Universidade Federal de Santa Catarina
Programa de Pós-Graduação em Engenharia
Elétrica

Controle Supervisório Hierárquico de Sistemas a Eventos Discretos: Uma Abordagem Baseada na Agregação de Estados

Tese submetida à
Universidade Federal de Santa Catarina
como requisito para a
obtenção do grau de Doutor em Engenharia Elétrica

César Rafael Claure Torrico

Florianópolis, Março de 2003.

Controle Supervisório Hierárquico de Sistemas a Eventos Discretos: Uma Abordagem Baseada na Agregação de Estados

César Rafael Claure Torrico

Esta Tese foi julgada adequada para obtenção do Título de Doutor em Engenharia Elétrica, Área de Concentração em *Controle, Automação e Informática Industrial*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.

Prof. José Eduardo Ribeiro Cury, Dr. d'Etat.
orientador

Prof. Edson Roberto de Pieri, Dr.
Coordenador do curso de Pós-Graduação em Engenharia Elétrica
da Universidade Federal de Santa Catarina.

Banca Examinadora

Prof. José Eduardo Ribeiro Cury, Dr. d'Etat.
orientador

Prof. Guilherme Bittencourt, Dr.

Prof. Vitor Juliano De Negri, Dr.

Prof. Luis Allan Künzle, Dr.

Prof. Carlos Eduardo Trabuco Dorea, Dr.

Prof. Eduardo Camponogara, Dr.

Si para ascender en la vida
hace falta pisar a otros
sin importar su muerte;

Si para alimentarnos mejor
es necesario dejar morir de hambre
al resto de la gente;

Si para conseguir la paz
es necesaria una guerra;

Nada ha aprendido el ser humano !

(Anonimo)

A mi abuelita Benigna*(in memorian 21/08/1994)*

Agradecimentos

Inicialmente quero agradecer a minha familia, por seu carinho, compreensão e apoio. Eles são Bismark, Scarlet, Carol e a minha mãe Rosa.

Agradeço a meu orientador José E. R. Cury, pelo ensinamento e incentivo ao longo do trabalho.

Aos membros da banca examinadora, que contribuíram opinando e sugerindo.

A CAPES pelo financiamento econômico.

A minha namorada Edileusa Berns pela correção do português.

Ao grupo SEDs, pelas discussões. Principalmente a Antônio da Cunha, Max de Queiroz, André Leal.

Aos amigos do LCMI, pelo convívio nestes quatro anos. Especialmente a Karina Barbosa, Cristiane Paim, Michelle Wangham, Tatiana, Rafael Obelheiro, Eugênio Castelan, Evandro Cantú, Rodrigo Sumar, Rodrigo Goytia, Alberto Arispe, Fernando Passold, Karen Campana, Raul Alves.

A Wilson Costa, pela amizade e auxilio nos tramites burocráticos da Pós-Graduação.

A todos que contribuíram direta ou indiretamente para o desenvolvimento desta tese.

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

Controle Supervisório Hierárquico de Sistemas a Eventos Discretos: Uma Abordagem Baseada na Agregação de Estados

César Rafael Claure Torrico

Março/2003

Orientador	: José Eduardo Ribeiro Cury
Área de Concentração	: Automação e Sistemas
Palavras-chave	: Sistemas a Eventos Discretos, Controle Hierárquico, Controle Modular, Estruturas de Controle
Número de Páginas	: 136

Este trabalho introduz um novo modelo para controle supervisório de Sistemas a Eventos Discretos, onde o sistema é representado por uma linguagem prefixo fechada e uma estrutura de controle dependente das cadeias com marcação dinâmica associada. Este modelo mostra-se apropriado para modelagem e controle de sistemas num alto nível de abstração. Baseado nesta abordagem propõe-se um novo modelo para controle supervisório hierárquico por agregação de estados. O modelo consiste de dois níveis de hierarquia, um baixo nível representado pelo modelo clássico de Ramadge-Wonham e um alto nível obtido pela agregação dos seus estados. Neste modelo, os eventos do alto nível são um subconjunto dos eventos de baixo nível, mas para síntese do controlador, este nível será dotado de estruturas de controle avançadas. Posteriormente, o modelo de controle hierárquico por agregação de estados foi utilizado para integração com o controle modular clássico. Apresenta-se uma combinação das arquiteturas mencionadas, e são dadas condições necessárias e suficientes para que supervisores projetados individualmente sobre a planta, quando atuando conjuntamente, levem a uma solução consistente e não bloqueante. Para facilitar a modelagem das extensões previamente desenvolvidas, estendeu-se uma ferramenta computacional denominada *Grail*.

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

Hierarchical Supervisory Control of Discrete Event Systems: An Aproach Based on State Aggregation

César Rafael Claure Torrico

March/2003

Advisor	: José Eduardo Ribeiro Cury
Area of concentration	: Automation and Systems
Keywords	: Discrete Event Systems, Hierarchical Control, Modular Control , Control Structures
Number of Pages	: 136

This work introduces a new model for supervisory control of discrete event systems where the system is described by a prefix closed language and a string-dependent control structures with dynamic marking. The proposed model is shown to be suitable for modeling and control problems where the system is considered at a high level of abstraction. Therefore, based on this approach is presented a new hierarchical control model for discrete event systems based on state aggregation and advanced control structures. The proposed model consists of two levels, a low-level represented by the classical model of Ramadge-Wonham, and a high-level, obtained by state aggregation. In this model, the high-level events are a subset of the low-level events, but for controller synthesis, this level will be endowed by advanced control structures. Subsequently, this hierarchical control model was integrated with the classical modular control. A combination of the mentioned architectures is presented, and necessary and sufficient conditions are given such that the supervisor´s components designed individually when acting jointly over the plant lead to a nonblocking and consistent solution. To facilitate the modeling of the extensions previously developed, a computational tool called *Grail* was extended.

Sumário

Notação

SEDs	:	Sistemas a Eventos discretos
Σ	:	Alfabeto
Σ^*	:	Conjunto de todas as cadeias de elementos de Σ
Σ^+	:	Conjunto de todas as cadeias não vazias
ϵ	:	Cadeia nula
L	:	Linguagem
$\overline{L}$	:	Prefixo fechamento de uma linguagem
$\emptyset$	:	Linguagem vazia
P_i	:	Projeção natural
P_i^{-1}	:	Projeção inversa
A	:	Autômato
G, C_i	:	Geradores
Q	:	Conjunto de estados
Q_m	:	Conjunto de estados marcados
$Q(C_i)$	:	Conjunto de estados do gerador C_i
δ	:	Função de transição de estados
δ_{C_i}	:	Função de transição de estados do gerador C_i
$\tilde{G}$	:	Autômato de Moore
$L(G)$	:	Linguagem gerada
Γ	:	Estrutura de controle
2^{Σ}	:	Conjunto de partes de Σ
f	:	Supervisor
$L(f/G)$	:	Comportamento em malha fechada
$\sup \mathcal{C}$	:	Máxima linguagem controlável
$D = (L, \Gamma)$	:	Sistema a eventos discretos controlado
$\sup \mathcal{CM}$	:	Máxima linguagem Γ-compatível
Ω	:	Operador monótono
G^A	:	Autômato agregado
Σ^A	:	Conjunto de eventos relevantes para especificação
X_i	:	Conjunto de estados de um bloco (estado de um autômato agregado)
$I(X_i)$	:	Conjunto de estados de entrada de um bloco X_i
H_i	:	Gerador de um bloco X_i
H_{ij}	:	Subsistema de H_i para a entrada $x_j \in I(X_i)$
H_i^+	:	Gerador aumentado

H_{ij}^{+} : Subsistema de H_i^{+} para a entrada $x_j \in I(X_i)$
$\mathcal{S}_{ij}$: Conjunto de subautômatos entre H_{ij} e H_{ij}^{+}
$\mathcal{C}_{ij}$: Conjunto de linguagens supremo controláveis de $\mathcal{S}_{ij}$
$\mathcal{D}_{ij}$: Sublinguagens de $\mathcal{C}_{ij}$ sem passar por $L_m(H_{ij})$
$\mathcal{C}_{Nij}$: Conjunto de sublinguagens supremo controlável dos elementos de $\mathcal{D}_{ij}$
E_{ij} : Conjunto total de elementos supremo controláveis de $\mathcal{S}_{ij}$
G_s^A : Autômato agregado com estruturas de controle dependentes do estado
G_e^A : Autômato agregado com estruturas de controle dependentes do estado, evento de entrada e estado anterior
f^A : Supervisor do nível agregado
$\textcircled{\cap}$: Operador de conjunção

Capítulo 1

Introdução

1.1 Motivação

O controle de Sistemas a Eventos Discretos (SED) é uma área de pesquisa de interesse atual, estimulada pela diversidade de possíveis aplicações, como é o caso dos sistemas de manufatura, sistemas de tráfego, sistemas de gerência de base de dados, protocolos de comunicação, entre outros.

De um ponto de vista formal, um SED define-se como um sistema dinâmico a estado discreto que evolui conforme a ocorrência assíncrona de certas transições discretas, chamadas *eventos*. Tais eventos podem ser a chegada de uma peça numa fila, uma tarefa realizada, a transmissão de uma mensagem numa rede de comunicação, o envio de um comando, etc.

As características principais dos SEDs são: o espaço de estados é discreto e o mecanismo de transição de estados dirigido por eventos. Estas propriedades contrastam com os sistemas a variáveis contínuas, que se caracterizam pela continuidade das variáveis de estado e cujo mecanismo de transição é dirigido pelo tempo. Nesses últimos podem ser usados modelos de equações diferenciais, onde o tempo é uma variável independente natural. A não-linearidade dos SEDs está inerente nas descontinuidades das transições de estado.

A idéia do controle supervisório para modelar sistemas a eventos discretos, originalmente proposta por Ramadge e Wonham (1989)(R-W), é expressa em termos da observação e inibição de eventos controláveis, realizadas por um "supervisor" minimamente restritivo sobre um gerador espontâneo de eventos chamado "planta". A

planta ou sistema a controlar, reflete o comportamento fisicamente possível do sistema, i.e., todas as seqüências de eventos que o sistema é capaz de gerar na ausência de controle. O *supervisor* ou elemento controlador está encarregado de confinar o comportamento da planta a certa especificação desejada, através de ações de controle, baseado no comportamento passado da planta. Nesta abordagem, aparecem alguns inconvenientes na modelagem, por exemplo, explosão de estados, quando se trata de modelar sistemas de maior envergadura. Para solucionar este problema, surgem extensões, dentre as quais pode-se citar, por exemplo, o controle modular e extensões (Wonham e Ramadge 1988, de Queiroz e Cury 2000), o controle com observação parcial (Lin e Wonham 1990), o controle descentralizado (Rudie e Wonham 1992), o controle hierárquico (Zhong e Wonham 1990, Wong e Wonham 1996, Caines e Hubbard 1998a) e a exploração da simetria em controle supervisório (Eyzell 2000, Eyzell e Cury 1998).

No controle hierárquico, proposto inicialmente por Zhong e Wonham (1990), situam-se dois níveis de hierarquia, cujas estruturas de controle são adotadas conforme a abordagem de Ramadge e Wonham (1989). Posteriormente, Wong e Wonham (1996) e Pu (2000) generalizam esta abordagem, considerando estruturas de controle avançadas e a noção de tarefas completas no nível hierarquicamente superior. De outro ponto de vista, Caines e Hubbard (1998a) propõem o controle por agregação de estados, mantendo a estrutura de controle tradicional de Ramadge e Wonham (1989) para os dois níveis de hierarquia. Em todas estas abordagens, o problema é obter um modelo para o alto nível, tal que o comportamento esperado devido à ação do supervisor no alto nível seja igual ao comportamento obtido pela ação do controle hierárquico (*Consistência Hierárquica*). Para garantir um comportamento com consistência hierárquica, as abordagens de Wong e Wonham (1996), Pu (2000) e Caines e Hubbard (1998a) fornecem condições conservadoras na construção dos modelos abstratos. Além de que os modelos de alto nível estão definidos sobre um alfabeto distinto do de baixo nível.

Por outro lado sistemas de grande porte em sua maioria podem ser representados pela composição de subsistemas menores, o que levou ao desenvolvimento das abordagens de controle modular (Wonham e Ramadge 1988, de Queiroz 2000) e controle descentralizado (Rudie e Wonham 1992). Na abordagem de controle modular (Wonham e Ramadge 1988), ao invés de se projetar um único supervisor que satisfaça todas as especificações, procura-se construir um controlador para cada especificação, de tal forma que, atuando em conjunto, os supervisores satisfaçam todas as especificações. O controle descentralizado (Rudie e Wonham 1992) impõe restrições de troca de informação entre supervisores e planta, permitindo a supervisores locais o controle apenas de uma parte do processo, para a realização de uma tarefa global.

Muitas vezes, especificações para projetar supervisores em sistemas de grande porte, contemplam somente parte dos eventos do alfabeto do sistema, estes eventos podendo ser considerados como *eventos relevantes para especificação*.

O critério da seleção destes eventos relevantes está associado a eventos necessários para representar múltiplas especificações no alto nível. Por um lado, isso leva a pensar numa divisão da ação de controle em dois níveis de hierarquia, um alto nível preocupado unicamente com o comportamento dos eventos relevantes e um baixo nível encarregado da implementação seqüencial de subtarefas consideradas fixas, a fim de garantir o comportamento desejado no alto nível. Por outro lado, leva a pensar numa divisão da tarefa de controle em várias subtarefas, e combinar os subcontroladores resultantes de modo a solucionar o problema original (controle modular).

Esta motivação incentiva o desenvolvimento de uma extensão da teoria de controle hierárquico e sua integração com a abordagem modular.

1.2 Objetivos do Trabalho

O objetivo de pesquisa deste trabalho é contribuir à teoria de controle supervisório, a fim de reduzir a complexidade no projeto de supervisores de sistemas de grande porte. As principais contribuições estão divididas em três etapas:

- Apresentação de uma nova abordagem de controle hierárquico de SEDs por agregação de estados, que permita a construção do modelo abstrato de uma forma mais simples em relação às abordagens mencionadas anteriormente.
- Integração da nova abordagem desenvolvida com a abordagem de controle modular.
- Implementação de uma ferramenta computacional para modelagem das abordagens desenvolvidas e aplicação a um exemplo prático.

1.3 Organização do trabalho

Este documento está organizado da forma seguinte:

O capítulo 2 apresenta de forma resumida os conceitos básicos da teoria de controle supervisório de Sistemas a Eventos Discretos introduzidas por Ramadge e Wonham.

O capítulo 3 apresenta os principais resultados das abordagens iniciais em Controle Hierárquico, dentre os quais destacam-se o controle hierárquico proposto por Zhong e Wonham (1990), Wong e Wonham (1996), e o controle hierárquico por agregação de estados proposto por Caines e Hubbard (1998a). O objetivo deste capítulo é apresentar as limitações e os problemas destas abordagem no âmbito do controle hierárquico.

No capítulo 4 apresenta-se um novo modelo formal para controle de SEDs baseado em estruturas de controle avançadas dependentes da trajetória passada com poder de marcação, de forma a facilitar a obtenção de um modelo de alto nível com boas propriedades e garantir condições menos conservadoras, no sentido de garantir consistência hierárquica. Apresenta-se um algoritmo para obtenção da máxima sub-linguagem que atenda as especificações desejadas. Este capítulo representa uma parte da nossa contribuição à teoria de SEDs (Cury et al. 2001, Cury et al. 2002).

O capítulo 5 apresenta a primeira parte da proposta, uma nova abordagem para modelagem de Controle Hierárquico por Agregação de Estados. Em particular apresentam-se métodos de agregação de estados em função de um conjunto de eventos relevantes para especificação. Através de um procedimento bem estabelecido dota-se o modelo agregado de uma nova estrutura de controle, conforme estabelecido no capítulo 4. Este capítulo contribuiu à teoria de SEDs com os artigos (Torrico e Cury 2001, Torrico e Cury 2002c)

No capítulo 6 apresentam-se métodos para redução do modelo abstrato com estruturas de controle dependente do estado apresentado no capítulo 5.

O capítulo 7 apresenta o controle supervisório hierárquico modular por agregação de estados. Este capítulo resulta da integração entre o controle hierárquico apresentado no capítulo 5, e no controle modular desenvolvido por Wonham e Ramadge (1988). O objetivo deste capítulo é explorar a modularidade numa estrutura hierárquica, ou seja, dada uma planta no nível gerencial, procuram-se condições para que dois controladores no nível gerencial implementados no baixo nível sejam não bloqueantes. A contribuição deste capítulo à teoria de SEDs deu-se com os artigos (Torrico e Cury 2002a) e (Torrico e Cury 2002b).

No capítulo 8 com a ajuda da ferramenta computacional implementada, apresenta-se uma aplicação das abordagens desenvolvidas a um processo de manufatura.

Finalmente no capítulo 9 apresentam-se as principais conclusões e as perspectivas para futuros trabalhos.

Capítulo 2

Controle de Sistemas a Eventos Discretos

Este capítulo apresenta a teoria básica sobre controle de Sistemas a Eventos Discretos (SEDs) introduzida por Ramadge e Wonham (1987) e a extensão para controle modular proposto por Wonham e Ramadge (1988). Esta apresentação é uma versão simplificada, mas introduz os principais conceitos e estabelece a notação a ser utilizada no desenvolvimento deste documento.

A origem do material apresentado neste capítulo baseia-se em (Carroll e Long 1989) e (Hopcroft e Ullman 1979) no que se trata de Linguagens e Autômatos, e em (Ramadge e Wonham 1987), (Wonham e Ramadge 1987), (Ramadge e Wonham 1989), (Wonham e Ramadge 1988), (Wonham 1998), e (Kumar e Garg 1995) para Supervisão de SEDs. Uma recopilação conjunta pode ser encontrada em (Ziller 1993) e (Claure 1999).

2.1 Linguagens e autômatos

A teoria de linguagens e autômatos é um formalismo matemático muito utilizado para representação de sistemas a eventos discretos. Nesta seção serão apresentados alguns conceitos relativos a linguagens regulares, as quais podem ser representadas por autômatos finitos.

2.1.1 Linguagens

Dado um conjunto finito de símbolos Σ denominado *alfabeto*, seja Σ^* o conjunto de todas as cadeias finitas de elementos de Σ, incluindo a cadeia nula ϵ. Define-se uma *linguagem* L sobre o alfabeto Σ como sendo um subconjunto de Σ^*. Segundo esta definição, tanto Σ^* quanto $\emptyset$ são linguagens. Cabe notar que a linguagem vazia $\emptyset$, é diferente da linguagem formada pela cadeia nula ϵ. Também define-se Σ^+ como o conjunto de todas as palavras "não vazias" que podem ser formadas a partir de Σ. A relação entre Σ^* e Σ^+ é: $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$.

O *prefixo* de uma cadeia w sobre um alfabeto Σ é qualquer cadeia $u \in \Sigma^*$ que possa ser completada com outra cadeia $v \in \Sigma^*$ para formar a cadeia w, denota-se $u \leq w$, u prefixo de w. O prefixo fechamento de L é dado por $\overline{L} = \{u \ : \ \exists v \in \Sigma^* \wedge uv \in L\}$. Diz-se, então, que uma linguagem é prefixo fechada sempre que $L = \overline{L}$.

Dada uma linguagem $L \subseteq \Sigma^*$ e uma cadeia $s \in \Sigma^*$, define-se o conjunto de eventos ativos em L após s como $\Sigma_L(s) = \{\sigma \in \Sigma : s\sigma \in L\}$.

Considerando a evolução seqüencial de um Sistema a Eventos Discretos e um alfabeto Σ correspondendo ao conjunto de eventos que afetam o sistema, pode-se afirmar que se um determinado sistema produziu uma cadeia qualquer w, então produziu anteriormente todos os seus prefixos. Portanto, o comportamento lógico de qualquer sistema a eventos discretos em que não ocorram eventos simultâneos, pode ser representado por uma linguagem prefixo fechada.

2.1.2 Projeção natural

Sejam dois alfabetos Σ_i e Σ, com $\Sigma_i \subset \Sigma$. Define-se $P_i : \Sigma^* \rightarrow \Sigma_i^*$, a *projeção natural* de Σ em Σ_i de acordo com:

1. $P_i(\epsilon) = \epsilon$
2. $P_i(\sigma) = \begin{cases} \epsilon & \text{se} \quad \sigma \notin \Sigma_i \\ \sigma & \text{se} \quad \sigma \in \Sigma_i \end{cases}$
3. $P_i(s\sigma) = P_i(s)P_i(\sigma)$ para $s \in \Sigma^*$, $\sigma \in \Sigma$

A ação de P_i sobre uma cadeia s é apenas a de apagar de s todas as ocorrências de σ tal que $\sigma \notin \Sigma_i$. Assim P_i é a projeção natural de Σ^* em Σ_i^*. O conceito de projeção natural pode ser estendido para linguagens regulares como: $P_i(L) = \{s_i \in \Sigma_i^* \ : \ s_i = P_i(s) \text{ para algum } s \in L\}$. A *projeção inversa*, denotada por P_i^{-1}, é, então, definida como: $P_i^{-1}(L_i) = \{s \ : \ s \in \Sigma^* \wedge P_i(s) \in L_i\}$. Observe que, em geral, $P_i^{-1}(P_i(L)) \supseteq L$.

2.1.3 Composição de linguagens

Sejam duas linguagens $L_1 \subseteq \Sigma_1^*, L_2 \subseteq \Sigma_2^*$, onde é possível que $\Sigma_1 \cap \Sigma_2 \neq \phi$, e seja $\Sigma = \Sigma_1 \cup \Sigma_2$. Define-se o *Produto Síncrono* $L_1||L_2 \subset \Sigma^*$, como $L_1||L_2 = P_1^{-1}(L_1) \cap P_2^{-1}(L_2)$.

Quando $\Sigma_1 \cap \Sigma_2 = \phi$, diz-se que $L_1||L_2$ é um *produto assíncrono*, e quando $\Sigma_1 = \Sigma_2 = \Sigma$, $L_1||L_2$ corresponde à *interseção das linguagens* L_1 e L_2.

2.1.4 Autômatos

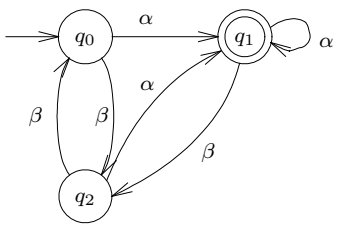
Um *autômato finito determinístico*, às vezes simplesmente chamado de *autômato*, é um modelo matemático de uma máquina de estados, que aceita um conjunto particular de cadeias sobre um alfabeto Σ (Carroll e Long 1989).

Formalmente, um *autômato finito determinístico* é uma quíntupla $A = (\Sigma, Q, \delta, q_0, Q_m)$, onde, Σ é um alfabeto, Q é um conjunto finito não-vazio de estados, $\delta : Q \times \Sigma \rightarrow Q$ é uma função de transição de estados, $q_0 \in Q$ é o estado inicial, $Q_m \subseteq Q$ é o conjunto de estados marcados. Entende-se aqui que a função de transição é *completa*, ou seja, definida para todo par $(q, \sigma) \in Q \times \Sigma$.

Um autômato pode ser representado a partir de um *diagrama de transição de estados* ou ainda por uma *tabela de transição de estados*. Num diagrama de transição de estados, os estados da máquina podem ser representados por vértices do diagrama, e as transições por arcos conectando esses vértices. A seguir mostra-se um exemplo de autômato.

Exemplo 2.1 *Na Figura 2.1 apresenta-se o diagrama de transição de estados de um autômato A.*

A:



$\Sigma = \{\alpha, \beta\}$
$Q = \{q_0, q_1, q_2\}$
$\delta(q_0, \alpha) = q_1, \quad \delta(q_0, \beta) = q_2,$
$\delta(q_1, \alpha) = q_1, \quad \delta(q_1, \beta) = q_2,$
$\delta(q_2, \alpha) = q_1, \quad \delta(q_2, \beta) = q_0,$
$q_0 \rightarrow$ *estado inicial*
$Q_m = \{q_1\}$

Figura 2.1: Diagrama de transição de estados.

A tabela 2.1 representa a respectiva tabela de transição de estados.

$q\backslash\sigma$	α	β
q_0	q_1	q_2
q_1	q_1	q_2
q_2	q_1	q_0

Tabela 2.1: Tabela de transição de estados.

A função de transição δ pode ser naturalmente estendida para cadeias de eventos. Assim, a função de transição estendida denotada por $\hat{\delta}$, é uma função $\hat{\delta} : Q \times \Sigma^* \to Q$ tal que, para $q \in Q$, $s \in \Sigma^*$ e $\sigma \in \Sigma$, $\hat{\delta}(q, \epsilon) = q$ e $\hat{\delta}(q, s\sigma) = \delta(\hat{\delta}(q, s), \sigma)$. Será referenciado $\hat{\delta}$ como δ, caso o contexto não torne necessário evidenciar.

2.1.5 Geradores

Utiliza-se o termo *Gerador* quando se fala de *autômatos com função de transição parcial*.

Um *Gerador* é uma quíntupla $G = (\Sigma, Q, \delta, q_0, Q_m)$, onde, Σ é um alfabeto, Q é um conjunto finito não-vazio de estados, $\delta : Q \times \Sigma \to Q$ é uma função de transição parcial de estados, $q_0 \in Q$ é o estado inicial, $Q_m \subseteq Q$ é o conjunto de estados marcados.

Em contraposição aos autômatos, os geradores têm a função de transição definida apenas para um subconjunto de eventos em cada estado do gerador. Assim como os autômatos, os geradores também podem 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.

A *linguagem gerada* por um gerador G é definida como $L(G) = \{s \ : \ s \in \Sigma^* \wedge \hat{\delta}(q_0, s) \text{ esteja definida}\}$ e a *linguagem marcada* como $L_m(G) = \{s \ : \ s \in \Sigma^* \wedge \hat{\delta}(q_0, s) \in Q_m\}$.

De forma geral, um gerador pode ter estados inacessíveis, isto é, estados que jamais podem ser alcançados a partir do estado inicial. Diz-se que um gerador G é *acessível* se todos os estados $q \in Q$ são alcançáveis a partir do estado inicial. Também diz-se que um gerador G é *co-acessível* se a partir de todos os estados $q \in Q$ é possível alcançar algum estado marcado $q' \in Q_m$.

Um gerador G é *Trim* se é acessível e co-acessível.

Dado um gerador G, define-se o conjunto ativo de eventos após um estado $q \in Q$ como $\Sigma_G(q) = \{\sigma \in \Sigma \ : \ \delta(q, \sigma) \text{ definido }\}$.

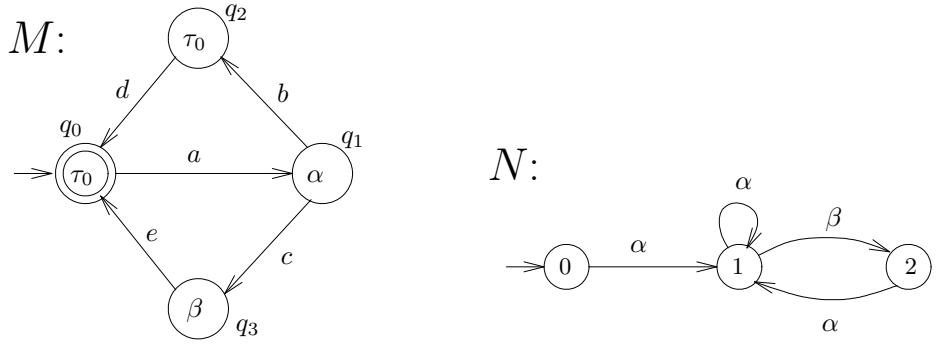
2.1.6 Autômato de Moore

O autômato de Moore é um gerador que carrega uma linguagem de saída definida sobre um alfabeto de saída $T_0 = T \cup \tau_0$, onde τ_0 é um símbolo que não pertence a T e é interpretado como o evento mudo. Este autômato é representado por

$$\tilde{G} = (\Sigma, T_0, \tilde{Q}, \tilde{\delta}, w, \tilde{q}_0, \tilde{Q}_m)$$

onde os ítens escritos com til realizam a mesma função que no gerador G, e $w : \tilde{Q} \to T_0$ é o mapeamento estado-saída.

Exemplo 2.2 *Na Figura 2.2 apresenta-se um exemplo de autômato de Moore com a sua respectiva linguagem de saída.*



(a) Autômato de Moore. (b) Saída do autômato de Moore.

onde:
$\Sigma = \{a, b, c, d, e\}$; $Q = \{q_0, q_1, q_2, q_3\}$; $T_0 = \{\tau_0, \alpha, \beta\}$
$w(q_0) = \tau_0$; $w(q_1) = \alpha$; $w(q_2) = \tau_0$; $w(q_3) = \beta$

Figura 2.2: Autômato de Moore e sua respectiva linguagem de saída.

2.2 Supervisão de Sistemas a Eventos Discretos

No modelo de Ramadge-Wonham, um sistema a eventos discretos, de linguagem gerada L (seqüências parciais) e linguagem marcada L_m (tarefas completadas) pode ser representado por um gerador G tal que $L(G) = L$ e $L_m(G) = L_m$.

Neste modelo se assume que a evolução de um sistema é seqüencial. Conseqüentemente toda seqüência gerada de eventos é "prefixo fechada" e define um conjunto que

pode ser descrito por uma linguagem prefixo fechada. A linguagem marcada representa as tarefas completas do sistema e também é parte da linguagem gerada.

