rangle$ um supervisor tal que $L(S/G) = K$. Então, pela definição 4.5,

$$L_c(S/G) = K \cap L_m(G).$$

Suficiência (se):

Seja $\langle S, \Phi \rangle$ seja um supervisor tal que $L_c(S/G) = K \cap L_m(G)$. Então

$$\overline{L_c(S/G)} = \overline{K \cap L_m(G)}.$$

Pelo teorema 5.1, $\langle S, \Phi \rangle$ é não bloqueante e, pelas definições 4.8 e 5.1,

$$L(S/G) = K. \quad \blacklozenge$$

É natural perguntar em seguida se esta equivalência continua válida se as condições de existência de supervisores deixarem de ser satisfeitas, isto é:

(i) Dado um problema do tipo PCSI com uma linguagem-alvo $E \subseteq L_m(G)$, é possível construir um supervisor tal que $L_c(S/G) = \sup CF(E)$ resolvendo-se o problema PCSII para alguma linguagem prefixo-fechada contida em $L(G)$?

(ii) Dado um problema do tipo PCSII com uma linguagem-alvo $E \subseteq L(G)$, é possível construir um supervisor tal que $L(S/G) = \sup CT(E)$ resolvendo-se o problema PCSI para alguma linguagem marcada contida em $L_m(G)$?

A resposta a ambas as perguntas é sim. As proposições 5.4 a 5.9 servirão de suporte à demonstração do teorema que estabelece esta dualidade.

Prop. 5.4: Dada uma linguagem prefixo-fechada $K \subseteq L(G)$, a linguagem $K \cap L_m(G)$ é L_m -fechada.

Dem.:

(\subseteq):

$K \cap L_m(G) \subseteq \overline{K \cap L_m(G)}$ e $K \cap L_m(G) \subseteq L_m(G)$. Logo,

$$K \cap L_m(G) \subseteq \overline{K \cap L_m(G)} \cap L_m(G).$$

(\supseteq):

$$K = \overline{K}$$

$$K \supseteq \overline{K \cap L_m(G)}$$

$$K \cap L_m(G) \supseteq \overline{K \cap L_m(G)} \cap L_m(G). \quad \blacklozenge$$

Prop. 5.5: Dada uma linguagem $E \subseteq L_m(G)$, se E é $L_m(G)$ -fechada, então $\sup C(E)$ também o é.

Dem.: Seja K uma sublinguagem $L(G)$ -controlável arbitrária (não necessariamente $L_m(G)$ -fechada) de E , de maneira que $K \in C(E)$. Então, como $K \subseteq \overline{K}$ e $K \subseteq E \subseteq L_m(G)$, vale

$$\overline{K} \cap L_m(G) \supseteq K.$$

Além disso,

$$\overline{K} \subseteq \overline{E}$$

$$\overline{K} \cap L_m(G) \subseteq \overline{E} \cap L_m(G)$$

e, por ser E $L_m(G)$ -fechada,

$$\overline{K} \cap L_m(G) \subseteq E.$$

Pela proposição 5.4, $\overline{K} \cap L_m(G)$ é $L_m(G)$ -fechada.

Esta linguagem é também $L(G)$ -controlável, conforme mostrado a seguir:

$$\overline{[\overline{K} \cap L_m(G)] \Sigma_u \cap L(G)} \subseteq [\overline{K} \cap L_m(G)] \Sigma_u \cap L(G)$$

$$\subseteq \overline{K} \Sigma_u \cap L(G)$$

$$\subseteq \overline{K}$$

(devido à $L(G)$ -controlabilidade de K)

$$\subseteq \overline{K \cap L_m(G)}$$

(por ser $K \subseteq \overline{K} \cap L_m(G)$).

Isto significa que, para toda linguagem não $L_m(G)$ -fechada $K \in C(E)$ existe uma linguagem $L_m(G)$ -fechada e $L(G)$ -controlável $\overline{K} \cap L_m(G) \supseteq K$, também pertencente a $C(E)$. Como $\sup C(E)$ é o elemento supremo dessa classe, $\sup C(E)$ é $L_m(G)$ -fechada. \blacklozenge

Prop. 5.6: Dada uma linguagem $E \subseteq L_m(G)$, $\sup CF(E) = \sup C[\sup F(E)]$.

Dem.: Pelas definições de $\sup F(E)$ (corolário 4.1) e de $\sup CF(E)$ (corolário 4.2),

$$\sup CF(E) \subseteq \sup F(E),$$

donde

$$\sup CF(E) = \sup CF[\sup F(E)].$$

Pela proposição 5.5,

$$\sup C[\sup F(E)] = \sup CF[\sup F(E)]$$

e o resultado é imediato. \blacklozenge

A proposição seguinte dá uma fórmula para a máxima sublinguagem $L_m(G)$ -compatível de uma dada linguagem:

Prop. 5.7: Dada a linguagem prefixo-fechada $E \subseteq L(G)$, $\sup T(E) = \overline{E \cap L_m(G)}$.

Dem.:

(i) $\sup T(E) \subseteq \overline{E \cap L_m(G)}$:

$$\sup T(E) = \overline{\sup T(E) \cap L_m(G)} \subseteq \overline{E \cap L_m(G)}.$$

(ii) $\sup T(E) \supseteq \overline{E \cap L_m(G)}$: Demonstrar esta relação equivale a mostrar que $\overline{E \cap L_m(G)}$ é $L_m(G)$ -compatível, isto é, que $\overline{\overline{E \cap L_m(G)} \cap L_m(G)} = \overline{E \cap L_m(G)}$.

$$(\subseteq): \quad \overline{\overline{E \cap L_m(G)} \cap L_m(G)} \subseteq \overline{E \cap L_m(G) \cap L_m(G)} \subseteq \overline{E \cap L_m(G)}$$

$$(\supseteq): \quad \overline{E \cap L_m(G)} \supseteq E \cap L_m(G)$$

$$\overline{E \cap L_m(G) \cap L_m(G)} \supseteq E \cap L_m(G)$$

$$\overline{\overline{E \cap L_m(G)} \cap L_m(G)} \supseteq \overline{E \cap L_m(G)}. \quad \blacklozenge$$

Prop. 5.8: Dada uma linguagem $E \subseteq L_m(G)$,

$$\sup CF(E) = \sup CT[\sup F(E)] \cap L_m(G).$$

Dem.:

(\subseteq):

$$\sup CF(E) \subseteq \sup F(E)$$

$$\overline{\sup CF(E)} \subseteq \overline{\sup F(E)}.$$

Sendo $\overline{\sup CF(E)}$ $L(G)$ -controlável,

$$\overline{\sup CF(E)} \in C[\overline{\sup F(E)}].$$

Por outro lado, $\overline{\sup CF(E)}$ é $L_m(G)$ -compatível, pois

$$\overline{\sup CF(E)} \cap L_m(G) = \sup CF(E)$$

implica

$$\overline{\overline{\sup CF(E)} \cap L_m(G)} = \overline{\sup CF(E)},$$

de modo que

$$\overline{\sup CF(E)} \in T[\overline{\sup F(E)}].$$

Portanto,

$$\overline{\sup CF(E)} \subseteq \sup CT[\overline{\sup F(E)}].$$

Tomando a interseção com $L_m(G)$ e considerando que $\sup CF(E)$ é $L_m(G)$ -fechada,

$$\sup CF(E) \subseteq \sup CT[\overline{\sup F(E)}] \cap L_m(G).$$

(\supseteq):

$$\sup CT[\overline{\sup F(E)}] \subseteq \overline{\sup F(E)}.$$

Tomando a interseção com $L_m(G)$ e considerando que $\sup F(E)$ é $L_m(G)$ -fechada,

$$\sup CT[\overline{\sup F(E)}] \cap L_m(G) \subseteq \sup F(E).$$

Mas $\sup CT[\overline{\sup F(E)}] \cap L_m(G)$ é $L(G)$ -controlável, pois

$$\overline{\sup CT[\overline{\sup F(E)}] \cap L_m(G)} = \sup CT[\overline{\sup F(E)}], \text{ que é } L(G)\text{-controlável. Logo,}$$

$$\sup CT[\overline{\sup F(E)}] \cap L_m(G) \subseteq \sup C[\sup F(E)] \quad \text{e, pela proposição 5.6,}$$

$$\sup CT[\overline{\sup F(E)}] \cap L_m(G) \subseteq \sup CF(E), \quad \text{o que demonstra a proposição. } \blacklozenge$$

Prop. 5.9: Dada uma linguagem prefixo-fechada $E \subseteq L(G)$,

$$\sup CT(E) = \overline{\sup CF[E \cap L_m(G)]}.$$

Dem.:

(\subseteq):

$$\sup CT(E) \subseteq E$$

$$\sup CT(E) \cap L_m(G) \subseteq E \cap L_m(G).$$

$\sup CT(E) \cap L_m(G)$ é $L_m(G)$ -fechada pela proposição 5.4. Logo,

$$\sup CT(E) \cap L_m(G) \subseteq \sup F[E \cap L_m(G)].$$

$\sup CT(E) \cap L_m(G)$ é $L(G)$ -controlável, pois, sendo $\sup CT(E)$ $L_m(G)$ -compatível,

$$\overline{\sup CT(E) \cap L_m(G)} = \sup CT(E), \text{ que é } L(G)\text{-controlável. Logo,}$$

$$\sup CT(E) \cap L_m(G) \subseteq \sup C\{\sup F[E \cap L_m(G)]\}. \quad \text{Pela proposição 5.6,}$$

$$\sup CT(E) \cap L_m(G) \subseteq \sup CF[E \cap L_m(G)]. \quad \text{Tomando o prefixo-fechamento,}$$

$$\sup CT(E) \subseteq \overline{\sup CF[E \cap L_m(G)]}.$$

(\supseteq):

$$\sup CF[E \cap L_m(G)] \subseteq E \cap L_m(G)$$

$\overline{\sup CF[E \cap L_m(G)]} \subseteq \overline{E \cap L_m(G)}$.	Pela proposição 5.7,
$\overline{\sup CF[E \cap L_m(G)]} \subseteq \sup T(E)$	e portanto
$\overline{\sup CF[E \cap L_m(G)]} \in T(E)$.	Por outro lado,
$\overline{\sup CF[E \cap L_m(G)]} \subseteq \overline{E} = E$.	Sendo $\overline{\sup CF[E \cap L_m(G)]}$ $L(G)$ -controlável,
$\overline{\sup CF[E \cap L_m(G)]} \in C(E)$	e assim
$\overline{\sup CF[E \cap L_m(G)]} \subseteq \sup CT(E)$.	♦

Teo. 5.4 (Teorema da Dualidade):

(i) Dado um problema do tipo PCSI com uma linguagem-alvo $E \subseteq L_m(G)$, um supervisor tal que $L_c(S/G) = \sup CF(E)$ pode ser encontrado resolvendo-se o PCSII para a linguagem-alvo $\overline{\sup F(E)}$.

(ii) Dado um problema do tipo PCSII com uma linguagem-alvo prefixo-fechada $E \subseteq L(G)$, um supervisor tal que $L(S/G) = \sup CT(E)$ pode ser encontrado resolvendo-se PCSI para a linguagem-alvo $E \cap L_m(G)$.

Dem.:

(i) Resolvendo-se PCSII para a linguagem-alvo $\overline{\sup F(E)}$, encontra-se um supervisor tal que

$$L(S/G) = \sup CT[\overline{\sup F(E)}]. \quad \text{Pela definição 4.5,}$$

$$L_c(S/G) = \sup CT[\overline{\sup F(E)}] \cap L_m(G). \quad \text{Pela proposição 5.8,}$$

$$L_c(S/G) = \sup CF(E).$$

(ii) Resolvendo-se PCSI para a linguagem-alvo $E \cap L_m(G)$, encontra-se um supervisor tal que

$$L_c(S/G) = \sup CF[E \cap L_m(G)].$$

Pelo teorema 4.2, este supervisor é não bloqueante, de modo que

$$L(S/G) = \overline{\sup CF[E \cap L_m(G)]}. \quad \text{e, pela proposição 5.9,}$$

$$L(S/G) = \sup CT(E). \quad \text{♦}$$

5.6 CONCLUSÃO

A principal contribuição do capítulo é a introdução da definição de L -compatibilidade, a partir da qual se pode estabelecer um novo critério para a ausência de bloqueio e uma nova classe de linguagens, a saber a classe das sublinguagens $L_m(G)$ -compatíveis de uma dada linguagem, que tem propriedades análogas às das classes de sublinguagens $L_m(G)$ -fechadas e $L(G)$ -controláveis de uma dada linguagem, apresentadas no capítulo 4. A máxima sublinguagem $L_m(G)$ -compatível e $L(G)$ -controlável de uma dada linguagem permite obter a solução para o problema proposto, PCSII.

Além disso, é estabelecida ainda uma dualidade entre as formulações em termos de linguagens marcadas e em termos de linguagens geradas, mostrando que é possível empregar PCSI para resolver PCSII e vice-versa. A utilização de PCSII para resolver PCSI se mostra especialmente útil em casos onde a formulação em termos de linguagens marcadas não é adequada. Um exemplo deste tipo é encontrado no capítulo 6, que estende as contribuições aqui feitas ao controle descentralizado.

CAPÍTULO 6

CONTRIBUIÇÕES AO MODELO RW DESCENTRALIZADO

Este capítulo estende os resultados obtidos no capítulo 5 ao modelo RW descentralizado que, conforme citado na subseção 2.5.10, é uma das variações do modelo básico. Utiliza-se como exemplo um protocolo de comunicação de dados, conhecido como *protocolo do bit alternante*. Esta aplicação não tem somente caráter ilustrativo, mas é também uma crítica do material contido em [RW89b] e [Rudi92], no que diz respeito ao emprego de linguagens geradas e marcadas na formulação e na solução de problemas. Tal posição se justifica pelas seguintes razões:

- nas referências citadas são apresentados dois problemas de controle supervisorio descentralizado, denominados em [Rudi92] de LP (Local Problem) e GP (Global Problem)¹. Enquanto que LP é formulado e resolvido em termos de linguagens geradas, GP mistura linguagens geradas e marcadas. Embora não seja incorreta, esta formulação é criticável por limitar desnecessariamente as soluções que podem ser obtidas em aplicações práticas;
- um estudo do material mencionado permite concluir que o que se pretende é formular e resolver GP em termos de linguagens marcadas. No entanto, sua solução consiste em reduzi-lo ao problema LP, que, conforme dito acima, é formulado e resolvido em termos de linguagens geradas, de modo que formular GP em termos de linguagens marcadas tornaria os dois problemas incompatíveis. Daí o emprego de ambas as linguagens neste problema e as limitações citadas;
- conclui-se também que estas limitações não foram observadas na literatura original, que faz um uso inconsistente solução obtida, quando a aplica ao exemplo do protocolo do bit alternante;
- após a análise dos problemas tais como encontrados na literatura, GP é reapresentado, utilizando-se a formulação em termos de linguagens marcadas, como originalmente pretendido. Para eliminar a incompatibilidade com LP, transforma-se GP em um problema equivalente, formulado em termos de linguagens geradas. Esta transformação é feita com base no teorema 5.4, uma das contribuições do capítulo 5. Encontra-se assim uma solução formalmente correta que abrange os resultados originais;

¹Os mesmos problemas, restritos a um caso particular, aparecem também em [RW89b], onde são denominados DSP1 e DSP2 (Decentralized Control Problem 1 e 2), respectivamente. As críticas feitas neste capítulo se aplicam a ambos os casos.

- o exemplo do protocolo é resolvido empregando esta nova solução, confirmando-se os resultados obtidos na literatura, que estão corretos, apesar de obtidos de uma maneira inconsistente com a formulação do problema GP;
- na literatura original, o problema fica sem solução no caso geral, sendo ainda a solução obtida para um caso particular criticável por ser pouco prática. Introduce-se então o conceito de *caminho opcional* no grafo de um gerador, que é utilizado para melhorar a solução da literatura e também para obter uma solução para o caso geral. O método sugerido pode também ser útil em outros problemas de controle supervisorio descentralizado.

6.1 NOÇÕES PRELIMINARES

Esta seção apresenta resultados e conceitos necessários à compreensão do restante do capítulo, baseando-se em [RW89b], [Rudi92], [LW88a], [LW88b] e [LW90].

6.1.1 SUPERVISORES GLOBAIS E SUPERVISORES LOCAIS

Nos problemas de controle supervisorio tratados nos capítulos 4 e 5, os supervisores em questão observam os eventos produzidos pela planta, determinando, em resposta, as entradas de controle. Assume-se ali que os supervisores sempre podem observar todos os eventos ocorridos e habilitar ou inibir qualquer evento do conjunto de eventos controláveis. Supervisores com esta característica são ditos *supervisores globais*. Existem situações, porém, em que é desejável ou mesmo necessário controlar um sistema sem que se possa observar todos os eventos ocorridos e tendo acesso somente a um subconjunto do conjunto de eventos controláveis. Um supervisor que trabalhe nestas condições é dito *supervisor local*.