2.2.1 Supervisores

O controle de um SED consiste no chaveamento das entradas de controle, em função do comportamento passado do sistema. Por analogia com a teoria de controle clássica, este comportamento controlado é denominado de *comportamento em malha fechada*.

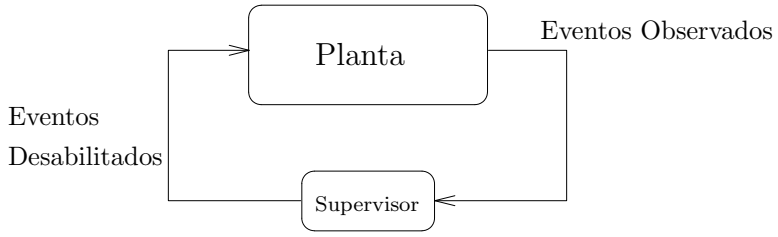


Figura 2.3: SED em malha fechada.

Este conceito leva à distinção do *sistema a controlar* (Planta) e do *agente de controle* (Supervisor), permitindo assim distinguir o comportamento fisicamente possível do sistema e as restrições ligadas a comportamentos não desejados.

Para associar a um SED estruturas de controle segundo Ramadge e Wonham (1989), particiona-se o alfabeto Σ em um conjunto Σ_c de eventos controláveis que podem ser inibidos de ocorrer, e um conjunto Σ_u de eventos não controláveis, sobre os quais o agente de controle não tem influência. O conjunto de *entradas de controle válidas* associado a G, é dado por $\Gamma = \{\gamma \in 2^{\Sigma} \ : \ \Sigma_u \subseteq \gamma \subseteq \Sigma\}$ (2^{Σ} denota o conjunto de partes). A condição $\Sigma_u \subseteq \gamma$, indica simplesmente que os eventos não controláveis são necessariamente habilitados.

Formalmente, um *Supervisor* f, é um mapeamento $f : L \rightarrow \Gamma$ que especifica, para cada cadeia possível de eventos gerados $w \in L$, um conjunto de eventos habilitados (entrada de controle) $\gamma = f(w) \in \Gamma$.

O sistema a eventos discretos G controlado por f é denotado por f/G.

O comportamento do sistema sob a ação do supervisor, definido pela linguagem $L(f/G) \subseteq L(G)$, satisfaz: i) $\epsilon \in L(f/G)$, e ii) $w\sigma \in L(f/G)$ se e somente se $w \in L(f/G)$, $w\sigma \in L(G)$ e $\sigma \in f(w)$

O comportamento marcado de f/G é definido por $L_m(f/G) = L(f/G) \cap L_m(G)$. A linguagem $L_m(f/G)$ representa 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 $L_m(f/G)$ representa as tarefas que podem ser completadas sob supervisão.

Diz-se que um supervisor f para a planta G é *não bloqueante* se e somente se $\overline{L_m(f/G)} = L(f/G)$. Isso implica que, de qualquer estado do comportamento em malha fechada da planta, uma tarefa pode ser completada no sistema.

2.2.2 Condições para síntese de supervisores

As condições para existência de supervisores são descritas usando as noções de controlabilidade e L_m-fechamento.

Dada uma planta G definida sobre o alfabeto $\Sigma = \Sigma_u \cup \Sigma_c$ e uma sub-linguagem K de $L(G)$, a linguagem K é dita *controlável* em relação a $L(G)$ se $\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}$. Esta definição exige que qualquer prefixo w de uma palavra de K ($w \in \overline{K}$), quando seguido de um evento não controlável $\sigma \in \Sigma_u$, tal que $w\sigma \in L(G)$, deve ser ainda prefixo de uma palavra de K ($w\sigma \in \overline{K}$). A classe de linguagens controláveis contidas numa linguagem K, denotada por $\mathcal{C}(K) = \{E \subseteq K \ : \ \overline{E}\Sigma_u \cap L(G) \subseteq \overline{E}\}$, é não vazia e fechada para operação de união de conjuntos. Sendo assim, $\mathcal{C}(K)$ contém um único elemento supremo, chamado de $\sup \mathcal{C}(K, L(G))$.

Diz-se que uma linguagem K é $L_m(G)$-*fechada*, se $K = \overline{K} \cap L_m(G)$, ou seja, todo prefixo de qualquer palavra de K que pertença a $L_m(G)$ é também uma palavra de K.

A proposição a seguir resume as condições necessárias e suficientes de existência de supervisores (Wonham 1998).

Proposição 2.1 *Seja um gerador G e uma linguagem alvo K.*

1. Para $K \subseteq L_m(G)$, $K \neq \emptyset$, existe um supervisor não bloqueante f tal que $L_m(f/G) = K$ se e somente se K é L_m-fechada e L-controlável.

2. Para $K \subseteq L(G)$, $K \neq \emptyset$, existe um supervisor f tal que $L(f/G) = K$ se e somente se K é prefixo-fechada e L-controlável

No caso em que a linguagem K que especifica o comportamento desejado não é

controlável, é possível projetar uma *aproximação de K* que é a *máxima linguagem contida em K*, (denotado por $K\uparrow$). Além disso, sob certas condições, esta linguagem satisfaz os requisitos de fechamento impostos pela proposição estabelecida, e portanto as condições de um supervisor associado. A este supervisor chamaremos de *Supervisor minimamente restritivo* ou *ótimo* (Wonham 1998, Ramadge e Wonham 1989, Kumar e Garg 1995).

2.3 Controle Modular

Na abordagem clássica de Ramadge e Wonham (1989) projetava-se um supervisor para a especificação resultante da composição síncrona de diferentes especificações parciais, abordagem conhecida como *abordagem monolítica*. Na *abordagem modular* (Wonham e Ramadge 1988, Wonham 1998, de Queiroz 2000) projetam-se supervisores para cada especificação em separado, de tal forma que, atuando em conjunto satisfaçam todas as especificações. A arquitetura do controle modular é ilustrada na Figura 2.4.

Figura 2.4: Arquitetura do controle modular.

A abordagem modular, desta forma, permite diminuir a complexidade computacional fazendo com que problemas complexos possam ser decompostos em módulos mais simples. Além desta vantagem, a abordagem modular aumenta a flexibilidade em caso de mudanças. Por exemplo, se uma sub-tarefa for alterada, só é necessário projetar novamente o subcontrolador correspondente.

Infelizmente, estas vantagens não são sempre possíveis de serem exploradas. A abordagem modular pode ser usada somente se as especificações forem *modulares* ou não conflitantes (Wonham 1998).

Na próxima seção são apresentados os principais resultados sobre controle modular de Wonham e Ramadge (1988) para problemas com duas especificações.

2.3.1 Conjunção de supervisores

Sejam f_1 e f_2 supervisores próprios[1] para G tais que cada um dos supervisores é controlável em relação a G e (f_1/G), (f_2/G) são não bloqueantes, isto é:

$$\overline{L_m(f_1/G)} = L(f_1/G), \qquad \overline{L_m(f_2/G)} = L(f_2/G).$$

A conjunção de f_1 e f_2, é denotada por $f_1 \wedge f_2$ cuja ação supervisória é habilitar um evento somente quando este for habilitado por f_1 e f_2 simultaneamente. A ação conjunta dos dois supervisores f_1 e f_2 não é necessariamente não bloqueante, mesmo ambos sendo não bloqueantes individualmente.

Para $K_1,\ K_2 \subseteq \Sigma^*$ sempre é verdade que $\overline{K_1 \cap K_2} \subseteq \overline{K}_1 \cap \overline{K}_2$, isto é, o prefixo de uma cadeia comum a K_1 e a K_2 é também prefixo de K_1 e K_2. Então, K_1 e K_2 são ditas *modulares* se $\overline{K_1 \cap K_2} = \overline{K}_1 \cap \overline{K}_2$. Isso quer dizer que quando K_1 e K_2 compartilharem um prefixo, compartilham uma palavra que contém esse prefixo.

Para descrever a modularidade da ação de dois supervisores f_1 e f_2 sobre uma planta comum, são apresentadas as seguintes proposições:

Proposição 2.2 *(Wonham e Ramadge 1988)* Seja G uma planta e $K_1,\ K_2 \subset \Sigma^*$ controláveis e L_m*-fechadas*, e $f_1,\ f_2$ supervisores não-bloqueantes tais que $L_m(f_i/G) = K_i$, $i = 1, 2$. Então, $L_m((f_1 \wedge f_2)/G) = K_1 \cap K_2$ e $f_1 \wedge f_2$ é não-bloqueante se e somente se $K_1,\ K_2$ são modulares.

Observa-se que se $f_1,\ f_2$ são geradores *trim* tais que $L_m(f_i) = K_i$, então $L_m(f_1||f_2) = K_1 \cap K_2$.

Demonstrou-se também que a modularidade das máximas linguagens controláveis calculadas localmente para cada uma das especificações garante que a síntese modular é ótima.

Proposição 2.3 *(Wonham e Ramadge 1988)* Seja G uma planta e $K_1,\ K_2 \subseteq L(G) \subseteq \Sigma^*$. Então, se $\sup \mathcal{C}(K_1, G)$ e $\sup \mathcal{C}(K_2, G)$ são modulares, $\sup \mathcal{C}(K_1, G) \cap \sup \mathcal{C}(K_2, G) = \sup \mathcal{C}(K_1 \cap K_2, G)$.

Sendo $\sup \mathcal{C}(K_1, G)$ e $\sup \mathcal{C}(K_2, G)$ modulares, a ação conjunta dos supervisores construídos para essas linguagens, gera a máxima linguagem controlável de $K_1 \cap K_2$.

[1] Um supervisor para uma planta G é próprio se é não bloqueante e completo, ou seja, a função de transição do supervisor é definida para todo evento habilitado fisicamente possível.

Quando a condição de modularidade entre as realizações de dois controladores é verificada, o controle modular mostra-se bastante vantajoso em relação ao monolítico em termos da implementação da estrutura de controle e da complexidade do processo de síntese (de Queiroz 2000).

2.4 Conclusão

Apresentou-se neste capítulo uma revisão dos conceitos básicos da teoria de controle supervisório de SEDs (Ramadge e Wonham 1987). Foram apresentadas as condições para síntese de supervisores e as condições para que dois ou mais supervisores sejam projetados modularmente.

Capítulo 3

Controle Supervisório Hierárquico e Agregação de Estados

Uma arquitetura hierárquica pode ser descrita, de forma geral, como uma divisão da ação de controle em níveis de hierarquia. Em geral associam-se representações do sistema para os diferentes níveis, onde o nível de abstração cresce quando sobe-se na hierarquia.

O formalismo sobre controle hierárquico de SEDs foi iniciado por Zhong e Wonham (1990) dentro do contexto de Ramadge e Wonham (1989). Nesta abordagem propõe-se uma hierarquia de dois níveis, um nível associado ao operador e um outro nível associado ao gerente. O problema de controle hierárquico é formulado para especificações prefixo fechadas e refere-se a projetar um controlador no nível gerencial tal que o comportamento esperado neste nível, seja igual ao comportamento obtido pela aplicação do controle hierárquico. Esta condição define a *consistência hierárquica*.

Em Wong e Wonham (1996), se estabelece uma versão generalizada de Zhong e Wonham (1990). Propõem-se estruturas de controle avançadas para o alto nível. Introduz-se o conceito de *consistência de controlabilidade* para garantir mais uma restrição no modelo de Zhong, a *consistência hierárquica forte*. Também se introduz a formulação do problema para linguagens com marcação, para a qual é necessário introduzir os conceitos de *consistência de marcação* e *observadores*, a fim de tratar o problema de não bloqueio.

De outro ponto de vista, em Caines e Hubbard (1998a) propõe-se o controle hierárquico por agregação de estados. Da mesma forma que nas abordagens citadas anteriormente, propõe-se uma hierarquia de dois níveis, cujas estruturas de controle são

adotadas conforme a abordagem de Ramadge e Wonham (1989). Introduzem-se conceitos de *Trilha-CD* e *controlabilidade IBC-não bloqueante* a fim de garantir a consistência hierárquica.

Em da Cunha (2001), encontra-se uma recopilação detalhada acerca da problemática do controle hierárquico.

Segundo a natureza do modelo abstrato de alto nível, neste capítulo, apresenta-se o controle hierárquico como proposto por Zhong e Wonham (1990), a extensão proposta por Wong e Wonham (1996) e o controle hierárquico por agregação de estados como proposto em (Caines e Hubbard 1998a, Caines e Hubbard 1998b, Caines et al. 1997).

3.1 Controle supervisório hierárquico

Partindo da abordagem de Zhong e Wonham (1990), consideram-se dois níveis de hierarquia, uma planta G_{op} e um controlador C_{op} de baixo nível, junto com uma planta G_{ge} e um controlador C_{ge} de alto nível. Estes são acoplados como mostrados na Figura 3.1.

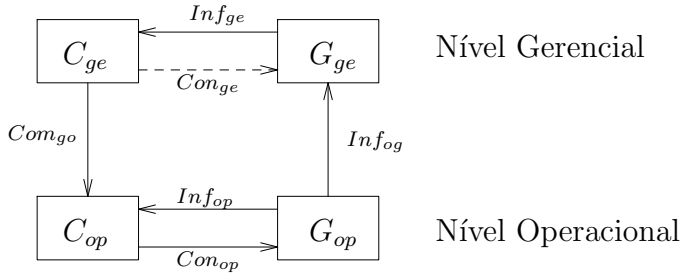


Figura 3.1: Arquitetura Hierárquica.

O sistema G_{op} é a planta local a ser controlada no mundo real por C_{op}, o operador, enquanto que G_{ge} é um modelo, abstraído de G_{op}, utilizado para tomada de decisão no mundo ideal de C_{ge}, o gerente. Evoluções na planta G_{op} são informadas ao nível do gerente por meio do canal de informação Inf_{og}, atualizando o modelo G_{ge}, e ao operador de baixo nível C_{op} pelo canal Inf_{op}. No alto nível, considerando que G_{ge} é dotada de uma estrutura de controle que faz sentido para C_{ge} conceber o exercício de controle sobre o comportamento de G_{ge} por meio do canal virtual Con_{ge}, como resposta à informação recebida da planta G_{ge} pelo canal Inf_{ge}, obtém-se $L(C_{ge}/G_{ge})$, o comportamento em malha fechada para o nível gerencial. O supervisor C_{op}, do nível operacional, controla a planta G_{op} via canal de controle Con_{op}, utilizando a informação do canal Inf_{op} e o comando vindo do gerente Com_{go}. Deseja-se que, o comportamento

em malha fechada no nível operacional, reportado ao nível gerencial $Inf(L(C_{op}/G_{op}))$, seja coerente com o comportamento em malha fechada do nível gerencial $L(C_{ge}/G_{ge})$.

3.1.1 Modelo operacional e gerencial

Para G_{op}, considera-se o modelo tradicional, a quíntupla $G_{op} = (\Sigma, Q, \delta, q_0, Q_m)$, tal como descrito na Seção 2.1.5. O alfabeto Σ, é particionado em elementos controláveis Σ_c e elementos não controláveis Σ_u. Na abordagem de Zhong e Wonham (1990) considera-se o caso quando $Q = Q_m$, isto é, linguagens prefixo fechadas.

Para o nível gerencial assume-se um outro alfabeto T não vazio, que pode ser interpretado como os eventos percebidos pelo gerente, que descreverão o modelo de alto nível G_{ge}. O canal de informação Inf_{og}, é modelado por um mapeamento $\theta : L(G_{op}) \rightarrow T^*$ com as propriedades seguintes:

$$\theta(\epsilon) = \epsilon$$

$$\theta(s\sigma) = \begin{cases} \theta(s) & \text{ou} \\ \theta(s)\tau & \text{para algum } \tau \in T \end{cases}$$

Desta forma, θ é *causal* no sentido de não ser antecipativo, ou seja se $s \leq s'$, i.e. s é um prefixo de s', então $\theta(s) \leq \theta(s')$. Em da Cunha (2001), denota-se θ de *mapa repórter*.

A seguir, combina-se θ com G_{op} num autômato de Moore, tendo como alfabeto de saída $T_0 = T \cup \tau_0$, onde τ_0 é um novo símbolo que não pertence a T e é interpretado como o evento mudo. Este autômato é representado por $\tilde{G}_{op} = (\Sigma, T_0, \tilde{Q}, \tilde{\delta}, w, \tilde{q}_0, \tilde{Q}_m)$, onde os ítens escritos com til realizam a mesma função que em G_{op}, e $w : \tilde{Q} \rightarrow T_0$ é o *mapa vocal* ou mapeamento estado-saída, tal que $w(\tilde{q}_0) = \tau_0$, $w(\tilde{\delta}(\tilde{q}_0, s\sigma)) = \tau_0 \quad$ se $\theta(s\sigma) = \theta(s)$, e $w(\tilde{\delta}(\tilde{q}_0, s\sigma)) = \tau \quad$ se $\quad \theta(s\sigma) = \theta(s)\tau$, isso quer dizer que w silencia se θ não informa algo novo, caso contrário w gera a informação $\tau \in T$. $\tilde{G}_{op}$ é construído tal que $\tilde{\delta}(\tilde{q}_0, s)$ é definido se e somente se $\delta(q_0, s)$ é definido, ou seja $L(G_{op}) = L(\tilde{G}_{op})$.

Para introduzir o modelo do gerente, primeiramente cria-se uma extensão do mapa repórter para linguagens $\hat{\theta} : 2^{L(G_{op})} \rightarrow 2^{T^*}$, tal que para $E \subseteq L(G_{op})$, $\hat{\theta}(E) = \{t \in T^* : (\exists s \in E), t = \theta(s)\}$. O mapa $\hat{\theta}$ possui a propriedade de ser prefixo preservante, i.e. para qualquer $E \subseteq L(G_{op})$, $\hat{\theta}(\overline{E}) = \overline{\hat{\theta}(E)}$, isto é, toda linguagem prefixo fechada no nível operacional possui imagem prefixo fechada, e toda linguagem que seja prefixo fechada no nível gerencial é imagem de uma linguagem prefixo fechada do nível operacional. O mapeamento $\hat{\theta}$ será referenciado como θ, caso o contexto não torne necessário evidenciar.

O comportamento livre no nível gerencial é representado pela linguagem $L_{ge} = \theta(L(G_{op})) \subseteq T^*$. O modelo do nível gerencial pode ser representado por um autômato determinístico $G_{ge} = (T, Z, \xi, z_0, Z_m)$ tal que $L(G_{ge}) = L_m(G_{ge}) = L_{ge}$.

Exemplo 3.1 *Na Figura 3.2, ilustra-se um exemplo do autômato de Moore, onde* $\Sigma = \{a, b, c, d, e, f, g, h\}$, $\Sigma_u = \{b, c, e\}$, $T = \{\alpha, \beta\}$, $\tilde{Q} = Q = \{0, 1, 2, 3, 4, 5\}$, $\tilde{q}_0 = q_0 = 0$, $w(0) = w(1) = w(2) = \tau_0$, $w(3) = \alpha$, $w(4) = w(5) = \beta$. *No diagrama de transição de estados as transições que levam um traço indicam transições controláveis. Na Figura 3.3, ilustra-se o modelo de* G_{ge}.

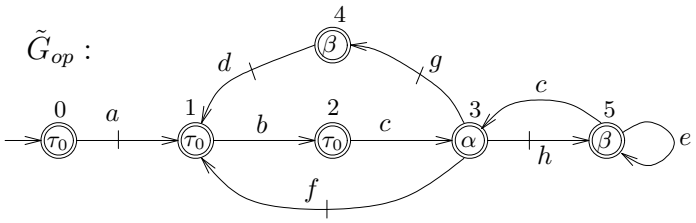


Figura 3.2: Autômato de Moore.

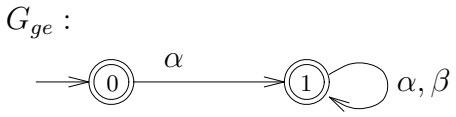


Figura 3.3: Modelo do nível gerencial.

A seguir, indica-se como refinar as descrições de G_{op} e G_{ge}, a fim de equipar G_{ge} com uma estrutura de controle, tal que, um controlador do alto nível C_{ge} que observe somente os estados de G_{ge}, possa tomar decisões significativas. Para G_{ge}, adota-se a mesma estrutura de controle que para G_{op}, e para isso será necessário refinar a estrutura de estados de G_{op} para estender e particionar o alfabeto de eventos T satisfatoriamente dentro de um subconjunto de eventos controláveis e não controláveis.

Para continuar refira-se ao Exemplo 3.1. Supondo que o gerente deseje desabilitar um evento em G_{ge}, através de um comando ao nível operacional. Observe que os eventos α e β do nível gerencial, podem ou não ser desabilitados como próxima saída do autômato de Moore dependendo do estado corrente em G_{op}. Por exemplo, em $\tilde{G}_{op}$, β pode ser desabilitado a partir do estado vocal[1] 3 pois $g, h \in \Sigma_c$ mas não no estado vocal 5 pois $e \in \Sigma_u$. Portanto, neste modelo, o nível gerencial não possui uma estrutura natural de controle. Diz-se, então, que o modelo *não possui Consistência de Controle.*

[1]Entenda-se por estado vocal, estados $q \in Q$ tal que o mapa vocal $w(q) \neq \tau_0$.

Os modelos de G_{op} e G_{ge} podem ser refinados de modo a definir uma estrutura consistente de controle para o nível gerencial. Para isso será necessário o refinamento da estrutura de estados de G_{op}, a extensão do alfabeto T e uma partição de T em T_c e T_u.

Conceitualmente, o procedimento a ser realizado é o seguinte: inicialmente, constrói-se a árvore de alcançabilidade para G_{op}, para o exemplo, mostra-se na Figura 3.4.

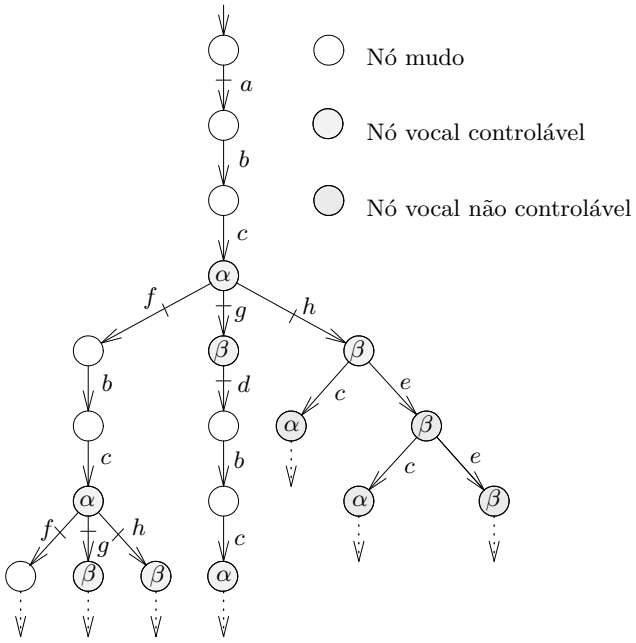


Figura 3.4: Árvore de alcançabilidade.

A seguir renomeia-se $\tau \in T$, em τ_c e τ_u segundo seu nó na árvore seja controlável ou não controlável, obtendo assim um novo alfabeto T_{ext}. O novo mapa vocal $w_{ext} : \mathcal{N} \to T_{ext}$, onde $\mathcal{N}$ é o conjunto de nós na árvore de alcançabilidade, estará definido como, $w_{ext}(n) = \tau_0$ se n é mudo, $w_{ext}(n) = \tau_c$ se $w(n) = \tau \in T$ e n é um nó vocal controlável, e $w_{ext}(n) = \tau_u$ se $w(n) = \tau \in T$ e n é um nó vocal não controlável. Conseqüentemente w_{ext}, determina uma extensão do mapa repórter Inf_{og} em $\theta_{ext} : L(G_{op}) \to T^*_{ext}$. Finalmente, define-se o novo autômato de Moore estendido $G_{op,ext} = \{\Sigma, T_{ext}, Q_{ext}, \delta_{ext}, w_{ext}, q_0, Q_{ext}\}$, com a particularidade que $G_{ge,ext}$ possui consistência de controle. O resultado mostra-se nas Figuras 3.5 e 3.6.

3.1.2 Consistência hierárquica

Nesta seção, descreve-se o controle supervisório definido no nível gerencial, para um controle apropriado a ser exercido no nível operacional.

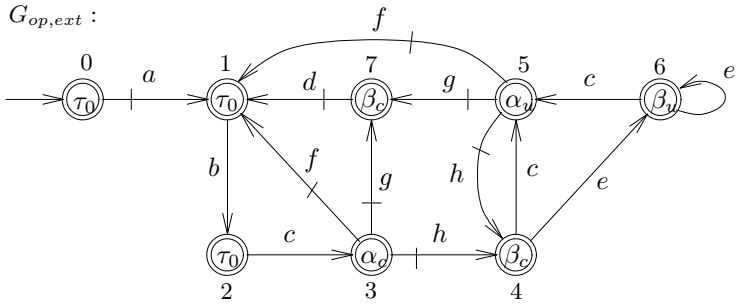


Figura 3.5: Autômato de Moore estendido com consistência de controle.

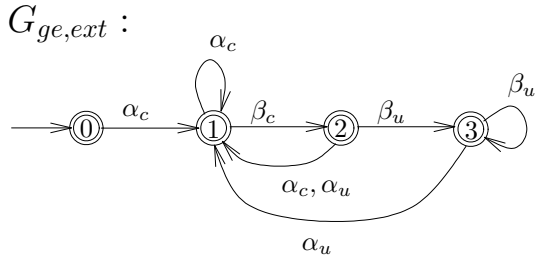


Figura 3.6: Modelo estendido do gerente.

Seja $K_{ge} \subset T^*$, uma linguagem controlável em relação a $L(G_{ge})$. Dentro do contexto tradicional, C_{ge} pode ser projetado de modo que C_{ge} sintetize K_{ge} em malha fechada, ou seja $L(C_{ge}/G_{ge}) = K_{ge}$. O controle supervisório do nível gerencial é determinado pela seleção de eventos de alto nível a serem desabilitados em função do comportamento passado. Assim, C_{ge} é definido por um mapeamento $\gamma_{ge} : L(G_{ge}) \rightarrow 2^T$, tal que $\gamma_{ge}(t) \supseteq T_u$ para qualquer $t \in L(G_{ge})$, indicando que os eventos não controláveis sempre serão habilitados. Dentro do ciclo de controle hierárquico, a ação de controle de C_{ge} em G_{ge} deve ser realizada via $Com_{go} \rightarrow C_{op} \rightarrow Con_{op}$, onde C_{op} interpreta o comando em Com_{go}, gerando a ação de controle Con_{op} e permitindo obter a linguagem gerada no nível operacional, $L(C_{op}/G_{op})$ e, conseqüentemente, no nível gerencial, $\theta(L(C_{op}/G_{op}))$. A questão é, qual a relação entre $L(C_{ge}/G_{ge})$ e $\theta(L(C_{op}/G_{op}))$?

Seja $\Delta_{ge} : L(G_{ge}) \rightarrow 2^{T_c}$, um mapeamento que define o conjunto de eventos desabilitados do alto nível, tal que $\Delta_{ge}(t) = T - \gamma_{ge}(t)$, para todo $t \in L(G_{ge})$.

Correspondentemente pode ser definido o mapeamento de desabilitação para o nível operacional, $\Delta_{op} : L(G_{op}) \times L(G_{ge}) \rightarrow 2^{\Sigma_c}$, tal que

$$\Delta_{op}(s,t) = \{\sigma \in \Sigma_c \ : \ (\exists s' \in \Sigma_u^*)\ s\sigma s' \in L(G_{op}) \ \wedge \ w(s\sigma s') \in \Delta_{ge}(t) \\ \wedge\ (\forall s'')s'' < s' \Rightarrow w(s\sigma s'') = \tau_0\}$$

Para o exemplo da Figura 3.5 e 3.6, assuma-se que G_{op} está no estado 5 e recebe $\gamma_{ge} = \{\alpha_u, \beta_u\}$, ou seja $\Delta_{ge} = \{\alpha_c, \beta_c\}$, a entrada aplicada no nível operacional será

então $\gamma_{op} = \{a, b, c, d, e\}$, ou seja C_{op} inibe $\{f, g, h\}$ em G_{op}.

Para introduzir os resultados obtidos em Zhong e Wonham (1990), considera-se a seguinte definição: seja, $K_{ge} \subseteq L(G_{ge})$, define-se $\theta^{-1}(K_{ge})$ por

$$\theta^{-1}(K_{ge}) = \{s \in L(G_{op}) \ : \ (\exists t \in K_{ge})\theta(s) = t\}$$

$\theta^{-1}(K_{ge})$ representa a maior linguagem em $L(G_{op})$, cuja imagem pelo mapa reporter θ é a linguagem K_{ge}.

Proposição 3.1 *(Zhong e Wonham 1990)* Seja uma estrutura hierárquica com *consistência de controle*. Seja $K_{ge} \subset T^*$ uma linguagem controlável em relação a G_{ge}, e seja C_{ge} tal que $L(C_{ge}/G_{ge}) = K_{ge}$. Se o controle definido por C_{ge} for implementado por C_{op}, então $L(C_{op}/G_{op}) = \sup \mathcal{C}(K_{op})$, onde $K_{op} = \theta^{-1}(K_{ge})$.

A proposição anterior indica que a implementação do controle hierárquico como definido não garante a controlabilidade de K_{op}, ou seja, em geral $\theta(L(C_{op}/G_{op})) \neq K_{ge}$, mas garante que $\theta(L(C_{op}/G_{op})) \subseteq K_{ge}$. Esta propriedade é conhecida como *consistência hierárquica de baixo nível*.

Então surge uma outra questão: "sob que condições se garante que $\theta(L(C_{op}/G_{op})) = K_{ge}$?", sendo que K_{ge} é controlável em relação a $L(G_{ge})$ e C_{op} implementado a partir de C_{ge}, com $L(C_{ge}/G_{ge}) = K_{ge}$. Para resolver esta questão utiliza-se o seguinte conceito.

Definição 3.1 *Dois nós vocais controláveis, n_1 e n_2, da árvore de alcançabilidade de G_{op}, são ditos* parceiros, *se os mesmos verificam as seguintes condições:*

- *Seus caminhos mudos começam na raiz ou num mesmo nó vocal.*
- *Dividem um segmento inicial $s'\sigma$, $\sigma \in \Sigma_c$, e pelo menos uma das continuações até um dos nós está em Σ_u.*

Na Figura 3.7, mostra-se um exemplo de nós parceiros. Observe que desabilitar τ' após τ implica desabilitar τ''.

A seguir são dadas condições para consistência hierárquica.

Proposição 3.2 *(Zhong e Wonham 1990)* Uma estrutura hierárquica possui *consistência hierárquica*, i.e. $\theta(L(C_{op}/G_{op})) = K_{ge}$, ou $\theta(\theta^{-1}(K_{ge}) \uparrow) = K_{ge}$, se a árvore de alcançabilidade de G_{op} não contiver nós vocais parceiros.

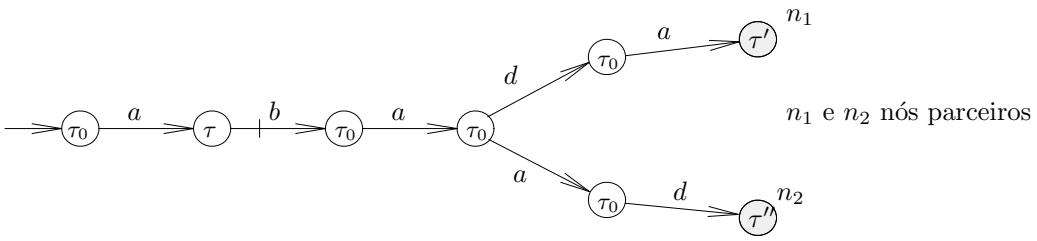


Figura 3.7: Exemplo de nós parceiros.

Na literatura considerada, esta condição é conhecida como *consistência de controle estrita*. Caso uma estrutura hierárquica não possua consistência hierárquica, pode-se torná-la consistente pela vocalização do nó que segue o evento controlável σ, reconsiderando a controlabilidade dos nós parceiros. Na Figura 3.8 mostra-se a eliminação dos nós parceiros, introduzindo um nó controlável n_3 vocalizado com τ^*, com n_1 e n_2, deixando de ser controláveis, e conseqüentemente nós parceiros.

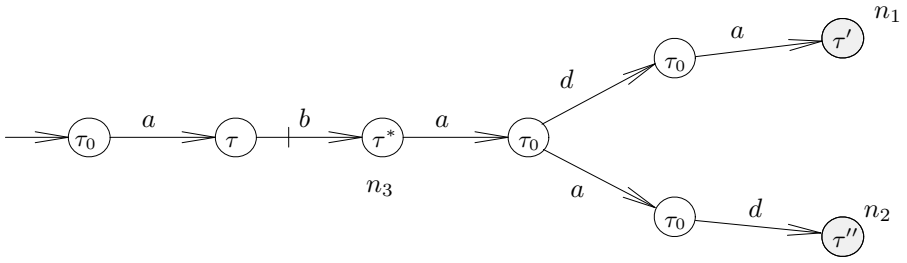


Figura 3.8: Exemplo de eliminação de nós parceiros.

Resumindo, se a *linguagem esperada* no nível gerencial, $L(C_{ge}/G_{ge})$, resulta ser igual à *linguagem obtida* no nível gerencial pela aplicação do controle hierárquico, $\theta(L(C_{op}/G_{op}))$, se diz que a estrutura hierárquica possui *consistência hierárquica*.

3.1.3 Extensão do controle hierárquico clássico

Nesta seção, apresenta-se em forma sucinta a generalização proposta por Wong e Wonham (1996), ao controle hierárquico de Zhong e Wonham (1990).

A consistência hierárquica em Zhong significa que toda linguagem implementada no nível gerencial é imagem de um comportamento implementado no nível do operador. Entretanto, nada se pode afirmar de uma linguagem não realizável no nível gerencial.

Considere a estrutura hierárquica da Figura 3.9. Verifica-se que a especificação $E_{ge} = \overline{(\alpha_c\beta_c)^*}$, não é controlável, pois, não há como impedir a ocorrência de α_u após a

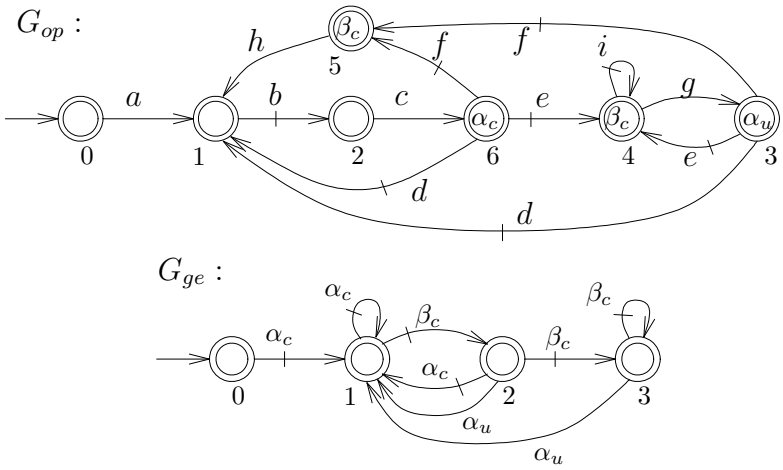


Figura 3.9: Estrutura hierárquica com consistência de controle.

primeira ocorrência de $\alpha_c\beta_c$ em G_{ge}. Considere então a especificação $E_{op} = \overline{a(bcfh)^*}$, para o nível do operador, verifica-se que E_{op} é controlável, prefixo fechada e que $\theta(E_{op}) = E_{ge}$. Assim, E_{ge} poderia ser implementada por um supervisor do operador C_{op} tal que $L(C_{op}/G_{op}) = E_{op}$. Este fato mostra que, para uma linguagem não realizável no nível gerencial, existe uma linguagem realizável no nível operacional, tal que, a imagem seja aquela linguagem não realizável do nível gerencial. Esta é a diferença entre a consistência hierárquica e consistência hierárquica forte.

Sejam $\mathcal{C}(L(G_{op}))$ o conjunto de todas as linguagens controláveis contidas em $L(G_{op})$ e $\mathcal{C}(L(G_{ge}))$ o conjunto de todas as linguagens controláveis contidas em $L(G_{ge})$, a definição a seguir apresenta a consistência hierárquica forte.

Definição 3.2 *(Wong e Wonham 1996)* Diz-se que uma estrutura hierárquica possui *consistência hierárquica forte* ou chamada também de *consistência de controlabilidade* se $\theta(\mathcal{C}(L(G_{op}))) = \mathcal{C}(L(G_{ge}))$.

A consistência de controlabilidade indica que todas as linguagens controláveis no nível operacional são mapeadas em linguagens controláveis do nível gerencial, e todas as linguagens controláveis no nível gerencial são imagem de linguagens controláveis do nível operacional.

Então surge uma questão: "Caso uma estrutura hierárquica não possua consistência de controlabilidade, como construir uma estrutura hierárquica que satisfaça essa condição?"

Considera-se o conceito de *mapa repórter observador*: o mapa repórter θ é um

observador se,

$$(\forall s \in L(G_{op}))(\forall \tau \in T) : \theta(s)\tau \in L(G_{ge}) \Rightarrow (\exists u \in \Sigma^+) : su \in L(G_{op}) \wedge \theta(su) = \theta(s)\tau$$

Interpreta-se como segue: toda palavra no nível operacional cuja imagem correspondente no nível gerencial possui um dado conjunto de eventos ativos, deve ser possível de ser estendida por trechos vocais para palavras que vocalizem aquele exato conjunto de eventos.

Para obter uma estrutura hierárquica com consistência de controlabilidade conforme apresentada em Wong e Wonham (1996), além de consistência de controle estrita, é preciso construir um mapa repórter que seja observador. Em da Cunha (2001), propõe-se um procedimento para o cálculo do observador.

Outro ponto explorado na abordagem de Wong e Wonham (1996) é a formulação do controle hierárquico para linguagens marcadas. O comportamento no nível operacional é representado por um par de linguagens $L(G_{op})$ e $L_m(G_{op})$, onde $L_m(G_{op})$ representa a linguagem marcada de G_{op}. O comportamento no nível gerencial é dado pelo par de linguagens $L(G_{ge})$ e $L_m(G_{ge})$, tal que $L(G_{ge}) = \theta(L(G_{op}))$ e $L_m(G_{ge}) = \theta(L_m(G_{op}))$. O fato que a linguagem marcada no nível gerencial seja imagem das cadeias marcadas do nível operacional traz as suas conseqüências como visto a seguir.

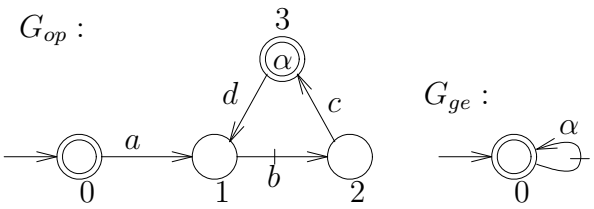


Figura 3.10: Estrutura hierárquica com consistência de controle estrita e mapa repórter observador.

Considere a estrutura hierárquica da Figura 3.10. Seja a especificação no nível gerencial, $E_{ge} = \epsilon$, verifica-se que esta é controlável e está contida na linguagem marcada no nível gerencial. A linguagem $\theta^{-1}(E_{ge}) = \epsilon + a + ab$, não é controlável, não está contida nem é fechada em relação a linguagem marcada no nível do operador. A sua máxima linguagem controlável $\sup \mathcal{C}(\theta^{-1}(E_{ge})) = \epsilon + a$, não é $L_m(G_{op})$-fechada. A linguagem $\theta^{-1}(E_{ge}) \cap L_m(G_{op}) = \epsilon$, é $L_m(G_{op})$-fechada, mas não é controlável. Finalmente $\sup \mathcal{C}(\theta^{-1}(E_{ge}) \cap L_m(G_{op})) = \emptyset$. Desta forma verifica-se que não é possível encontrar um supervisor não bloqueante no nível operacional que implemente no nível gerencial E_{ge}.

Em Wong e Wonham (1996), o que se pretende é tornar $\sup \mathcal{C}(\theta^{-1}(E_{ge}))$ contida

em $L_m(G_{op})$ e $L_m(G_{op})$-fechada se e somente se E_{ge} estiver contida em $L_m(G_{ge})$ e for $L_m(G_{ge})$-fechada. Assim, garantida a consistência de controlabilidade, é possível encontrar um supervisor não bloqueante no nível operacional tal que a linguagem marcada em malha fechada seja $\sup \mathcal{C}(\theta^{-1}(E_{ge}))$ e sua imagem no nível gerencial seja E_{ge}. Para resolver este problema, considera-se o conceito de *consistência de marcação*. Uma estrutura hierárquica possui *consistência de marcação* quando $\theta^{-1}(L_m(G_{ge})) = L_m(G_{op})$. Interpreta-se, se um nó vocal for marcado, então todos os seus caminhos que continuam por nós não vocais devem estar marcados também.

Dada uma estrutura hierárquica com consistência de controle estrita, e θ um observador, em Wong e Wonham (1996) mostra-se que, incluindo a condição de consistência de marcação, para qualquer $E_{ge} \subseteq L_m(G_{ge})$ existe um supervisor não bloqueante C_{op} tal que, $L_m(C_{op}/G_{op}) = \sup \mathcal{C}(\theta^{-1}(E_{ge}))$ e $\theta(L_m(C_{op}/G_{op})) = E_{ge}$ se e somente se E_{ge} for controlável e $L_m(G_{ge})$-fechada.

Observe que, se na Figura 3.10 os eventos a e d fossem controláveis, sempre seria possível obter um supervisor não bloqueante no nível operacional que atendesse qualquer especificação realizável no nível gerencial, mesmo que a estrutura hierárquica não possua consistência de marcação. Isto mostra que a condição de consistência de marcação é conservadora.

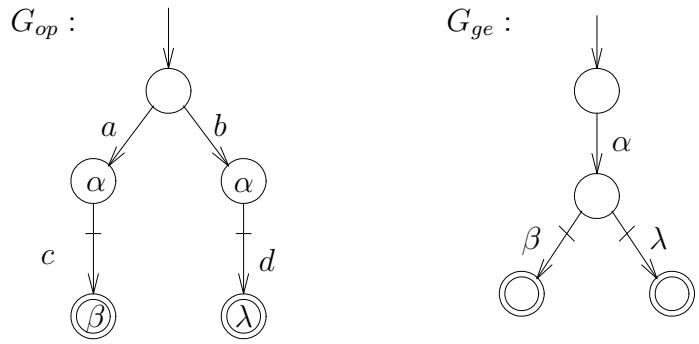


Figura 3.11: Estrutura hierárquica com consistência de controle estrita e consistência de marcação.

Por outro lado, a condição de mapa reporter ser observador também é conservadora. Veja a estrutura hierárquica da Figura 3.11, ela possui consistência de controle estrita e consistência de marcação mas o mapa reporter não é um observador. Seja a linguagem $E_{ge} = \alpha\beta$, ela está contida em $L_m(G_{ge})$, é $L_m(G_{ge})$-fechada e controlável, mas não existe linguagem $E_{op} \subset L_m(G_{op})$ que seja $L_m(G_{op})$-fechada, controlável tal que $\theta(E_{op}) = E_{ge}$. Se os eventos a e b fossem controláveis, para qualquer especificação realizável E_{ge} no nível gerencial, existiria uma realização no nível operacional controlável e $L_m(G_{op})$-fechada que garantiria $\theta(E_{op}) = E_{ge}$, mesmo o mapa reporter não sendo observador.

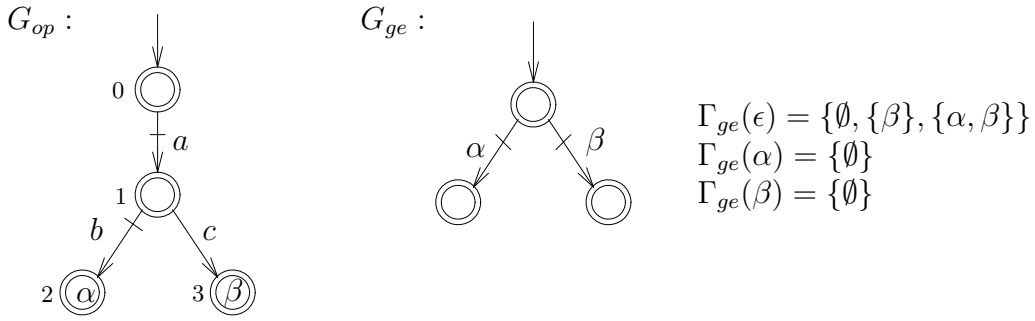


Figura 3.12: Estrutura hierárquica com estruturas de controle avançadas.

Para finalizar esta seção, em Wong e Wonham (1996), no nível gerencial, introduzem-se de forma não muito detalhada, estruturas de controle avançadas dependentes da trajetória passada, de tal forma a garantir a condição de consistência de controle estrita. Por exemplo, seja a estrutura hierárquica da Figura 3.12, verifica-se que não possui consistência de controle estrita, uma vez que os nós 2 e 3 de G_{op} são parceiros. Assumindo uma estrutura de controle, como aquela descrita na mesma Figura 3.12, garante-se a consistência de controle estrita. Em Pu (2000), apresenta-se uma extensão desta abordagem onde o problema de controle hierárquico é resolvido com garantia de consistência hierárquica apenas.

3.2 Controle hierárquico por agregação de estados

Esta abordagem foi desenvolvida inicialmente por Caines e Hubbard (1998a), onde apresenta-se uma estrutura hierárquica em dois níveis. Um *baixo nível* que representa o comportamento real do sistema e um *alto nível (nível π)* obtido pela agregação de estados do baixo nível. A agregação é feita de tal modo que o conjunto de estados agregados dentro de um mesmo bloco tenham um significado ou algum sentido em comum, por exemplo, conjunto de estados onde a peça encontra-se dentro de um buffer B, ou, estados onde a peça está na esteira, etc. Assim, no nível π, só interessa a interpretação dos estados, ou seja, as especificações consideradas são do tipo estados proibidos.

A seguir desenvolve-se em forma sucinta as considerações mais relevantes desta abordagem.

3.2.1 Agregação de estados em controle supervisório

Na abordagem de Caines e Hubbard (1998a) o sistema está modelado por um gerador,

$$G = (\Sigma, X, \delta, Q_0, Q_m) \tag{3.1}$$

onde Σ, X, δ, e Q_m, são definidos como na Seção 2.1.5 e Q_0 é o conjunto de estados iniciais admissíveis ($Q_0 \subseteq X$).

Uma abstração do sistema pode ser feita através de partições do conjunto de estados. A partição π é uma coleção de subconjuntos X_k (chamados blocos-π) obtidos através da partição do conjunto de estados X, sendo formalmente denotada por $\pi = \{X_1, X_2, ..., X_N\}$ com $\bigcup X_i = X$, $X_i \neq \emptyset$ e $X_i \cap X_j = \emptyset$ para $i \neq j$. Define-se o complemento de X_i, por $X_i^C = \{\bigcup X_j \in \pi : X_j \neq X_i\}$

A seguir são apresentadas algumas definições a fim de descrever a dinâmica entre os blocos particionados.

Definição 3.3 $I(X_i, Q_0)$

O conjunto de estados de entrada "In-set" $I(X_i, Q_0)$ *de um elemento* X_i *de uma partição* π*, é o conjunto de estados em* X_i *que ou são estados iniciais ou são estados diretamente acessíveis(acessível num passo) a partir do complemento de* X_i*, isto é:*

$$x \in I(X_i, Q_0) \longleftrightarrow (x \in (Q_0 \cap X_i)) \vee \exists x' \in X_i^C \ , \ \exists \sigma \in \Sigma \ , \ \delta(x', \sigma) = x$$

Definição 3.4 $\langle X_i, X_j \rangle_u$

A relação $\langle X_i, X_j \rangle_u$ *é satisfeita para* $i \neq j$ *sempre que exista pelo menos uma trajetória não controlável de* X_i *a* X_j*, isto é:*

$$\langle X_i, X_j \rangle_u \longleftrightarrow \forall x \in I(X_i, Q_0) \ , \ \exists s \in \Sigma_u^* \ , \ \delta(x, s) \in X_j \wedge \forall s' < s, \delta(x, s') \in X_i$$

Definição 3.5 $\langle X_i, X_j \rangle_d$

A relação $\langle X_i, X_j \rangle_d$ *é satisfeita para* $i \neq j$ *sempre que todas as trajetórias de* X_i *a* X_j *são controláveis, isto é:*

$$\begin{aligned} \langle X_i, X_j \rangle_d \longleftrightarrow \forall x \in I(X_i, Q_0), \ \exists s \in \Sigma^*, \ \delta(x, s) \in X_j \wedge \forall s' < s, \ \delta(x, s') \in X_i \\ \wedge \ \ \forall x \in I(X_i, Q_0) \ \neg \exists s \in \Sigma_u^* \ , \ \delta(x, s) \in X_j \wedge \forall s' < s \ , \ \delta(x, s') \in X_i \end{aligned}$$

Note que $\langle X_i, X_j \rangle_u$ e $\langle X_i, X_j \rangle_d$ não podem ser satisfeitas simultaneamente. Note ainda que blocos X_i, X_j, mesmo estando ligados por algumas transições podem não

satisfazer nenhuma das relações $\langle X_i, X_j \rangle_u$ e $\langle X_i, X_j \rangle_d$. Neste ponto é possível definir formalmente o autômato partição-π.

Definição 3.6 Autômato partição-π

Definimos o autômato partição-π

$$G^\pi = (\Sigma^\pi, \pi, \delta^\pi, Q_0^\pi, Q_m^\pi)$$

onde Σ^π *é o conjunto de eventos, i.e.* $\Sigma^\pi = \Sigma_u^\pi \cup \Sigma_c^\pi$, *tal que* $\Sigma_u^\pi = \{V_i^j : \langle X_i, X_j \rangle_u,\ i,j = 1,\dots N\}$ *e* $\Sigma_c^\pi = \{U_i^j : \langle X_i, X_j \rangle_d,\ i,j = 1,\dots N\}$. *Ainda,* $\pi = \{X_1, ...X_N\}$ *é conjunto finito de estados definido para* G^π, *obtido após a partição de* X, $\delta^\pi : \pi \times \Sigma^\pi \longrightarrow \pi$ *é a função de transição tal que* $\delta^\pi(X_i, U_i^j) = X_j$ *ou* $\delta^\pi(X_i, V_i^j) = X_j$ *para as relações* $\langle X_i, X_j \rangle_d$ *e* $\langle X_i, X_j \rangle_u$ *tomadas entre* X_i *e* X_j, *respectivamente.* $Q_0^\pi = \{X_i \in \pi : X_i \cap Q_0 \neq \emptyset\}$ *é o novo conjunto de estados iniciais e* $Q_m^\pi = \{X_i \in \pi : X_i \cap Q_m \neq \emptyset\}$ *é o novo conjunto de estados marcados.*

A Figura 3.13 mostra os autômatos G e G^π, e como os estados originais foram agregados dentro de blocos-π.

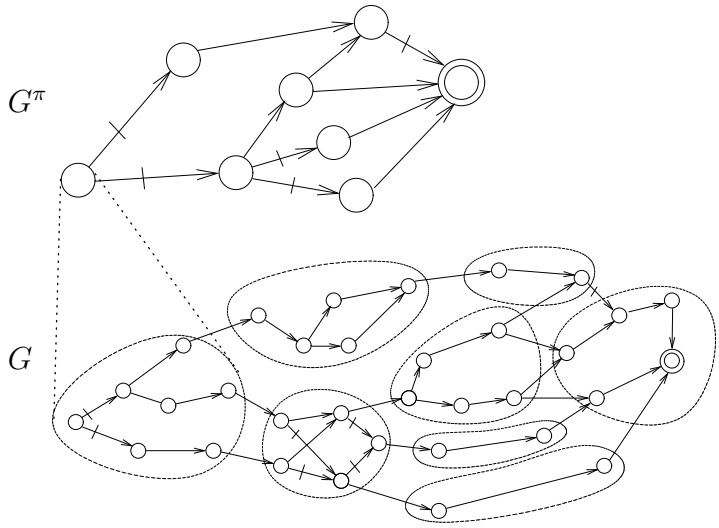


Figura 3.13: Agregação e autômato partição-π.

Definição 3.7 Trilha-CD

Um autômato partição-π é dito ser trilha-CD se e somente se:

$$\forall\langle X_i, X_j\rangle \ , \ \Big(\langle X_i, X_j\rangle_u \vee \langle X_i, X_j\rangle_d \vee \neg[\exists\sigma \in \Sigma \ , \ \exists x \in X_i \ , \ \exists x' \in X_j \ , \ \delta(x,\sigma) = x']\Big)$$

Ou seja, um autômato partição-π é dito ser trilha-CD se para quaisquer pares de blocos-π $\langle X_i, X_j\rangle$, ou está definida uma transição de X_i para X_j controlável ou não-controlável, ou então não existe nenhuma transição definida no autômato G partindo de algum estado de X_i para algum estado de X_j. De outra maneira, um autômato π não é trilha-CD se entre algum par de blocos X_i, X_j existem transições em G que os ligam diretamente, mas nem a condição $\langle X_i, X_j\rangle_d$ nem a condição $\langle X_i, X_j\rangle_u$ é satisfeita.

Esta definição garante que as transições no nível π sejam realizadas sem ambigüidades por trajetórias no baixo nível e que as trajetórias do baixo nível sejam sempre representadas no autômato partição-π.

Definimos o mapeamento canônico para:

(estados) $\Theta_\pi : X \longrightarrow \pi$,

$$\Theta_\pi(x) = X_i \text{ se } x \in X_i, \qquad 1 \leq i \leq |\pi|$$

e estendemos esta definição de forma natural a domínios sucessivos de maior complexidade:

(conjunto de estados) $\Theta_\pi : 2^X \longrightarrow 2^\pi$,

$$\Theta_\pi(R) = \{X \in \pi \ : \ X \cap R \neq \emptyset\}$$

Para um estado inicial, $q_0 \in Q_0$, continua-se com a definição de mapeamento para:

(cadeias de eventos) $\Theta_\pi : L(G) \longrightarrow L(G^\pi)$,
para $s \in \Sigma^*$ e $\sigma \in \Sigma$, $\Theta_\pi(\epsilon) = \epsilon$ e

$$\Theta_\pi(s\sigma) = \begin{cases} \Theta_\pi(s)U_i^j & \textbf{se } \hat{\delta}(q_0, s) \in X_i,\ \delta(\hat{\delta}(q_0, s), \sigma) \in X_j, \text{ e } \langle X_i, X_j\rangle_d \\ \Theta_\pi(s)V_i^j & \textbf{se } \hat{\delta}(q_0, s) \in X_i,\ \delta(\hat{\delta}(q_0, s), \sigma) \in X_j, \text{ e } \langle X_i, X_j\rangle_u \\ \Theta_\pi(s) & \textbf{se } \hat{\delta}(q_0, s) \in X_i, \text{ e } \delta(\hat{\delta}(q_0, s), \sigma) \in X_i \end{cases}$$

(Linguagens) $\Theta_\pi : 2^{L(G)} \longrightarrow 2^{L(G^\pi)}$,

$$\Theta_\pi(K) = \{\Theta_\pi(t) \ : \ t \in K\}$$

3.2.2 Controle do Autômato-π

Nesta abordagem cada bloco X_i tem definido *localmente* um supervisor $f : X_i \longrightarrow 2^\Sigma$. A função do supervisor é especificar o subconjunto de transições que são habilitadas após um determinado estado x ser alcançado dentro de X_i, e para qualquer $x \in X_i$, tem-se que $\Sigma_u \subseteq f(x)$. Tal condição implica que todos os eventos não controláveis estão sempre habilitados para ocorrer.

Assume-se este tipo de controle por realimentação de estados a fim de poder definir um supervisor individualmente para cada bloco X_i.

Já para o nível π um supervisor é dado pelo mapeamento $f^\pi : \Sigma^{\pi *} \longrightarrow 2^{\Sigma^\pi}$, o qual especifica o subconjunto de transições que são habilitadas pelo supervisor f^π depois da execução de uma seqüência de eventos $s \in \Sigma^{\pi *}$.

Exemplo 3.2 *O autômato da Figura 3.14(c) mostra um autômato no baixo nível:*

$G = (\{v_1, \ldots, v_6\} \cup \{u_1, u_2, u_3\}, \{x_1, \ldots, x_8\}, \delta, \{x_1, x_2\}, \{x_6\})$

Para uma partição $\pi = \{\{x_1, \ldots, x_5\}, \{x_6\}, \{x_7\}, \{x_8\})$*, temos o autômato partição-*π*,* G^π*, como mostrada na Figura 3.14(a):*

$G^\pi = (\{U_1^2, U_1^3\} \cup \{V_1^4\}, \{X_1, \ldots, X_4\}, \delta^\pi, \{X_1\}, \{X_2\})$.

O autômato G^π ilustrado na Figura 3.14(a) é Trilha-CD, pois todos os controles no nível-π são realizáveis no baixo nível, isto é, não há nenhum controle exigido por G^π que não seja realizável por G.

Na Figura 3.14(c) , é visualizada a partição do conjunto de estados de G e a agregação destes para formar o autômato G^π. Para entender claramente as representações das transições no nível π, basta recorrer as definições de $\langle X_i, X_j \rangle_d$ e $\langle X_i, X_j \rangle_u$.

Para conseguir as possíveis entradas de controle no nível π para o bloco X_1, no baixo nível, terá que se exercer controle a partir dos estados x_3 e x_5, já que até atingir estes estados todas as transições em G são não controláveis.