Dada uma planta de alfabeto (global) $\Sigma = \Sigma_c \cup \Sigma_u$, definem-se então os alfabetos locais Σ_l e Σ_{lc} . Estes são, respectivamente, os conjuntos de eventos observáveis e controláveis por um supervisor local, valendo as relações

$$\Sigma_l \subseteq \Sigma \quad (6.1)$$

$$\Sigma_{lc} \subseteq \Sigma_c \quad (6.2)$$

Os eventos em $\Sigma - \Sigma_{lc}$ são considerados não controláveis em nível local.

Além disso, a solução dos problemas de síntese de supervisores globais consiste em encontrar um único supervisor para a planta. No caso dos supervisores locais, pode-se tanto formular problemas com o objetivo de controlar a planta com um único supervisor local - caso tratado em [LW88a], como controle supervisorio sob observação parcial - como formular problemas cuja solução seja um conjunto de dois ou mais supervisores. É este o caso do controle descentralizado, aplicado a sistemas distribuídos, cuja topologia, além de permitir apenas a observação parcial de eventos, não admite o uso de um único supervisor global.

O conceito de projeção, apresentado na seção 3.4, é utilizado para tratar formalmente sistemas com observação parcial de eventos, e portanto também com controle descentralizado. As projeções são utilizadas para determinar o comportamento de um sistema do ponto de vista local: dada a projeção $P: \Sigma^* \rightarrow \Sigma_1^*$, se $L(G)$ é a linguagem gerada pela planta $\langle G, \Gamma \rangle$, então $PL(G)$ é a linguagem observada localmente. Um gerador G_1 tal que $L(G_1) = PL(G)$ representa, portanto, o comportamento do sistema do ponto de vista local.

6.1.2 NORMALIDADE E DECOMPONIBILIDADE

Os conceitos de projeção e de projeção inversa podem ser empregados para definir *normalidade* e *decomponibilidade* de uma linguagem, conceitos importantes para o tratamento de problemas de controle descentralizado.

Def. 6.1: Dados uma linguagem $L \subseteq \Sigma^*$, uma sublinguagem $K \subseteq L$, um alfabeto local $\Sigma_1 \subseteq \Sigma$ e a projeção $P: \Sigma^* \rightarrow \Sigma_1^*$, a sublinguagem K é dita *normal com relação a L e P* ou simplesmente *normal sse*

$$K = L \cap P^{-1}PK.$$

Informalmente, K é normal sse é possível recuperá-la a partir de sua versão local PK , do conhecimento da projeção inversa de P (isto é, do conhecimento do núcleo de P) e dos limites impostos por ser $K \subseteq L$.

Def. 6.2: Dados uma linguagem $L \subseteq \Sigma^*$, uma sublinguagem $K \subseteq L$, alfabetos locais $\Sigma_{11}, \Sigma_{12} \subseteq \Sigma$ e as projeções $P_1: \Sigma^* \rightarrow \Sigma_{11}^*$ e $P_2: \Sigma^* \rightarrow \Sigma_{12}^*$, K é dita *decomponível com relação a L , P_1 e P_2* ou simplesmente *decomponível sse*

$$K = L \cap P_1^{-1}PK \cap P_2^{-1}PK.$$

Informalmente, K é decomponível se é possível recuperá-la a partir de suas versões locais P_1K e P_2K , do conhecimento dos núcleos de P_1 e P_2 e dos limites impostos por ser $K \subseteq L$.

Prop. 6.1: Sejam uma linguagem $L \subseteq \Sigma^*$, uma sublinguagem $K \subseteq L$ e projeções $P_1: \Sigma^* \rightarrow \Sigma_{11}^*$ e $P_2: \Sigma^* \rightarrow \Sigma_{12}^*$ sobre alfabetos locais $\Sigma_{11}, \Sigma_{12} \subseteq \Sigma$. Se K é normal com respeito a L e P_1 e/ou com respeito a L e P_2 , então K é decomponível.

Dem.: A inclusão $K \subseteq L \cap P_1^{-1}PK \cap P_2^{-1}PK$ é sempre satisfeita, por ser $K \subseteq L$ e pela equação 3.11. Para provar a inclusão reversa, observe que

$$L \cap P_1^{-1}PK \cap P_2^{-1}PK \subseteq L \cap P_1^{-1}PK \quad \text{e que, se } K \text{ é normal com respeito a } L \text{ e } P_1,$$

$$L \cap P_1^{-1}PK \cap P_2^{-1}PK \subseteq K.$$

A demonstração para L e P_2 é análoga. ◆

Note que a recíproca da proposição anterior não é necessariamente verdadeira, isto é, não é necessário que K seja normal para que seja decomponível.

6.1.3 CONJUNÇÃO DE SUPERVISORES

A ação conjunta de dois (ou mais) supervisores sobre uma mesma planta pode ser descrita por uma operação denominada *conjunção de supervisores* e definida como segue:

Def. 6.3: Sejam $\langle S, \Phi \rangle$ e $\langle T, \Psi \rangle$ supervisores atuando sobre a planta $\langle G, \Gamma \rangle$, com $S = \langle \Sigma, X, \xi, x_0, X_m \rangle$, $T = \langle \Sigma, Y, \eta, y_0, Y_m \rangle$ e $G = \langle \Sigma, Q, \delta, q_0, Q_m \rangle$. A *conjunção de $\langle S, \Phi \rangle$ e $\langle T, \Psi \rangle$* é o supervisor

$$\langle S \times T, \Phi * \Psi \rangle,$$

definido por

$$S \times T = \langle \Sigma, X \times Y, \xi \times \eta, (x_0, y_0), X_m \times Y_m \rangle, \text{ com}$$

$$(\xi \times \eta)(\sigma, x, y) = \begin{cases} (\xi(\sigma, x), \eta(\sigma, y)) & \text{se } \xi(\sigma, x) \wedge \eta(\sigma, y) \\ \text{indefinida} & \text{de outra forma} \end{cases}$$

e

$$\Phi * \Psi: X \times Y \rightarrow \Gamma = \Phi \cap \Psi.$$

O supervisor resultante da conjunção de $\langle S, \Phi \rangle$ e $\langle T, \Psi \rangle$ tem sua função de transição definida exatamente para os pares (evento, estado) em que as funções ξ e η estão ambas definidas; um evento controlável σ está habilitado no estado (x, y) se e somente se $\sigma \in \Phi(x) \wedge \sigma \in \Psi(y)$.

Utiliza-se a notação $L(S \wedge T / G)$ para a linguagem gerada pela planta $\langle G, \Gamma \rangle$ quando sob supervisão de $\langle S \times T, \Phi * \Psi \rangle$. As notações para as linguagens controlada e marcada são análogas. Conforme demonstrado em [LW88b] (lema A.1), vale:

$$L(S \wedge T / G) = L(S / G) \cap L(T / G) \quad (6.3).$$

Além disso, de acordo com [Rudi92],

$$L_c(S \wedge T / G) = L_c(S / G) \cap L_c(T / G) \quad (6.4)$$

e

$$L_m(S \wedge T / G) = L_m(S / G) \cap L_m(T / G) \quad (6.5).$$

A operação de conjunção goza da propriedade associativa, podendo ser estendida a um número arbitrário de supervisores.

6.1.4 EXTENSÃO GLOBAL DE UM SUPERVISOR LOCAL

A observação da definição 6.3 permite concluir que a operação de conjunção só se aplica a supervisores globais, uma vez que se supõe que o alfabeto dos supervisores operandos seja igual ao alfabeto global da planta. Por outro lado, é necessária uma operação que permita determinar o comportamento global de um sistema controlado por supervisores locais. A fim de tornar isto possível, um supervisor local pode ser estendido de acordo com a seguinte definição:

Def. 6.4: Dados:

- uma planta $\langle G, \Gamma \rangle$ de alfabeto $\Sigma = \Sigma_c \cup \Sigma_u$;
- um alfabeto local $\Sigma_l \subseteq \Sigma$;
- um conjunto de eventos localmente controláveis $\Sigma_{lc} \subseteq \Sigma_c$;
- a projeção $P: \Sigma^* \rightarrow \Sigma_l^*$;
- um supervisor local $\langle S, \Phi \rangle$ com:

$$S = \langle \Sigma_l, X, \xi, x_0, X_m \rangle \quad \text{e}$$

$$\Phi = \Phi: X \rightarrow \Gamma, \quad \text{onde } \Gamma, \text{ é o conjunto de entradas de controle válidas definido por } \Sigma \text{ e } \Sigma_{lc},$$

a *extensão global* de $\langle S, \Phi \rangle$ é o supervisor $\langle \tilde{S}, \tilde{\Phi} \rangle$, onde

$$\tilde{S} = \langle \Sigma, X, \tilde{\xi}, x_0, X_m \rangle \quad \text{e}$$

$$\tilde{\Phi} = \tilde{\Phi}: X \rightarrow \Gamma, \quad \text{com}$$

$\tilde{\xi}$ definida de modo que $L(\tilde{S}) = P^{-1}L(S)$ e

$\tilde{\Phi}$ exercendo sobre os eventos de Σ_{lc} a mesma ação de controle que Φ e habilitando permanentemente os eventos em $\Sigma - \Sigma_l$.

Com esta definição, $\langle \tilde{S}, \tilde{\Phi} \rangle$ é um supervisor global que executa as mesmas transições que $\langle S, \Phi \rangle$ para os eventos em Σ_l , permanecendo no mesmo estado quando ocorrem eventos de $\Sigma - \Sigma_l$.

6.2 TIPOS DE PROBLEMAS DE CONTROLE DESCENTRALIZADO

Esta seção apresenta os dois tipos de problemas de controle descentralizado tratados na literatura. Considere a figura 6.1, onde a planta $\langle G, \Gamma \rangle$ é controlada por n supervisores locais.

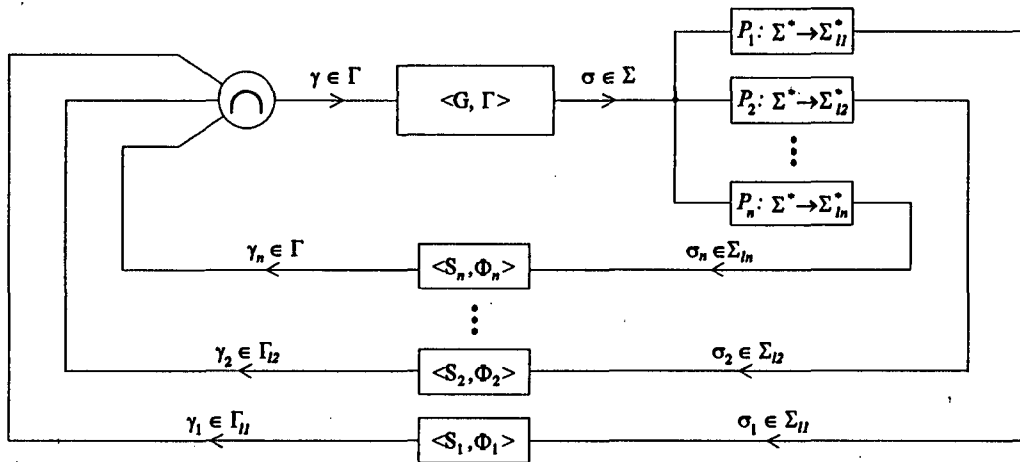


Fig. 6.1 - Controle supervisorio descentralizado

Os supervisores locais $\langle S_i, \Phi_i \rangle$, $i = 1, 2, \dots, n$ observam a planta $\langle G, \Gamma \rangle$ através das projeções P_i . Localmente, a planta tem por modelo o par $\langle G_i, \Gamma_{i\ell} \rangle$, com $L(G_i) = P_i L(G)$ e $\Gamma_{i\ell}$ definido por Σ e $\Sigma_{i\ell}$. Desta forma, estão também definidas as linguagens geradas locais $L(S_i / G_i)$, $i = 1, 2, \dots, n$.

Do ponto de vista global, tudo se passa como se a planta estivesse sendo controlada por um supervisor global único, igual à conjunção das extensões globais dos n supervisores locais, dado por:

$$\bigwedge_{i=1}^n \tilde{S}_i = \langle \tilde{S}_1 \times \tilde{S}_2 \times \dots \times \tilde{S}_n, \tilde{\Phi}_1 * \tilde{\Phi}_2 * \dots * \tilde{\Phi}_n \rangle \tag{6.6}$$

de acordo com a definição 6.3. Portanto, está definida também a linguagem $L\left(\bigwedge_{i=1}^n \tilde{S}_i / G\right)$, que representa o comportamento global do sistema sob ação n supervisores $\langle S_i, \Phi_i \rangle$. A seguinte equação, retirada de [LW90] e consequência da equação 6.3, permite expressar o comportamento global em termos dos comportamentos locais:

$$L\left(\bigwedge_{i=1}^n \tilde{S}_i / G\right) = L(G) \cap \bigcap_{i=1}^n P_i^{-1} L(S_i / G_i) \tag{6.7}$$

Com isso, é possível formular dois tipos distintos de problemas de controle descentralizado, a saber:

i. *Problemas com especificações locais*, onde o objetivo é construir os supervisores locais $\langle S_i, \Phi_i \rangle$ de modo a satisfazer de forma minimamente restritiva um conjunto de linguagens-alvo locais $K_i \subseteq L(G_i)$. Neste caso, o comportamento global resultante, dado pela equação 6.7, é mera consequência dos comportamentos locais;

ii. *Problemas com especificação global*, onde o objetivo é construir os supervisores locais $\langle S_i, \Phi_i \rangle$ de modo que $L\left(\bigwedge_{i=1}^n \tilde{S}_i / G\right)$ satisfaça de forma minimamente restritiva uma linguagem-alvo global $K \subseteq L(G)$.

Os problemas de controle descentralizado mencionados no início do capítulo correspondem aos dois tipos descritos acima. LP (DSP1) é um problema com especificações locais e GP (DSP2) é um problema com especificação global.

Cabe observar que as versões DSP1 e DSP2 ([RW89b]) diferem das versões LP e GP ([Rudi92]) no seguinte aspecto: assume-se em [RW89b] que os eventos localmente controláveis são também localmente observáveis, isto é, que

$$\Sigma_{loc} = \Sigma_c \cap \Sigma_{H_i}, \quad i = 1, 2, \dots, n \quad (6.8)$$

e portanto

$$\Sigma_{loc} \subseteq \Sigma_{H_i}, \quad i = 1, 2, \dots, n \quad (6.9).$$

Este é um caso particular da formulação adotada em [Rudi92], onde os supervisores locais podem controlar eventos sem que seja necessário que possam observá-los.

O presente trabalho se limita a tratar as soluções dos problemas para este caso particular. Não obstante, as críticas feitas à formulação de DSP2 e GP na seção seguinte se aplicam também no caso geral.

Note ainda que ambos os tipos de problemas podem ser formulados tanto em termos de linguagens marcadas quanto em termos de linguagens geradas. No caso das linguagens marcadas, as linguagens-alvo locais são tais que $K_i \subseteq L_m(G_i) \subseteq L(G_i)$ e a linguagem-alvo global tal que $K \subseteq L_m(G) \subseteq L(G)$. No caso das linguagens geradas, $K_i = \bar{K}_i \subseteq L(G_i)$ e $K = \bar{K} \subseteq L(G)$.

6.3 A FORMULAÇÃO ENCONTRADA NA LITERATURA

Esta seção apresenta os problemas LP e GP conforme encontrados em [Rudi92]. LP é apresentado com a solução para o caso particular citado na seção 6.2, seguido do enunciado de GP, apontando-se as falhas encontradas.

6.3.1 PROBLEMA LOCAL (LP)

A formulação do problema LP, feitas as mudanças cabíveis na notação, é a seguinte:

Problema Local (LP): Dados uma planta $\langle G, \Gamma \rangle$ de alfabeto Σ , alfabetos locais $\Sigma_1, \Sigma_2 \subseteq \Sigma$, projeções $T_1: \Sigma^* \rightarrow \Sigma_{11}^*$ e $T_2: \Sigma^* \rightarrow \Sigma_{12}^*$, linguagens-alvo $E_1 \subseteq T_1L(G)$ e $E_2 \subseteq T_2L(G)$, linguagens minimamente adequadas $A_1 \subseteq E_1$ e $A_2 \subseteq E_2$ e conjuntos $\Sigma_{1c1}, \Sigma_{11} \subseteq \Sigma_1$ e $\Sigma_{1c2}, \Sigma_{12} \subseteq \Sigma_2$, construir supervisores locais (completos) $\langle S_1, \Phi_1 \rangle$ e $\langle S_2, \Phi_2 \rangle$ tais que

$$L(G) \cap \bigcap_{i=1}^n T_i^{-1} A_i \subseteq L(\tilde{S}_1 \wedge \tilde{S}_2 / G) \subseteq L(G) \cap \bigcap_{i=1}^n T_i^{-1} E_i.$$

Aqui, para $i = 1, 2$, $\langle S_i, \Phi_i \rangle$ é um supervisor que observa somente eventos em Σ_{1i} e controla apenas os eventos em Σ_{1ci} . $\langle \tilde{S}_i, \tilde{\Phi}_i \rangle$ é a extensão global de $\langle S_i, \Phi_i \rangle$.