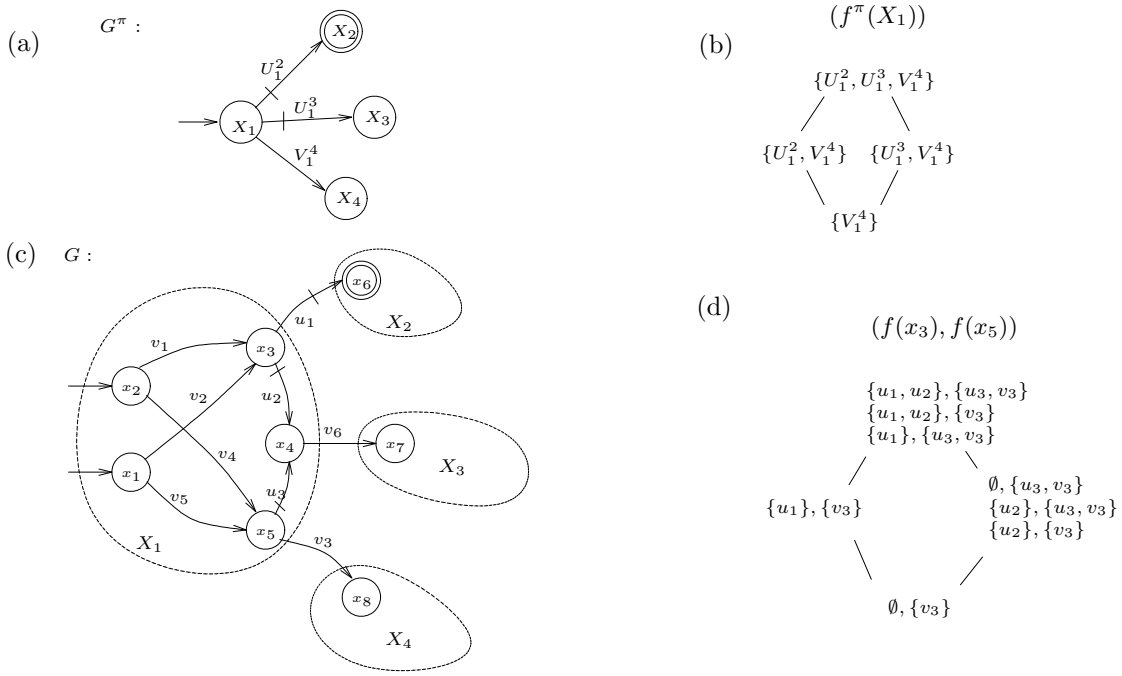


Figura 3.14: G e G^π com entradas de controle por realimentação de estados.

Na Figura 3.14(d) é ilustrada a ação de controle pelo supervisor f para os estados x_3 e x_5. Na estrutura de ação de controle obtida, os elementos do topo indicam que nenhuma transição é desabilitada. Desta forma, no nível π todas as transições entre X_1 e demais blocos-π estão habilitadas para ocorrer. Os elementos do topo da Figura 3.14(b), refletem esta idéia. Voltando à estrutura de controle de G (Fig. 3.14(d)), os elementos do meio representam desabilitações parciais de transições dentro de X_1, e os elementos da parte inferior representam a desabilitação de todos as transições controláveis em X_1. Na verdade, os elementos desta estrutura representam realizações possíveis no baixo nível, requeridas pelo nível π.

Exemplo 3.3 *A Figura 3.15 mostra um autômato diferente no baixo nível. Neste exemplo, uma das entradas de controle do nível π não é realizável no baixo nível, i.e. para a entrada de controle $f^\pi(X_i) = \{U_1^2, V_1^4\}$ no nível π, não é possível realizar no baixo nível.*

Define-se um autômato local para cada bloco X_i, a fim de analisar o comportamento interno de cada bloco e poder evitar o problema de alguma falta de realização de um comando de alto nível no baixo nível.

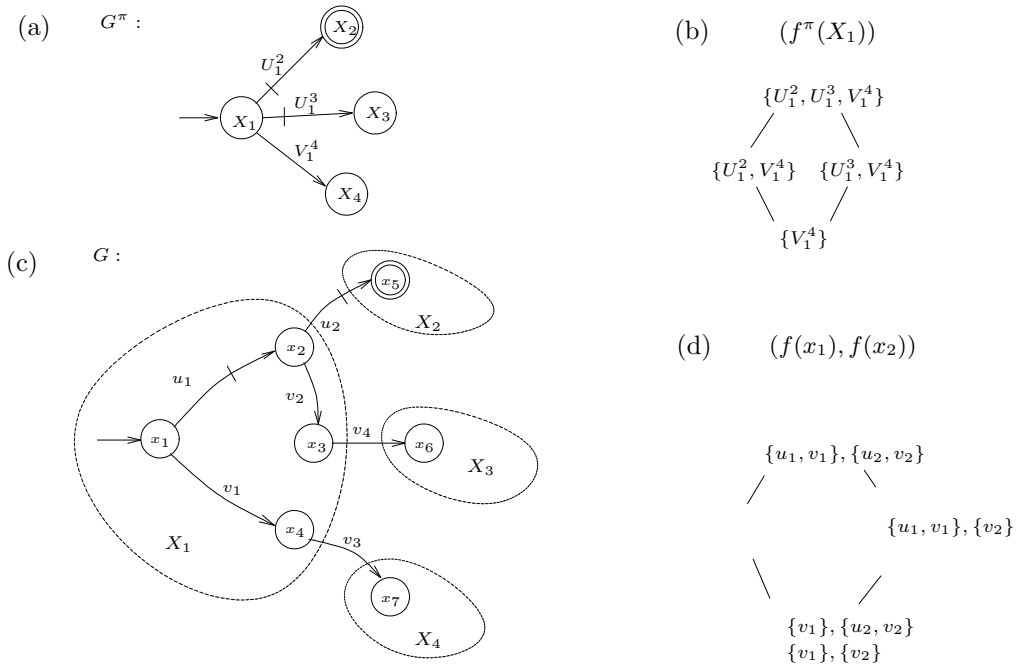


Figura 3.15: Um bloco com falta de uma realização.

Definição 3.8 $G_{X_i^+}(x, Q)$

Definimos o sub-autômato $G_{X_i^+}(x, Q)$ *do autômato* $G = (\Sigma, X, \delta, Q_0, Q_m)$ *por:*

$$G_{X_i^+}(x, Q) = (\Sigma, X_i^+, \delta^{X_i^+}, x, Q)$$

onde $X_i^+ = X_i \cup \{x' \in X_i^C \ : \ \exists \sigma \in \Sigma \ , \ \exists y \in X_i \ , \ \delta(y, \sigma) = x'\}$ *i.e. o bloco* X_i *e os pontos alcançáveis de* X_i *em uma transição.* $\delta^{X_i^+}$ *é* δ *com o domínio restrito ao conjunto* $X^i \times \Sigma$. x *e* Q *são os parâmetros do sub-autômato representando o estado inicial e estados finais respectivamente.*

Com a seguinte definição, pode-se formular a questão de controlabilidade em termos de controle por realimentação de estados e acessibilidade de estados objetivo.

Definição 3.9 Conjunto de estados controlável e não-bloqueante.

Um conjunto $R \subseteq X$ *é* controlável e não-bloqueante *com relação ao autômato* $G = (X, \Sigma, \delta, x_0, Q_m)$ *se satisfaz o seguinte:*

1. $\forall x \in R \ , \ \exists s \in \Sigma^* \ , \ \delta(x_0, s) = x \wedge \forall s' \leq s \ , \ \delta(x_0, s') \in R$ *(R-alcançável)*

2. $\forall x \in R\ ,\ \neg\exists \sigma \in \Sigma_u\ ,\ \delta(x,\sigma) \in (X - R)$ *(fechado com relação a* Σ_u*)*

   Adicionalmente, R *é não-bloqueante com relação ao autômato* G *se a seguinte condição satisfaz*

3. $\forall x \in R\ ,\ \exists s \in \Sigma^*, \delta(x,s) \in Q_m,\ \forall s' \leq s\ ,\ \delta(x,s') \in R$ *(não-bloqueante)*

Definição 3.10 Linguagens Controláveis e Não-Bloqueio

Uma linguagem $K \subseteq L(G)$ *é* controlável *com relação ao autômato* $G = (\Sigma_u \cup \Sigma_c, X, \delta, x_0, Q_m)$ *se* $\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}$*. A linguagem* K *é não bloqueante se* $\overline{K} = \overline{K \cap L_m(G)}$*.*

As entradas de controle no nível π são desabilitações das transições do nível π, mas estas devem ser realizadas por meio de supervisores locais no baixo nível. Isso implica que, a cada supervisor local de um bloco X_i, está associado um conjunto de estados controláveis no baixo nível dentro do sub-autômato $G_{X_i^+}(x, Q)$.

Para assegurar que um tal conjunto de estados em cada bloco seja controlável, definimos a seguinte condição para um bloco X_i e a coleção de blocos $P_i^d = \{X_j : \langle X_i, X_j\rangle_d\}$. $P_i^u = \{X_j : \langle X_i, X_j\rangle_u\}$.

Definição 3.11 Controlabilidade IBC-não bloqueante.

Um autômato partição trilha-CD é IBC-não bloqueante se para todos os blocos X_i*, ambas condições a seguir são satisfeitas.*

(i) $\forall X_j \in P_i^d\ ,\ \forall x \in I(X_i, Q_0)\ ,\ \exists R_{i,x}^j \subseteq X_i^+$ *tal que.*

1. $\forall k\ ,\ [X_k \cap R_{i,x}^j \neq \emptyset] \Longleftrightarrow [k = i \vee k = j \vee X_k \in P_i^u]$

   e

2. $R_{i,x}^j$ *é controlável e não bloqueante em relação ao sub-autômato* $G_{X_i^+}(x, P_i^u \cup [X_j \cap X_i^+])$.

(ii) $[X_i \cap Q_m \neq \emptyset] \Longrightarrow \forall x \in I(X_i, Q_0)\ ,\ \exists R_{i,x}^{Q_m} \subseteq X_i$

tal que

1. $Q_m \cap R_{i,x}^{Q_m} \neq \emptyset$*, e*
2. $R_{i,x}^{Q_m}$ *é controlável e não bloqueante com relação ao sub-autômato* $G_{X_i^+}(x, Q_m \cap X_i)$.

A condição (*ii*) *IBC-Não bloqueante* assegura que, dentro de cada estado marcado no nível π , a alcançabilidade de estados marcados é garantida.

Na Figura 3.16 mostra-se um exemplo de uma partição IBC não bloqueante

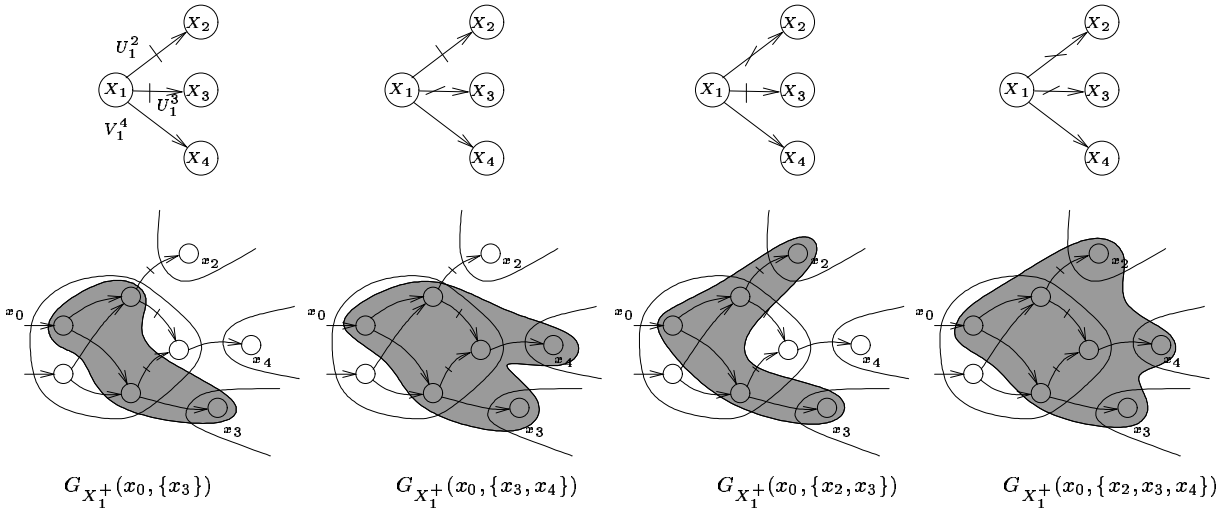


Figura 3.16: Condição IBC não bloqueante.

Pode-se notar que os blocos conformados por um estado satisfazem trivialmente os requisitos da definição IBC-não bloqueante, desta maneira, a partição identidade $\pi^{id} = \{\{x\} \ : \ x \in X\}$ é IBC-não bloqueante.

A definição 3.11 dá a condição de existência de um conjunto de estados controláveis. Assim para um dado autômato partição-π que seja IBC-não bloqueante, para cada $\langle X_i, X_j \rangle_d$, $(1 \leq i,j \leq |\pi|)$ e $x \in I(X_i, Q_0)$, podemos calcular o máximo conjunto de estados controláveis e não-bloqueantes, o qual será etiquetado por $R^{X_j}_{X_{i,x}}$ resultado da união de todos os conjuntos que satisfaçam a condição de controlabilidade e não-bloqueio. De igual forma, para cada bloco $X_i \in Q^{\pi}_m (1 \leq i \leq |\pi|)$ e estado $x \in I(X_i, Q_0)$, etiquetamos o máximo conjunto de estados controláveis e não-bloqueantes por $R^{Q_m}_{X_{i,x}}$.

Etiquetamos os supervisores os quais realizam o máximo conjunto de estados controláveis $R^{X_j}_{X_{i,x}}$ e $R^{Q_m}_{X_{i,x}}$ por $f^{X_j}_{X_{i,x}} : X_i \longrightarrow 2^{\Sigma}$ e $f^{Q_m}_{X_{i,x}} : X_i \longrightarrow 2^{\Sigma}$ respectivamente.

Para uma linguagem de especificação no nível π, $K^{\pi} \subseteq L(G^{\pi})$, o seguinte esquema traduz a entrada de controle $H^{\pi} : L(G^{\pi}) \longrightarrow 2^{\Sigma^{\pi}}$, a qual sintetiza K^{π} em G^{π}, para uma entrada de controle no baixo nível $h_{low} : L(G) \longrightarrow 2^{\Sigma}$. Este esquema é ilustrado na Figura 3.17

Dada uma partição π IBC-não bloqueante, a seguir mostra-se a tradução da ação

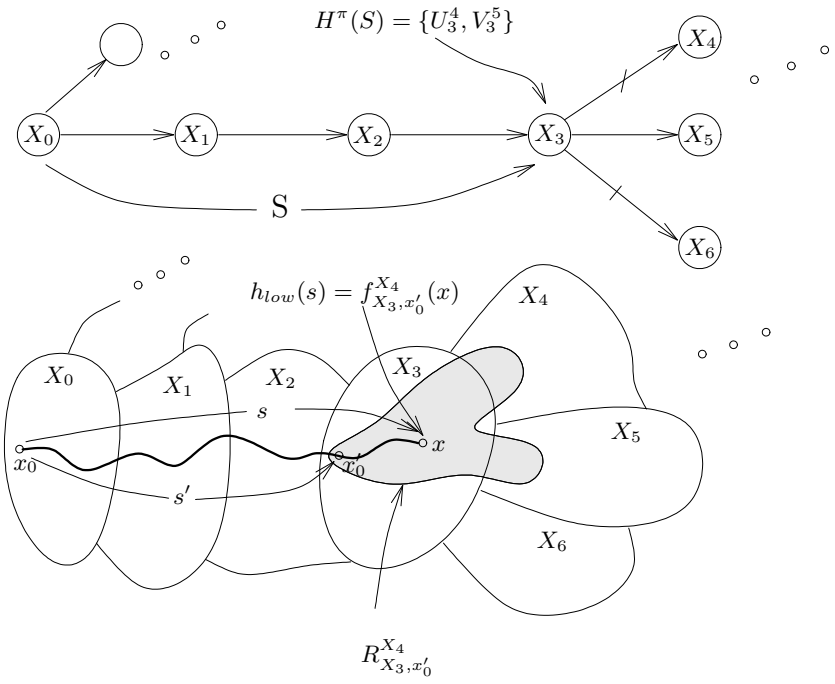


Figura 3.17: Tradução do controle do alto para baixo nível.

do supervisor do nível π no baixo nível.

Entrada: H^π

[a] Para $s \in L(G)$ tal que $\Theta_\pi(s) = \epsilon$,

$$h_{low}(s) = \bigcup_{W \in H^\pi(\epsilon)} f^{\delta^\pi(X_0,W)}_{X_0,x_0}(\delta(x_0,s)).$$

[b] Para $s \in L(G)$ tal que $\Theta_\pi(s) = S$,

$$h_{low}(s) = \bigcup_{W \in H^\pi(S)} f^{\delta^\pi(X',W)}_{X',x_0'}(\delta(x_0,s)).$$

onde $X' = \delta^\pi(X_0, S)$, $x_0' = \delta(x_0, s')$ e s' é a mínima cadeia tal que $\Theta_\pi(s') = S$ e $s' \leq s$ (i.e. para algum outro s'' satisfazendo aquele requerimento, $s' \leq s'' \leq s$).

[c] Finalmente, para $s \in L(G)$ tal que $\Theta_\pi(s) = S \in L_m(G^\pi)$,

$$h_{low}(s) = f^{Q_m}_{X',x_0'}(\delta(x_0,s)),$$

onde, outra vez, $X' = \delta^\pi(X_0, S)$, $x_0' = \delta(x_0, s')$ e s' é a mínima cadeia tal que $\Theta(s') = \Theta_\pi(s)$ e $s' \leq s$.

Saída : h_{low}

Etiquetamos a linguagem de baixo nível resultado do controle h_{low} como $K_{low}^{IBC}(K^{\pi})$.

3.2.3 Consistência do autômato π

Comparando com a teoria de controle hierárquico proposto por Wong e Wonham (1996), a seguir mostra-se que a condição de *IBC-não bloqueante* consegue o mesmo resultado de consistência que aquele de consistência de controlabilidade; isto é, uma linguagem do nível π controlável é imagem de uma linguagem controlável e toda linguagem controlável no baixo nível tem uma imagem controlável.

Teorema 3.1 *(Caines e Hubbard 1998a) Considerando uma partição-π Trilha-CD de X no autômato G e qualquer $x_0 \in Q_0$. Se uma linguagem K é não bloqueante e controlável com relação a G, então $\Theta_{\pi}(K)$ é não bloqueante e controlável com relação a G^{π}*

Pode-se notar que para qualquer subconjunto de estados R de X que seja não-bloqueante e controlável, existe uma única máxima linguagem controlável e não-bloqueante L_R que tem R como conjunto de estados alcançáveis. Também pode-se notar que para qualquer linguagem K não-bloqueante e controlável, o conjunto de estados alcançados deve também ser não-bloqueante e controlável. Desta forma temos a seguinte corolário:

Corolário 3.1 *(Caines e Hubbard 1998a) Considerando uma partição-π Trilha-CD de X no autômato G e qualquer $x_0 \in Q_0$. Se $R \subseteq X$ é não bloqueante e controlável em relação a G, então $\Theta_{\pi}(R)$ é não bloqueante e controlável em relação a G^{π}*

Teorema 3.2 *(Caines e Hubbard 1998a) Considerando uma partição-π IBC-não bloqueante de X no autômato G e qualquer $x_0 \in Q_0$. Se K^{π} é não-bloqueante e controlável em relação a G^{π} então $K_{low}^{IBC}(K^{\pi})$ é não-bloqueante e controlável em relação a G e $\Theta_{\pi}(K_{low}^{IBC}(K^{\pi})) = K^{\pi}$.*

Comparando com a abordagem de Wong e Wonham (1996) pode-se afirmar que se G^{π} é IBC-não bloqueante, então o par $(\tilde{G}_{\Theta_{\pi}}, G^{\pi})$ possui consistência hierárquica forte. O inverso nem sempre é verdade, i.e. consistência hierárquica forte não implica IBC não bloqueante.

3.3 Conclusão

Neste capítulo foram apresentadas diversas abordagens para controle supervisório hierárquico, das quais serão comparadas as principais vantagens e desvantagens.

Com a abordagem de Zhong e Wonham (1990), limitada a linguagens prefixo fechadas, se obtêm condições de consistência hierárquica de baixo nível e consistência hierárquica. Com estas condições consegue-se que qualquer especificação controlável no nível gerencial seja imagem de uma linguagem controlável do nível operacional.

Wong e Wonham (1996) estendem a abordagem anterior. Com a consistência hierárquica forte garante-se que qualquer linguagem controlável no nível gerencial seja imagem de uma linguagem controlável no nível operacional, e qualquer linguagem controlável no nível operacional, tenha a sua imagem controlável no nível gerencial. Além disso, trata-se o problema de controle hierárquico para linguagens marcadas. Conforme mostrou-se na seção 3.1.3, as condições estabelecidas por Wong e Wonham (1996) para conseguir consistência hierárquica forte com linguagens marcadas são condições conservadoras, como é o caso de mapa repórter observador e consistência de marcação.

Um dos maiores problemas destas abordagens é o refinamento do canal de informação, a fim de obter uma estrutura hierárquica com todas as boas propriedades.

Por outro lado, a abordagem de controle hierárquico por agregação de estados proposta por Caines e Hubbard (1998a), também tem as suas limitações. Numa agregação nem sempre a condição de trilha-CD é satisfeita, e caso seja, requer-se mais uma condição de consistência hierárquica denominada IBC-não bloqueante. A dificuldade desta abordagem está em se obter uma partição que seja trilha-CD e IBC-não bloqueante, uma vez que não se apresenta uma forma sistemática para cálculo de modelos que satisfaçam essas condições. Outra limitação desta abordagem é que a especificação só pode ser expressa em termos de estados proibidos.

O que se pretende nesta tese é propor uma arquitetura hierárquica de tal modo que as especificações não se limitem a estados proibidos e que o modelo do nível abstrato seja obtido através de um algoritmo que garante consistência hierárquica forte, além de relaxar as condições restritivas de observabilidade e consistência de marcação.

Todas as abordagens apresentadas neste capítulo consideram a estrutura de controle "convencional" particionada em eventos controláveis e não controláveis. No próximo capítulo apresenta-se um novo modelo para controle supervisório de SEDs com estruturas de controle avançadas, que será considerado como base para a proposta de controle hierárquico a ser apresentada posteriormente.

Capítulo 4

Uma Nova Abordagem para Controle Supervisório

Na teoria de controle supervisório para Sistemas a Eventos Discretos introduzida por Ramadge e Wonham (1987) o objetivo é controlar o comportamento lógico do sistema descrito em termos de linguagens e autômatos. Nesta abordagem o sistema é representado por uma linguagem gerada e uma linguagem marcada. O mecanismo de controle está baseado na partição do conjunto de eventos em dois subconjuntos, eventos controláveis e não controláveis. Eventos controláveis são aqueles que podem ser desabilitados por um agente externo. Eventos não controláveis são aqueles cuja ocorrência não pode ser evitada.

A abordagem de Ramadge e Wonham foi estendida para tratar sistemas de grande escala. Algumas destas extensões são: SEDs com eventos forçados (Golaszewski e Ramadge 1987), SEDs temporizados (Brandin e Wonham 1994), controle hierárquico de SEDs (Pu 2000, Wong e Wonham 1996), entre outros. Um fato comum de tais extensões é o desenvolvimento de diferentes formas para definir a controlabilidade dos eventos. No entanto, nenhum deles explora formas para definir as possíveis tarefas em malha fechada, especificamente, quando observamos um sistema num alto nível de abstração, como é o caso do controle hierárquico de SEDs. Nesse caso a definição do controle dos eventos e as possíveis tarefas em malha fechada requereriam uma abordagem alternativa à abordagem tradicional de Ramadge e Wonham para uma descrição apropriada do sistema.

Este capítulo introduz um novo modelo formal para controle supervisório de Sistemas a Eventos Discretos proposto por Cury et al. (2001) onde o sistema é representado por uma linguagem prefixo fechada e uma estrutura de controle. A linguagem descreve

o conjunto de todas as cadeias de eventos que o sistema pode gerar. A estrutura de controle é uma função que associa a cada cadeia um conjunto de padrões de controle, cada padrão de controle define o conjunto de eventos habilitados após a cadeia e um atributo de marcação que define se a cadeia é uma tarefa completa ou não. Uma estrutura de controle dependente do estado é definida de tal forma que, além de sua ação de habilitação de eventos a mesma também encapsula dinamicamente o atributo de marcação do sistema. O modelo proposto mostra-se apropriado para modelagem e controle de sistemas num alto nível de abstração, como visto em (da Cunha e Cury 2002, Torrico e Cury 2002c). Neste caso, esta abordagem proporciona uma representação compacta resultando numa economia do espaço de estados para descrever o sistema e permite modelar o sistema no seu nível de abstração. Neste capítulo apresenta-se a teoria de controle supervisório para este modelo, são dadas as condições para existência de supervisores, apresenta-se um método para síntese do supervisor ótimo, e para fechar o capítulo apresenta-se um exemplo de aplicação.

4.1 Representação de um SED Controlado

Introduz-se uma representação para SED controlado como um par $D = (L, \Gamma)$, onde L é uma linguagem prefixo fechada sobre um alfabeto de eventos Σ, e Γ é uma estrutura de controle. Sejam $\{M, N\}$ etiquetas de marcação, a estrutura de controle é um mapeamento $\Gamma : L \rightarrow 2^{2^{\Sigma} \times \{M,N\}}$ que associa a cada cadeia $s \in L$ um conjunto de padrões de controle

$$\Gamma(s) = \Big\{(\gamma, \#) \in 2^{\Sigma} \times \{M, N\}\Big\}$$

tal que um padrão de controle $(\gamma, \#) \in \Gamma(s)$ significa:

1. $\gamma \subset \Sigma$, é um conjunto de eventos habilitados após s;
2. $\# = M$, significa que, quando selecionado γ, a cadeia s é aceita como uma tarefa do sistema, e
3. $\# = N$, significa que, quando selecionado γ, a cadeia s não é considerada uma tarefa do sistema.

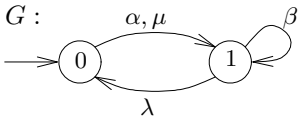
Exemplo 4.1 *Para ilustrar um SED controlado, considera-se o alfabeto* $\Sigma = \{\alpha, \beta, \mu, \lambda\}$ *e o sistema* D *definido pela linguagem* $L = [(\alpha + \mu)\beta^*(\epsilon + \lambda)]^*$, *e a estrutura de controle* Γ *definido como:*

- $\Gamma(\epsilon) = \{(\emptyset, M),\ (\{\alpha\}, M)\}$,

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

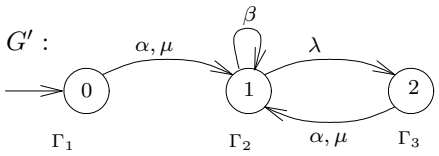
Percebe-se que estes três controles apresentados anteriormente caracterizam uma classe de cadeias que podem ser consideradas marcadas ou não dependendo do padrão de controle selecionado, este é um fato diferente da abordagem tradicional de Ramadge e Wonham.

Dado um SED controlado $D = (L, \Gamma)$, representa-se L por um gerador $G = (\Sigma, Q, \delta, q_0, Q_m)$ como definido na seção 2.1.5, com Q_m fixado em $\emptyset$, tal que $L(G) = \{s \in \Sigma^* : \delta(q_0, s) \text{ definido } \} = L$ e $L_m(G) = \{s \in L(G) : \delta(q_0, s) \in Q_m\} = \emptyset$. Também representa-se a estrutura de controle Γ como uma tabela que armazena o correspondente conjunto de padrões de controle para cada cadeia de L. Na Figura 4.1(a) representa-se o gerador para L do exemplo 4.1.



cadeias	padrões de controle
$\Gamma(\epsilon) = \Gamma_1$	$\{(\emptyset, M),\ (\{\alpha\}, M)\}$
$\Gamma(((\alpha + \mu)\beta^*\lambda)^*(\alpha + \mu)) = \Gamma_2$	$\{(\{\lambda, \beta\}, N)\}$
$\Gamma(((\alpha + \mu)\beta^*\lambda)^+) = \Gamma_3$	$\{(\emptyset, N), (\{\alpha\}, M), (\{\alpha, \mu\}, M)\}$

(a) Gerador e a tabela para Γ.



(b) Gerador Γ dependente do estado .

Figura 4.1: Geradores para L.

Um SED controlado D pode também ser representado por um gerador G' com estruturas de controle dependentes do estado. Cada estado de G' é a intersecção entre as classes de equivalência de Nerode (Carroll e Long 1989) e as classes de Γ-equivalência para L, onde a Γ-equivalência (R_Γ) relaciona cadeias em L com igual conjunto de padrões de controle, isto é, $(\forall s_1, s_2 \in \overline{L}) : s_1 \equiv s_2 \bmod R_\Gamma \Leftrightarrow \Gamma(s_1) = \Gamma(s_2)$. Para o caso regular, isto leva a uma representação finita, uma vez que a Γ-equivalência tem um *rank* finito, desde que Σ seja finito. Para o Exemplo 4.1, na Figura 4.1(b) mostra-se o refinamento do gerador da Figura 4.1(a) tal que o autômato resultante representa uma estrutura de controle dependente do estado.