Note que as linguagens-alvo especificadas são locais e, embora a linguagem representando o comportamento global do sistema, $L(\tilde{S}_1 \wedge \tilde{S}_2 / G)$, apareça na formulação, nenhuma restrição lhe é imposta além daquelas decorrentes das restrições locais, de acordo com a equação 6.7.

A solução para LP no caso particular em que $\Sigma_{11} = \Sigma_1$ e $\Sigma_{12} = \Sigma_2$, valendo portanto as equações 6.8 e 6.9 (para $n = 2$), dada em [Rudi92], utiliza-se das projeções $P_1: \Sigma^* \rightarrow \Sigma_{11}^*$ e $P_2: \Sigma^* \rightarrow \Sigma_{12}^*$ que são, neste caso, iguais a T_1 e T_2 , respectivamente.

Primeiramente, a solução do problema PCSIG (v. capítulo 4) é empregada para encontrar supervisores locais tais que

$$P_1L(G) \cap A_1 \subseteq L(S_1 / G_1) \subseteq P_1L(G) \cap E_1 \quad (6.10)$$

e

$$P_2L(G) \cap A_2 \subseteq L(S_2 / G_2) \subseteq P_2L(G) \cap E_2 \quad (6.11).$$

Em seguida, $\langle S_1, \Phi_1 \rangle$ e $\langle S_2, \Phi_2 \rangle$ são estendidos de acordo com a definição 6.4. Então, de acordo com a equação 6.7,

$$L(\tilde{S}_1 \wedge \tilde{S}_2 / G) = L(G) \cap P_1^{-1}L(S_1 / G_1) \cap P_2^{-1}L(S_2 / G_2). \quad \text{Pelas equações 6.10 e 6.11,}$$

$$L(\tilde{S}_1 \wedge \tilde{S}_2 / G) \subseteq L(G) \cap P_1^{-1}P_1L(G) \cap P_1^{-1}E_1 \cap P_2^{-1}P_2L(G) \cap P_2^{-1}E_2.$$

Considerando a equação 3.11,

$$L(\tilde{S}_1 \wedge \tilde{S}_2 / G) \subseteq L(G) \cap P_1^{-1}E_1 \cap P_2^{-1}E_2.$$

De forma análoga, pode-se mostrar que

$$L(G) \cap P_1^{-1}A_1 \cap P_2^{-1}A_2 \subseteq L(\tilde{S}_1 \wedge \tilde{S}_2 / G), \quad \text{de modo que os supervisores locais}$$

construídos satisfazem o enunciado do problema.

Note que esta solução para LP não leva em conta a questão do bloqueio: a utilização da solução de PCSIG exige que se marquem todos os estados da planta, não sendo possível considerar uma marcação diferente. Em princípio, poder-se-ia utilizar a solução do problema PCSII, introduzido no capítulo 5, que garantiria supervisores locais não bloqueantes. Mas, de acordo com [LW88b], não basta que $\langle S_1, \Phi_1 \rangle$ e $\langle S_2, \Phi_2 \rangle$ sejam supervisores não bloqueantes em nível local para que o sistema seja não bloqueante em nível global. Para isto, é necessário que as linguagens $L_c(\tilde{S}_1/G)$ e $L_c(\tilde{S}_2/G)$ sejam *não conflitantes*, condição expressa na referência citada como

$$\overline{L_c(\tilde{S}_1/G) \cap L_c(\tilde{S}_2/G)} = \overline{L_c(\tilde{S}_1/G)} \cap \overline{L_c(\tilde{S}_2/G)} \quad (6.12).$$

Não existe, porém, na literatura, uma abordagem para solucionar LP levando em conta esta condição. Este também não é o objetivo deste trabalho, de modo que a questão do bloqueio é deixada em aberto, prosseguindo-se com a apresentação do problema GP.

6.3.2 PROBLEMA GLOBAL (GP)

A formulação dada em [Rudi92] para o problema GP, feitas as mudanças cabíveis na notação, é a seguinte:

Problema Global (GP): Dados uma planta $\langle G, \Gamma \rangle$ de alfabeto Σ , uma linguagem-alvo $E \subseteq L_m(G)$, uma linguagem minimamente adequada $A \subseteq E$ e conjuntos $\Sigma_{1c1}, \Sigma_{1c2}, \Sigma_{1n}, \Sigma_{12} \subseteq \Sigma$, construir supervisores locais (completos) $\langle S_1, \Phi_1 \rangle$ e $\langle S_2, \Phi_2 \rangle$ tais que $\langle \tilde{S}_1 \times \tilde{S}_2, \tilde{\Phi}_1 * \tilde{\Phi}_2 \rangle$ seja um supervisor próprio para $\langle G, \Gamma \rangle$ e tal que

$$A \subseteq L(\tilde{S}_1 \wedge \tilde{S}_2 / G) \subseteq E.$$

Aqui, para $i = 1, 2$, o supervisor $\langle S_i, \Phi_i \rangle$ pode observar somente os eventos em Σ_{in} e controlar somente os eventos em Σ_{1ci} ; $\langle \tilde{S}_i, \tilde{\Phi}_i \rangle$ denota a extensão global de $\langle S_i, \Phi_i \rangle$. Subentende-se que o conjunto de eventos não controláveis é $\Sigma - (\Sigma_{1c1} \cup \Sigma_{1c2})$.

O problema acima não pode ser classificado como sendo um problema formulado em termos de linguagens marcadas nem em termos de linguagens geradas, uma vez que mistura ambas as formulações: enquanto que a linguagem-alvo $E \subseteq L_m(G)$ é uma linguagem marcada, o objetivo do problema é expresso em termos da linguagem gerada $L(\tilde{S}_1 \wedge \tilde{S}_2 / G)$.

Note agora que $L(\tilde{S}_1 \wedge \tilde{S}_2 / G)$ é uma linguagem prefixo-fechada, de forma que qualquer solução para o problema está limitada à linguagem

$$\sup P(E) = \{s : s \in E \wedge \text{Pre}(s) \subseteq E\} \quad (6.13),$$

que é a *máxima sublinguagem prefixo-fechada de E*. Para ilustrar, suponha que a linguagem-alvo E seja a linguagem marcada do gerador da figura 6.2.

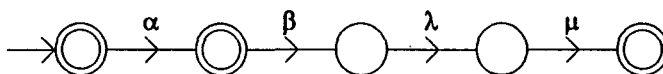


Fig. 6.2 - Gerador para ilustrar as limitações de GP

Tem-se então $E = \{\epsilon, \alpha, \alpha\beta\lambda\mu\}$. A máxima sublinguagem prefixo-fechada de E é $\text{sup } P(E) = \{\epsilon, \alpha\}$. Não obstante, poderia haver supervisores $\langle S_1, \Phi_1 \rangle$ e $\langle S_2, \Phi_2 \rangle$ tais que $L_c(\tilde{S}_1 \wedge \tilde{S}_2 / G) = E$. Esta solução, embora desejável, não seria permitida pela própria formulação do problema, uma vez que violaria a condição $L(\tilde{S}_1 \wedge \tilde{S}_2 / G) \subseteq E$.

Note ainda que, se $\epsilon \notin E$, isto é, se o estado inicial do gerador para E não for um estado marcado, então $\text{sup } P(E) = \emptyset$ e o problema não tem solução e que, por outro lado, a igualdade $L(\tilde{S}_1 \wedge \tilde{S}_2 / G) = E$ só é atingida no caso particular em que E é prefixo-fechada, isto é, se todos os estados do gerador forem marcados.

Embora não seja incorreta, a formulação de GP é criticável por ser pouco prática. O que se deseja ao resolver um problema desse tipo é, em geral, obter a máxima sublinguagem implementável da linguagem-alvo, o que, em geral, não é possível com as limitações impostas. Dado que a linguagem-alvo é uma sublinguagem de $L_m(G)$, o mais natural seria formular o problema com o objetivo de encontrar supervisores tais que $A \subseteq L_c(\tilde{S}_1 \wedge \tilde{S}_2 / G) \subseteq E$, isto é, especificar a linguagem controlada do sistema sob supervisão e não sua linguagem gerada.

Um estudo do exemplo do protocolo do bit alternante, utilizado em [Rudi92] para ilustrar a aplicação de GP, permite afirmar que o que se pretende ali corresponde exatamente a esta formulação: a solução apresentada é tal que $L_c(\tilde{S}_1 \wedge \tilde{S}_2 / G) = E$. Mais ainda, como naquele exemplo E não é prefixo-fechada, a condição $L(\tilde{S}_1 \wedge \tilde{S}_2 / G) \subseteq E$ não é obedecida, o que permite dizer que o emprego do problema não é consistente com sua formulação.

Outro ponto falho com relação a esse problema está nas considerações sobre o bloqueio. Embora se afirme em [Rudi92] (p. 42) que "soluções para ... GP são sempre não bloqueantes, dado que um par de supervisores $\langle S_1, \Phi_1 \rangle$ e $\langle S_2, \Phi_2 \rangle$ é considerado uma solução ... somente se $\langle \tilde{S}_1 \times \tilde{S}_2, \tilde{\Phi}_1 * \tilde{\Phi}_2 \rangle$ for um supervisor próprio para $\langle G, \Gamma \rangle$ ", não existe nada na solução apresentada que garanta que o comportamento global do sistema seja de fato livre de bloqueio. Por isso, nas versões do problema apresentadas na seção seguinte, o termo "próprio" é substituído por "completo e não rejeitante". Pode-se verificar que a definição da função de transição de $\langle \tilde{S}_1 \times \tilde{S}_2, \tilde{\Phi}_1 * \tilde{\Phi}_2 \rangle$ corresponde à de um supervisor completo e que se pode sempre escolher $X_m = X$, de modo que não há rejeição.

6.4 AS FORMULAÇÕES PROPOSTAS

Esta seção propõe duas novas versões para GP, a primeira formulada em termos de linguagens marcadas e a segunda em termos de linguagens geradas. Parte-se da formulação da subseção 6.3.2, modificada de modo a se tornar consistente com o uso pretendido em [Rudi92], o que leva à formulação em termos de linguagens marcadas:

GP para linguagens marcadas (GPM): Dados uma planta $\langle G, \Gamma \rangle$ de alfabeto Σ , uma linguagem-alvo $E_m \subseteq L_m(G)$, uma linguagem minimamente adequada $A_m \subseteq E_m$ e conjuntos $\Sigma_{lc1}, \Sigma_{lc2}, \Sigma_{l1}, \Sigma_{l2} \subseteq \Sigma$, construir supervisores locais (completos) $\langle S_1, \Phi_1 \rangle$ e $\langle S_2, \Phi_2 \rangle$ tais que $\langle \tilde{S}_1 \times \tilde{S}_2, \tilde{\Phi}_1 * \tilde{\Phi}_2 \rangle$ seja um supervisor completo e não rejeitante para $\langle G, \Gamma \rangle$ e tal que

$$A_m \subseteq L_c(\tilde{S}_1 \wedge \tilde{S}_2 / G) \subseteq E_m.$$

Para $i = 1, 2$, $\langle S_i, \Phi_i \rangle$ e $\langle \tilde{S}_i, \tilde{\Phi}_i \rangle$ são definidos como no problema GP. Subentende-se que o conjunto de eventos não controláveis é $\Sigma - (\Sigma_{lc1} \cup \Sigma_{lc2})$.

As modificações introduzidas acima têm uma consequência importante: a solução apresentada em [Rudi92] para GP, que consiste em reduzi-lo a um problema do tipo LP, não se aplica a GPM, por se tratar agora com a linguagem controlada do sistema e não mais com sua linguagem gerada. Por isso, formula-se o seguinte problema:

GP para linguagens geradas (GPG): Dados uma planta $\langle G, \Gamma \rangle$ de alfabeto Σ , uma linguagem-alvo prefixo-fechada $E \subseteq L(G)$, uma linguagem minimamente adequada $A \subseteq E$ e conjuntos $\Sigma_{lc1}, \Sigma_{lc2}, \Sigma_{l1}, \Sigma_{l2} \subseteq \Sigma$, construir supervisores locais (completos) $\langle S_1, \Phi_1 \rangle$ e $\langle S_2, \Phi_2 \rangle$ tais que $\langle \tilde{S}_1 \times \tilde{S}_2, \tilde{\Phi}_1 * \tilde{\Phi}_2 \rangle$ seja um supervisor completo e não rejeitante para $\langle G, \Gamma \rangle$ e tal que

$$A \subseteq L(\tilde{S}_1 \wedge \tilde{S}_2 / G) \subseteq E.$$

Para $i = 1, 2$, $\langle S_i, \Phi_i \rangle$ e $\langle \tilde{S}_i, \tilde{\Phi}_i \rangle$ são definidos como no problema GP. Subentende-se que o conjunto de eventos não controláveis é $\Sigma - (\Sigma_{lc1} \cup \Sigma_{lc2})$.

De acordo com o teorema 5.4, a solução para GPM pode ser obtida resolvendo-se GPG com $E = \overline{\sup F(E_m)}$. A solução para GPG, semelhante àquela dada em [Rudi92] para GP, é dada pelo teorema seguinte:

Teo. 6.1: Seja $\Sigma_{lc1} \subseteq \Sigma_{li}$, $i = 1, 2$. Dadas projeções $P_1: \Sigma^* \rightarrow \Sigma_{l1}^*$ e $P_2: \Sigma^* \rightarrow \Sigma_{l2}^*$, as seguintes condições são suficientes para a existência de uma solução para GPG:

(i) para o alfabeto local Σ_{li} e a versão local da linguagem da planta $P_i L(G)$ deve valer $A_i \subseteq \sup C(E_i)$, $i = 1, 2$;

(ii) $\sup C(E_i) \neq \emptyset$;

(iii) A e E são decomponíveis com relação a $L(G)$, P_1 e P_2 .

Dem.: Sejam $A_1 = P_1A$, $A_2 = P_2A$, $E_1 = P_1E$ e $E_2 = P_2E$. Pondo ainda $\Sigma_1 = \Sigma_{11}$ e $\Sigma_2 = \Sigma_{12}$, a hipótese $\Sigma_{1cl} \subseteq \Sigma_{11}$, $i = 1, 2$ permite aplicar a solução do problema LP dada na subseção 6.3.1 a estas linguagens. As condições (i) e (ii) acima garantem a existência de uma solução tal que

$$L(G) \cap P_1^{-1}A_1 \cap P_2^{-1}A_2 \subseteq L(\tilde{S}_1 \wedge \tilde{S}_2 / G) \subseteq L(G) \cap P_1^{-1}E_1 \cap P_2^{-1}E_2.$$

Substituindo as definições de A_i e E_i , $i = 1, 2$, vem:

$$L(G) \cap P_1^{-1}P_1A \cap P_2^{-1}P_2A \subseteq L(\tilde{S}_1 \wedge \tilde{S}_2 / G) \subseteq L(G) \cap P_1^{-1}P_1E \cap P_2^{-1}P_2E.$$

Pela condição (iii) do teorema,

$$A \subseteq L(\tilde{S}_1 \wedge \tilde{S}_2 / G) \subseteq E. \quad \blacklozenge$$

Este resultado mostra que, se as linguagens A e E forem decomponíveis, então GPG pode ser reduzido a um problema do tipo LP, cuja solução é conhecida.

A seção seguinte apresenta um exemplo de aplicação deste problema, apontando e corrigindo as falhas existentes na versão encontrada na literatura.

6.5 O PROBLEMA DA TRANSMISSÃO SEQUENCIAL DE DADOS

Nesta seção se discute um problema de comunicação, conhecido como problema da transmissão sequencial de dados e originário da área da Ciência da Computação. Em vista do apresentado neste capítulo, este problema pode também ser tratado como um problema de controle supervisorio descentralizado. O seguinte conceito é importante para a discussão que segue:

Def. 6.5: Um *protocolo de comunicação de dados* ou simplesmente *protocolo* é um conjunto de regras que determinam quais ações os agentes da comunicação, denominados de *transmissor* e *receptor*, devem executar e sob que condições, para que este seja capaz de reproduzir sem erro as informações entregues àquele.

Solucionar um problema de transmissão de dados consiste, em geral, em encontrar um protocolo para os agentes de comunicação de modo que o sistema obedeça a certas especificações globais, tais como garantia de entrega de mensagens, repetição em caso de perda e ausência de duplicatas. Estas são características de um problema descentralizado (uma vez que, por hipótese, receptor e transmissor só têm acesso a uma parte dos eventos do sistema) com especificações globais (formuladas em termos do comportamento global desejado). A partir destas especificações e do comportamento fisicamente possível do sistema, deve-se então encontrar especificações locais para o receptor e o transmissor tais que o comportamento global resultante atenda as especificações feitas. As soluções locais encontradas constituem o protocolo de comunicação.

O problema tratado tem o seguinte enunciado:

"Um agente de comunicação, denominado transmissor, deve enviar uma seqüência arbitrariamente longa de mensagens, uma de cada vez, a um outro agente, denominado receptor. Este deve entregar as mensagens no destino na seqüência correta e sem duplicatas.

As seguintes limitações físicas devem ser consideradas:

- transmissor e receptor só podem se comunicar por mensagens (isto é, não se admite o uso de variáveis compartilhadas);
- a comunicação é assíncrona (não tem relação com qualquer tipo de referência de tempo);
- todas as mensagens (os dados no sentido transmissor→receptor e as confirmações no sentido inverso) são transmitidas por um canal semi-duplex (canal bidirecional que pode ser utilizado alternadamente em cada direção);
- o canal pode perder mensagens, mas não é capaz de corrompê-las, alterar sua ordem ou duplicá-las;
- o transmissor pode adicionar aos dados um bit de controle e o receptor pode responder com uma mensagem de reconhecimento de um bit."

Este problema tem uma solução clássica, conhecida como *protocolo do bit alternante*, sendo o objetivo da análise em [RW89b] e [Rudi92] verificar até que ponto o problema GP pode ser utilizado para sintetizar uma solução como esta. Conforme mencionado na seção 6.3, a solução apresentada na literatura é inconsistente com o problema GP. O exemplo é resolvido aqui de acordo com o problema GPG, introduzido na seção 6.4, apontando-se as diferenças em relação ao material original.

A figura 6.3 mostra as partes componentes do sistema de comunicação, a saber o transmissor (Tx), o canal de comunicação e o receptor (Rx).

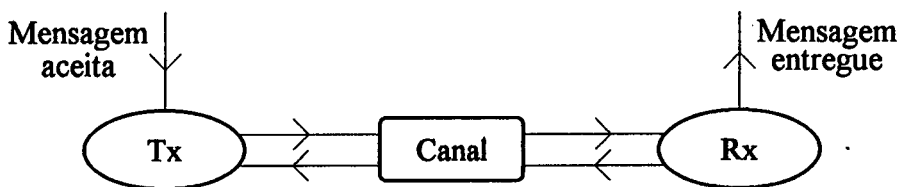


Fig. 6.3 - Topologia do Sistema de Comunicação

Para construir uma planta correspondente a este sistema é necessário descrever cada um dos componentes acima e então combiná-los de forma a obter uma representação de seu funcionamento conjunto. Em função das especificações do problema e das limitações físicas impostas, tal descrição deve se dar em termos do seguinte alfabeto de eventos:

$$\Sigma = \{g, t_0, t_1, rt_0, rt_1, a_0, a_1, ra_0, ra_1, p, l\} \quad (6.14),$$

sendo estes assim interpretados:²

g : transmissor aceita mensagem para transmitir;

t_0 : transmissor envia mensagem com bit de controle igual a 0;

t_1 : transmissor envia mensagem com bit de controle igual a 1;

rt_0 : receptor recebe mensagem com bit de controle igual a 0;

rt_1 : receptor recebe mensagem com bit de controle igual a 1;

a_0 : receptor envia mensagem de reconhecimento com bit de controle igual a 0;

a_1 : receptor envia mensagem de reconhecimento com bit de controle igual a 1;

ra_0 : transmissor recebe mensagem de reconhecimento com bit de controle igual a 0;

ra_1 : transmissor recebe mensagem de reconhecimento com bit de controle igual a 1;

p : receptor entrega mensagem no destino;

l : perda de mensagem no canal.

Das limitações físicas impostas, conclui-se também que o transmissor não pode observar as ações do receptor e vice-versa. Além disso, assume-se que o transmissor tem um mecanismo de temporização que lhe permite saber se houve perda de mensagem no canal. Este mecanismo se baseia na hipótese de que toda mensagem enviada pelo transmissor é confirmada pelo receptor. Cada vez que envia uma mensagem, o transmissor inicia a temporização e aguarda até que chegue a confirmação da mensagem enviada - caso em que tem certeza de que a mensagem atingiu o receptor - ou então até que vença a temporização, quando então assume que uma das mensagens - a que enviou ou a confirmação - foi perdida. Assume-se ainda que a temporização é ajustada de tal modo que o tempo de espera seja maior do que o maior atraso possível no canal, de maneira que não é necessário considerar a possibilidade de chegar uma confirmação com atraso, após vencida a temporização.

Com isso, os alfabetos locais para o transmissor e o receptor são, respectivamente:

$$\Sigma_{11} = \{g, t_0, t_1, ra_0, ra_1, l\} \quad (6.15)$$

e

$$\Sigma_{12} = \{a_0, a_1, rt_0, rt_1, p\} \quad (6.16).$$

Consideram-se definidas ainda as projeções

$$P_1: \Sigma^* \rightarrow \Sigma_{11}^* \quad (6.17)$$

²Embora as letras utilizadas para os eventos não sejam sempre as iniciais de uma palavra em português que se pudesse associar ao seu significado, preferiu-se manter a simbologia de [Rudi92], para facilitar eventuais comparações.

e

$$P_2: \Sigma^* \rightarrow \Sigma_{12}^* \tag{6.18},$$

cujos núcleos são, respectivamente:

$$\ker_1 = \Sigma_{12} = \{a_0, a_1, rt_0, rt_1, p\} \tag{6.19}$$

e

$$\ker_2 = \Sigma_{11} = \{g, t_0, t_1, ra_0, ra_1, l\} \tag{6.20}.$$

Os conjuntos de eventos controláveis do transmissor e do receptor são, respectivamente:

$$\Sigma_{lc1} = \{g, t_0, t_1\} \tag{6.21}$$

e

$$\Sigma_{lc2} = \{a_0, a_1, p\} \tag{6.22}.$$

Note que, para $i = 1, 2$, $\Sigma_{lc_i} \subseteq \Sigma_{ii}$, satisfazendo a hipótese inicial do teorema 6.1.

O gerador "CANAL" da figura 6.4 será empregado para representar o canal de comunicação em todos os casos analisados.

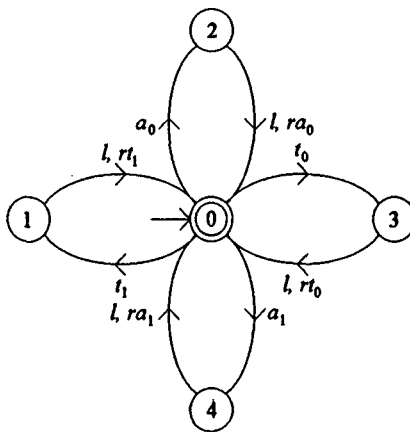


Fig. 6.4 - Gerador "CANAL"

A representação acima significa que o canal pode estar ou vazio (estado 0) ou então conter uma das quatro mensagens definidas no sistema, que deixam o canal ou por perda ou pela recepção no destino correspondente. Conclui-se daí que os agentes de comunicação não podem enviar duas mensagens consecutivas sem que ocorra entre elas uma perda ou a recepção no destino. Esta limitação não restringe o poder ilustrativo do exemplo.

A fim de investigar a utilidade do problema GPG na síntese de protocolos de comunicação, são considerados dois modelos diferentes para o transmissor e dois para o receptor, da mesma forma que em [RW89b] e [Rudi92]. Combinados de todas as formas possíveis, resultam em quatro casos a analisar. Os dois primeiros modelos, denominados *transmissor solução* e *receptor solução*, correspondem à solução conhecida para o problema

(o protocolo do bit alternante) e estão representados pelos geradores STX e SRX das figuras 6.5 e 6.6, respectivamente; os dois outros, denominados *transmissor genérico* e *receptor genérico*, são uma versão menos especializada dos primeiros, empregada para verificar se é possível chegar até aquela solução sem seu prévio conhecimento e são representados pelos geradores GTX e GRX das figuras 6.7 e 6.8.

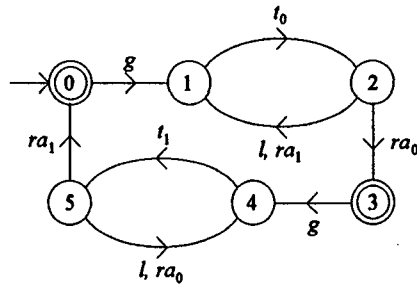


Fig. 6.5 - Gerador STX - Transmissor solução

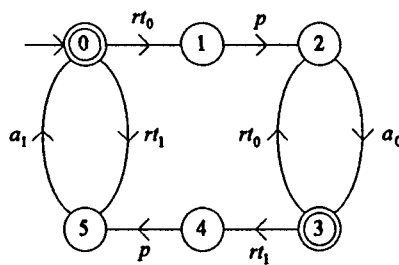


Fig. 6.6 - Gerador SRX - Receptor solução

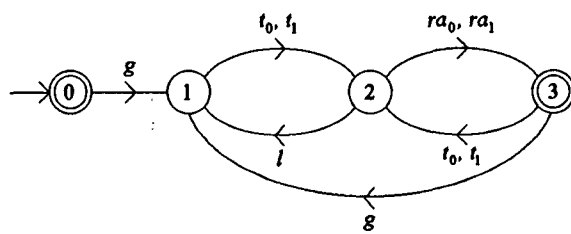


Fig. 6.7 - Gerador GTX - Transmissor genérico

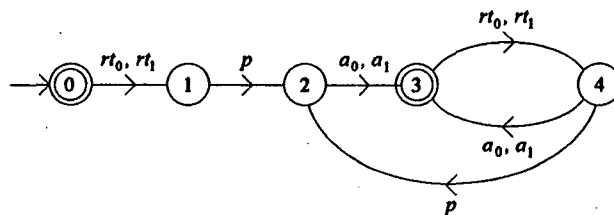


Fig. 6.8 - Gerador GRX - Receptor genérico

Os quatro casos analisados correspondem à utilização das quatro representações obtidas para a planta, combinando-se:

- (i) transmissor solução, canal, receptor solução;
- (ii) transmissor solução, canal, receptor genérico;
- (iii) transmissor genérico, canal, receptor solução;
- (iv) transmissor genérico, canal, receptor genérico.

Em qualquer dos casos, o gerador da linguagem-alvo E é dado pelo produto síncrono do gerador da linguagem da planta com o gerador "LEGAL" da figura 6.9, que traduz a especificação do problema que diz que todas as mensagens devem ser entregues em ordem e sem duplicação.

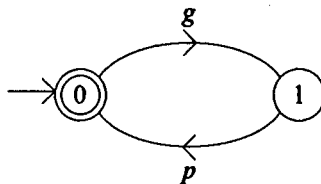


Fig. 6.9 - Gerador "LEGAL"

O objetivo da análise é verificar, em primeiro lugar, se a linguagem-alvo é decomponível, isto é, se satisfaz a condição (iii) do teorema 6.1. Da demonstração desse teorema, conclui-se que isto pode ser feito obtendo-se um gerador $TEST$ tal que

$$L(TEST) = L(G) \cap P_1^{-1} P_1 E \cap P_2^{-1} P_2 E \tag{6.23}$$

e verificando se

$$L(TEST) = E \tag{6.24}$$

onde E é a linguagem-alvo em questão.

Estando esta condição satisfeita, verifica-se a condição (ii). Quanto à condição (i), no caso do presente problema, não existe especificação para uma linguagem minimamente aceitável. Esta situação não é incomum: em casos como este, uma linguagem minimamente aceitável mais restritiva do que a linguagem-alvo, se encontrada, poderia ser utilizada ela própria como linguagem-alvo. Desta forma, toma-se $A = \emptyset$. Torna-se então necessário substituir a condição (i) do teorema 6.1 por outro critério de aceitação para a solução encontrada. No caso do presente problema, este critério consiste em definir a seguinte *projeção de teste*:

$$P_i: \Sigma^* \rightarrow \Sigma_i^* \tag{6.25}$$

com

$$\Sigma_i = \Sigma - \{g, p\} \tag{6.26}$$

e aplicá-la à linguagem $L(\tilde{S}_1 \wedge \tilde{S}_2 / G)$ que, de acordo com a equação 6.7, representa o comportamento do sistema em nível global. A projeção P_i elimina de $L(\tilde{S}_1 \wedge \tilde{S}_2 / G)$ todos os eventos, com exceção de g e p . Assim, a solução obtida será considerada aceitável se

$$P_i L(\tilde{S}_1 \wedge \tilde{S}_2 / G) = L(LEGAL) \quad (6.27).$$

Para realizar as operações necessárias com os geradores, utiliza-se o programa TCT³. A tabela 6.1 descreve as funções utilizadas:

Nome da função	Resultado
$\text{sync}(G_1, G_2)$	Produto síncrono de G_1 e G_2 , conforme definido na seção 3.5
$\text{meet}(G_1, G_2)$	Gerador G tal que $L(G) = L(G_1) \cap L(G_2)$ e $L_m(G) = L_m(G_1) \cap L_m(G_2)$
$\text{project}(G_1, \text{ker})$	Gerador G tal que $L(G) = PL(G_1)$, sendo P a projeção de núcleo ker
$\text{selfloop}(G_1, \text{ker})$	Gerador G tal que $L(G) = P^{-1}L(G_1)$, sendo P a projeção de núcleo ker
$\text{min}(G_1)$	Gerador G que corresponde ao gerador com o menor número de estados tal que $L(G_1) = L(G)$. Esta operação é aplicada antes de se comparar geradores, para garantir que dois geradores com linguagens idênticas sejam reconhecidos como equivalentes
$\text{isomorph}(G_1, G_2)$	Permite determinar se dois geradores são idênticos a menos da numeração dos estados
$\text{condat}(G_{\text{planta}}, G_{\text{alvo}})$	Mapa de controle para o supervisor que implementa a linguagem do gerador G_{alvo} . Esta função pode ser empregada para determinar se $L(G_{\text{alvo}})$ é controlável, verificando-se se o mapa obtido requer ou não a inibição de eventos não controláveis
$\text{supcon}(G_{\text{planta}}, G_{\text{alvo}})$	Gerador G cuja linguagem gerada é a máxima sublinguagem controlável e $L_m(G)$ -compatível de $L(G_{\text{alvo}})$ e cuja linguagem marcada é igual a $L(G) \cap L_m(G_{\text{planta}})$

Tab. 6.1 - Algumas funções do programa TCT

³Preparado e cedido pelo Systems Control Group do Departamento de Engenharia Elétrica da Universidade de Toronto, Canadá.

6.5.1 CASO 1: TRANSMISSOR SOLUÇÃO E RECEPTOR SOLUÇÃO

Os geradores obtidos para este caso são:

- G_1 = sync(CANAL, sync(STX, SRX))
- G_{11} = project(G_1 , ker₁)
- G_{12} = project(G_1 , ker₂)
- E_1 = meet(G_1 , LEGAL)
- E_{11} = project(E_1 , ker₁)
- E_{12} = project(E_1 , ker₂)
- IE_{11} = selfloop(E_{11} , ker₁)
- IE_{12} = selfloop(E_{12} , ker₂)
- $TEST_1$ = meet(G_1 , meet(IE_{11} , IE_{12}))
- $PTEST_1$ = project($TEST_1$, ker₁).

O gerador G_1 da figura 6.10 representa a planta, obtida pela composição síncrona do transmissor solução, do canal de comunicação e do receptor solução.

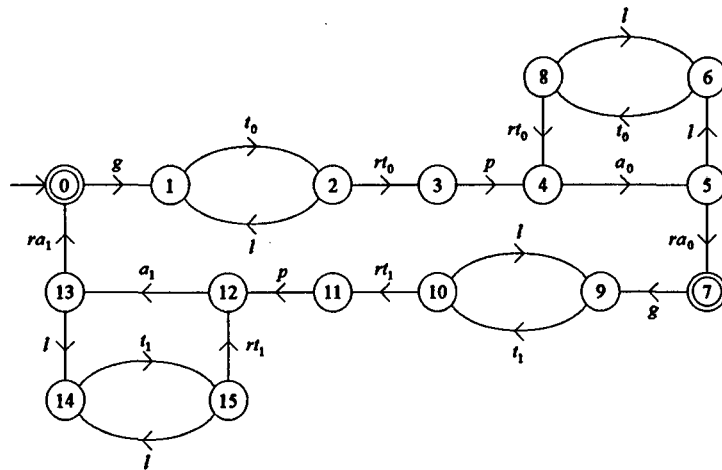


Fig. 6.10 - Gerador G_1

E_1 é o gerador da linguagem-alvo, que resulta ser idêntico a G_1 . Isto não é surpreendente neste caso, uma vez que se sabe que o transmissor e o receptor empregados são soluções para o problema.

E_{11} e E_{12} são as projeções da linguagem-alvo, de acordo com os pontos de vista do transmissor e do receptor, respectivamente. As figuras 6.11 e 6.12 mostram estes geradores, sendo clara a sua semelhança com os geradores STX e SRX.

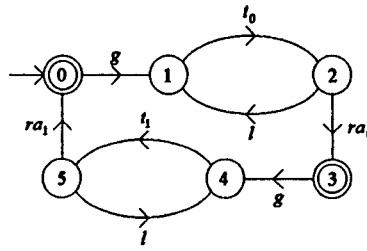


Fig. 6.11 - Gerador E_{11}

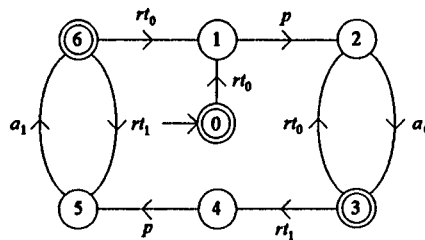


Fig. 6.12 - Gerador E_{12}

IE_{11} e IE_{12} são as projeções inversas de E_{11} e E_{12} , respectivamente, utilizados para determinar $TEST_1$, de acordo com o segundo membro da equação 6.23. Estes geradores não são reproduzidos aqui.