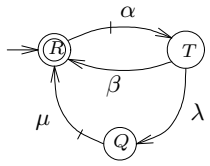
Pode ser provado que este modelo proposto para SEDs controlados serve como uma generalização dos seguintes modelos: modelo clássico de Ramadge e Wonham (1987), SEDs com eventos forçados (Golaszewski e Ramadge 1987), SEDs temporizados (Brandin e Wonham 1994), controle hierárquico de SEDs (Zhong e Wonham 1990, Wong e Wonham 1996, Caines et al. 1997, Pu 2000) e controle supervisório com supervisor marcador (Wonham 1998). Na teoria clássica de Ramadge e Wonham um SED é representado por um par (L, L_m), onde $L \subseteq \Sigma^*$ é uma linguagem prefixo fechada e Ł$_m \subseteq L$ define as tarefas do sistema. O controle é definido pela partição de Σ dentro de subconjuntos controláveis (Σ_c) e não controláveis(Σ_u). A representação de tal modelo, dentro desta proposta, é dada pela linguagem L e uma estrutura de controle Γ tal que $\forall s \in L$, $\Gamma(s) = \{(\gamma, \#) \in 2^{\Sigma} \times \{M, N\} : \Sigma_u \subseteq \gamma \subseteq \Sigma \ \wedge \ \# = M \text{ se } s \in L_m \ \wedge \ \# = N \text{ se } s \in (L - L_m)\}$. No caso de controle supervisório com poder de desmarcação, o supervisor pode selecionar as tarefas em malha fechada de L_m, então a única diferença com o caso anterior é que na estrutura de controle, o atributo #, pode ser escolhido M ou N, para $s \in L_m$. Na continuação apresenta-se um exemplo deste caso.

Exemplo 4.2 *Na Figura 4.2(a) apresenta-se o modelo genérico de uma máquina de três estados utilizado na abordagem de Ramadge e Wonham. Os estados da máquina são repouso (R), trabalhando (T) e quebrada (Q), e dirigida pelos eventos: início de operação (α), final de operação (β), quebra (λ) e reparo (μ). A Figura 4.2(b) apresenta o modelo para a mesma máquina de três estados dentro da abordagem proposta com estruturas de controle dependentes do estado. Além disso, na Figura 4.2(c) apresenta-se o mesmo modelo dentro desta abordagem considerando o supervisor marcador. Nota-se que a diferença em relação à Figura 4.2(b) é a disponibilidade de padrões de controle com atributo de marcação M e N para cada estado marcado correspondente ao modelo original.*

Para o controle de um SED controlado impõe-se que o conjunto de padrões de controle $\Gamma(s)$ deve satisfazer os seguintes requisitos, para todo $s \in L$,

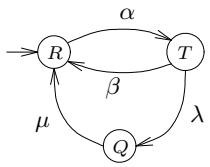
1. $(\gamma_1, N), (\gamma_2, N) \in \Gamma(s) \ \rightarrow \ (\gamma_1 \cup \gamma_2, N) \in \Gamma(s)$.
2. $(\gamma_1, M), (\gamma_2, \#) \in \Gamma(s) \ \rightarrow \ (\gamma_1 \cup \gamma_2, M) \in \Gamma(s), \# = M, N$.

Os requisitos anteriores têm duas interpretações. Primeiro, o conjunto $\gamma \subset \Sigma$ tal que $(\gamma, \#) \in \Gamma(s)$ deve ser fechado para operação união de conjuntos para todo $s \in L$, um requerimento usual para existência de um supervisor ótimo na literatura (Golaszewski e Ramadge 1987, Ramadge e Wonham 1989, Lin et al. 1998). Segundo, os atributos de



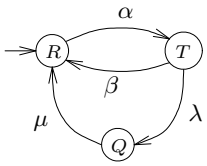
$\Sigma = \{\alpha, \beta, \lambda, \mu\}$
$\Sigma_u = \{\beta, \lambda\}$

(a) Representação dentro da abordagem de Ramadge e Wonham.



$\Gamma(R) = \{(\emptyset, M), (\{\alpha\}, M)\}$
$\Gamma(T) = \{(\{\beta, \lambda\}, N)\}$
$\Gamma(Q) = \{(\emptyset, N), (\{\mu\}, N)\}$

(b) Representação dentro do modelo proposto.



$\Gamma(R) = \{(\emptyset, N), (\emptyset, M), (\{\alpha\}, N), (\{\alpha\}, M)\}$
$\Gamma(T) = \{(\{\beta, \lambda\}, N)\}$
$\Gamma(Q) = \{(\emptyset, N), (\{\mu\}, N)\}$

(c) Representação dentro do modelo proposto, considerando supervisor marcador.

Figura 4.2: Representações da máquina de três estados.

marcação são fechados no seguinte sentido: para uma dada palavra, se (γ_1, N) e (γ_2, N), são padrões de controle válidos, então o padrão de controle não marcado, $(\gamma_1 \cup \gamma_2, N)$, deve ser válido; por outro lado, se (γ_1, M) e $(\gamma_2, \#)$, onde $\# = M$ ou N, são padrões de controle válidos, o padrão de controle marcado, $(\gamma_1 \cup \gamma_2, M)$, deve ser padrão de controle válido.

4.2 Supervisores, Γ-Compatibilidade e Existência de Supervisores

Nesta seção, define-se um supervisor, o comportamento em malha fechada do sistema, e apresenta-se a condição para existência de supervisores.

Dado um SED controlado $D = (L, \Gamma)$, um supervisor f para D é um mapeamento $f : L \to 2^{\Sigma} \times \{M, N\}$. Para um dado $s \in L$, o supervisor seleciona um padrão de controle $f(s) = (\gamma, \#) \in \Gamma(s)$, tal que em malha fechada, o conjunto de eventos ativos após s é reduzido a $\gamma \cap \Sigma_L(s)$, e a cadeia s é considerada marcada se $\# = M$ e não marcada se $\# = N$.

O comportamento do sistema em malha fechada f/D é representado por um par de linguagens, uma linguagem prefixo fechada $L(f/D)$ e uma linguagem marcada $L_m(f/D)$, tal que $L_m(f/D) \subseteq Ł(f/D) \subseteq L$. A linguagem prefixo fechada $L(f/D)$ é definida recursivamente como:

1. $\epsilon \in L(f/D)$

2. $s\sigma \in L(f/D) \longleftrightarrow s \in L(f/D) \wedge s\sigma \in L \wedge \sigma \in \gamma$, onde $f(s) = (\gamma, \#)$

e a linguagem marcada $L_m(f/D)$ como

$$s \in L_m(f/D) \longleftrightarrow s \in L(f/D) \wedge f(s) = (\gamma, M)$$

O comportamento em malha fechada $L(f/D)$ é uma linguagem prefixo fechada e contém cadeias em L permitidas por f sob supervisão, e o comportamento marcado em malha fechada $L_m(f/D)$ contém as cadeias em $L(f/D)$ onde o supervisor escolheu o padrão de controle com atributo de marcação "M". Em geral $\overline{L_m(f/D)} \subseteq L(f/D)$, e o supervisor é dito ser não bloqueante se $\overline{L_m(f/D)} = L(f/D)$. O comportamento em malha fechada pode ser representado por um gerador S onde $L(S) = L(f/D)$ e $L_m(S) = L_m(f/D)$.

Neste ponto pode ser introduzido o problema de controle supervisório (PCS) como dado a seguir.

(PCS):*Dado um SED controlado $D = (L, \Gamma)$ e uma linguagem de especificação $K \subseteq L$, encontrar um supervisor não bloqueante tal que $\emptyset \subset L_m(f/D) \subseteq K$.*

De modo a propor uma solução para o PCS, introduz-se o conceito de Γ-*compatibilidade*.

Definição 4.1 Γ-compatibilidade

A linguagem $K \subseteq L$ é Γ-compatível *se, e somente se, $K = \emptyset$ ou ;*

1. $(\forall s \in K)\ (\exists(\gamma, M) \in \Gamma(s))\ :\ \gamma \cap \Sigma_L(s) = \Sigma_K(s)$

2. $(\forall s \in \overline{K} - K)\ (\exists(\gamma, N) \in \Gamma(s))\ :\ \gamma \cap \Sigma_L(s) = \Sigma_K(s)$

O teorema a seguir mostra que Γ-*compatibilidade* é uma condição necessária e suficiente para existência de um supervisor não bloqueante que permita implementar uma linguagem K num SED controlado $D = (L, \Gamma)$.

Teorema 4.1 *Dado $D = (L, \Gamma)$ e dada uma linguagem $K \subseteq L$, com $K \neq \emptyset$, existe um supervisor não bloqueante f para D, tal que $L_m(f/D) = K$, se e somente se K é* Γ-compatível.

Prova: (Se)

Seja K, Γ-*compatível*, mostraremos que $L(f/D) = \overline{K}$.

$\big(L(f/D) \subseteq \overline{K}\big)$

Supõe-se que $w\sigma \in L(f/D)$, i.e. $w \in L(f/D)$, $\sigma \in \gamma$, onde, $f(w) = (\gamma, \#)$, e $\sigma \in \Sigma_L(w)$.

$\epsilon \in \overline{K}$,

por indução, se $w \in \overline{K}$ tem-se: para $\sigma \in \gamma$ e $\sigma \in \Sigma_L(w)$

$\Rightarrow\ \sigma \in (\gamma \cap \Sigma_L(w))$

$\Rightarrow\ \sigma \in \Sigma_K(w)$ por Γ-*compatibilidade* de K.

$\Rightarrow\ w\sigma \in \overline{K}$ Portanto $L(f/D) \subseteq \overline{K}$

$\big(L(f/D) \supseteq \overline{K}\big)$

Supõe-se que $w\sigma \in \overline{K}$, i.e. $\sigma \in \Sigma_K(w)$, e $\sigma \in \Sigma_L(w)$.

$\epsilon \in L(f/D)$,

por indução, se $w \in L(f/D)$ tem-se: para $\sigma \in \Sigma_K(w)$

$\Rightarrow\ \sigma \in (\gamma \cap \Sigma_L(w))$ por Γ-*compatibilidade* de K.

$\Rightarrow\ \sigma \in \gamma$ e $\sigma \in \Sigma_L(w)$

$\Rightarrow\ w\sigma \in L(f/D)$ Portanto $L(f/D) \supseteq \overline{K}$

Então $L(f/D) = \overline{K}$

$\forall w \in K,\ \exists(\gamma, M) \in \Gamma(w)$ pela primeira condição de Γ-*compatibilidade* (Def 4.1).

$\Rightarrow$ $L_m(f/D) = K$ pela definição de comportamento marcado em malha fechada (pág 44).

$\Rightarrow \overline{L_m(f/D)} = \overline{K}$

$\Rightarrow \overline{L_m(f/D)} = L(f/D)$ Portanto não bloqueante.

(Somente se)

Seja f um supervisor não bloqueante tal que $L_m(f/D) = K$, i.e. $\overline{L_m(f/D)} = \overline{K} = L(f/D)$, mostraremos que K é Γ-*compatível*.

$w\sigma \in L(f/D) \leftrightarrow w \in L(f/D) \wedge w\sigma \in L \wedge \sigma \in \gamma$ pela definição de comportamento em malha fechada (pág 44).

$\Rightarrow\ w\sigma \in L(f/D) \leftrightarrow w \in L \wedge w\sigma \in L \wedge \sigma \in \gamma$

$\Rightarrow\ w\sigma \in \overline{K} \leftrightarrow w\sigma \in L \wedge \sigma \in \gamma$

$\Rightarrow\ \sigma \in \Sigma_K(w) \leftrightarrow \sigma \in \Sigma_L(w) \wedge \sigma \in \gamma$

$\Rightarrow\ \ \Sigma_K(w)\ =\ \Sigma_L(w) \cap \gamma$

Também $\forall w \in K,\ \exists(\gamma, M)\ \wedge\ \forall w \in (\overline{K} - K)\ \exists(\gamma, N)$ pela definição de comportamento marcado em malha fechada (pág 44).

Portanto satisfaz os dois requisitos de Γ-*compatibilidade*

$\diamondsuit$

Como pode ser visto no teorema anterior, dentro das condições do controle supervisório de Ramadge e Wonham, a Γ-*compatibilidade* de uma linguagem é equivalente às condições de controlabilidade e L_m-fechamento (Wonham 1998).

Nesta seção será mostrado que caso uma linguagem K, a qual especifica o comportamento desejado, não seja Γ-*compatível*, existe uma aproximação de K que é única e ótima, no sentido de ser minimamente restritiva. Além disso, no caso regular, é apresentado um algoritmo para o cálculo desta aproximação.

Dado um SED controlado $D\ =\ (L, \Gamma)$ e uma especificação $K \subseteq L$, seja $\mathcal{CM}(K)$ o conjunto de linguagens Γ-*compatíveis* contidas em K, $\mathcal{CM}(K) = \{E \subset K\ :\ E$ é Γ-compatível$\}$.

Teorema 4.2 *$\mathcal{CM}(K)$ é fechado para operação união de conjuntos.*

Prova:

Seja $K_1, K_2\ \subset\ L$ duas linguagens Γ-*compatíveis* contidas em $\mathcal{CM}(K)$, então, é suficiente provar que $K_1 \cup K_2$ é Γ-*compatível*.

- Para $s \in (K_1 \cup K_2)$, podem ocorrer as seguintes situações;

  a) $s \in K_1\ \wedge\ s \notin \overline{K}_2$

  $\Rightarrow\ \exists(\gamma_1, M) \in \Gamma(s)\ :\ \gamma_1 \cap \Sigma_L(s) = \Sigma_{K_1}(s)$. Conseqüentemente satisfaz a primeira condição de Γ-*compatibilidade*

  b) $s \in K_1\ \wedge\ s \in K_2$

  $\Rightarrow\ \exists(\gamma_1, M) \in \Gamma(s)\ :\ \gamma_1 \cap \Sigma_L(s) = \Sigma_{K_1}(s)$ **e**
  $\exists(\gamma_2, M) \in \Gamma(s)\ :\ \gamma_2 \cap \Sigma_L(s) = \Sigma_{K_2}(s)$

  $\Rightarrow\ (\gamma_1 \cap \Sigma_L(s)) \cup (\gamma_2 \cap \Sigma_L(s)) = \Sigma_{K_1}(s) \cup \Sigma_{K_2}(s)$

  $\Rightarrow\ (\gamma_1 \cup \gamma_2) \cap \Sigma_L(s) = \Sigma_{K_1 \cup K_2}(s)$

$\Rightarrow\ \exists((\gamma_1 \cup \gamma_2), M) \in \Gamma(s)$, por causa do "segundo" requerimento para $\Gamma(s)$. Conseqüentemente satisfaz a primeira condição de Γ-*compatibilidade*.

c) $s \in K_1\ \wedge\ s \in (\overline{K}_2 - K_2)$

$\Rightarrow\ \exists(\gamma_1, M) \in \Gamma(s)\ :\ \gamma_1 \cap \Sigma_L(s) = \Sigma_{K_1}(s)$ **e**
$\exists(\gamma_2, N) \in \Gamma(s)\ :\ \gamma_2 \cap \Sigma_L(s) = \Sigma_{K_2}(s)$

$\Rightarrow\ (\gamma_1 \cap \Sigma_L(s)) \cup (\gamma_2 \cap \Sigma_L(s)) = \Sigma_{K_1}(s) \cup \Sigma_{K_2}(s)$

$\Rightarrow\ (\gamma_1 \cup \gamma_2) \cap \Sigma_L(s) = \Sigma_{K_1 \cup K_2}(s)$

$\Rightarrow\ \exists((\gamma_1 \cup \gamma_2), M) \in \Gamma(s)$, por causa do "segundo" requerimento para $\Gamma(s)$. Conseqüentemente satisfaz a primeira condição de Γ-*compatibilidade*.

- Para $s\ \in\ (\overline{K_1 \cup K_2} - K_1 \cup K_2)$, podem ocorrer as seguintes situações;

a) $s\ \in\ (\overline{K}_1 - K_1)\ \wedge\ s \notin \overline{K}_2$

$\Rightarrow\ \exists(\gamma_1, N) \in \Gamma(s)\ :\ \gamma_1 \cap \Sigma_L(s) = \Sigma_{K_1}(s)$. Conseqüentemente satisfaz a segunda condição de Γ-*compatibilidade*.

b) $s\ \in\ (\overline{K}_1 - K_1)\ \wedge\ s \in (\overline{K}_2 - K_2)$

$\Rightarrow\ \exists(\gamma_1, N) \in \Gamma(s)\ :\ \gamma_1 \cap \Sigma_L(s) = \Sigma_{K_1}(s)$ **e**
$\exists(\gamma_2, N) \in \Gamma(s)\ :\ \gamma_2 \cap \Sigma_L(s) = \Sigma_{K_2}(s)$

$\Rightarrow\ (\gamma_1 \cap \Sigma_L(s)) \cup (\gamma_2 \cap \Sigma_L(s)) = \Sigma_{K_1}(s) \cup \Sigma_{K_2}(s)$

$\Rightarrow\ (\gamma_1 \cup \gamma_2) \cap \Sigma_L(s) = \Sigma_{K_1 \cup K_2}(s)$

$\Rightarrow\ \exists((\gamma_1 \cup \gamma_2), N) \in \Gamma(s)$, por causa do "primeiro" requerimento para $\Gamma(s)$. Conseqüentemente satisfaz a segunda condição de Γ-*compatibilidade*.

$\diamondsuit$

O teorema anterior e o fato de $\mathcal{CM}(K)$ ser não vazia levam à existência de um único elemento supremo de $\mathcal{CM}(K)$, dado por $\sup \mathcal{CM}(K) = \bigcup\{E\ :\ E \in \mathcal{CM}(K)\}$. Portanto este resultado é usado para trazer uma solução ao PCS como apresentado na proposição a seguir.

Proposição 4.1 (Solução ao PCS) *O problema de controle supervisório tem solução se e somente se* $\sup \mathcal{CM}(K) \neq \emptyset$

Prova:

(Se)

Supondo que $\sup \mathcal{CM}(K) \neq \emptyset$, então, a partir do teorema 4.1 existe um supervisor não bloqueante f para D tal que $\emptyset \subset L_m(f/D) = \sup \mathcal{CM}(K) \subseteq K$, conseqüentemente o problema de controle supervisório tem solução.

(Somente se)

Supondo que o problema de controle supervisório tem solução, isto é, existe um supervisor não bloqueante f' para D tal que $\emptyset \subset L_m(f'/D) \subseteq K$. Seja $L_m(f'/D) = K'$. A partir do teorema 4.1, K' é Γ- compatível e $K' \neq \emptyset$. Então, o conjunto $\mathcal{CM}(K)$ contém linguagens diferentes de $\emptyset$, e a partir do teorema 4.2, existe uma máxima linguagem não vazia Γ-compatível contida em K, isto é, $\sup \mathcal{CM}(K) \neq \emptyset$.

$\diamondsuit$

O resultado da máxima linguagem Γ-compatível é usado como uma política de controle ótima para resolver o PCS, no sentido de ser máximamente permissivo. Na próxima seção apresenta-se o algoritmo para encontrar esta linguagem no caso de sistemas representáveis por geradores finitos.

4.3 Síntese de um Supervisor Ótimo

Esta seção apresenta um algoritmo para calcular a máxima linguagem Γ-compatível contida numa dada linguagem.

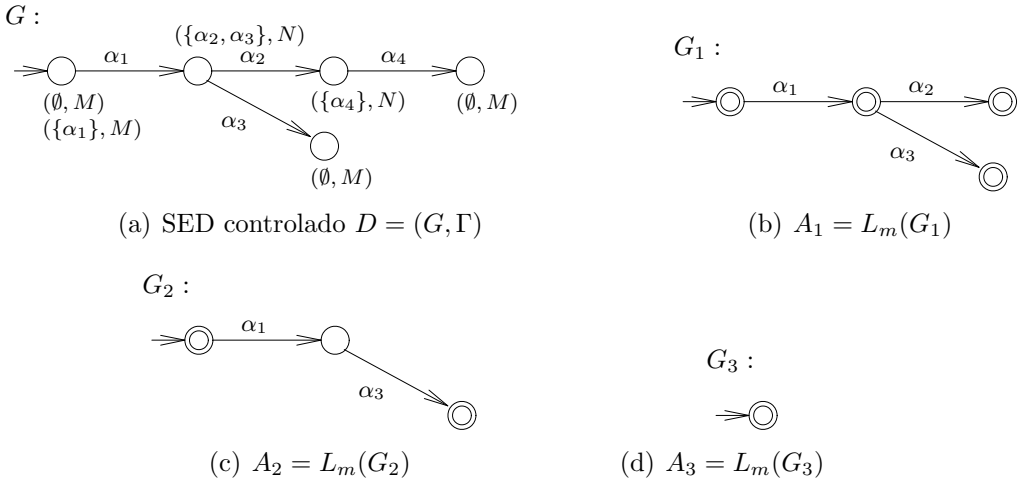
Dado um SED controlado $D = (L, \Gamma)$ sobre Σ e uma linguagem $\emptyset \subset K \subseteq L$, o problema é achar a máxima linguagem Γ-compatível contida em K. Caracteriza-se o cálculo como a obtenção do máximo ponto fixo de um *operador monótono*.

Define-se o operador $\Omega : 2^L \rightarrow 2^L$ para $A \subseteq L$ como:

$$\begin{aligned}\Omega(A) \;=\; & \{s \in A : [\exists(\gamma, M) \in \Gamma(s) \text{ tal que } \gamma \cap \Sigma_L(s) \subseteq \Sigma_A(s)] \text{ e} \\ & [\forall s' \leq s, \text{ com } s' = s''\sigma \text{ e } \sigma \in \Sigma, \\ & (\text{se } s'' \in A, \text{ então } \exists(\gamma, \#) \in \Gamma(s''),\ \# \in \{M, N\}, \text{ tal que } \gamma \cap \Sigma_L(s'') \subseteq \Sigma_A(s'') \text{ e } \sigma \in \gamma) \text{ ou} \\ & (\text{se } s'' \in \overline{A} - A, \text{ então } \exists(\gamma, N) \in \Gamma(s''), \text{ tal que } \gamma \cap \Sigma_L(s'') \subseteq \Sigma_A(s'') \text{ e } \sigma \in \gamma)]\}. \qquad (4.1)\end{aligned}$$

No exemplo a seguir apresenta-se o efeito da aplicação sucessiva do operador Ω.

Exemplo 4.3 *Dado o SED controlado D apresentado na Figura 4.3(a) e uma linguagem $A_1 = L_m(G_1)$, Figura 4.3(b), o objetivo é ilustrar a aplicação sucessiva do operador Ω, equação 4.1.*



(a) SED controlado $D = (G, \Gamma)$ (b) $A_1 = L_m(G_1)$

(c) $A_2 = L_m(G_2)$ (d) $A_3 = L_m(G_3)$

Figura 4.3: Aplicações sucessivas do operador Ω.

- *$\Omega(A_1)$ é calculado a partir da seguinte análise sobre as cadeias em A_1 e seus prefixos:*
  - *$\epsilon \in \Omega(A_1)$ porque $\exists(\{\alpha_1\}, M) \in \Gamma(\epsilon)$ tal que $\{\alpha_1\} \cap \{\alpha_1\} \subseteq \{\alpha_1\}$*
  - *$\alpha_1 \notin \Omega(A_1)$ porque $\nexists(\gamma, M) \in \Gamma(\alpha_1)$ tal que $\gamma \cap \{\alpha_2, \alpha_3\} \subseteq \{\alpha_2, \alpha_3\}$*
  - *$\alpha_1\alpha_2 \notin \Omega(A_1)$ porque $\nexists(\gamma, M) \in \Gamma(\alpha_1\alpha_2)$ tal que $\gamma \cap \{\alpha_4\} \subseteq \emptyset$*
  - *$\alpha_1\alpha_3 \in \Omega(A_1)$ porque*
    - *para $\alpha_1\alpha_3$, $\exists(\emptyset, M) \in \Gamma(\alpha_1\alpha_3)$ tal que $\emptyset \cap \emptyset \subseteq \emptyset$ e*
    - *para $\alpha_1 \in \overline{\alpha_1\alpha_3}$, $\exists(\{\alpha_2, \alpha_3\}, N) \in \Gamma(\alpha_1)$ tal que $\{\alpha_2, \alpha_3\} \cap \{\alpha_2, \alpha_3\} \subseteq \{\alpha_2, \alpha_3\}$ e $\alpha_3 \in \{\alpha_2, \alpha_3\}$ e*
    - *para $\epsilon \in \overline{\alpha_1\alpha_3}$, $\exists(\{\alpha_1\}, M) \in \Gamma(\epsilon)$ tal que $\{\alpha_1\} \cap \{\alpha_1\} \subseteq \{\alpha_1\}$ e $\alpha_1 \in \{\alpha_1\}$.*

  A aplicação do operador Ω para A_1 recorta *as cadeias α_1 e $\alpha_1\alpha_2$ de A_1, e $\Omega(A_1) = A_2 = L_m(G_2)$ é mostrado na Figura 4.3(c).*

- *Aplicando Ω para A_2, tem-se:*

– $\epsilon \in \Omega(A_2)$ *porque* $\exists(\{\alpha_1\}, M) \in \Gamma(\epsilon)$ *tal que* $\{\alpha_1\} \cap \{\alpha_1\} \subseteq \{\alpha_1\}$

– $\alpha_1\alpha_3 \notin \Omega(A_2)$ *porque para* $\alpha_1 \in \overline{\alpha_1\alpha_3}$, $\nexists(\gamma, \#) \in \Gamma(\alpha_1)$ *tal que* $\gamma \cap \{\alpha_2, \alpha_3\} \subseteq \{\alpha_3\}$.

A Figura 4.3(d) apresenta $\Omega(A_2) = A_3 = L_m(G_3)$.

- *Finalmente, aplicando mais uma vez o operador* Ω *resulta* $\Omega(A_3) = A_3$.

Uma linguagem $A \subseteq L$ é dita ser um *ponto fixo* de Ω sempre que $\Omega(A) = A$. O teorema a seguir é uma propriedade importante para pontos fixos do operador Ω.

Teorema 4.3 *Dado um SED controlado* $D = (L, \Gamma)$ *sobre* Σ, *os pontos fixos do operador* Ω *são as sub-linguagens* Γ*-compatíveis de* D.

Prova:

A prova deste teorema corresponde a provar que para qualquer linguagem $A \subseteq L$, $A = \Omega(A)$ se e somente se A é Γ-compatível em relação a L.

Considera-se a linguagem auxiliar $L_b(A)$ contendo as cadeias de A recortadas pelo operador Ω, isto é $L_b(A) = A - \Omega(A)$, definido por

$$
\begin{aligned}
L_b(A) \;=\; & \{s \in A : [\forall(\gamma, M) \in \Gamma(s),\ \gamma \cap \Sigma_L(s) \not\subseteq \Sigma_A(s)] \text{ ou} \\
& [\exists s' \leq s \text{ com } s' = s''\sigma \text{ e } \sigma \in \Sigma, \text{ para que} \\
& (s'' \in A \text{ e } \forall(\gamma, \#) \in \Gamma(s''),\ \# \in \{M, N\}, \text{ também } \gamma \cap \Sigma_L(s'') \not\subseteq \Sigma_A(s'') \text{ ou } \sigma \notin \gamma) \text{ ou} \\
& (s'' \in \overline{A} - A \text{ e } \forall(\gamma, N) \in \Gamma(s''), \text{ também } \gamma \cap \Sigma_L(s'') \not\subseteq \Sigma_A(s'') \text{ ou } \sigma \notin \gamma)]\}. \qquad (4.2)
\end{aligned}
$$

(Se)

Partindo da definição 4.1, se A é Γ-compatível, então para qualquer $s \in \overline{A}$,

- se $s \in A$, $\exists(\gamma, M) \in \Gamma(s)$ tal que $\gamma \cap \Sigma_L(s) = \Sigma_A(s)$ ou
- se $s \in \overline{A} - A$, $\exists(\gamma, N) \in \Gamma(s)$ tal que $\gamma \cap \Sigma_L(s) = \Sigma_A(s)$.

As condições acima podem ser alternativamente escritas como, para qualquer $s \in A$:

- $\exists(\gamma, M) \in \Gamma(s)$ tal que $\gamma \cap \Sigma_L(s) = \Sigma_A(s)$, e
- para qualquer $s' \leq s$, com $s' = s''\sigma$ e $\sigma \in \Sigma$
  - se $s'' \in A$, então $\exists(\gamma, M) \in \Gamma(s'')$ tal que $\gamma \cap \Sigma_L(s'') = \Sigma_A(s'')$ e $\sigma \in \gamma$, ou
  - se $s'' \in \overline{A} - A$, $\exists(\gamma, N) \in \Gamma(s'')$ tal que $\gamma \cap \Sigma_L(s'') = \Sigma_A(s'')$ e $\sigma \in \gamma$.

Pelo último conjunto de expressões, $L_b(A) = \emptyset$ e $A = \Omega(A)$, isto é, A é um ponto fixo de Ω.