Um teste com a função isomorph mostra que $L(TEST_1) = L(E_1)$ e que, portanto, $L(E_1)$ é decomponível.

Um teste com a função condmat mostra que E_{11} e E_{12} são localmente controláveis. Com isso, a condição (ii) do teorema 6.1 está satisfeita e, portanto, existem supervisores locais $\langle S_1, \Phi_1 \rangle$ e $\langle S_2, \Phi_2 \rangle$ tais que

$$L(S_1 / G_1) = L(E_{11})$$

e

$$L(S_2 / G_2) = L(E_{12}).$$

S_1 e S_2 são idênticos a E_{11} e E_{12} , respectivamente. Os mapas de controle Φ_1 e Φ_2 são vazios neste caso, uma vez que $L(G_1) = L(E_1)$, não havendo necessidade de se inibir qualquer evento.

Note que, neste caso, de acordo com a equação 6.7, $TEST_1$ representa também o comportamento global obtido para o sistema. Como $L(TEST_1) = L(E_1)$, a solução obtida é obviamente aceitável. Conforme esperado, a aplicação da função isomorph à projeção $PTEST_1$ mostra que

$$L(PTEST_1) = L(LEGAL).$$

Fica demonstrado assim que o problema tem solução e que o comportamento global resultante corresponde àquele representado pela linguagem-alvo $L(E_1)$. Isto está de acordo com a formulação do problema GPG, que pede que $A \subseteq L(\tilde{S}_1 \wedge \tilde{S}_2 / G) \subseteq E$, com $E = L(E_1)$.

Neste ponto, pode-se chamar a atenção para a inconsistência do uso do problema GP em [Rudi92]: a solução obtida ali é idêntica à apresentada acima, o que contraria o enunciado do problema GP, que pede que $A \subseteq L(\tilde{S}_1 \wedge \tilde{S}_2 / G) \subseteq E$, onde $E = L_m(E_1)$. A observação da figura 6.10 (lembrando que $G_1 = E_1$) deixa claro que a linguagem gerada desse gerador não está contida na sua linguagem marcada, uma vez que há estados não marcados.

O fato se repete nos demais casos analisados, permitindo concluir que o objetivo daquele trabalho era resolver o problema como feito aqui, com a introdução de GPG.

Embora a solução acima esteja correta, não se pode afirmar que tenha havido a síntese de uma solução para o problema da transmissão seqüencial de dados, uma vez que se partiu de uma solução conhecida. O próximo passo é então utilizar uma planta menos estruturada, que não contém essa solução, e tentar obter o mesmo resultado.

6.5.2 CASO 2: TRANSMISSOR SOLUÇÃO E RECEPTOR GENÉRICO

Nesta seção combinam-se o transmissor STX e o receptor GRX para formar a planta. Pode-se argumentar que esta situação é muito artificial e que, na hipótese de um projetista chegar, por qualquer meio, ao transmissor solução, chegaria também sem muito esforço adicional ao receptor solução. Não obstante, este caso é o que melhor ilustra a síntese de supervisores neste exemplo. É também neste caso que são mais visíveis as conseqüências do uso inconsistente das linguagens geradas e marcadas em [Rudi92], conforme mostrado a seguir.

Os geradores obtidos para este caso são:

$$G_2 = \text{sync}(\text{CANAL}, \text{sync}(\text{STX}, \text{GRX}))$$

$$G_{21} = \text{project}(G_2, \text{ker}_1)$$

$$G_{22} = \text{project}(G_2, \text{ker}_2)$$

$$E_2 = \text{meet}(G_2, \text{LEGAL})$$

$$E_{21} = \text{project}(E_2, \text{ker}_1)$$

$$E_{22} = \text{project}(E_2, \text{ker}_2)$$

$$IE_{21} = \text{selfloop}(E_{21}, \text{ker}_1)$$

$$IE_{22} = \text{selfloop}(E_{22}, \text{ker}_2)$$

$$TEST_2 = \text{meet}(G_2, \text{meet}(IE_{21}, IE_{22})).$$

Como antes, G_2 representa a planta, mostrada na figura 6.13.

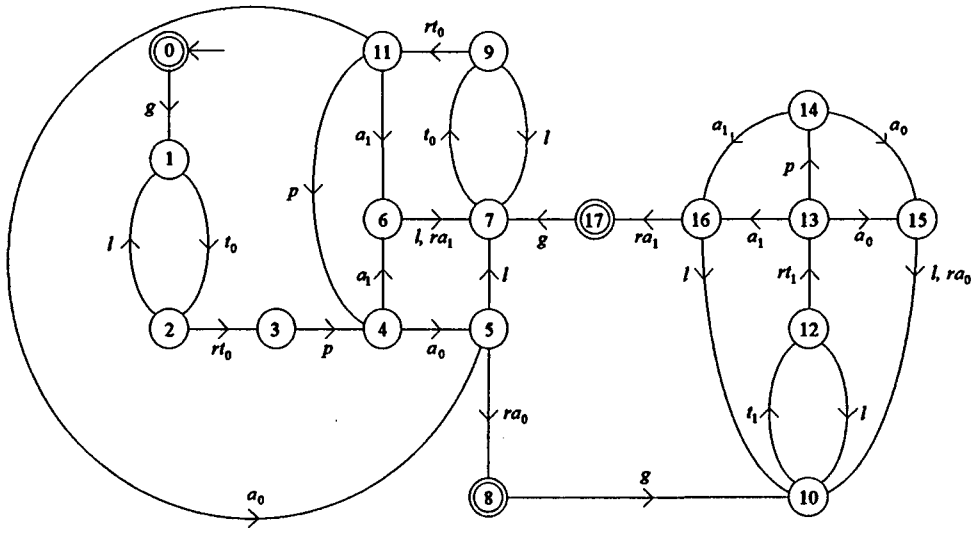


Fig. 6.13 - Gerador G_2

A linguagem-alvo é a do gerador E_2 , apresentado na figura 6.14.

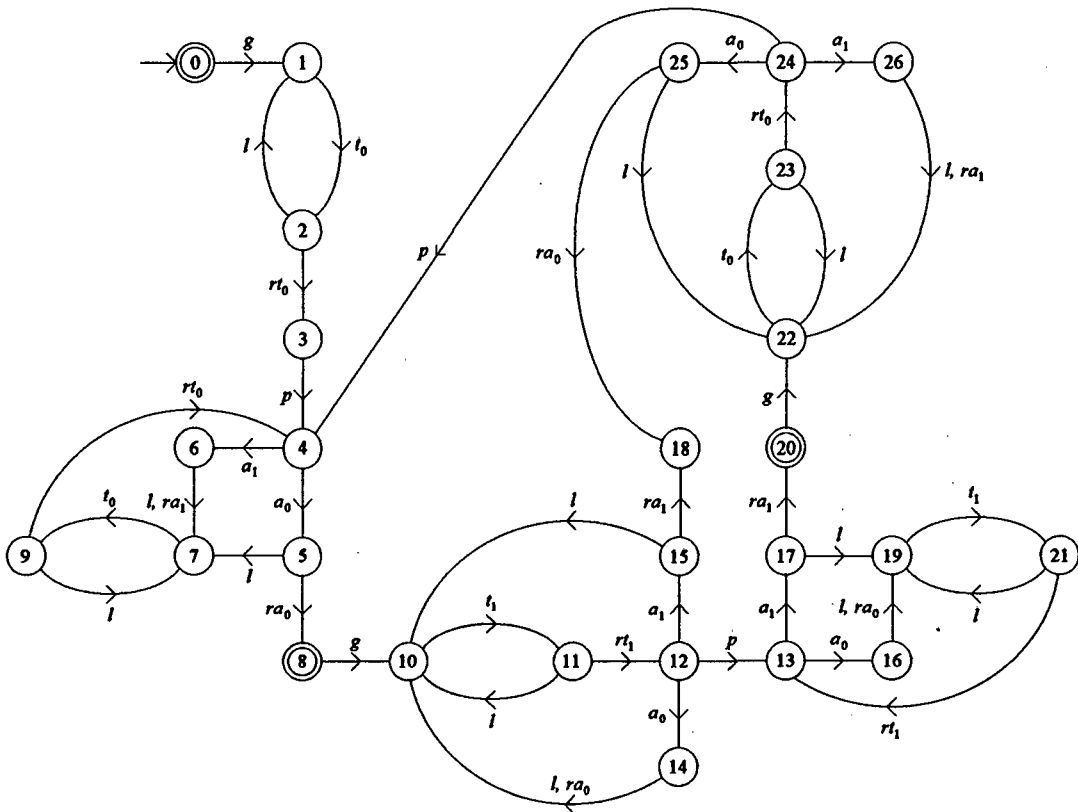


Fig. 6.14 - Gerador E_2

Obtêm-se em seguida as projeções E_{21} e E_{22} , mostradas nas figuras 6.15 e 6.16. Note que E_{21} é idêntico a STX.

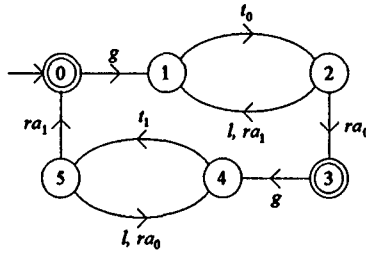


Fig. 6.15 - Gerador E_{21}

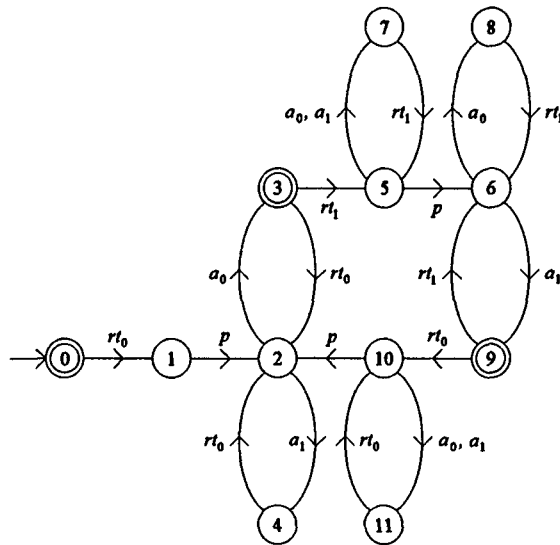


Fig. 6.16 - Gerador E_{22}

IE_{21} e IE_{22} , são as projeções inversas desses geradores, utilizadas para determinar $TEST_2$. Um teste com a função isomorph mostra que $L(TEST_2) \neq L(E_2)$ e que, portanto, E_2 não é decomponível. A figura 6.17 mostra o gerador $TEST_2$, destacando (linhas pontilhadas) suas diferenças com relação a E_2 .

É neste ponto que aparece a consequência mais importante do uso inconsistente das linguagens geradas e marcadas em [RW89b] e [Rudi92]. Conforme mencionado na seção 6.3, a solução do problema GP consiste em reduzi-lo a LP, que é formulado e resolvido em termos de linguagens geradas. Com isso, o teste de decomponibilidade a ser feito é o mesmo que se estabeleceu aqui, de acordo com as equações 6.23 e 6.24. As referências citadas, porém, não fazem esse teste pela comparação de linguagens geradas, mas sim através de demonstrações que levam em conta apenas a linguagem marcada dos geradores E_2 e $TEST_2$, ignorando as diferenças entre as linguagens geradas.

Acontece que, por coincidência,

$$L_m(TEST_2) = L_m(E_2) \tag{6.28}$$

o que leva [RW89b] e [Rudi92] à falsa conclusão de que E_2 é decomponível.

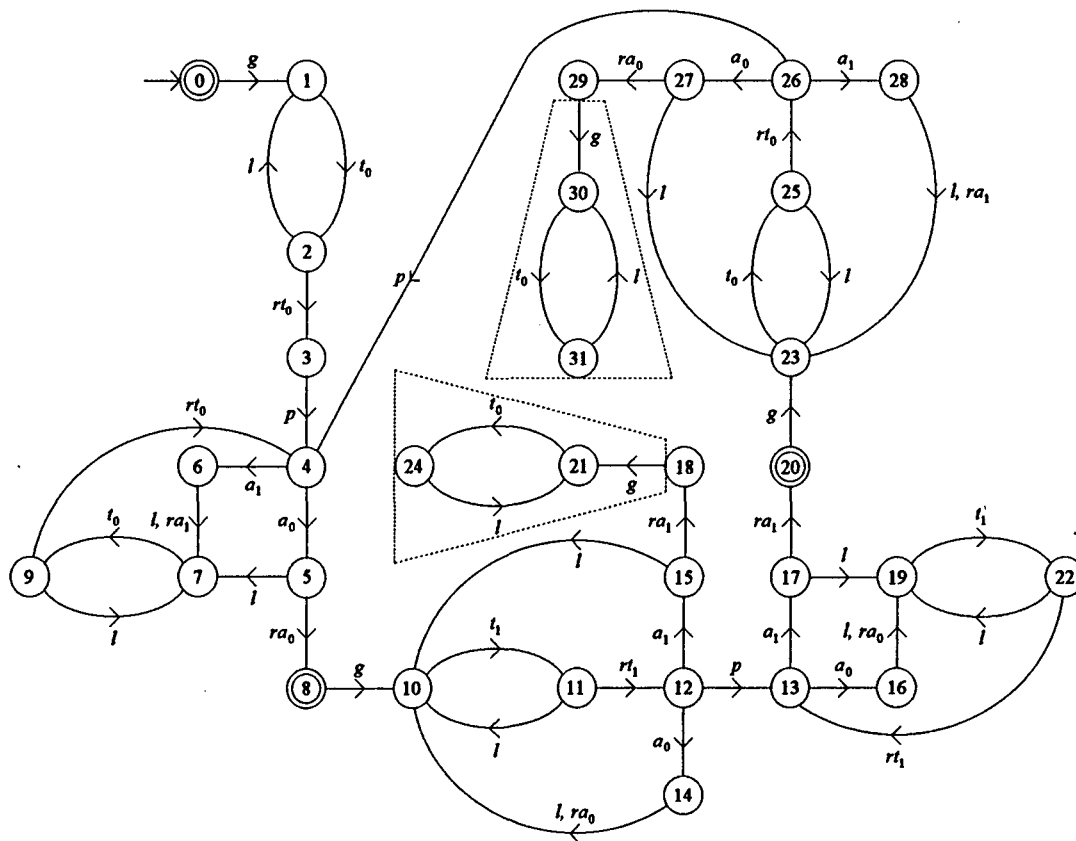


Fig. 6.17 - Gerador $TEST_2$

Com isso, a solução de [RW89b] e [Rudi92] prossegue. Verifica-se que $L(E_{21})$ é controlável, existindo portanto um supervisor $\langle S_1, \Phi_1 \rangle$ que implementa essa linguagem. Quanto ao receptor, um teste com condad mostra que $L(E_{22})$ é não controlável. Obtém-se então o gerador

$$SUPE_{22} = \text{supcon}(G_{22}, E_{22}),$$

mostrado na figura 6.18.

O comportamento global resultante, que segundo a equação 6.7 é dado por

$$L(RES_2) = L(G) \cap P_1^{-1} E_{21} \cap P_2^{-1} SUPE_{22} \tag{6.29}$$

é o representado pelo gerador RES_2 da figura 6.19. Este é obtido com os seguintes passos:

$$ISUPE_{22} = \text{selfloop}(SUPE_{22}, \text{ker}2)$$

$$RES_2 = \text{meet}(G_2, \text{meet}(IE_{21}, ISUPE_{22})).$$

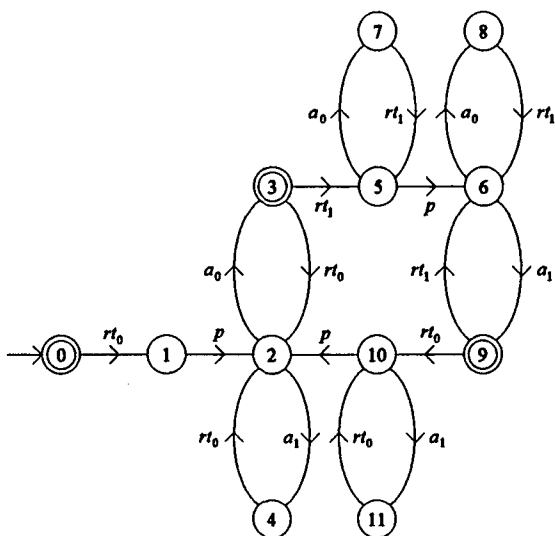


Fig. 6.18 - Gerador $SUPE_{22}$

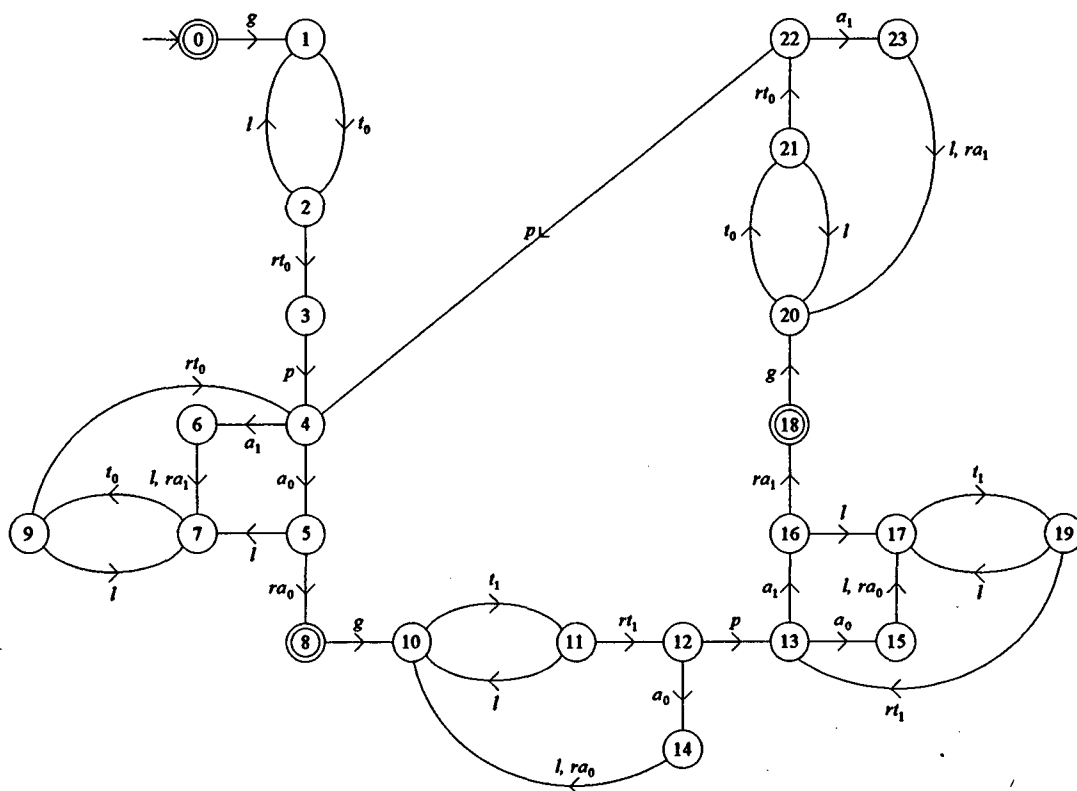


Fig. 6.19 - Gerador RES_2

O critério de aceitação definido pela equação 6.27 é satisfeito por RES_2 , de forma que o par $(E_{21}, SUPE_{22})$ é considerado em [RW89b] e [Rudi92] como solução para este caso.

Embora funcione, a obtenção dessa solução é questionável, uma vez que a condição de decomponibilidade não é satisfeita.

Portanto, também o problema GPG não tem uma solução para este caso. No entanto, é possível introduzir medidas para obter uma linguagem-alvo decomponível, baseadas no seguinte raciocínio: observando o gerador da figura 6.14, vê-se que este possui um estado sem transições de saída (estado 18) e que, portanto, é não coacessível e, com isso, bloqueante. É razoável supor que, na maioria das aplicações, se deseja que o sistema seja não bloqueante, donde segue que, ao invés de utilizar $L(E_2)$ como linguagem-alvo, poder-se-ia utilizar sua máxima sublinguagem $L_m(G)$ -compatível, denotada por $\text{sup } T[L(E_2)]$, definida no capítulo 5. Um gerador para esta linguagem pode ser obtido do gerador E_2 eliminando-se o estado 18 e os eventos a este ligados, ra_0 e ra_1 . Surge, então, um outro problema: estes eventos são não controláveis, de maneira que $\text{sup } T[L(E_2)]$ é uma linguagem não controlável. Isto significa que não há supervisores locais que possam implementá-la. Segue daí a idéia de se tomar como linguagem-alvo a máxima sublinguagem $L_m(G)$ -compatível e $L(G_2)$ -controlável de E_2 , cuja existência é assegurada pelo corolário 5.2.

Um gerador para esta linguagem, denominado E_{2tc} , pode ser obtido com a função supcon :

$$E_{2tc} = \text{supcon}(G_2, E_2).$$

Este gerador coincide com o gerador RES_2 , apresentado na figura 6.19. Empregando-se este gerador para representar a linguagem-alvo, obtêm-se, como antes, os demais geradores:

$$\begin{aligned} E_{2tc1} &= \text{project}(E_{2tc}, \ker_1) \\ E_{2tc2} &= \text{project}(E_{2tc}, \ker_2) \\ IE_{2tc1} &= \text{selfloop}(E_{2tc1}, \ker_1) \\ IE_{2tc2} &= \text{selfloop}(E_{2tc2}, \ker_2) \\ TEST_{2tc} &= \text{meet}(G_2, \text{meet}(IE_{2tc1}, IE_{2tc2})) \\ PTEST_{2tc} &= \text{project}(TEST_{2tc}, \ker_1). \end{aligned}$$

E_{2tc1} e E_{2tc2} são as projeções locais de E_{2tc} . Resulta que estes geradores são idênticos a E_{21} e $SUPE_{22}$, respectivamente (figuras 6.15 e 6.18) e portanto $L(E_{2tc1})$ e $L(E_{2tc2})$ são controláveis.

As projeções inversas IE_{2tc1} e IE_{2tc2} são utilizadas para determinar $TEST_{2tc}$, com o que se pode testar a decomponibilidade de $L(E_{2tc})$. A aplicação da função isomorph mostra que

$$L(TEST_{2tc}) = L(E_{2tc})$$

e que, portanto, a linguagem-alvo é decomponível.

Assim, existem supervisores $\langle S_1, \Phi_1 \rangle$ e $\langle S_2, \Phi_2 \rangle$ tais que

$$L(S_1 / G_1) = L(E_{2tc1})$$

e

$$L(S_2 / G_2) = L(E_{2tc2}).$$

$TEST_{2tc}$ representa também o comportamento global do sistema sob ação desses supervisores. Conforme verificado acima, a solução obtida é aceitável, uma vez que

$$L(PTEST_{2tc}) = L(LEGAL).$$

A utilização do problema GPG no lugar do problema GP e o emprego da máxima sublinguagem $L_m(G)$ -compatível e $L(G_2)$ -controlável de $L(E_2)$ como linguagem-alvo fornecem portanto uma maneira formalmente correta de se chegar à solução proposta na literatura. Outras sugestões para melhorar esta solução são dadas na seção 6.6.

6.5.3 CASO 3: TRANSMISSOR GENÉRICO E RECEPTOR SOLUÇÃO

Investiga-se a seguir a combinação do transmissor GTX (figura 6.7) e do receptor SRX (figura 6.6), para o que se obtêm os seguintes geradores:

$$G_3 = \text{sync}(\text{CANAL}, \text{sync}(\text{GTX}, \text{SRX}))$$

$$G_{31} = \text{project}(G_3, \text{ker}_1)$$

$$G_{32} = \text{project}(G_3, \text{ker}_2)$$

$$E_3 = \text{meet}(G_3, \text{LEGAL})$$

$$E_{31} = \text{project}(E_3, \text{ker}_1)$$

$$E_{32} = \text{project}(E_3, \text{ker}_2)$$

$$IE_{31} = \text{selfloop}(E_{31}, \text{ker}_1)$$

$$IE_{32} = \text{selfloop}(E_{32}, \text{ker}_2)$$

$$TEST_3 = \text{meet}(G_3, \text{meet}(IE_{31}, IE_{32})).$$

Os geradores acima não são apresentados aqui. Um teste com isomorph mostra que $L(TEST_3) \neq L(E_3)$ e que, portanto, $L(E_3)$ não é decomponível. Pode-se novamente aplicar a sugestão da subseção anterior, tomando a máxima sublinguagem $L_m(G)$ -compatível e $L(G_3)$ -controlável de $L(E_3)$ como alternativa para a linguagem-alvo, mas esta se mostra igualmente não decomponível e portanto, neste caso, o problema fica sem solução.

6.5.4 CASO 4: TRANSMISSOR GENÉRICO E RECEPTOR GENÉRICO

Utilizam-se aqui o transmissor GTX e o receptor GRX para formar a planta. Como já no caso anterior o problema não apresentou solução, é de se esperar que o mesmo aconteça aqui. Os geradores necessários para testar a decomponibilidade da linguagem-alvo são:

$$\begin{aligned}
G_4 &= \text{sync}(\text{CANAL}, \text{sync}(\text{GTX}, \text{SRX})) \\
G_{41} &= \text{project}(G_4, \text{ker}_1) \\
G_{42} &= \text{project}(G_4, \text{ker}_2) \\
E_4 &= \text{meet}(G_4, \text{LEGAL}) \\
E_{41} &= \text{project}(E_4, \text{ker}_1) \\
E_{42} &= \text{project}(E_4, \text{ker}_2) \\
IE_{41} &= \text{selfloop}(E_{41}, \text{ker}_1) \\
IE_{42} &= \text{selfloop}(E_{42}, \text{ker}_2) \\
TEST_4 &= \text{meet}(G_4, \text{meet}(IE_{41}, IE_{42})).
\end{aligned}$$

De fato, um teste com isomorph mostra que $L(TEST_4) \neq L(E_4)$. Além disso, $L(E_4)$ é $L(G_4)$ -controlável e $L_m(G)$ -compatível, de modo que não adianta tentar aplicar a sugestão desenvolvida na subseção 6.5.2, ficando o problema por ora sem solução.

Os geradores G_4 e E_4 são apresentados na seção 6.6 (figuras 6.22 e 6.23), onde se continua a discussão deste caso.

6.6 CAMINHOS OPCIONAIS

A solução obtida na subseção 6.5.2 é criticável sob o seguinte aspecto: embora o receptor sintetizado, representado pelo gerador $SUPE_{22}$ (figura 6.18), atenda as especificações do problema, ele pode fazer com que o sistema demore um tempo arbitrariamente longo para conseguir atingir um estado marcado e, com isso, completar uma tarefa (que consiste, neste caso, em aceitar uma mensagem na origem e entregá-la no destino). Por exemplo, o receptor poderia ficar oscilando entre os estados 2 e 4, escolhendo sempre a mensagem a_1 para confirmar uma recepção com rt_0 , o que obrigaria o transmissor a repetir sempre a mesma mensagem, impedindo o progresso da transmissão. Na prática, esta solução seria, portanto, inaceitável. Conforme observado em [Rudi92], subseção 5.2.4, isto se dá porque a formulação do problema da transmissão seqüencial de dados não especifica o que se entende pelo progresso da transmissão. De fato, o modelo RW não inclui outra forma de especificar um objetivo além da linguagem-alvo e da linguagem mínima aceitável.

Esta seção introduz dois novos conceitos na teoria, a saber o de *caminho opcional* no grafo de um gerador e o de *componente eficaz de um gerador*, que se mostram úteis para realizar esse tipo de especificação. O método desenvolvido, embora computacionalmente complexo, não se aplica somente ao problema da transmissão seqüencial de dados, mas também a problemas semelhantes de síntese em controle supervisorio.

Parte-se da observação do gerador da figura 6.18. Do estado 2 desse gerador saem os eventos a_0 e a_1 . O primeiro leva a um estado marcado, o que significa que este caminho leva a uma tarefa completada. Já o segundo leva ao estado 4, a partir do qual só se pode retornar ao estado 2. Como nenhum destes estados é marcado, não se completa qualquer tarefa com a realização desse ciclo. Além disso, o estado 3, que é o próximo estado marcado, pode ser atingido passando-se zero ou mais vezes pelo ciclo 2 - 4 - 2, sempre com o mesmo resultado. Naturalmente, o sistema será mais eficiente se este ciclo for evitado. Uma vez que o evento a_1 , que dá início ao ciclo, é controlável, pode-se modificar o mapa de controle do supervisor local que implementa $L(SUPE_{22})$ e inibi-lo, retirando assim o estado 4 do gerador. Raciocínio análogo se aplica aos estados 7, 8 e 11. Isto é motivação para as definições que seguem.

Def. 6.6: Dado um gerador acessível $G = \langle \Sigma, Q, \delta, q_0, Q_m \rangle$, um *caminho opcional* no grafo deste gerador é qualquer caminho fechado da forma $q_i \sigma_1 q_{i+1} \sigma_2 q_{i+2} \dots \sigma_n q_f$, $q_f = q_i \in Q$, se e somente se o conjunto de estados marcados do gerador

$$G' = Ac(\langle \Sigma, Q, \delta', q_0, Q_m \rangle)^4, \text{ com}$$

$$\delta'(\sigma, q) = \begin{cases} \text{indefinida se } \sigma = \sigma_1 \wedge q = q_i \\ \delta(\sigma, q) & \text{nos demais casos} \end{cases}$$

for idêntico a Q_m .

Portanto, um caminho opcional é um caminho que começa e termina em um mesmo estado e tal que a retirada do seu primeiro evento do diagrama de transição de estados não impede o acesso a qualquer um dos estados marcados.

Utiliza-se deste ponto em diante a terminologia "eliminação de um caminho opcional" para indicar a remoção da transição $q_i \xrightarrow{\sigma_1} q_i$, seguida da retirada dos estados que tenham se tornado inacessíveis e das transições a estes ligadas.

Ex. 6.1: Considere o gerador da figura 6.20.

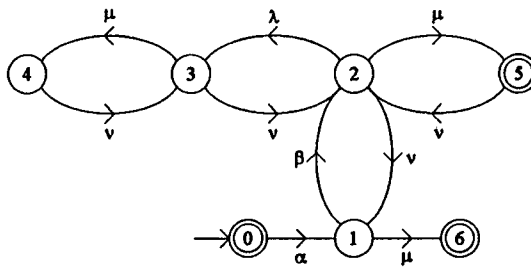


Fig. 6.20 - Gerador com caminhos opcionais

⁴Para um gerador qualquer H , $Ac(H)$ denota a componente acessível de H , de acordo com a definição 3.23.

O caminho $2\lambda 3\mu 4\nu 3\nu 2$ é opcional, pois a sua eliminação (que implica na remoção dos estados 3 e 4) não impede o acesso aos estados marcados. O mesmo vale para o caminho $3\mu 4\nu 3$. Já os caminhos que incluem o evento β , tais como $1\beta 2\lambda 3\nu 2\nu 1$ ou $2\nu 1\beta 2$, não são opcionais, uma vez que sua eliminação impediria o acesso ao estado 5. ♦

Def. 6.7: Um gerador é dito *irreduzível com respeito a caminhos opcionais* ou simplesmente *irreduzível* sse não possuir qualquer caminho opcional.

Def. 6.8: Dado um gerador acessível $G = \langle \Sigma, Q, \delta, q_0, Q_m \rangle$, uma *componente irreduzível de G* é qualquer gerador obtido de G por eliminação sucessiva de caminhos opcionais, até que o gerador resultante seja irreduzível com respeito a caminhos opcionais.

A exigência de que a eliminação dos caminhos opcionais seja feita em seqüência é necessária porque um caminho considerado opcional pode deixar de sê-lo após a eliminação de outro. Nestes casos, o gerador em questão tem mais de uma componente irreduzível e os caminhos opcionais são ditos *dependentes*. O exemplo seguinte ilustra essa situação.

Ex. 6.2: Seja o gerador da figura 6.21.

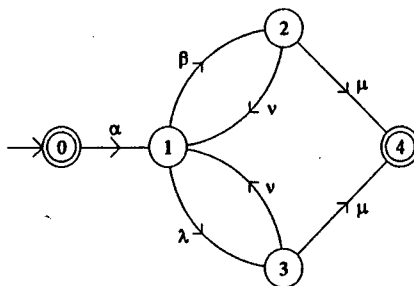


Fig. 6.21 - Gerador com duas componentes irreduzíveis

Considere os caminhos opcionais $1\beta 2\nu 1$ e $1\lambda 3\nu 1$. Não se pode eliminar ambos ao mesmo tempo, pois a retirada de qualquer um deles faz com que o outro deixe de ser opcional. Assim, o gerador acima tem duas componentes irreduzíveis, obtidas eliminando-se ou um ou outro dos caminhos citados. ♦

Um caso particular importante é aquele em que o primeiro evento de um caminho opcional é controlável, levado em conta pelas definições seguintes:

Def. 6.9: Um caminho opcional num gerador G é dito um *caminho opcional controlável* sse o primeiro evento deste caminho for controlável.

Def. 6.10: Um gerador é dito *irreduzível com relação a caminhos controláveis* sse não possuir qualquer caminho opcional controlável.

Def. 6.11: Dado um gerador acessível $G = \langle \Sigma, Q, \delta, q_0, Q_m \rangle$, uma *componente irreduzível controlável de G* é qualquer gerador obtido de G por eliminação sucessiva de caminhos opcionais controláveis, até que o gerador resultante seja irreduzível com relação a caminhos controláveis.

Com isso, pode-se melhorar a solução encontrada para o receptor na subseção 6.5.2, substituindo o gerador da figura 6.18 por sua componente irreduzível com relação a caminhos controláveis⁵ que, neste caso, é única. Resulta daí a eliminação dos estados 4, 7, 8 e 11. O gerador obtido pode ainda ser minimizado, com o que se obtém a mesma solução que no caso 1, ou seja, o gerador da figura 6.12.