(Somente se)

A é um ponto fixo de Ω, então $L_b(A) = \emptyset$. Consideram-se as seguintes situações:

1. Para qualquer $s \in A$ existe (γ, M) em $\Gamma(s)$ tal que $\gamma \cap \Sigma_L(A) = \Sigma_A(s)$.

   - Caso 1 – Supondo que $\not\exists s' \in A$ tal que $s < s'$, então $\Sigma_A(s) = \emptyset$. Se $\exists(\gamma, M) \in \Gamma(s)$ tal que $\gamma \cap \Sigma_L(s) \subseteq \Sigma_A(s)$, então $\gamma \cap \Sigma_L(s) = \emptyset$ e a igualdade satisfaz.
   - Caso 2 – Supondo que $\exists s' \in A$ tal que $s < s'$. Por contradição, supondo que para qualquer $(\gamma, M) \in \Gamma(s)$ tal que $\gamma \cap \Sigma_L(s) \subseteq \Sigma_A(s)$, a inclusão é estrita, isto é, existe $\sigma \in \Sigma_A(s)$ tal que $\sigma \notin \gamma$. Se $\sigma \in \Sigma_A(s)$, existe $s' \in A$ tal que $s < s'$ e $s\sigma \leq s'$. Portanto, para qualquer $(\gamma, M) \in \Gamma(s)$ tal que $\gamma \cap \Sigma_L(s) \subseteq \Sigma_A(s)$, tem-se $\sigma \notin \gamma \cap \Sigma_L(s)$, então $\sigma \notin \gamma$. Isto contradiz a hipótese que A é um ponto fixo de Ω, uma vez que $\exists s' \in A$ para que $\exists s\sigma \leq s'$, $s \in A$ e $\not\exists(\gamma, M) \in \Gamma(s)$ (desde que $s \in A$, (γ, N) não satisfaz), tal que $\gamma \cap \Sigma_L(s) \subseteq \Sigma_A(s)$ e $\sigma \in \gamma$.

2. Para qualquer $s \in \overline{A} - A$ existe (γ, N) em $\Gamma(s)$ tal que $\gamma \cap \Sigma_L(s) = \Sigma_A(s)$.

   Por hipótese, uma vez que $s \in \overline{A} - A$. Portanto, pelo fato que A é um ponto fixo de Ω, $\exists(\gamma, N) \in \Gamma(s)$ tal que $\gamma \cap \Sigma_L(s) \subseteq \Sigma_A(s)$ e $\sigma \in \gamma$. Por contradição, supondo que para qualquer $(\gamma, N) \in \Gamma(s)$ para que $\gamma \cap \Sigma_L(s) \subseteq \Sigma_A(s)$ a inclusão é estrita, isto é, $\gamma \cap \Sigma_L(s) \subset \Sigma_A(s)$. Então existe $\sigma \in \Sigma_A(s)$ e $s' \in A$ para que $s\sigma \leq s'$ e $\forall(\gamma, N) \in \Gamma(s)$ tal que $\gamma \cap \Sigma_L(s) \subseteq \Sigma_A(s)$, $\sigma \notin \gamma \cap \Sigma_L(s)$, isto é $\sigma \notin \gamma$. Isto contradiz o fato que A é um ponto fixo de Ω.

Portanto, a partir das duas situações apresentadas acima, se A é um ponto fixo de Ω então A é Γ-compatível, o que completa a prova.

$\diamondsuit$

Corolário 4.1 *Dado $K \subseteq L$, o maior ponto fixo de Ω contido em K é a máxima linguagem Γ-compatível contida em K.*

Prova:

Desde que as linguagens Γ-compatíveis de $D = (L, \Gamma)$ são pontos fixos de Ω o maior ponto fixo de Ω contido em K é a máxima linguagem Γ-compatível contida em K.

$\diamondsuit$

Dada uma linguagem $K \subseteq L$, denota-se $\{K_i\}$ como a seqüência de linguagens geradas pela aplicação sucessiva do operador Ω sobre K. A seqüência $\{K_i\}$ é definida por $K_0 = K$ e $K_{i+1} = \Omega(K_i)$.

O teorema seguinte caracteriza uma propriedade para a seqüência $\{K_i\}$.

Teorema 4.4 *A seqüência $\{K_i\}$ converge para a máxima linguagem Γ-compatível contida em K.*

Prova:

Para provar que a seqüência $\{K_i\}$ converge para a máxima linguagem Γ-compatível, demonstra-se primeiramente que a seqüência converge para um ponto fixo de Ω contido em K, a qual, pelo teorema 4.3 é Γ-compatível. Então demonstra-se que a aplicação sucessiva do operador Ω não pula nenhuma linguagem Γ-compatível contida em K, e então, a seqüência converge para a máxima linguagem Γ-compatível contida em K.

O operador Ω é monótono, isto é, para $A \subseteq L$, $\Omega(A) \subseteq A$, desde que $A - \Omega(A) = L_b(A)$. Portanto, a seqüência converge para um ponto fixo de Ω contida em K.

Para um arbitrário i, considerando as linguagens K_i e K_{i+1} de $\{K_i\}$, tal que $K_{i+1} = \Omega(K_i)$. Afirma-se que se K_i não é Γ-compatível, então para qualquer $A \subseteq K$, tal que $K_{i+1} \subset A \subseteq K_i$, A não é Γ-compatível.

Pela equação 4.2, tem-se que $K_i - K_{i+1} = L_b(K_i)$, portanto, qualquer $A \subseteq K$ tal que $K_{i+1} \subset A \subseteq K_i$ pode ser obtido por adição de cadeias de $L_b(K_i)$ a K_{i+1}. Mas, qualquer linguagem construída pela adição de cadeias de $L_b(K_i)$ a K_{i+1}, não é Γ-compatível, desde que as cadeias em $L_b(K_i)$, por definição, são cadeias em K_i que violam a condição de Γ-compatibilidade. Portanto, se K_i não é Γ-compatível, então para qualquer $A \subseteq K$ tal que $K_{i+1} \subset A \subseteq K_i$, A também não é Γ-compatível.

Conclui-se que a seqüência $\{K_i\}$ converge para o maior ponto fixo de Ω contido em K, isto é, a máxima linguagem Γ-compatível contida em K.

$\diamondsuit$

O algoritmo a seguir calcula a máxima linguagem Γ-compatível contida em K.

Algoritmo 4.1 *Calcula* $\sup\mathcal{CM}(K)$ *no caso regular*

Dados: $D = (G, \Gamma)$: Um SED controlado, com Γ dependente do estado.
S : Um gerador Trim tal que $L_m(S) = K \subset L(G)$.

Obter G', um gerador Trim tal que $L(G') = L_m(G') = L(G)$ (Marcar todos os estados de G)

Construir $C_i = G' \,||\, S$, a composição síncrona entre G' e S, com $i = 0$

Identificar os maus estados (q, p) de C_i e/ou as más transições da forma a seguir

laço

Primeiro, identificar os *possíveis* maus estados de C_i. São estados (q, p) de C_i tais que:

- $(q, p) \in Q_m(C_i) \wedge \nexists(\gamma, M) \in \Gamma(q) \; : \; \gamma \cap \Sigma_G(q) = \Sigma_{C_i}((q, p))$

- $(q, p) \in (Q(C_i) - Q_m(C_i)) \wedge \nexists(\gamma, N) \in \Gamma(q) \; : \; \gamma \cap \Sigma_G(q) = \Sigma_{C_i}((q, p))$

Dentre estes possíveis maus estados, separar os bons. São estados $(q, p) \in C_i$ tais que:

para todo $(q, p) \in Q_m(C_i)$ **faça**

se $\exists(\gamma, N) \in \Gamma(q) : \gamma \cap \Sigma_{G'}(q) = \Sigma_{C_i}((q, p))$ **então**

(q, p) é um bom estado e desmarcar (q, p)

senão

Buscar o máximo $(\gamma', \#) \in \Gamma(q) : \gamma' \cap \Sigma_{G'}(q) \subseteq \Sigma_{C_i}((q, p))$

caso:

- $(\gamma', M) \wedge (\gamma', N)$ existe, então (q, p) é um bom estado mantendo a marcação. As más transições serão: $bad_t(q, p) = \Sigma_{C_i}((q, p)) - (\gamma' \cap \Sigma_{G'}(q))$.

- (γ', M) existe, então (q, p) é um bom estado mantendo a marcação. As más transições serão: $bad_t(q, p) = \Sigma_{C_i}((q, p)) - (\gamma' \cap \Sigma_{G'}(q))$.

- (γ', N) existe, então (q, p) é um bom estado e desmarcar (q, p). As más transições serão: $bad_t(q, p) = \Sigma_{C_i}((q, p)) - (\gamma' \cap \Sigma_{G'}(q))$.

- Outro caso $(q, p) \in Q_m(C_i)$ é um mau estado.

fim se

fim para

para todo $(q, p) \in Q(C_i) - Q_m(C_i)$ **faça**

Buscar o máximo $(\gamma', N) \in \Gamma(q) : \gamma' \cap \Sigma_{G'}(q) \subseteq \Sigma_{C_i}((q, p))$

se (γ', N) existe **então**

(q, p) é um bom estado e as más transições serão $bad_t(q, p) = \Sigma_{C_i}((q, p)) - (\gamma' \cap \Sigma_{G'}(q))$

senão

$(q, p) \in Q_m(C_i)$ é um mau estado

fim se

fim para

Obter C_i' apagando os maus estados e as más transições de C_i e fazer $C_{i+1} = Trim(C_i')$
se $C_{i+1} = C_i$ **então** parar!, e fixar $I = i$. **senão**, fixar $i = i + 1$ **fim se**
fim laço
Saída: C_I

Proposição 4.2 *O resultado do algoritmo 4.1 é tal que $L_m(C_I)$ é a máxima linguagem Γ-compatível em relação a $L(G)$ contida em K.*

Prova:

Considerando a ação do operador Ω sobre a linguagem K, a eliminação das cadeias feito pelo operador Ω corresponde no algoritmo 4.1 à eliminação de maus estados e más transições, a desmarcar possíveis maus estados, e à operação *trim*. Portanto o algoritmo 4.1 gera a seqüência

$$\begin{aligned} L_m(C_0) &= L_m(S) \\ L_m(C_{i+1}) &= \Omega(L_m(C_i)) \end{aligned} \tag{4.3}$$

onde $L_m(S) = K = L_m(C_0)$.

O algoritmo pára quando a seqüência converge, portanto conclui-se que $L_m(C_I)$ é a máxima linguagem Γ-compatível em relação a $L(G)$ contida em K.

$\diamondsuit$

Também pode ser provado que o algoritmo 4.1 converge em um número finito de passos. Isto porque o algoritmo itera recortando estados e transições a partir de um gerador inicial C_0. A cada ciclo do algoritmo, C_{i+1} é um gerador possivelmente com menor número de estados, transições e estados marcados que C_i. O algoritmo termina quando $C_{i+1} = C_i$, no pior dos casos C_{i+1} e C_i serão vazios. O que demonstra que o algoritmo termina em um número finito de passos.

Para análise da complexidade computacional do algoritmo, considera-se que G tem n estados, S tem m estados e Σ tem e eventos. O algoritmo manipula o produto síncrono entre G e S ($C_0 = G||S$), o qual tem no máximo caso nm estados. No pior caso, a cada ciclo do algoritmo um estado de C_i é removido, e todo o espaço de estados deverá ser testado, resultando n^2m^2 passos. Também, considera-se que, para cada estado é realizada uma busca binária para o controle ótimo com uma complexidade $\mathcal{O}(\log_g(p))$ para uma lista ordenada de p elementos, sendo p limitado por 2^{e+1}, o tamanho máximo do número de padrões de controle por estado, isto é $2^{\Sigma} \times \{M, N\}$. Portanto a complexidade estimada do algoritmo 4.1 é da ordem $\mathcal{O}((e+1)m^2n^2)$.

4.4 Exemplo de aplicação.

Sejam um gato e um rato dentro de um labirinto como mostrado na Figura 4.4, uma extensão ao problema clássico apresentado em (Wonham e Ramadge 1987). Os animais podem se movimentar dentro de quatro quartos comunicados por dutos privativos representados pelas linhas que interconectam os quartos mostrados em ambos lados da Figura 4.4. Os dutos são unidirecionais, indicados pelas setas, e o acesso de um animal aos dutos pode ser inibido por portas internas, como indicado pelos traços nas setas. A inibição do acesso aos quartos pode ser interdependente, como por exemplo, quando o rato está no quarto 3, e o acesso para o quarto 1 não pode ser inibido sem que se iniba o acesso ao quarto 2. O gato encontra-se inicialmente no quarto 1 e o rato no quarto 3. Os animais enquanto se encontram no labirinto devem ser alimentados, e a comida encontra-se disponível dentro de algum dos dutos mostrados por estrelas na Figura 4.4. O acesso ao duto é concomitante ao acesso à comida.

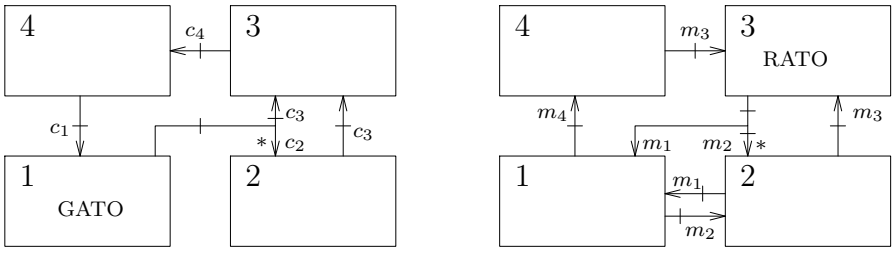


Figura 4.4: Labirinto para o Gato e o Rato

A tarefa é projetar um supervisor que permita que o gato e o rato sobrevivam dentro do labirinto, e nunca permitir que os dois se encontrem juntos dentro de um mesmo quarto, sob a hipótese que o gato pode comer o rato, e sempre permitir ambos acessarem suas comidas, senão, os animais morrerão de fome. A lei de controle deve ser ótima, no sentido de ser a menos restritiva à movimentação dos animais.

Primeiramente o comportamento do gato e do rato é modelado por geradores individuais mostrados na Figura 4.5. Os eventos c_i e m_j significam, a entrada do gato e do rato aos quartos i e j, respectivamente, e a etiqueta dos estados no gerador corresponde ao número de quarto. O comportamento do sistema completo, denotado por G, é obtido pela composição síncrona dos geradores dos animais, o resultado apresenta 16 estados e 48 transições.

Constrói-se uma estrutura de controle dependente do estado para G. Dado um estado (i, j) de G, onde a numeração do estado indica os quartos ocupados pelo gato

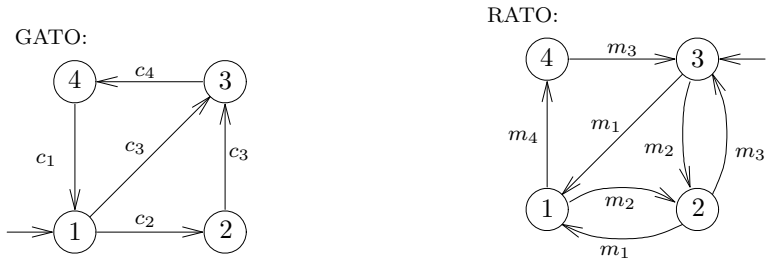


Figura 4.5: Comportamento do gato e do rato, respectivamente

e rato respectivamente, um padrão de controle $(\gamma, \#) \in \Gamma(i,j)$ é tal que: $\gamma \subseteq \Sigma_G(i,j)$ contém um possível conjunto de eventos habilitados no estado (i,j); e $\# = M$ se algum evento habilitado em γ permite acessar a comida; caso contrário $\# = N$. Por exemplo, dado o estado $(1,3)$ de G, o correspondente conjunto de padrões de controle é:

$$\begin{aligned}\Gamma((1,3)) \;=\; & \{(\emptyset,N),(\{m_1\},N),(\{m_1,m_2\},M),(\{c_2\},M),(\{c_2,m_1\},M),(\{c_2,m_1,m_2\},M),\\ & (\{c_2,c_3\},M),(\{c_2,c_3,m_1\},M),(\{c_2,c_3,m_1,m_2\},M)\}\end{aligned}$$

A especificação é construída como um gerador H apagando os estados de G onde o gato e o rato compartilham o mesmo quarto, e marca-se os estados onde é possível dar acesso à comida para qualquer um dos animais. Pelo algoritmo 4.1, determina-se a linguagem suprema Γ-*compatível* de $K = L(H)$ e o resultado é mostrado na Figura 4.6. Note que a marcação dos estados $(1,3)$ e $(4,3)$ é gerada pelos padrões de controle $(\{c_2\},M)$ e $(\{m_1,m_2,c_1\},M)$, correspondendo ao acesso à comida para o gato e o rato respectivamente.

O problema é naturalmente resolvido pelo uso do método de síntese de controle apresentado para o nível de abstração que foi proposto. Se fosse utilizada a teoria tradicional de Ramadge e Wonham para resolver este problema, o número de estados e transições do modelo original seria incrementado a fim de expressar os aspectos abstratos da estrutura de controle e marcação (5 estados e 6 transições para modelar o gato; 6 estados e 9 transições para modelar o rato).

4.5 Conclusão

Neste capítulo foi discutido o problema de controle supervisório para um novo modelo generalizado dos modelos usuais na literatura (Ramadge e Wonham 1989, Golaszewski e Ramadge 1987, Zhong e Wonham 1990, Wong e Wonham 1996, Caines e

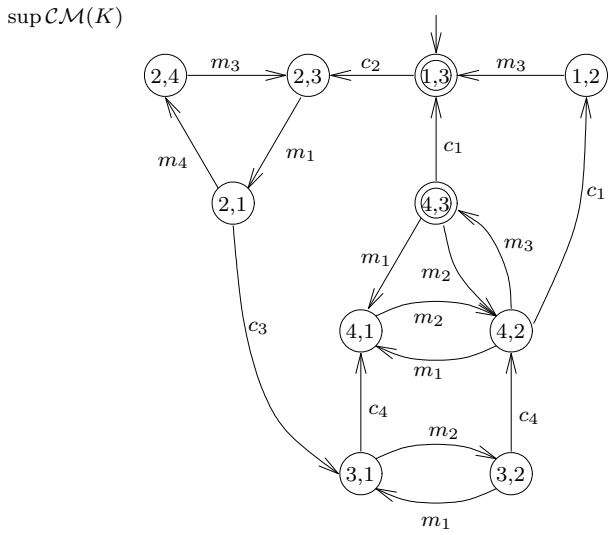


Figura 4.6: Linguagem suprema Γ-*compatível*

Hubbard 1998a, Pu 2000). Foram apresentadas as condições para existência de supervisores e um algoritmo para o cálculo da máxima linguagem Γ-*compatível*

Este trabalho foi inserido na pesquisa relacionada ao controle supervisório hierárquico por agregação de estados que será apresentado no capítulo a seguir.

Capítulo 5

Controle Supervisório Hierárquico por Agregação de Estados

Neste capítulo apresenta-se um dos resultados obtidos da pesquisa desenvolvida em relação ao problema de controle supervisório hierárquico de sistemas a eventos discretos baseado na agregação de estados e na utilização de estruturas de controle avançadas.

A complexidade que envolve os sistemas a eventos discretos por causa da explosão combinatória motiva uma abordagem hierárquica para controle supervisório. Neste capítulo formulamos uma extensão das teorias para controle supervisório hierárquico propostas por Zhong e Wonham (1990), Wong e Wonham (1996) e Caines e Hubbard (1998a), em combinação com a nova abordagem apresentada no capítulo anterior (Cury et al. 2001).

Verificou-se que a abordagem proposta por Zhong e Wonham (1990) e estendida por Wong e Wonham (1996) apresenta condições conservadoras para garantir consistência hierárquica forte, além do que o canal de informação em geral tem que ser refinado várias vezes até se conseguir consistência hierárquica forte. Por outro lado a abordagem de Caines e Hubbard (1998a) para controle hierárquico por agregação de estados também apresenta uma condição de consistência hierárquica forte (IBC-não bloqueante), mas não apresenta uma forma sistemática de obter esta condição. Além disso esta última só permite especificações expressas em termos de estados proibidos.

Neste capítulo, será introduzido um novo formalismo para o controle hierárquico de sistemas a eventos discretos usando dois modelos de estruturas de controle. Consideram-se dois níveis de hierarquia, baixo e alto nível: para o baixo nível recorre-se ao modelo tradicional de Ramadge e Wonham, no qual o sistema é definido sobre um

alfabeto particionado em eventos controláveis e não controláveis; no alto nível, será utilizado o modelo proposto por Cury et al. (2001), baseado em estruturas de controle avançadas e marcação dinâmica associada. Portanto, o que se pretende é propor um modelo para controle hierárquico que permita especificações no alto nível, dentro de um subconjunto dos eventos de baixo nível considerados como relevantes, tal que o sistema possua consistência hierárquica forte e o modelo hierárquico precise de refinamento mínimo para alcançar esta propriedade.

Em (da Cunha 2001), encontra-se um trabalho relacionado. Nesse trabalho apresenta-se o controle hierárquico como uma generalização das abordagens de Zhong e Wonham (1990), Wong e Wonham (1996), e Pu (2000), usando o novo modelo de Cury et al. (2001) para síntese de controladores no alto nível. Em particular, no trabalho de da Cunha (2001) constrói-se o modelo de alto nível através da vocalização de cadeias de baixo nível, em oposição à abordagem aqui considerada.

5.1 O problema de controle supervisório hierárquico

Dada uma planta G definida sobre um alfabeto Σ, e dado um alfabeto de eventos relevantes $\Sigma^A \subset \Sigma$ para possíveis especificações, o problema é obter um modelo agregado para o alto nível, definido sobre Σ^A, tal que: *i)* para qualquer especificação realizável no alto nível, a linguagem sintetizada com o modelo agregado seja igual à imagem da implementação por meio do baixo nível; *ii)* que toda linguagem realizável no baixo nível tenha uma imagem realizável no alto nível.

O critério para a identificação dos eventos relevantes, baseia-se na escolha dos eventos necessários para representar múltiplas especificações coerentes no alto nível.

Na estrutura hierárquica, o baixo nível consiste de uma planta G e um supervisor f definidos sobre um alfabeto Σ, e o alto nível (nível agregado) é formado por uma planta G^A e um supervisor f^A definidos sobre um alfabeto $\Sigma^A \subset \Sigma$. Estes são acoplados como mostrado na Figura 5.1.

Na Figura 5.1, G é a planta a ser controlada no mundo real por f, o operador, enquanto que G^A é uma abstração de G obtida pela agregação de seus estados que será controlada por um supervisor f^A, o gerente. A planta G informa ao nível agregado somente alguns eventos considerados relevantes ($\sigma \in \Sigma^A$), atualizando o modelo G^A, mas também informa ao operador f a ocorrência de qualquer evento $\sigma \in \Sigma$. A

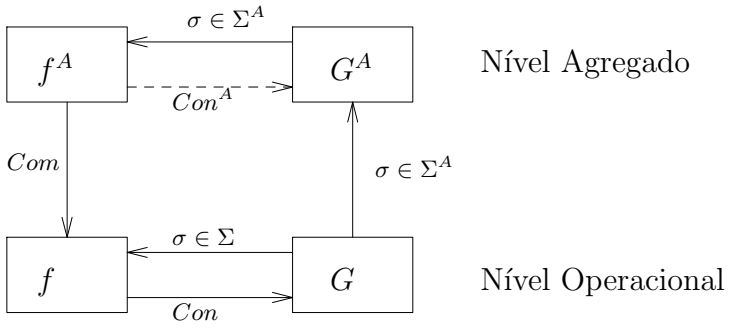


Figura 5.1: Arquitetura hierárquica.

cada ocorrência de um evento $\sigma \in \Sigma^A$, o gerente f^A envia um comando Com a ser implementado pelo operador f. O operador f recebe duas informações - o comando Com do nível agregado e a ocorrência de eventos $\sigma \in \Sigma$ gerados pela planta G - e com estes dados envia uma entrada de controle Con para a planta G. A entrada de controle Con^A, enviada pelo gerente f^A para G^A, resulta ser uma entrada de controle virtual, uma vez que o comportamento de G^A é determinado totalmente pelo comportamento de G.

5.2 Agregação de estados

No nível operacional, o autômato da planta é dado por uma quíntupla, $G = (\Sigma, X, \delta, q_0, Q_m)$, igual ao da seção 2.1.5.

Uma partição π[1] do conjunto de estados X é feita de tal forma que só eventos relevantes sejam mostrados, ou seja, dois estados pertencem a um mesmo bloco se estão ligados por alguma transição não relevante. A partir desta partição apresenta-se um modelo abstrato definido sobre um conjunto de eventos relevantes.

Definição 5.1 Autômato agregado-π

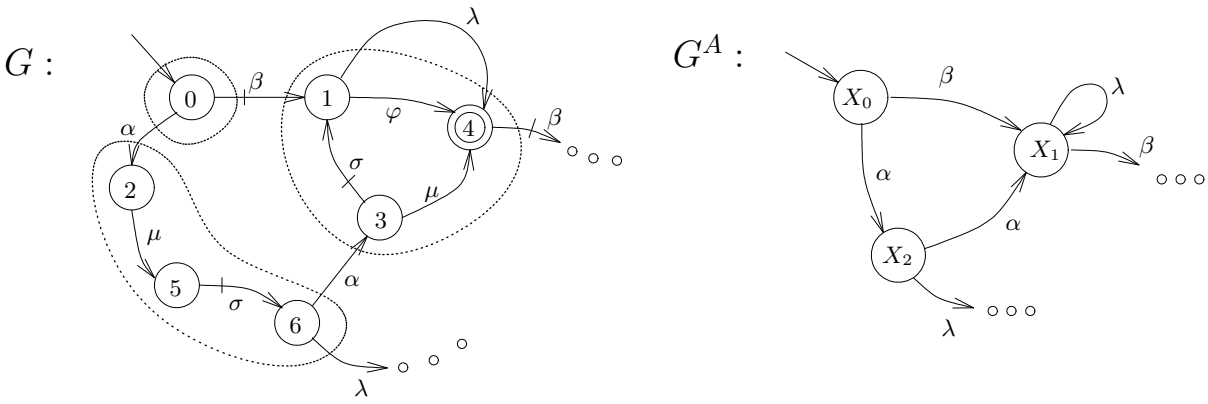
Dado um autômato G, um conjunto de eventos relevantes $\Sigma^A \subset \Sigma$, e uma partição $\pi = \{X_i : x_j, x_k \in X_i \leftrightarrow x_j, x_k \in X \wedge \exists s \in (\Sigma - \Sigma^A)^*, \delta(x_j, s) = x_k\}$, define-se o autômato agregado-π*,*

$$G^A = (\Sigma^A, \pi, \delta^A, q_0^A)$$

[1] O conceito de partição foi definida na Seção 3.2.1

onde Σ^A *é o conjunto de eventos,* π *é o conjunto finito de estados obtido após a partição de* X*,* $\delta^A : \pi \times \Sigma^A \to \pi$ *é a função de transição parcial de estados tal que* $\delta^A(X_i, \sigma) = X_j$ **se** $(\exists x \in X_i, \exists y \in X_j : \delta(x, \sigma) = y)$*, e* $q_0^A \in \pi$ *é o estado inicial, tal que* $q_0 \in q_0^A$*.*

Na Figura 5.2 apresentam-se autômatos G e G^A, mostrando como os estados originais foram agregados dentro de blocos-π.



$\Sigma = \{\alpha, \beta, \lambda, \varphi, \mu, \sigma\}$
$\Sigma_c = \{\beta, \sigma\}$
$\Sigma_u = \{\alpha, \lambda, \varphi, \mu\}$
$\Sigma^A = \{\alpha, \beta, \lambda\}$

(a)

$\Sigma^A = \{\alpha, \beta, \lambda\}$

(b)

Figura 5.2: (a)Agregação de estados e (b) autômato agregado-π.

Dado um autômato G, e um autômato agregado-π correspondente, G^A, definimos o mapeamento canônico Θ para vários objetos. Comete-se este abuso de notação a fim de simplificar o número de variáveis e sabendo que isso não dará lugar a confusão.

(cadeias de eventos) $\Theta : L(G) \longrightarrow L(G^A)$, para $s \in \Sigma^*$ e $\sigma \in \Sigma$,

$$\Theta(\epsilon) = \epsilon \text{ e}$$

$$\Theta(s\sigma) = \begin{cases} \Theta(s)\sigma & \textbf{se } \sigma \in \Sigma^A \\ \Theta(s) & \textbf{caso contrário} \end{cases}$$

Os exemplos a serem apresentados dentro do escopo de mapeamento refere-se à agregação mostrada na Figura 5.2.

Seja $\alpha\mu\sigma\alpha\mu$ *uma cadeia de* $L(G)$*, temos que,* $\Theta(\alpha\mu\sigma\alpha\mu) = \alpha\alpha$*.*

Esta operação, consiste em apagar da cadeia de $L(G)$ os eventos não relevantes.

(linguagens) $\Theta : 2^{L(G)} \longrightarrow 2^{L(G^A)}$,

$$\Theta(K) = \{\Theta(s) \ : \ s \in K\}$$

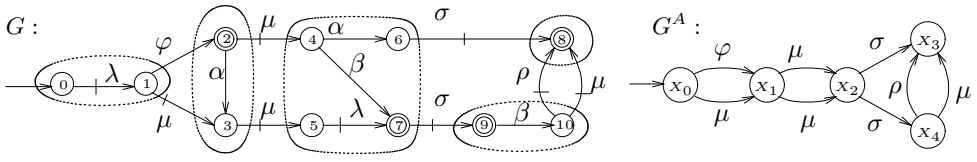
Seja $K = \{\varepsilon, \alpha, \alpha\mu, \alpha\mu\sigma, \alpha\mu\sigma\alpha, \alpha\mu\sigma\alpha\mu, \beta, \beta\varphi, \beta\lambda\}$, *temos que,* $\Theta(K) = \{\varepsilon, \alpha, \alpha\alpha, \beta, \beta\lambda\}$.

Esta operação agrupa o mapeamento de todas as cadeias da linguagem K.

O mapeamento para *estados* e *conjunto de estados* será considerado igual ao definido na Seção 3.2.1 (pág. 29).