Este exemplo mostra que a eliminação de caminhos opcionais do gerador solução pode ter um efeito altamente desejável, melhorando o desempenho da solução encontrada, que passa a ser aceitável do ponto de vista prático.

É natural perguntar em seguida o que acontece se o mesmo for feito com a linguagem-alvo global, antes do teste de decomponibilidade. Teoricamente, é possível que uma linguagem não decomponível passe a sê-lo quando se eliminam alguns de seus caminhos opcionais.

Cabe observar, porém, que mesmo que se consiga a decomponibilidade, o problema pode continuar sem solução. Um caminho opcional não é necessariamente um caminho supérfluo, e sua eliminação afeta as linguagens gerada e marcada do gerador em questão, podendo portanto fazer com que a linguagem resultante deixe de satisfazer o critério de aceitação estabelecido para o problema. Apesar disso, resultados interessantes podem ser conseguidos, conforme mostrado a seguir.

Seja novamente o problema da transmissão seqüencial de dados. Conforme mostrado na seção 6.5, não foi possível encontrar soluções para os casos 3 e 4, envolvendo o transmissor genérico. Destes, o mais difícil é sem dúvida o último, por não se apoiar, em qualquer ponto, no conhecimento do protocolo do bit alternante. O gerador G_4 , representando a planta para este caso, é mostrado na figura 6.22.

⁵Neste caso particular, poder-se-ia utilizar simplesmente a componente mínima com relação a caminhos opcionais desse gerador, com o mesmo resultado. No caso geral, porém, a linguagem do gerador assim obtido pode não ser controlável, exigindo sua restrição à máxima sublinguagem controlável, o que pode ou não ser aceitável, dependendo do problema em questão.

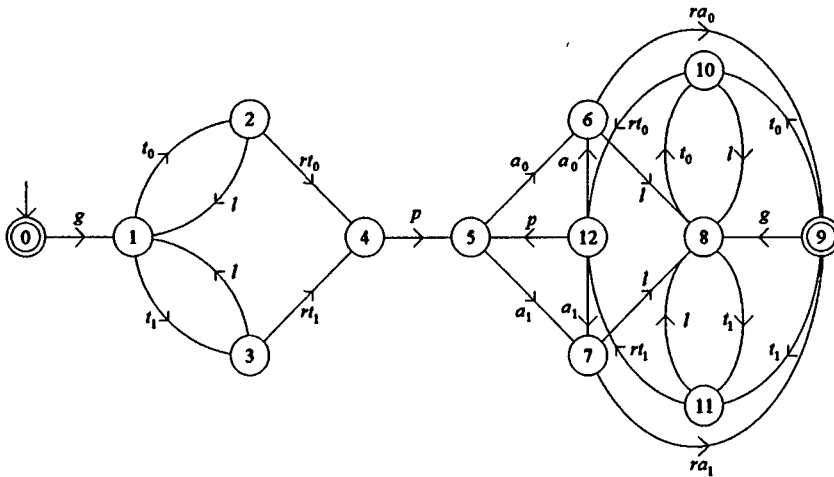


Fig. 6.22 - Gerador G_4

O gerador E_4 , que representa a linguagem-alvo global para este caso, é dado na figura 6.23.

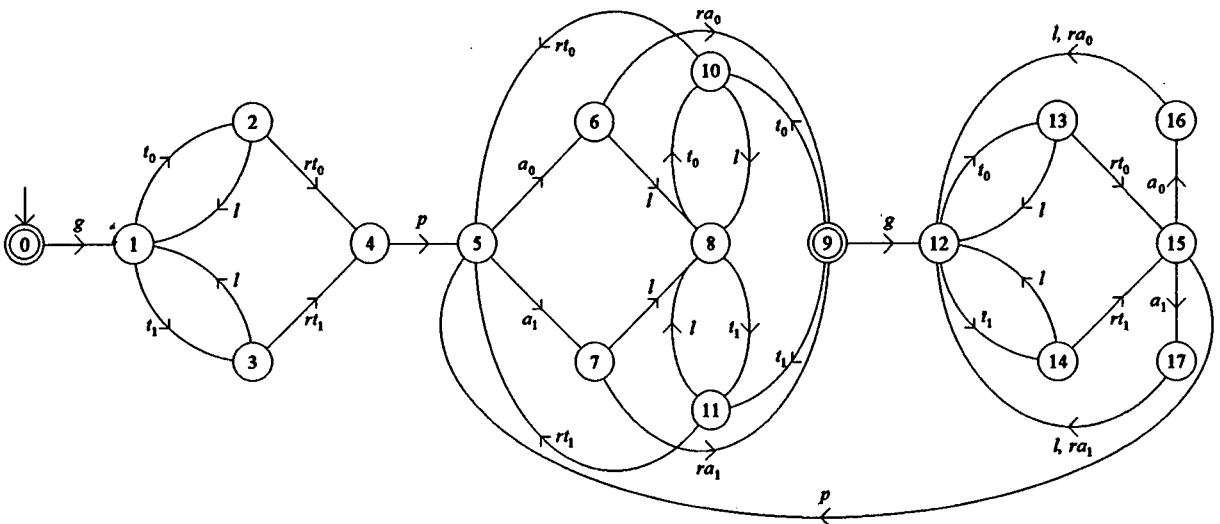


Fig. 6.23 - Gerador E_4

A observação desta figura permite concluir que, embora o gerador tenha vários caminhos opcionais controláveis, estes não são independentes, de modo que não existe uma componente irredutível única. Mesmo que fosse este o caso, a melhor solução poderia consistir na eliminação de apenas parte dos caminhos opcionais, e não de todos. Desta forma, não havendo, ao menos por ora, um critério para guiar a escolha dos caminhos opcionais a serem eliminados, o que se pode fazer é determinar todos os caminhos opcionais e eliminá-los em todas as combinações e seqüências possíveis, testando a decomponibilidade e a aceitabilidade da linguagem de cada gerador obtido. Não se discute aqui o custo computacional de tal

método, que certamente se torna proibitivo para sistemas com muitos caminhos opcionais, pois é visível que o número de alternativas a analisar cresce de forma não polinomial com o número desses caminhos.

Uma análise exaustiva do exemplo, como descrita acima, está fora do escopo do presente trabalho. Foram feitas algumas tentativas, sem que se conseguisse, porém, obter uma linguagem decomponível.

Isto, no entanto, não esgota ainda as possibilidades da aplicação da eliminação de caminhos opcionais. O gerador E_4 é um gerador minimizado, isto é, um gerador tal que não há outro equivalente e com menor número de estados. Há, porém, um número infinito de geradores equivalentes com mais estados, sendo um destes o gerador X_4 , obtido do anterior por duplicação dos estados 5 a 17 e mostrado na figura 6.24. A equivalência pode ser comprovada com o programa TCT: a minimização de X_4 leva de volta ao gerador E_4 .

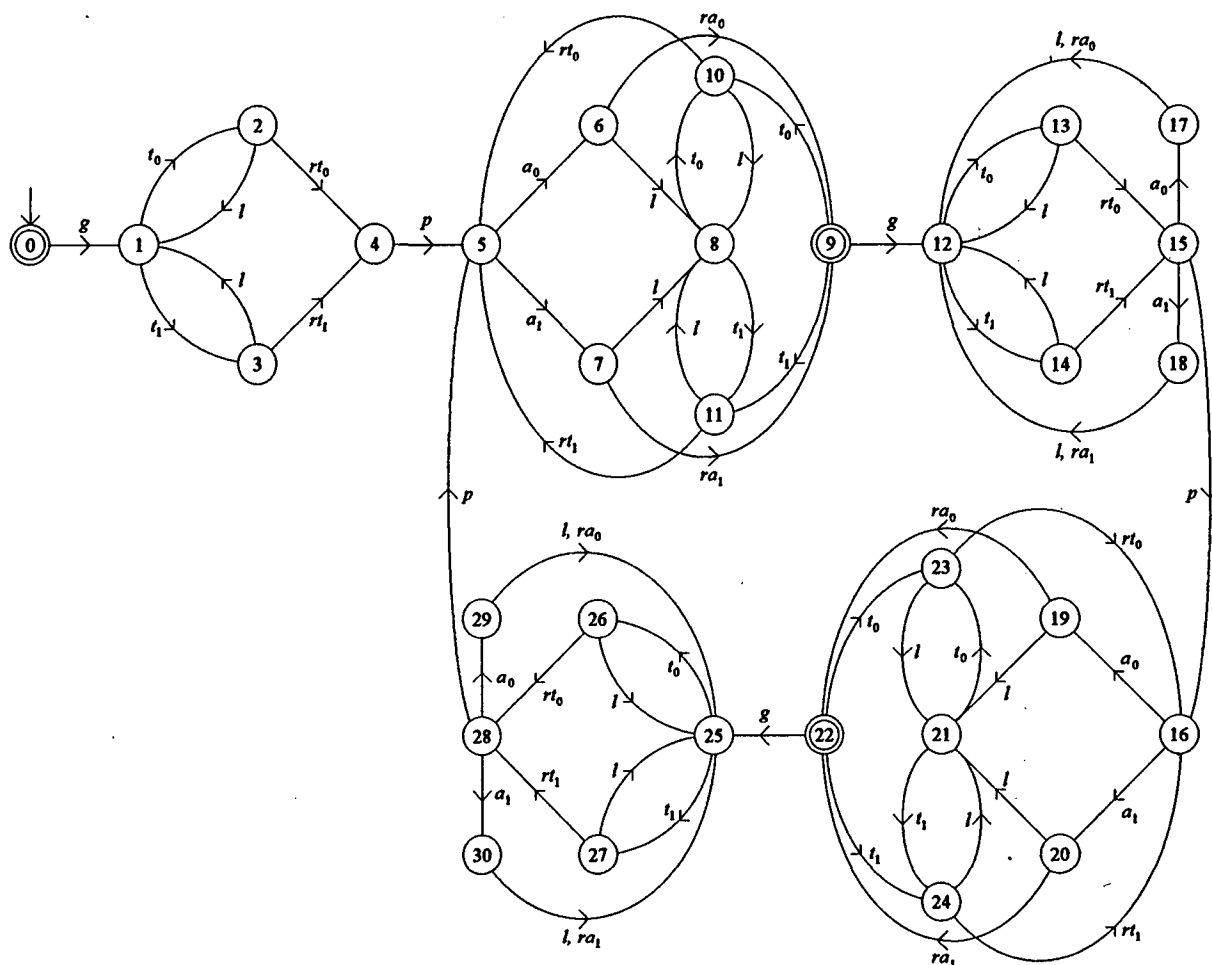


Fig. 6.24 - Gerador X_4

Portanto, pode-se aplicar a X_4 o procedimento exaustivo sugerido acima, o que, como no caso anterior, não é feito aqui. Considere, no entanto, o gerador Y_4 da figura 6.25, obtido

de X_4 por eliminação dos caminhos opcionais controláveis que começam com as seguintes transições:

- $1 \rightarrow t_1 \rightarrow 3$ $5 \rightarrow a_1 \rightarrow 7$ $8 \rightarrow t_1 \rightarrow 11$ $9 \rightarrow t_0 \rightarrow 10$ $12 \rightarrow t_0 \rightarrow 13$
- $15 \rightarrow a_0 \rightarrow 17$ $15 \rightarrow a_1 \rightarrow 18$ $16 \rightarrow a_0 \rightarrow 19$ $21 \rightarrow t_0 \rightarrow 23$ $22 \rightarrow t_1 \rightarrow 24$
- $25 \rightarrow t_1 \rightarrow 27$ $28 \rightarrow a_0 \rightarrow 29$ $28 \rightarrow a_1 \rightarrow 30$.

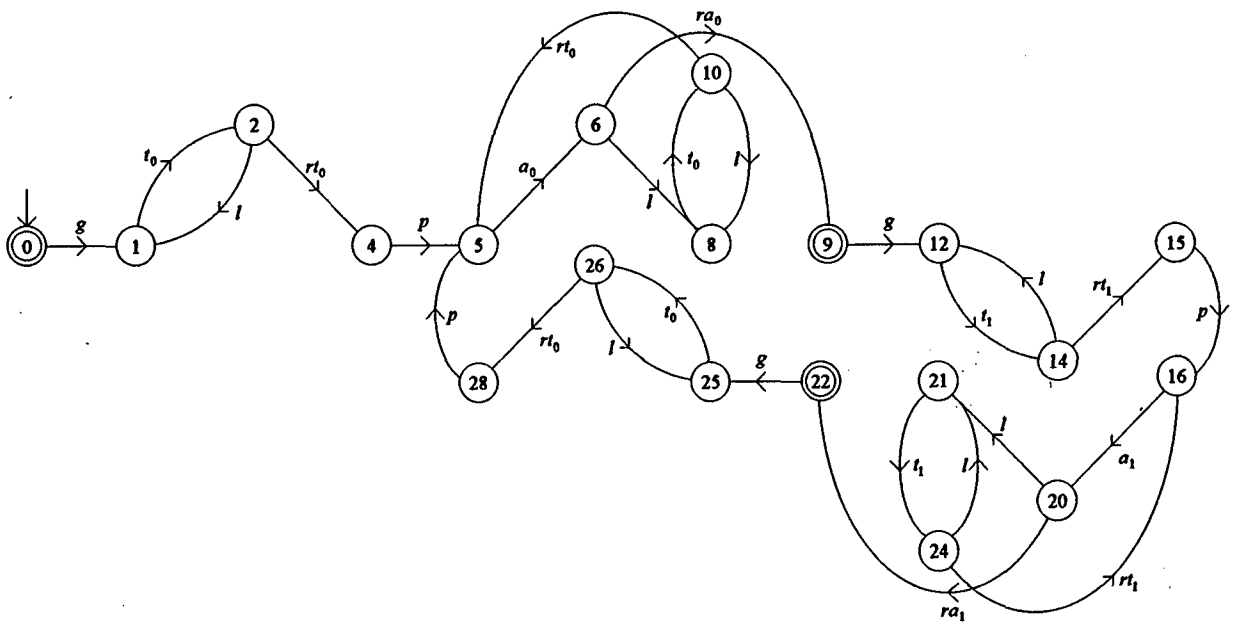


Fig. 6.25 - Gerador Y_4

Obtendo-se, como na seção 6.5, os geradores:

$$Y_{41} = \text{project}(Y_4, \ker_1)$$

$$Y_{42} = \text{project}(Y_4, \ker_2)$$

$$IY_{41} = \text{selfloop}(Y_{41}, \ker_1)$$

$$IY_{42} = \text{selfloop}(Y_{42}, \ker_2)$$

$$TESTY_4 = \text{meet}(G_4, \text{meet}(IY_{41}, IY_{42})).$$

$$PTESTY_4 = \text{project}(TESTY_4, \ker_1), \text{ verifica-se que}$$

$$L(TESTY_4) = L(Y_4) \quad \text{e que, portanto, esta linguagem é}$$

decomponível.

Além disso, $L(PTESTY_4) = L(LEGAL)$, de modo que a solução é aceitável. Os comportamentos locais, representados por Y_{41} para o transmissor e Y_{42} para o receptor,

coincidem com os geradores E_{11} e E_{12} , respectivamente (figuras 6.11 e 6.12), os quais representam a especificação do protocolo do bit alternante.

Desta forma, existem supervisores $\langle S_1, \Phi_1 \rangle$ e $\langle S_2, \Phi_2 \rangle$ tais que

$$L(S_1 / G_1) = L(Y_{41}) = L(E_{11})$$

e

$$L(S_2 / G_2) = L(Y_{41}) = L(E_{12}).$$

Os geradores S_1 e S_2 são portanto idênticos a E_{11} e E_{12} , respectivamente, de modo que se obtém o mesmo comportamento que no caso 1 (subseção 6.5.1). Os mapas de controle, porém, não são vazios como naquele caso, uma vez que a planta G_4 é capaz de realizar ações que violariam a especificação do problema e portanto requer que se inibam alguns eventos. Estes mapas são dados nas tabelas 6.2 e 6.3, respectivamente.

Φ_1	g	t_0	t_1
0	1	0	0
1	-	1	0
2	-	-	-
3	1	0	0
4	-	0	1
5	-	-	-

Tab. 6.2 - Mapa de controle Φ_1

Φ_2	p	a_0	a_1
0	-	-	-
1	1	0	0
2	0	1	0
3	-	-	-
4	1	0	0
5	0	0	1
6	-	-	-

Tab. 6.3 - Mapa de controle Φ_2

A observação de Y_4 permite concluir que este gerador é equivalente à expressão global do protocolo do bit alternante. De fato, foi com base nesta solução que se escolheram

os caminhos opcionais a eliminar para a sua obtenção. Mais ainda, uma minimização desse gerador com o programa TCT tem como resultado o gerador E_1 , obtido no caso 1 (subseção 6.5.1). Isto, porém, não afeta a generalidade da solução encontrada, uma vez que o gerador Y_4 pertence à família dos geradores obtidos de X_4 por eliminação de caminhos opcionais e seria encontrado também numa busca exaustiva. A principal contribuição desta seção está em mostrar que *é possível obter o protocolo do bit alternante partindo-se de uma planta que não incorpora essa solução, desde que se escolha uma linguagem-alvo adequada.*

Portanto, o mesmo procedimento pode ser uma alternativa interessante para tratar problemas semelhantes, sempre que o esforço computacional requerido for tolerável. Cabe frisar ainda que este esforço cresce com o número de caminhos opcionais, e não necessariamente com o número de estados.

6.7 CONCLUSÃO DO CAPÍTULO

O capítulo apresentou os conceitos necessários ao estudo do modelo RW descentralizado, voltando-se para os problemas abstratos de síntese LP e GP, encontrados na literatura. Contribui com a teoria ao analisar e reformular o problema de controle supervisorio descentralizado com especificação global (GP), que se mostra inconsistente com a aplicação encontrada na literatura.

Além disso, a introdução do conceito de *caminho opcional* oferece uma maneira de buscar soluções para problemas que, a princípio, têm uma linguagem-alvo não decomponível, conforme ilustrado com a solução do exemplo do protocolo do bit alternante no caso mais geral, não resolvido na literatura pesquisada.

O método sugerido, embora correto, tem a desvantagem de ser computacionalmente custoso, uma vez que o número de possibilidades a analisar cresce de forma não polinomial com o número de caminhos opcionais no grafo do gerador da linguagem-alvo. Em função disso, não se pode afirmar que a solução sugerida seja aceitável em todas as situações. Não obstante, ela o é para a classe de problemas onde o número de caminhos opcionais se mantém dentro dos limites impostos pelos recursos computacionais disponíveis. Como em outros problemas não polinomiais, pode-se ainda tentar encontrar métodos heurísticos para guiar a escolha dos caminhos opcionais a eliminar. Tal tarefa está fora do escopo da presente dissertação, ficando como sugestão para trabalhos futuros.

CONCLUSÃO

O trabalho trata uma classe de sistemas de grande interesse para a tecnologia moderna, conhecidos como sistemas a eventos discretos. Apesar dos muitos esforços, não se conseguiu ainda criar um modelo único para tratar formalmente esses sistemas. Dentre os modelos existentes, apresentados no capítulo 2, destaca-se o de Ramadge e Wonham (RW), objeto dos demais capítulos.

Com o estudo realizado foi possível adquirir um conhecimento profundo do modelo RW básico e do modelo descentralizado, além de noções gerais sobre as demais extensões, não abordadas aqui. O objetivo inicial de estudar as técnicas de obtenção de supervisores evoluiu para uma análise detalhada das condições de existência destes, o que levou às contribuições teóricas do capítulo 5. O capítulo 6, além de apresentar contribuições teóricas para o modelo de controle supervísório descentralizado, ilustra a aplicação do modelo RW a um problema da área da Informática.

As principais contribuições do presente trabalho estão resumidas no quadro abaixo:

- * a introdução da definição de compatibilidade;
- * o estabelecimento de um critério para a ausência de bloqueio empregando a linguagem gerada pelo sistema sob supervisão e a linguagem marcada da planta;
- * a utilização da informação representada pelos estados marcados na formulação de problemas em termos de linguagens geradas;
- * a introdução da classe de sublinguagens $L_m(G)$ -compatíveis de uma linguagem e a demonstração da existência de um elemento supremo nesta classe;
- * a obtenção de uma fórmula para a máxima sublinguagem $L_m(G)$ -compatível de uma dada linguagem;
- * a formulação e a solução de um novo problema de síntese de supervisores (PCSII);
- * a introdução do teorema que estabelece a dualidade entre a formulação de problemas encontrada na literatura e a formulação introduzida para PCSII ;
- * a análise crítica do problema de controle supervísório descentralizado, utilizando as contribuições anteriores para eliminar as inconsistências encontradas;
- * a introdução do conceito de caminhos opcionais no grafo de um gerador, com o qual se obtém a solução do problema do protocolo do bit alternante para o caso geral, sem solução na literatura pesquisada.

Em face dos resultados alcançados, consideram-se atingidos os objetivos inicialmente propostos.

Durante a preparação deste trabalho houve várias ocasiões em que foi necessário deixar de lado, em virtude das limitações de tempo e de abrangência de uma tese de mestrado, alternativas de pesquisa que pareceram interessantes e promissoras. Estas se traduzem nas seguintes sugestões para trabalhos futuros:

i. Investigar os algoritmos existentes para o cálculo das máximas sublinguagens de uma linguagem. Conforme mencionado na conclusão do capítulo 4, existem na literatura métodos de cálculo da máxima sublinguagem $L(G)$ -controlável, mas não se conhece ainda um algoritmo para a máxima sublinguagem $L(G)$ -controlável e $L_m(G)$ -fechada, que poderia ser desenvolvido.

ii. Estudar as extensões do modelo RW, descritas no capítulo 2. A intensa atividade de pesquisa nesta área permite afirmar que estas podem ser utilizadas como fonte para vários outros trabalhos, com boas chances de se fazer contribuições à teoria.

iii. Discutir a aplicabilidade do modelo RW à Engenharia de Software. Aqui há duas áreas a considerar: primeiro, a composição síncrona de geradores pode ser utilizada na formalização de especificações, sobretudo na obtenção de representações globais de sistemas modulares, a partir da representação do comportamento de cada módulo. Em segundo lugar, pode-se interpretar a máxima linguagem controlável como representando a parte realizável de uma especificação de comportamento. O fato de ser possível sintetizar essa linguagem pode facilitar ou mesmo dispensar a verificação de software, uma vez que a especificação obtida é sempre correta.

iv. Explorar o fato de ser indiferente habilitar ou não eventos fisicamente impossíveis para a minimização de supervisores (equação 4.16).

v. Investigar a questão do bloqueio no modelo RW descentralizado.

vi. Conforme sugerido no final do capítulo 6, tentar encontrar um método heurístico para guiar a escolha de caminhos opcionais a eliminar do gerador da linguagem-alvo, na tentativa de obter uma sublinguagem decomponível.

vii. Ainda com relação ao mesmo problema, demonstra-se em [Rudi92] que, dadas uma linguagem-alvo qualquer E e o conjunto das sublinguagens decomponíveis de E , denotado por $D(E)$, não existe, em geral, uma máxima sublinguagem decomponível dentro dessa linguagem, uma vez que a condição de decomponibilidade não é fechada sob a operação de união. No entanto, a união de dois elementos de $D(E)$ pode ser decomponível. Este fato permite conjecturar que pode haver vários "máximos locais" decomponíveis, obtidos pela união de alguns elementos de $D(E)$, mas não de todos. Seria interessante então verificar se o número desses máximos locais é finito e, se for, obter um método que produza os geradores das linguagens correspondentes. Estes poderiam ser então analisados quanto à sua utilidade como solução.

REFERÊNCIAS BIBLIOGRÁFICAS

- [BH90] Brave, Y.; Heymann, M.: "Stabilization of Discrete-Event Processes". *International Journal of Control*, 51(5):1101-1117 (1990)
- [BW92] Brandin, B. A.; Wonham, W. M.: "Supervisory Control of Timed Discrete-Event Systems". *Systems Control Group Report No. 9210*, Department of Electrical Engineering, University of Toronto, Toronto, Canada (1992)
- [BWB92] Brandin, B. A.; Wonham, W. M.; Benhabib, B.: "Modular Supervisory Control of Timed Discrete-Event Systems". *Allerton Communication and Control Conference*, Urbana-Champaign, Illinois (Sept. 1992)
- [BGK+90] Brandt, R. D.; Garg, V.; Kumar, R.; Lin, F.; Marcus, S. I.; Wonham, W. M.: "Formulas for Calculating Supremal Controllable and Normal Sublanguages". *Systems & Control Letters*, 15(2):111-117 (1990)
- [CDFV88] Cieslak, R.; Desclaux, C.; Fawaz, A. S.; Varaiya, P.: "Supervisory Control of Discrete-Event Processes with Partial Observations". *IEEE Transactions on Automatic Control*, 33(3):249-260 (1988)
- [CDQV85] Cohen, Guy; Dubois, Didier; Quadrat, Jean Pierre; Viot, Michel: "A Linear System Theoretic View of Discrete Event Processes and its use for Performance Evaluation in Manufacturing". *IEEE Transactions on Automatic Control*, 30(3):210-220 (1985)
- [CH90] Cao, Xi-Ren; Ho, Yu-Chi: "Models of Discrete Event Dynamic Systems". *IEEE Control Systems Magazine*, 69-76 (Jun. 1990)
- [Chri75] Christofides, Nicos: *GRAPH THEORY An Algorithmic Approach*. Academic Press, London, UK (1975) 400 pp.
- [Çinl75] Çinlair, Erhan: *Introduction to Stochastic Processes*. Prentice-Hall, Inc., Englewood Cliffs, N.J., USA (1975) 402 pp.
- [CL89] Carrol, John; Long, Darrell: *Theory of Finite Automata With an Introduction to Formal Languages*. Prentice-Hall International, Englewood Cliffs, USA (1989) 438 pp.
- [CR90] Cassandras, Christos G.; Ramadge, Peter. J.: "Toward a Control Theory for Discrete Event Systems". *IEEE Control Systems Magazine*, (Jun. 1990) p:66-68

- [Flem88] Fleming, W. H., Chair, "Future Directions in Control Theory - A Mathematical Perspective", *Report Panel on Future Directions in Control Theory*, 1988
- [Glyn89] Glynn, Peter W.: "A GSMP Formalism for Discrete Event Systems". *Proceedings of the IEEE*, 77(1):14-23 (1989)
- [HC77] Hughes, G. E.; Cresswell, M. J.: *An Introduction to Modal Logic*. Methuen & Co. Ltd., London (1985) 388 pp.
- [Heym90] Heymann, Michael: "Concurrency and Discrete Event Control". *IEEE Control Systems Magazine*, p:103-112 (Jun. 1990)
- [HK90] Holloway, L. E.; Krogh, B. H.: "Synthesis of Feedback Control Logic for a Class of Controlled Petri Nets". *IEEE Transactions on Automatic Control*, 35(5):514-523 (1990)
- [Ho87] Ho, Yu-Chi: "Basic Research, Manufacturing Automation, and Putting the Cart Before the Horse" editorial. *IEEE Transactions on Automatic Control*, 32(12):1042-1043 (1987)
- [Ho89] Ho, Yu-Chi: "Dynamics of Discrete Event Systems" - guest editorial. *Proceedings of the IEEE*, 77(1):3-6 (1989)
- [Hoar85] Hoare, Charles Antony Richard: *Communicating Sequential Processes*. Prentice-Hall International, UK, LTD., London, UK (1985) 256 pp.
- [HPS72] Hoel, Paul G.; Port, Sidney C.; Stone, Charles J.: *Introduction to Stochastic Processes*. Houghton Mifflin Company, USA (1972) 203 pp.
- [HU79] Hopcroft, John E.; Ullman, Jeffrey D.: *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, USA (1979) 418 pp.
- [IV88] Inan, Kemal; Varaiya, Pravin: "Finitely Recursive Process Models for Discrete Event Systems". *IEEE Transactions on Automatic Control*, 33(7):626-639 (1988)
- [KGM91] Kumar, R.; Garg, V.; Marcus, S. I.: "On Controllability and Normality of Discrete Event Dynamical Systems". *Systems & Control Letters*, 17:157-168 (1991)
- [KGM93] Kumar, R.; Garg, V.; Marcus, S. I.: "Language Stability and Stabilizability of Discrete Event Dynamical Systems". *SIAM Journal of Control & Optimization*, a ser publicado em set. 1993

- [KH91] Krogh, B. H.; Holloway, L. E.: "Synthesis of Feedback Control Logic for Discrete Manufacturing Systems". *Automatica*, 27(4):641-651 (1991)
- [Klei75] Kleinrock, Leonard: *Queueing Systems*, Volume I: Theory. John Wiley & Sons, USA, Canada (1975) 417 pp.
- [KP90] Knight, J. F.; Passino, K. M.: "Decidability for a Temporal Logic Used in Discrete-Event System Analysis". *International Journal of Control*, 52(6):1489-1506 (1990)
- [Krög87] Kröger, Fred: *Temporal Logic for Programs*. Springer Verlag, Berlin, Germany (1987)
- [Krog87] Krogh, Bruce H.: "Controlled Petri Nets and Maximally Permissive Feedback Logic", *Proceedings of the 25th Allerton Conference*, Urbana - IL, USA, pp. 317-326 (Sept. 1987)
- [LI90] Lin, J. Y.; Ionescu, D.: "A Generalized Temporal Logic Approach for Control Problems of a Class of Nondeterministic Discrete Event Systems". *Proceedings of the 29th CDC*, Honolulu, Hawaii (Dec. 1990)
- [LVW88] Lin, F.; Vaz, A. F.; Wonham, W. M.: "Supervisor Specification and Synthesis for Discrete Event Systems". *Int. J. Control*, 48(1):321-332 (1988)
- [LW88a] Lin, F.; Wonham, W. M.: "On Observability of Discrete-Event Systems". *Information Sciences*, 44(3):173-198 (1988)
- [LW88b] Lin, F.; Wonham, W. M.: "Decentralized Supervisory Control of Discrete-Event Systems". *Information Sciences*, 44(3):199-224 (1988)
- [LW90] Lin, F.; Wonham, W. M.: "Decentralized Control and Coordination of Discrete-Event Systems with Partial Observation". *IEEE Transactions on Automatic Control*, 35(12): 1330-1337 (1990)
- [LW91] Li, Yong; Wonham, W. M.: "Control of Vector Discrete-Event Systems". *Systems Control Group Report No. 9106 (Ph. D. Thesis)*, Department of Electrical Engineering, University of Toronto, Toronto, Canada (1992)
- [Miln80] Milner, Robin: "A Calculus of Communicating Systems". *Lecture Notes in Computer Science 92*, Springer Verlag, Berlin, Germany (1980) 171 pp.

- [Moor66] Moore, John T.: *Elements of Abstract Algebra*. The Macmillan Company, New York, USA (1966) 203 pp.
- [Mura89] Murata, Tadao: "Petri Nets: Properties, Analysis and Applications". *Proceedings of the IEEE*, 77(4):541-580 (1989)
- [Ostr89] Ostroff, J. S.: *Temporal Logic for Real-Time Systems*. Research Studies Press Ltd., UK; (1989) 209 pp.
- [OW90] Ostroff, J. S.; Wonham, W. M.: "A Framework for Real-Time Discrete Event Control". *IEEE Transactions on Automatic Control*, 35(4):386-397 (1990)
- [Pete81] Peterson, James Lyle: *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, Inc., Englewood Cliffs, N.J. 07632 (1981) 290 pp.
- [Rama89] Ramadge, P. J.: "Some tractable supervisory control problems for discrete-event systems modeled by Büchi automata". *IEEE Transactions on Automatic Control*, 34 (1989)
- [Rudi88] Rudie, Karen G.: "Software for the Control of Discrete Event Systems: A Complexity Study". *M. A. Sc. Thesis*, Department of Electrical Engineering, University of Toronto, Toronto, Canada (1988)
- [Rudi92] Rudie, Karen G.: "Decentralized Control of Discrete-Event Systems". *Ph. D. Thesis*, Department of Electrical Engineering, University of Toronto, Toronto, Canada (1992)
- [RW87] Ramadge, P. J.; Wonham, W. M.: "Supervisory Control of a Class of Discrete Event Processes". *SIAM J. Control and Optimization*, 25(1):206-230 (1987)
- [RW89a] Ramadge, P. J.; Wonham, W. M.: "The Control of Discrete Event Systems". *Proceedings of the IEEE*, 77(1):81-98 (1989)
- [RW89b] Rudie, Karen G.; Wonham, W. M.: "Supervisory Control of Communicating Processes". *Systems Control Group Report No. 8907*, Department of Electrical Engineering, University of Toronto, Toronto, Canada (1989)
- [Silv92] da Silva Júnior, Brás Isaías: "Lógica Temporal de Tempo Real Generalizada Aplicada ao Controle e Simulação de Sistemas Dinâmicos a Eventos Discretos". *Dissertação de Mestrado*, Departamento de Engenharia da Computação e Automação Industrial, Faculdade de Engenharia Elétrica, Universidade Estadual de Campinas, Campinas - SP, Brasil (1992)

-
- [This92] Thistle, J. G.: "Controllability subsets of live Rabin automata". *Proceedings of the 31st IEEE Conference on Decision and Control*, pp.3746-3747, IEEE Press, New York, USA (1992)
- [TW86] Thistle, J. G.; Wonham, W. M.: "Control Problems in a Temporal Logic Framework". *International Journal of Control*, 44(4):943-976 (1986)
- [WR87] Wonham, W. M.; Ramadge, P. J.: "On the Supremal Controllable Sublanguage of a Given Language". *SIAM J. Control and Optimization*, 25(3):637-659 (1987)
- [WW92] Wong, K. C.; Wonham, W. M.: "Hierarchical and Modular Control of Discrete-Event Systems". *Thirtieth Annual Allerton Conference*, Sept. - Oct. 1992
- [ZW90] Zhong, Hao; Wonham, W. M.: "On the Consistency of Hierarchical Supervision in Discrete-Event Systems". *IEEE Transactions on Automatic Control* 35(10):1125-1134 (1990)