Seja L^A o mapeamento para o nível agregado de uma linguagem $L(G)$, i.e., $L^A = \Theta(L(G))$, e seja $L(G^A)$ a linguagem gerada pelo autômato agregado-π. Note-se que L^A é diferente de $L(G^A)$ uma vez que algumas cadeias de $L(G^A)$ são impossíveis de serem obtidas através do mapeamento das cadeias de baixo nível. Por exemplo, na Figura 5.2, a cadeia $\beta\lambda\lambda \in L(G^A)$ nunca poderá ser representada pelo mapeamento de alguma cadeia de baixo nível, i.e. $\nexists s \in L(G) \ : \ \Theta(s) = \beta\lambda\lambda$. Pode-se mostrar que $L^A \subseteq L(G^A)$.

Nem sempre uma agregação assim obtida terá as boas propriedades que permitam a síntese de um supervisor hierárquico, como apresentado através do exemplo a seguir. Considerando a planta G da Figura 5.3(a), e o alfabeto de eventos relevantes para especificação $\Sigma^A = \{\varphi, \mu, \sigma, \rho\}$, obtém-se o modelo agregado da Figura 5.3(b).



(a) $\Sigma = \{\alpha, \beta, \lambda, \varphi, \mu, \sigma, \rho\}$, $\Sigma^A = \{\varphi, \mu, \sigma, \rho\}$ (b)

Figura 5.3: Problemas dentro de uma agregação.

Esta agregação apresenta alguns problemas, tais como: bloqueio, falta de consistência hierárquica, e não determinismo.

1. Não Determinismo:

   O fato de agregar estados poderá originar naturalmente um não determinismo no nível agregado.

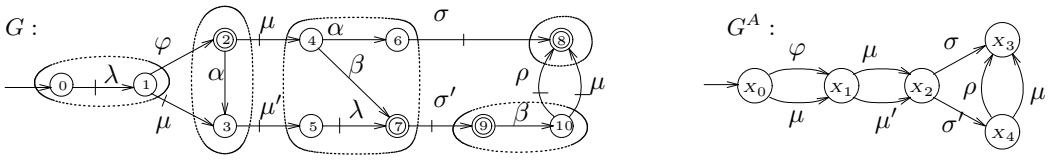
   Seja o autômato agregado $G^A = (\Sigma^A, \pi, \delta^A, q_0^A)$, a seguir consideram-se as duas situações possíveis de ocorrer:

   (a) G^A é propriamente não determinístico, isto é, existe $X_i, X_j, X_k \in \pi$ e $\sigma \in \Sigma^A$, tal que, $\delta^A(X_i, \sigma) = \{X_j, X_k\}$, com $X_j \neq X_k$. Na Figura 5.3a, as saídas do bloco X_2 com o evento σ ilustra um exemplo deste caso.

   (b) Existe $X_i, X_j \in \pi$ e $\sigma \in \Sigma^A$ tal que $\delta(X_i, \sigma) = X_j$, e tal que, existe $x, y \in X_i$ e $z, w \in X_j$, $z \neq w$, com $\delta(x, \sigma) = z$ e $\delta(y, \sigma) = w$. Na Figura 5.3a, as saídas do bloco X_1 com o evento μ, ilustra um exemplo deste caso.

   O caso (b) considera-se também como não determinismo porque as continuações após aquelas transições afetadas não necessariamente serão iguais.

   A princípio o não determinismo será eliminado renomeando os eventos $\sigma \in \Sigma$, que geram não determinismo, em eventos $\sigma, \sigma', \sigma'', \ldots$ que irão aumentar os alfabetos Σ e Σ^A. Por exemplo, o não determinismo da agregação ilustrada na Figura 5.3, eliminou-se renomeando o evento μ que sai do bloco X_1 em μ, μ', e o evento σ que sai do bloco X_2 em σ, σ'. Na Figura 5.4 apresenta-se o resultado desta renomeação.



(a) $\Sigma = \{\alpha, \beta, \lambda, \varphi, \mu, \mu', \sigma, \sigma', \rho\}$, $\Sigma^A = \{\varphi, \mu, \mu', \sigma, \sigma', \rho\}$ (b)

Figura 5.4: Solução ao problema de não determinismo.

No capítulo 6 apresentam-se métodos para diminuir a renomeação.

2. Bloqueio:

   Uma ação de controle no alto nível, pode levar o baixo nível ao bloqueio. Por exemplo, quando no autômato G^A da Figura 5.3(b) desabilitamos as duas saídas σ no bloco X_2 (isto é possível uma vez que σ é controlável), no baixo nível o sistema poderá ficar bloqueado no estado 6, não podendo completar nenhuma

tarefa. O mesmo problema acontece com o bloco X_4, quando os eventos ρ e μ são desabilitados.

3. Ausência de consistência hierárquica:

   Da mesma forma que na teoria de controle hierárquico de Zhong e Wonham (1990), aparece aqui o problema de inconsistência hierárquica. Na Figura 5.3, os eventos de saída do bloco X_0 (φ e μ) poderiam ser considerados controláveis no alto nível, uma vez que a ocorrência destes pode ser controlada indiretamente através do evento interno λ. Entretanto, ao desabilitar o evento φ no alto nível, o evento μ será também desabilitado, originando assim o problema de inconsistência hierárquica.

5.3 Obtenção do modelo agregado para controle supervisório

Nesta seção o modelo agregado será completado com a introdução de uma estrutura de controle com as características do modelo apresentado no capítulo 4, ou seja, este modelo apresentará estruturas de controle avançados com atributos de marcação associados. Serão inicialmente apresentadas algumas definições associadas aos blocos de uma dada partição.

Definição 5.2 $I(X_i)$**, Conjunto de estados de entrada**

O conjunto de estados de entrada $I(X_i)$ de um elemento X_i de uma partição π, é o conjunto de estados em X_i que, ou é o estado inicial ou são estados diretamente acessíveis com algum evento relevante, isto é:

$$x \in I(X_i) \;\longleftrightarrow\; (x \in X_i) \wedge ([x = q_0] \;\vee\; [\exists x' \in X \;,\; \exists \sigma \in \Sigma^A \;,\; \delta(x', \sigma) = x])$$

Cada bloco da partição resulta ser uma parte do sistema que pode ser representado por um outro gerador.

Definição 5.3 H_i**, Gerador de um bloco** X_i

Para cada bloco X_i*, elemento de uma partição* π*, é definido um gerador,*

$$H_i = ((\Sigma - \Sigma^A), X_i, I(X_i), \delta_i, Q_{im})$$

onde $(\Sigma - \Sigma^A)$ *é o alfabeto,* X_i *é o conjunto de estados,* $I(X_i)$ *é o conjunto de estados iniciais,* $\delta_i : X_i \times (\Sigma - \Sigma^A) \longrightarrow X_i$ *é a função de transição de estados tal que* $(\delta_i(x, \sigma) = \delta(x, \sigma)$ **se** $x \in X_i$*,* $\sigma \in (\Sigma - \Sigma^A))$*, e* Q_{im} *é o conjunto de estados marcados tal que* $Q_{im} = X_i \cap Q_m$*.*

Por exemplo, na Figura 5.5(a) mostra-se o gerador H_1 referente ao bloco X_1 da Figura 5.2.

Dado um gerador H_i, para cada estado inicial $x_j \in I(X_i)$ pode ser definido um subsistema da forma a seguir.

Definição 5.4 H_{ij}**, Subsistema de** H_i **para um estado de entrada** $x_j \in I(X_i)$

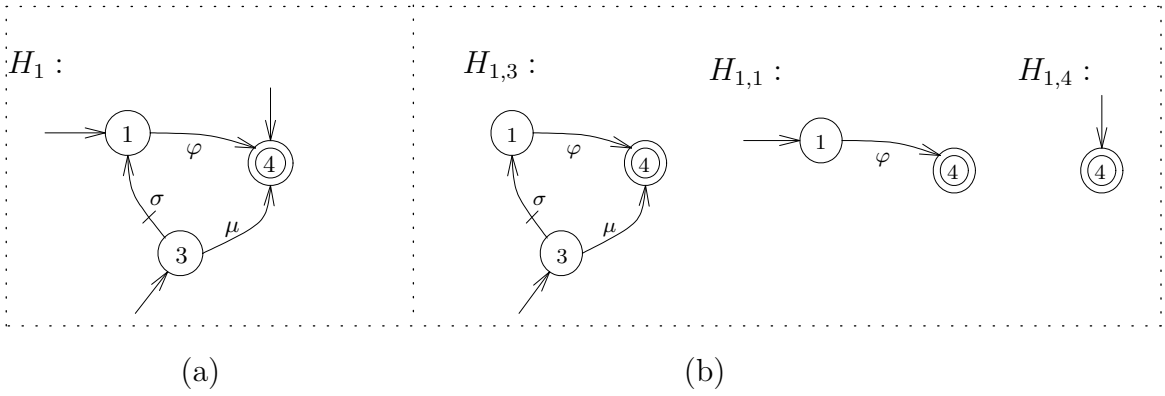
Para cada entrada $x_j \in I(X_i)$ *de* H_i *define-se um gerador,*

$$H_{ij} = ((\Sigma - \Sigma^A), X_{ij}, x_j, \delta_{ij}, Q_{ijm})$$

onde, $(\Sigma - \Sigma^A)$ *é o alfabeto,* x_j *é o estado inicial tal que* $x_j \in I(X_i)$*,* X_{ij} *é o conjunto de estados tal que* $X_{ij} = \{x \in X_i : x = \delta_i(x_j, u), u \in (\Sigma - \Sigma^A)^*\}$*,* δ_{ij} *é a função de transição de estados tal que* $(\delta_{ij}(x, \sigma) = \delta_i(x, \sigma)$ **se** $x \in X_{ij}$*,* $\sigma \in (\Sigma - \Sigma^A))$*, e* Q_{ijm} *é o conjunto de estados marcados tal que* $Q_{ijm} = Q_{im} \cap X_{ij}$*.*

Na Figura 5.5(b) mostram-se todos os subsistemas do gerador H_1 mostrado na Figura 5.5(a). A notação de H_{ij}, na hora da instanciação, será denotada por exemplo $H_{1,3}$, com $i = 1$ e $j = 3$, separados por vírgula a fim de evitar confusão na numeração.

Dado um bloco X_i e considerando os eventos de saída em Σ^A do mesmo, isto é, $\Sigma_{saida}(X_i) = \{\sigma \in \Sigma^A : \exists x \in X_i,\ \delta(x, \sigma)$ definido $\}$, define-se um gerador aumentado como segue.



(a) (b)

Figura 5.5: Subsistemas de um gerador H_i.

Definição 5.5 H_i^+**, Gerador aumentado**

Dado um gerador H_i*, define-se um gerador aumentado*

$$H_i^+ = (\Sigma, X_i^+, I(X_i), \delta_i^+, Q_{im}^+)$$

onde, Σ *é o alfabeto,* X_i^+ *é o conjunto de estados tal que* $X_i^+ = X_i \cup \{x^+\}$*,* $I(X_i)$ *é o conjunto de estados inicias,* δ_i^+ *é a função de transição de estados tal que* $\Big(\delta_i^+(x,\sigma) = \delta(x,\sigma)$ **se** $x \in X_i$*,* $\sigma \in (\Sigma - \Sigma^A)\Big)$ **e** $\Big(\delta_i^+(x,\sigma) = x^+$ **se** $x \in X_i$*,* $\sigma \in \Sigma^A$*,* $\delta(x,\sigma)$ *definido* $\Big)$*, e* Q_{im}^+ *é o conjunto de estados marcados tal que* $Q_{im}^+ = Q_{im} \cup \{x^+\}$*.*

O gerador aumentado de um bloco X_i, é representado pelo gerador H_i aumentado de um estado marcado x^+ para receber os eventos relevantes de saída do bloco em questão.

Na Figura 5.6, mostra-se o gerador aumentado do bloco X_1 descrito na Figura 5.2.

Analogamente, para cada estado de entrada $x_j \in I(X_i)$ define-se os subsistemas para H_i^+.

Definição 5.6 H_{ij}^+**, Subsistema de** H_i^+ **para um estado de entrada** $x_j \in I(X_i)$

Para cada estado de entrada $x_j \in I(X_i)$ *de* H_i^+ *define-se um autômato,*

$$H_{ij}^+ = (\Sigma, X_{ij}^+, \delta_{ij}^+, x_j, Q_{ijm}^+)$$

onde, Σ *é o alfabeto,* x_j *é o estado inicial tal que* $x_j \in I(X_i)$*,* X_{ij}^+ *é o conjunto de estados tal que* $X_{ij}^+ = \{x \in X_i^+ : x = \delta_i^+(x_j, u), u \in \Sigma^*\}$*,* δ_{ij}^+ *é a função de transição parcial tal*

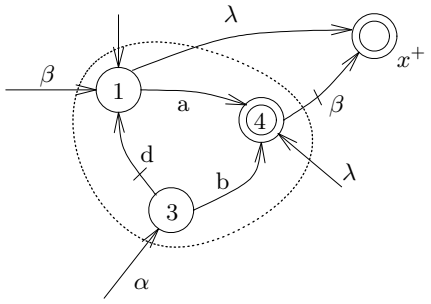


Figura 5.6: Gerador aumentado H_i^+.

que $\left(\delta_{ij}^+(x,\sigma) = \delta(x,\sigma)\ \textbf{se}\ x \in X_{ij},\ \sigma \in (\Sigma - \Sigma^A)\right)$ **e** $\left(\delta_{ij}^+(x,\sigma) = x^+\ \textbf{se}\ x \in X_{ij},\ \sigma \in \Sigma^A,\ \delta(x,\sigma)\ \textit{definido}\right)$ *e* Q_{ijm}^+ *é o conjunto de estados marcados tal que* $Q_{ijm}^+ = Q_{im}^+ \cap X_{ij}^+$.

Na continuação define-se o conjunto de autômatos cuja linguagem reconhecida L_r é $L(H_{ij}) \subseteq L_r \subseteq L(H_{ij}^+)$.

Definição 5.7 $\mathcal{S}_{ij}$**, Conjunto de sub-autômatos entre** H_{ij} **e** H_{ij}^+

Seja H_{ij} *um sub-autômato do bloco* X_i *para a entrada* $x_j \in I(X_i)$*, define-se o conjunto de sub-autômatos de* H_{ij}^+*, maiores ou iguais a* H_{ij}*,*

$$\mathcal{S}_{ij} = \{H_s\ :\ (L(H_{ij}) \subseteq L(H_s) \subseteq L(H_{ij}^+)) \wedge (L_m(H_s) = L_m(H_{ij}^+) \cap L(H_s))\}$$

O número de elementos de S_{ij} será o número de combinações das transições de saída com eventos relevantes em H_{ij}, isto é, $2^{|\Sigma_{saida(H_{ij})}|}$, onde $\Sigma_{saida(H_{ij})} = \{\sigma \in \Sigma^A\ :\ \exists x \in X_{ij},\ \delta(x,\sigma)\ \text{definido}\ \}$.

Para ilustrar esta definição, consideram-se os autômatos H_{ij} e H_{ij}^+ mostrados na Figura 5.7. Note que o conjunto de eventos relevantes é $\{a, b, c\}$.

Na Figura 5.8 mostram-se todos os elementos de $\mathcal{S}_{ij} = \{H_{s1}, H_{s2}, \ldots, H_{s8}\}$, considerados entre H_{ij} e H_{ij}^+ da Figura 5.7.

O conjunto de subsistemas entre H_{ij} e H_{ij}^+ é a combinação de todas as possíveis saídas com eventos relevantes de H_{ij} que alcançam o estado aumentado $x^+ \in X_{ij}^+$. No exemplo têm-se 3 saídas $\{a, b, c\}$, portanto o número de sub-autômatos será 2^3.

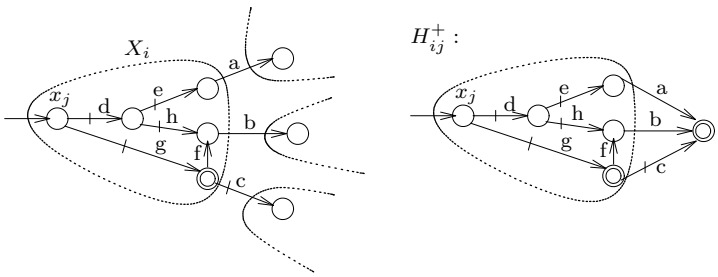


Figura 5.7: Parte de uma agregação para análise.

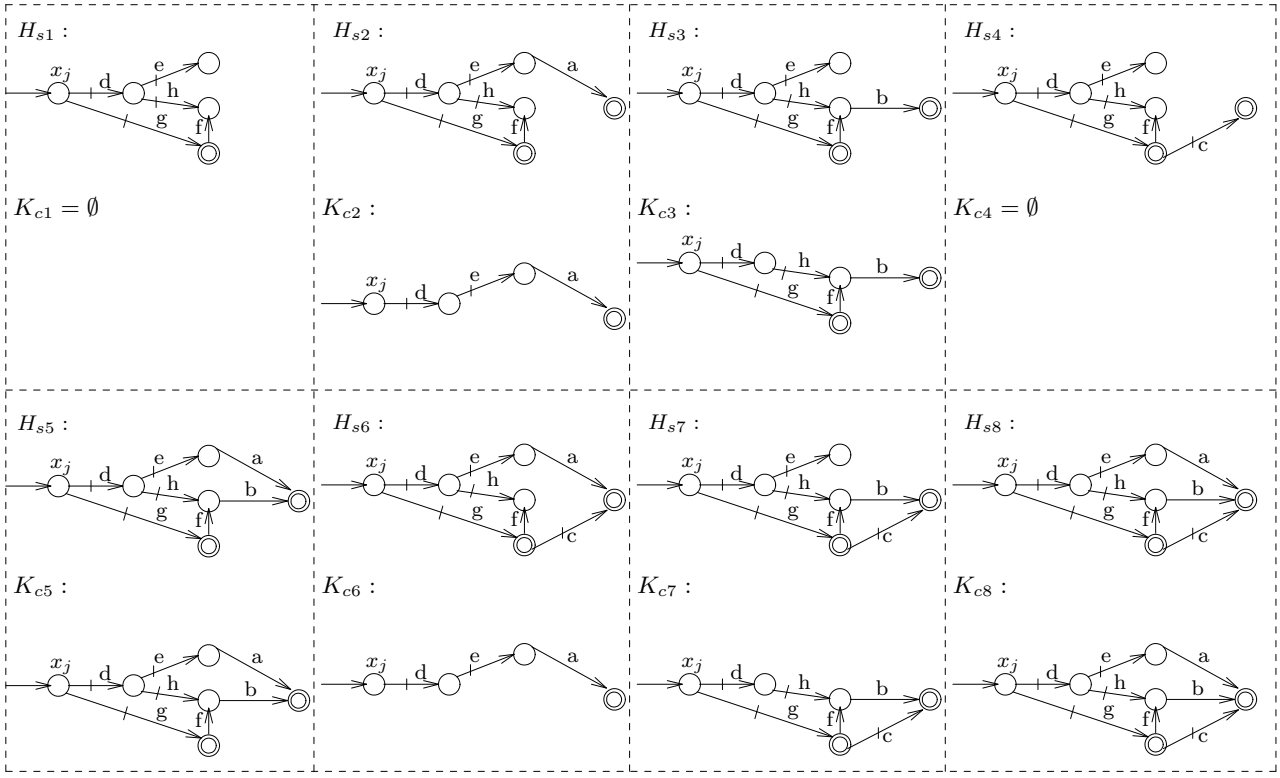


Figura 5.8: H_{si}, $i = 1, \ldots, 8$ - Conjunto de subsistemas entre H_{ij} e H_{ij}^+ ; K_{ci}, $i = 1, \ldots, 8$ - Respectivas máximas linguagens controláveis.

Definição 5.8 $\mathcal{C}_{ij}$, Conjunto de linguagens supremo controláveis dos elementos de $\mathcal{S}_{ij}$

A partir de cada elemento de $\mathcal{S}_{ij}$, constrói-se um conjunto de linguagens supremo controláveis em relação a $L(H_{ij}^{+})$,

$$\mathcal{C}_{ij} = \{K_c \subset \Sigma^* : K_c = \sup\mathcal{C}(L_m(H_s), L(H_{ij}^{+})) \text{ se } \sup\mathcal{C}(L_m(H_s), L(H_{ij}^{+})) \neq \emptyset \text{ e } H_s \in \mathcal{S}_{ij}\}$$

Ainda continuando com a Figura 5.8 também mostra-se na parte inferior de cada célula os elementos de $\mathcal{C}_{ij}$, sendo que neste exemplo $\mathcal{C}_{ij} = \{K_{c2}, K_{c3}, K_{c5}, K_{c6}, K_{c7}, K_{c8}\}$.

Definição 5.9 $\mathcal{D}_{ij}$, Sub-linguagens de $\mathcal{C}_{ij}$ sem passar por $L_m(H_{ij})$

Constrói-se $\mathcal{D}_{ij}$, retirando-se as cadeias que passam por $L_m(H_{ij})$ de cada elemento de $\mathcal{C}_{ij}$. Em outras palavras, retiram-se os estados marcados dos autômatos que representam os elementos de $\mathcal{C}_{ij}$ diferentes do estado aumentado.

$$\mathcal{D}_{ij} = \{K_d \subset \Sigma^* \ : \ K_d = K_c - \sup\mathcal{F}[(K_c - L_m(H_{ij})),\ L(H_{ij}^{+})], K_c \in \mathcal{C}_{ij}\}$$

Nesta equação $\sup\mathcal{F}$ representa a máxima linguagem L_m-fechada em relação a $L(H_{ij}^{+})$ (Ziller 1993).

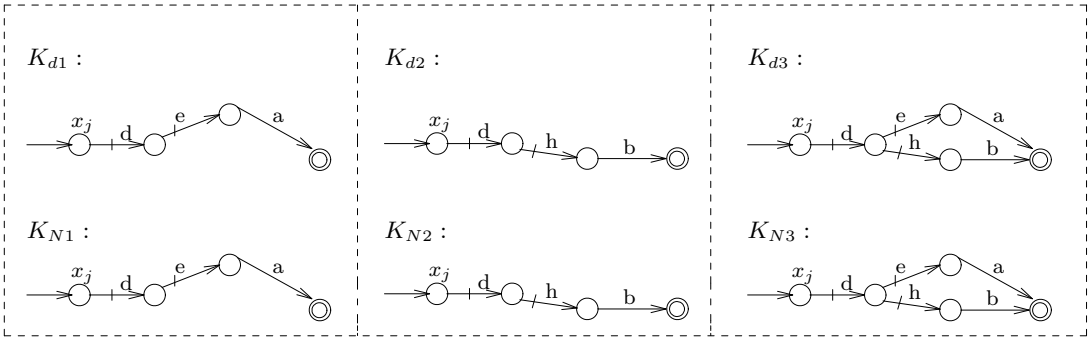


Figura 5.9: K_{di}, $i = 1, 2, 3$ - Conjunto de subsistemas de $\mathcal{C}_{ij}$ sem passar por $L_m(H_{ij})$; K_{Ni}, $i = 1, 2, 3$, Respectivas máximas linguagens controláveis.

Na Figura 5.9 mostram-se os elementos de $\mathcal{D}_{ij} = \{K_{d1}, K_{d2}, K_{d3}\}$ obtidos a partir de $\mathcal{C}_{ij}$ da Figura 5.8, lembrando que $\mathcal{C}_{ij} = \{K_{c2}, K_{c3}, K_{c5}, K_{c6}, K_{c7}, K_{c8}\}$.

Definição 5.10 $\mathcal{C}_{Nij}$**, Conjunto de sub-linguagens supremo controlável dos elementos de** $\mathcal{D}_{ij}$

A partir de cada elemento de $\mathcal{D}_{ij}$*, constrói-se um conjunto de linguagens supremo controláveis com relação a* $L(H_{ij}^{+})$*,*

$$\mathcal{C}_{Nij} = \{K_N \subset \Sigma^* : K_N = \sup \mathcal{C}(K_d, L(H_{ij}^{+})),\ se\ \sup \mathcal{C}(K_d, L(H_{ij}^{+})) \neq \emptyset\ e\ K_d \in \mathcal{D}_{ij}\}$$

Na parte inferior das células da Figura 5.9 mostram-se os elementos de $\mathcal{C}_{Nij}$, sendo $\mathcal{C}_{Nij} = \{K_{N1}, K_{N2}, K_{N3}\}$. Neste caso coincidentemente $\mathcal{D}_{ij}$ e $\mathcal{C}_{Nij}$ resultaram iguais.

Definição 5.11 E_{ij}**, Conjunto total de elementos supremo controlável**

$$E_{ij} = \mathcal{C}_{ij} \cup \mathcal{C}_{Nij}$$

Cada elemento $K \in E_{ij}$ no baixo nível corresponde a um comportamento menos restritivo dentro do bloco X_i que pode ser controlado a partir de $x_j \in I(X_i)$, para gerar $\Theta(K) - \{\epsilon\}$ como o próximo conjunto de eventos admissíveis do alto nível. Nenhum comportamento em $\mathcal{C}_{Nij}$ passa por estados marcados de X_i, enquanto que comportamentos em $\mathcal{C}_{ij}$ que não estão em $\mathcal{C}_{Nij}$ necessariamente passam pelo menos por um estado marcado de X_i.

Dado E_{ij} atribui-se ao nível agregado um conjunto de pares $\Gamma_{ij} \in 2^{\Sigma A} \times \{M, N\}$ como definido a seguir.

Definição 5.12 Γ_{ij}**, Conjunto de pares (mapeamento dos elementos de** E_{ij} **; marcação associada ao mapeamento)**

Constrói-se um conjunto de pares onde o primeiro elemento representa o mapeamento para o nível π *de um elemento de* E_{ij} *sem considerar a palavra* ϵ*, e o segundo elemento representa uma marcação associada ao primeiro elemento, i.e.*

$$\Gamma_{ij} = \{(\gamma, \#) : \gamma = [\Theta(K) - \epsilon],\ K \in E_{ij},\ e\ \# = M\ se\ (\epsilon \in \Theta(K)),\ senão\ \# = N\}$$

Para o exemplo da Figura 5.8 e Figura 5.9, lembrando que os eventos relevantes são $\{a, b, c\}$, tem-se
$\Gamma_{ij} = \{(\{a\}, N), (\{b\}, N), (\{b\}, M), (\{a, b\}, N), (\{a, b\}, M), (\{b, c\}, M), (\{a, b, c\}, M)\}$.

Observe que pela forma em que são obtidos os elementos de Γ_{ij}, nunca existirá um elemento $(\emptyset, N)$ de Γ_{ij}, uma vez que esse elemento representa uma condição de bloqueio no baixo nível.

Teorema 5.1 Γ_{ij} representa uma estrutura de controle como definido no Cap. 4 (Cury et al. 2001), isto é:

1. $(\gamma_1, N), (\gamma_2, N) \in \Gamma_{ij} \longrightarrow (\gamma_1 \cup \gamma_2, N) \in \Gamma_{ij}$
2. $(\gamma_1, M), (\gamma_2, \#) \in \Gamma_{ij} \longrightarrow (\gamma_1 \cup \gamma_2, M) \in \Gamma_{ij},\ \# = M, N$

Prova:

Primeiro demonstra-se o fechamento para união dos primeiros elementos de Γ_{ij}.

Seja:

$K_1 \in E_{ij} : (\Theta(K_1) - \epsilon) = \gamma_1 \ \longrightarrow \ (\gamma_1, \#) \in \Gamma_{ij}$

$K_2 \in E_{ij} : (\Theta(K_2) - \epsilon) = \gamma_2 \ \longrightarrow \ (\gamma_2, \#) \in \Gamma_{ij}$

Sabe-se que:

K_1 e K_2, são controláveis com relação a $L(H_{ij}^+)$.

então $(K_1 \cup K_2)$ é controlável com relação a $L(H_{ij}^+)$ e $(\Theta(K_1 \cup K_2) - \epsilon) = \gamma_1 \cup \gamma_2$.

Por construção de $\mathcal{S}_{ij}$, $\exists H_s \in \mathcal{S}_{ij} : (\Theta(L(H_s)) - \epsilon) = \gamma_1 \cup \gamma_2$

Seja $K_t = \sup \mathcal{C}(L_m(H_s), L(H_{ij}^+))$,

então $K_t \supseteq K_1 \cup K_2$,

$(\Theta(K_t) - \epsilon) = \gamma_1 \cup \gamma_2$ e $K_t \in \mathcal{C}_{ij} \longrightarrow K_t \in E_{ij}$

$\boxed{(\gamma_1 \cup \gamma_2, \#) \in \Gamma_{ij}}$ o que demonstra o fechamento para união dos primeiros elementos de Γ_{ij}

Em segundo lugar demonstra-se a consistência de marcação.

Seja:

$K_1 \in \mathcal{C}_{ij} \ : \ (\Theta(K_1) - \epsilon) = \gamma_1 \wedge \epsilon \notin \Theta(K_1) \ \longrightarrow \ (\gamma_1, N) \in \Gamma_{ij}$

$K_2 \in \mathcal{C}_{ij} \ : \ (\Theta(K_2) - \epsilon) = \gamma_2 \wedge \epsilon \notin \Theta(K_2) \ \longrightarrow \ (\gamma_2, N) \in \Gamma_{ij}$

$K_3 \in \mathcal{C}_{ij} \ : \ (\Theta(K_3) - \epsilon) = \gamma_3 \wedge \epsilon \in \Theta(K_3) \ \longrightarrow \ (\gamma_3, M) \in \Gamma_{ij}$

$K_4 \in \mathcal{C}_{ij} \ : \ (\Theta(K_4) - \epsilon) = \gamma_4 \wedge \epsilon \in \Theta(K_4) \ \longrightarrow \ (\gamma_4, M) \in \Gamma_{ij}$

Pelo passo anterior se sabe que $(\gamma_1 \cup \gamma_2, \#), (\gamma_2 \cup \gamma_3, \#), (\gamma_3 \cup \gamma_4, \#) \in \Gamma_{ij}$

$\epsilon \notin \Theta(K_1 \cup K_2) \ \longrightarrow \ (\gamma_1 \cup \gamma_2, N) \in \Gamma_{ij}$

$\epsilon \in \Theta(K_2 \cup K_3) \ \longrightarrow \ (\gamma_2 \cup \gamma_3, M) \in \Gamma_{ij}$

$\epsilon \in \Theta(K_3 \cup K_4) \ \longrightarrow \ (\gamma_3 \cup \gamma_4, M) \in \Gamma_{ij}$ O que demonstra o Teorema.

$\diamondsuit$

As estruturas de controle Γ_{ij} discutidas no Teorema 5.1 são dependentes do bloco X_i e do estado de entrada $x_j \in I(X_i)$. Uma vez que a informação do estado de entrada não poderá ser vista no autômato do nível agregado, propõem-se duas formas de representação de um autômato agregado-π com estruturas de controle, uma com estruturas de controle dependentes do estado e a outra com estruturas de controle dependentes do estado, evento, estado anterior, sendo esta última denotada por "(s, e,ps)-dependente".

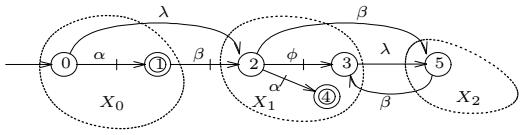
5.3.1 Autômato agregado-π com estruturas de controle dependentes do estado (G_s^A)

Para poder representar o autômato agregado-π com estruturas de controle dependentes do estado será necessário refiná-lo, fazendo uma divisão de cada bloco para cada estado de entrada que tenha um conjunto de padrões de controle diferente, garantindo desta forma uma única estrutura para cada bloco.

Como exemplo, na Figura 5.10, apresenta-se um autômato G com uma agregação dada e logo abaixo, mostra-se o autômato G_s^A resultante do refinamento da agregação original tal que as estruturas de controle sejam dependentes do estado , isto é, a estrutura de controle para o bloco X_1 entrando pelo estado 2 e pelo estado 3 são diferentes ($\Gamma_{1,2} \neq \Gamma_{1,3}$) conseqüentemente o bloco X_1 é refinado em X_1' e X_1''.

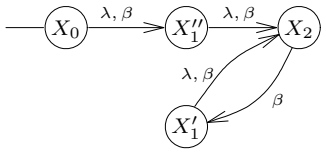
O número de estados do autômato resultante G_s^A, no pior caso, será igual ao somatório de todos os estados de entrada de todos os blocos. Um caso crítico acontece quando todos os estados do baixo nível são estados de entrada, e as estruturas de controle para cada entrada são diferentes, caso em que o autômato agregado e refinado terá o mesmo número de estados que o autômato do nível operacional.

G :



$$\Gamma_{0,0} = \{(\{\lambda\}, N), (\{\lambda\}, M), (\{\lambda, \beta\}, M)\}$$
$$\Gamma_{1,2} = \{(\{\beta\}, N), (\{\beta\}, M), (\{\lambda, \beta\}, N), (\{\lambda, \beta\}, M)\}$$
$$\Gamma_{1,3} = \{(\{\lambda\}, N)\}$$
$$\Gamma_{2,5} = \{(\{\beta\}, N)\}$$

G_s^A :



$$\Gamma_{X_0} = \Gamma_{0,0}$$
$$\Gamma_{X_1''} = \Gamma_{1,2}$$
$$\Gamma_{X_1'} = \Gamma_{1,3}$$
$$\Gamma_{X_2} = \Gamma_{2,5}$$

Figura 5.10: Autômato G_s^A, com estruturas de controle dependentes do estado.

Caso um autômato agregado-π possua estruturas de controle dependentes do estado, adapta-se a Definição 4.1 de Γ-compatibilidade como segue.

Definição 5.13 Γ-compatibilidade (dependente do estado)

Dada uma planta G_s^A com estruturas de controle dependentes do estado, e uma linguagem K tal que $K \subseteq L(G_s^A)$. Seja $G_K = \{\Sigma^A, Q_K, \delta_K, q_0, Q_{Km}\}$ o gerador que reconhece K. Tem-se $\forall s \in \overline{K}, (\exists q \in Q_K$ tal que $\delta(q_0, s) = q)$ então $(\exists X_i \in \pi$ tal que $\delta^A(X_0, s) = X_i)$.

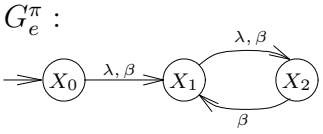
A linguagem K é Γ-compatível em relação a $L(G_s^A)$ se, e somente se, $K = \emptyset$ ou;

1. $(\forall q \in Q_{Km})(\exists(\gamma, M) \in \Gamma(X_i)) \ : \ \gamma \cap \Sigma_{G_s^A}(X_i) = \Sigma_{G_K}(q)$
2. $(\forall q \in (Q_K - Q_{Km}))(\exists(\gamma, N) \in \Gamma(X_i)) \ : \ \gamma \cap \Sigma_{G_s^A}(X_i) = \Sigma_{G_K}(q)$

5.3.2 Autômato π com estruturas de controle (s, e, ps)-dependente (G_e^A)

Adota-se esta forma de representação a fim de manter a estrutura de transição inicial do autômato agregado-π. Para poder representar um autômato agregado-π com estruturas de controle (s, e, ps)-dependente do tipo $\Gamma_{(X_i,\sigma,X_b)}$, atribui-se uma estrutura de controle Γ_{ij} para o bloco X_i com o estado de entrada $x_j \in I(X_i)$, tal que $\delta^A(X_b, \sigma) = X_i$ e $\exists x_b \in X_b$ tal que $\delta(x_b, \sigma) = x_j$. Observe que para algum $x_j \in I(X_i)$ podem existir eventos relevantes de entrada vindos de blocos diferentes, isto é, $(\exists x \in X_l, y \in X_k \ldots)$, com $l \neq k$ e $\exists \sigma_1, \sigma_2, \ldots \in \Sigma^A$ tal que $\delta(x, \sigma_1) = \delta(y, \sigma_2) = x_j$, a estrutura para essas transições será igual, ou seja, $\Gamma_{(X_i,\sigma_1,X_l)} = \Gamma_{(X_i,\sigma_2,X_k)} = \Gamma_{ij}$.

Continuando com a agregação da Figura 5.10, monta-se um autômato G_e^A com estruturas de controle (s, e, s)-dependente tal como mostrado na Figura 5.11.



$$\Gamma_{(X_0,\epsilon,X_0)} = \Gamma_{0,0}$$
$$\Gamma_{(X_1,\lambda,X_0)} = \Gamma_{1,2}$$
$$\Gamma_{(X_1,\beta,X_0)} = \Gamma_{1,2}$$
$$\Gamma_{(X_1,\beta,X_2)} = \Gamma_{1,3}$$
$$\Gamma_{(X_2,\beta,X_1)} = \Gamma_{2,5}$$
$$\Gamma_{(X_2,\lambda,X_1)} = \Gamma_{2,5}$$

Figura 5.11: Autômato G_e^A, com estruturas de controle (s, e, ps)-dependente.

Caso um autômato possua estruturas de controle (s, e, ps)-dependente, adapta-se a Definição 4.1 de Γ-compatibilidade como segue.

Definição 5.14 Γ-compatibilidade (s, e, ps)-dependente

Dada uma planta G_e^A com estruturas de controle (s, e, ps)-dependente, e uma linguagem K tal que $K \subseteq L(G_e^A)$. Seja $G_K = \{\Sigma^A, Q_K, \delta_K, q_0, Q_m\}$ o gerador que reconhece K. Tem-se $\forall s\sigma \in \overline{K}$, $s \in \Sigma^{A*}, \sigma \in \Sigma^A$, $(\exists q \in Q_K$ tal que $\delta(q_0, s\sigma) = q)$ então $(\exists X_i \in \pi$ tal que $\delta^A(X_0, s\sigma) = X_i$ e $\exists X_b \in \pi$ tal que $\delta^A(X_0, s) = X_b)$.

A linguagem K é Γ-compatível em relação a $L(G_e^A)$ se, e somente se, $K = \emptyset$ ou;

1. *(Se $q_0 \in Q_{Km}$), $(\exists(\gamma, M) \in \Gamma(X_0, \sigma, X_0)) : \gamma \cap \Sigma_{G_e^A}(X_0) = \Sigma_{G_K}(q_0)$*

   (Se $q_0 \in (Q_K - Q_{Km})$), $(\exists(\gamma, N) \in \Gamma(X_0, \sigma, X_0)) : \gamma \cap \Sigma_{G_e^A}(X_0) = \Sigma_{G_K}(q_0)$

2. $(\forall q \in Q_{Km}),\ (\exists(\gamma, M) \in \Gamma(X_i, \sigma, X_b)) : \gamma \cap \Sigma_{G_e^A}(X_i) = \Sigma_{G_K}(q)$

3. $(\forall q \in (Q_K - Q_{Km})),\ (\exists(\gamma, N) \in \Gamma(X_i, \sigma, X_b)) : \gamma \cap \Sigma_{G_e^A}(X_i) = \Sigma_{G_K}(q)$

5.3.3 Complexidade computacional para obtenção do modelo agregado

Como visto anteriormente, a complexidade para obtenção do modelo agregado situa-se basicamente no cálculo das estruturas de controle. Para analisar a complexidade computacional para a obtenção do modelo agregado, considera-se que a planta de baixo nível G, inicialmente particiona-se em b blocos e cada bloco no pior caso, tem m estados e n transições. Calculam-se estruturas de controle para cada bloco e cada estado de entrada no bloco, no pior caso, todos os estados do bloco são estados de entrada, conseqüentemente pode-se ter mb estruturas de controle. Considera-se que o alfabeto

de eventos relevantes para especificação Σ^A tem r eventos. Cada estrutura de controle, no pior caso, pode ter 2^r padrões de controle marcados e 2^r não marcados.

O número de padrões de controle do modelo agregado no pior caso será $2mb(2)^r$. A complexidade computacional para obtenção de um padrão de controle é $\mathcal{O}(m^2n^2)$ (Complexidade para obtenção da máxima linguagem controlável). Portanto a complexidade para obtenção do modelo agregado com estruturas de controle é da ordem $\mathcal{O}(2m^3n^2(2)^r)$. Observa-se que a complexidade é exponencial em relação ao número de eventos relevantes.

5.4 Resultados Principais

Nesta seção prova-se que, dada uma planta G determinística e um alfabeto para possíveis especificações Σ^A, sempre é possível obter um modelo agregado G^A definido sobre Σ^A que satisfaça as boas propriedades para controle supervisório.

Constatou-se que as estruturas de controle tal como construídas na Definição 5.12, garante o não bloqueio e consistência hierárquica. Para demonstrar este fato, inicialmente são definidos supervisores de alto e baixo nível e mostra-se como estes estão relacionados.

Dado um SED controlado $D = (L(G^A), \Gamma)$ representando a planta de alto nível com estruturas de controle, seja dependentes do estado ou (s, e, ps)-dependente, um supervisor f^A para D é definido como um mapa

$$f^A : L(G^A) \longrightarrow 2^\Sigma \times \{M, N\}$$

O comportamento em malha fechada do sistema f^A/D é representado por um par de linguagens, uma linguagem prefixo fechada $L(f^A/D)$ e uma linguagem marcada $L_m(f^A/D)$. A linguagem fechada $L(f^A/D)$ é definida recursivamente como

1. $\epsilon \in L(f^A/D)$
2. $s\sigma \in L(f^A/D) \longleftrightarrow s \in L(f^A/D) \wedge s\sigma \in L(G^A) \wedge \sigma \in \gamma$ com $f^A(s) = (\gamma, \#)$

e a linguagem marcada $L_m(f^A/D)$ como

$$s \in L_m(f^A/D) \longleftrightarrow s \in L(f^A/D) \wedge f^A(s) = (\gamma, M)$$

onde para estruturas de controle dependentes do estado $(\gamma, \#) \in \Gamma(X_i)$ para $\delta^A(q_0^A, s) = X_i$ e para estruturas de controle (s, e, ps)- dependente $(\gamma, \#) \in \Gamma(X_i, \sigma, X_b)$ para $s = s'\sigma$, com $\delta^A(q_0^A, s) = X_i$ e $\delta^A(q_0^A, s') = X_b$.

Em geral $\overline{L_m(f^A/D)} \subseteq L(f^A/D)$, e o supervisor é dito não bloqueante se $\overline{L_m(f^A/D)} = L(f^A/D)$

O teorema 4.1 apresentado no capítulo 4 mostra que a Γ-compatibilidade é uma condição necessária e suficiente para a existência de um supervisor não bloqueante para implementar uma dada linguagem K num SED controlado $D = (L(G^A), \Gamma)$

Por construção das estruturas de controle do nível agregado, para cada entrada de controle $(\gamma, \#) \in \Gamma_{ij}$, existe no baixo nível uma linguagem $K_c \subset L(H_{ij}^+)$ tal que $K_c \in E_{ij}$ e $[\theta(K_c) - \epsilon] = \gamma$. Observe que K_c é controlável em relação a $L(H_{ij}^+)$ por ser elemento de E_{ij}. Define-se o supervisor no baixo nível o qual realiza K_c como o mapeamento,

$$f_{ij}^{K_c} : K_c \longrightarrow 2^{\Sigma}$$

observe também que para cada elemento K_c de E_{ij} tem-se um supervisor local $f_{ij}^{K_c}$.

Definidos os supervisores para os dois níveis, o algoritmo a seguir traduz a implementação de f^A para um supervisor de baixo nível $f : L(G) \longrightarrow 2^{\Sigma}$. Este esquema é ilustrado na Figura 5.12.

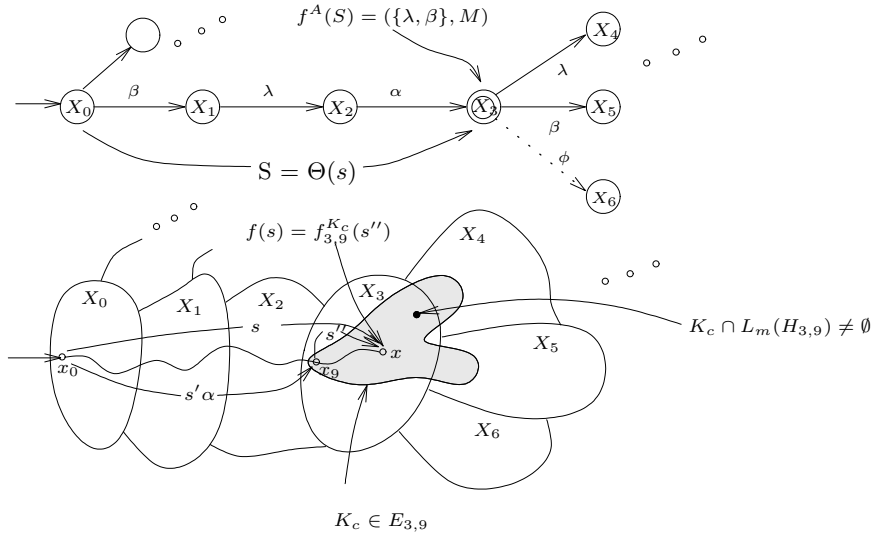


Figura 5.12: Tradução do controle do alto para baixo nível.

Algoritmo 5.1 *Obtenção do supervisor de baixo nível.*

Entrada: f^A

1. Para $s \in L(G)$, tal que $\Theta(s) = \epsilon$,
   $f(s) = f_{0,0}^{K_c}(s)$, onde $K_c \in E_{0,0}$ é tal que $(\Theta(K_c) - \epsilon) = \gamma$ **e** $\Big[(\epsilon \in \Theta(K_c)$ se $f^A(\epsilon) = (\gamma, M))$ **ou** $(\epsilon \notin \Theta(K_c)$ se $f^A(\epsilon) = (\gamma, N))\Big]$.
2. Para $s \in L(G)$, com $s = s'\alpha s''$, $s' \in \Sigma^*$, $\alpha \in \Sigma^A$, $s'' \in (\Sigma - \Sigma^A)^*$, tal que $\Theta(s) \in \overline{K}^A$, e $\delta(x_0, s'\alpha) = x_j \in I(X_i)$,
   $f(s) = f_{ij}^{K_c}(s'')$, onde $K_c \in E_{ij}$ é tal que $(\Theta(K_c) - \epsilon) = \gamma$ **e** $\Big[(\epsilon \in \Theta(K_c)$ se $f^A(\theta(s)) = (\gamma, M))$ **ou** $(\epsilon \notin \Theta(K_c)$ se $f^A(\theta(s)) = (\gamma, N))\Big]$.

Saída: f

Observe que a linguagem de baixo nível resultante da ação de f sobre G denotada $L(f/G)$, pode ser obtida pela montagem das linguagens realizadas pelos supervisores locais $f_{ij}^{K_c}$ envolvidos. A linguagem marcada é definida como $L_m(f/G) = L(f/G) \cap L_m(G)$.

Definidas as implementações de baixo e alto nível, o Teorema a seguir apresenta as condições de consistência hierárquica.

Teorema 5.2 Dada uma linguagem realizada pelo supervisor do nível agregado, isto é, $L_m(f^A/G^A) = K^A$, e a correspondente realização no baixo nível, $L_m(f/G) = K$, pode-se afirmar que $\Theta(K) = K^A$.

Prova:

Por ser $K^A \in L(G^A)$, Γ-*compatível* e por construção do supervisor de baixo nível, existe a correspondente realização no baixo nível ($\overline{K} \subset L(G)$), resultado da montagem de todas as linguagens controláveis $\overline{K}_c \subset L(H_{ij}^+)$ envolvidas, tal que $K_c \in E_{ij}$, onde cada $\overline{K}_c$ resulta ser a realização de um supervisor local $f_{ij}^{K_c}$, isto é, $L(f_{ij}^{K_c}/H_{ij}^+) = \overline{K}_c$. Por construção das estruturas de controle do nível agregado, o mapeamento das linguagens K_c, $(\Theta(K_c) - \epsilon)$ representam as continuações (entradas de controle) no nível agregado após ser atingido o bloco X_i pela entrada $x_j \in I(X_i)$. Portanto esta realização representa o comportamento em malha fechada do nível agregado, isto é, $L(f^A/G^A) = \overline{K}^A$.

Conclui-se que $\Theta(\overline{K}) = \overline{K}^A$.

Para qualquer $s \in K$, $\exists\Theta(s)$, tal que $f^A(\Theta(s)) = (\gamma, M)$, isto por construção de Γ e a condição de Γ-compatibilidade. Portanto $\Theta(s) \in K^A$.

Conclui-se que $\Theta(K) = K^A$

$\diamondsuit$

O teorema anterior garante que para qualquer linguagem K^A realizável no alto nível, existe no baixo nível uma linguagem controlável cuja imagem seja K^A, o que caracteriza a *consistência hierárquica*. Como conseqüência tem-se o Corolário a seguir.

Corolário 5.1 Dado K^A, Γ-compatível , $\exists K \subset L_m(G)$ controlável e L_m-fechada tal que $\Theta(K) = K^A$

Prova:

Pelo Teorema 5.2

$\diamondsuit$

Teorema 5.3 Toda linguagem $K \subseteq L_m(G)$ controlável e L_m-fechada tem imagem Γ-compatível no nível agregado.

Prova:

A linguagem K pode ser decomposta em pequenas linguagens locais K' correspondentes aos blocos envolvidos como segue.

- Para $s \in \overline{K}$, com $s \in (\Sigma - \Sigma^A)^*$. Seja $\sigma \in \Sigma$ um evento habilitado após s, tal que $s\sigma \in \overline{K}$.

$s\sigma \in \overline{K} \longleftrightarrow s\sigma \in \overline{K}'$

$s\sigma \in L(G) \longleftrightarrow s\sigma \in L(H^+_{0,0})$

Se $s\sigma \in (\overline{K}\Sigma_u) \wedge s\sigma \in L(G) \longrightarrow s\sigma \in \overline{K}$ por controlabilidade de K

Então $s\sigma \in (\overline{K}'\Sigma_u) \wedge s\sigma \in L(H^+_{0,0}) \longrightarrow s\sigma \in \overline{K}'$

Portanto $\overline{K}'\Sigma_u \cap L(H^+_{0,0}) \subseteq \overline{K}'$, ou seja K' é controlável em relação a $L(H^+_{0,0})$.

Por ser K, $L_m(G)$-fechado, isto é $K = \overline{K} \cap L_m(G)$, também pode-se afirmar que $K' = \overline{K}' \cap L_m(H^+_{0,0})$. Ou seja K' é $L_m(H^+_{0,0})$-fechado.

- Para $s \in \overline{K}$, com $s = s'\alpha s''$, $s' \in \Sigma^*$, $\alpha \in \Sigma^A$, $s'' \in (\Sigma - \Sigma^A)^*$, tal que $\delta(x_0, s'\alpha) = x_j \in I(X_i)$. Seja $\sigma \in \Sigma$ um evento habilitado após s tal que $s\sigma \in \overline{K}$.

$s\sigma \in \overline{K} \longleftrightarrow s''\sigma \in \overline{K}'$

$s\sigma \in L(G) \longleftrightarrow s''\sigma \in L(H^+_{ij})$

Se $s\sigma \in (\overline{K}\Sigma_u) \wedge s\sigma \in L(G) \longrightarrow s\sigma \in \overline{K}$ por controlabilidade de K

Então $s''\sigma \in (\overline{K}'\Sigma_u) \wedge s''\sigma \in L(H_{ij}^+) \longrightarrow s''\sigma \in \overline{K}'$

Portanto $\overline{K}'\Sigma_u \cap L(H_{ij}^+) \subseteq \overline{K}'$, ou seja K' é controlável em relação a $L(H_{ij}^+)$

Por ser K, $L_m(G)$-fechado, também pode-se afirmar que $K' = \overline{K}' \cap L_m(H_{ij}^+)$. Ou seja K' é $L_m(H_{0,0}^+)$-fechado.

Uma vez que $K' \subset L(H_{ij}^+)$ é controlável e L_m-fechado, então por construção das estruturas de controle do nível agregado existe $K_c \subseteq L(H_{ij})$, tal que $K_c \in E_{ij}$ e $K' \subseteq K_c$, com $\Theta(K') = \Theta(K_c)$.

Portanto para cada $s \in \overline{K}$, com $s \in (\Sigma - \Sigma^A)^*$, existe $K' \subseteq L(H_{0,0}^+)$, tal que $\exists(\gamma, \#)$, com $\gamma = \Theta(K') - \epsilon$ e $\# = M$ se $\epsilon \in \Theta(K')$, caso contrário $\# = N$.

Estendendo para as demais cadeias, para cada $s \in \overline{K}$, com $s = s'\alpha s''$, $s' \in \Sigma^*$, $\alpha \in \Sigma^A$, $s'' \in (\Sigma - \Sigma^A)^*$, tal que $\delta(x_0, s'\alpha) = x_j \in I(X_i)$, existe $K' \subseteq L(H_{ij}^+)$, tal que $\exists(\gamma, \#)$, com $\gamma = \Theta(K') - \epsilon$ e $\# = M$ se $\epsilon \in \Theta(K')$, caso contrário $\# = N$.

Finalmente, para cada cadeia $S \in \Theta(K)$, existe (γ, M) tal que $\gamma = \Sigma_K(S)$, e para cada cadeia $S \in (\Theta(K) - \overline{\Theta}(K))$, existe (γ, N) tal que $\gamma = \Sigma_K(S)$. O que caracteriza a Γ-compatibilidade de $\Theta(K)$.

$\diamondsuit$

O teorema anterior garante a *consistência hierárquica forte*.

Observe que os teoremas anteriores são válidos para autômatos agregados determinísticos, uma vez que a construção de Γ está restrita a autômatos determinísticos.

Caso uma linguagem K^A não seja Γ-compatível, procura-se uma sub-linguagem Γ-compatível contida em K^A, tal que seja minimamente restritiva.

5.4.1 Cálculo da máxima linguagem Γ-compatível $\sup\mathcal{CM}$

Dependendo do tipo da estrutura de controle que for usado para representação de um sistema agregado, apresentam-se algoritmos para o cálculo da máxima linguagem Γ-compatível.

Sistema com estruturas de controle dependentes do estado $D = (G_s^A, \Gamma)$,

Neste caso o algoritmo para obter a máxima linguagem Γ-compatível resulta ser o mesmo que o algoritmo 4.1 (Cury et al. 2001).

Sistema com estruturas de controle (s, e, ps)-dependente $D = (G_e^A, \Gamma)$,

A seguir apresenta-se o algoritmo para obter a máxima linguagem Γ-compatível no caso de sistemas com estruturas de controle (s, e, ps)-dependente.

Algoritmo 5.2 *Permite encontrar* $\sup \mathcal{CM}(K)$ *no caso (s, e, ps)-dependente.*

Dados:
$D = (G_e^A, \Gamma)$: Um SED com Γ (s, e, ps)-dependente.
S : Um gerador Trim tal que $L_m(S) = K \subseteq L(G)$.

Obter G', um gerador Trim tal que $L(G') = L_m(G') = L(G_e^A)$, (marcar todos os estados de G_e^A)

Construir $C_i = G' \,||\, S$, a composição síncrona entre G' e S, com $i = 0$

Identificar os maus estados (q, p) de C_i e/ou as más transições da forma a seguir.

laço

Primeiro identificar os *possíveis* maus estados de C_i. São estados $(q, p) \in C_i$ tais que:

- $(q, p) \in Q_m(C_i)$ e se para algum $(q', p') \in Q(C_i) : \delta_{C_i}((q', p'), \sigma) = (q, p),\ \nexists(\gamma, M) \in \Gamma(q, \sigma, q') : \gamma \cap \Sigma_{G'}(q) = \Sigma_{C_i}(q, p)$

- $(q, p) \in (Q(C_i) - Q_m(C_i))$ e se para algum $(q', p') \in Q(C_i) : \delta_{C_i}((q', p'), \sigma) = (q, p)$, $\nexists(\gamma, N) \in \Gamma(q, \sigma, q') : \gamma \cap \Sigma_{G'}(q) = \Sigma_{C_i}(q, p)$

Destes possíveis maus estados, refinar (q, p) como segue:

para todo $(q, p) \in Q_m(C_i)$ **faça**

para todo $(q', p') \in Q(C_i) : \delta_{C_i}((q', p'), \sigma) = (q, p)$, buscar o máximo $(\gamma', \#) \in \Gamma(q, \sigma, q')$ tal que $\gamma' \cap \Sigma_{G'}(q) \subseteq \Sigma_{Ci}(q, p)$,

- caso $(\gamma', M), (\gamma', N)$ existam, escolher (γ', M)

- caso para algum q', não exista nenhum $(\gamma', \#)$ tal que $\gamma' \cap \Sigma_{G'}(q) \subseteq \Sigma_{C_i}(q, p)$, então a transição de entrada $\delta_{C_i}((q', p'), \sigma) = (q, p)$ é uma má transição.

fim para

Para cada diferente $(\gamma', \#)$ achado, refinar o estado (q, p) em (q_k, p_k), com $k = 1, \ldots, T$, onde T é o número de $(\gamma', \#)$s diferentes.

As más transições de saída para cada (q_k, p_k), serão:

$$bad_t(q_k, p_k) = \Sigma_{C_i}(q, p) - (\gamma' \cap \Sigma_{G'}(q))$$

fim para

para todo $(q, p) \in (Q(C_i) - Q_m(C_i))$ **faça**

para todo $(q', p') \in Q(C_i) : \delta_{C_i}((q', p'), \sigma) = (q, p)$, buscar o máximo $(\gamma', N) \in \Gamma(q, \sigma, q')$ tal que $\gamma' \cap \Sigma_{G'}(q) \subseteq \Sigma_{Ci}(q, p)$,

- caso para algum q', não exista nenhum (γ', N) tal que $\gamma' \cap \Sigma_{G'}(q) \subseteq \Sigma_{C_i}(q, p)$, então a transição de entrada $\delta_{C_i}((q', p'), \sigma) = (q, p)$ é uma má transição.

fim para

Para cada diferente (γ', N) achado, refinar o estado (q, p) em (q_k, p_k), com $k = 1, \ldots, T$, onde T é o número de $(\gamma', \#)$s diferentes.

As más transições de saída para cada (q_k, p_k), serão:

$$bad_t(q_k, p_k) = \Sigma_{C_i}(q, p) - (\gamma' \cap \Sigma_{G'}(q))$$

fim para

Obter C_i' apagando as más transições e os estados não alcançáveis de C_i e fazer $C_{i+1} = Trim(C_i')$.

se $C_{i+1} = C_i$, **então** parar!, e fixar $I = i$ pois $L_m(C_I)$ é a máxima sub-linguagem Γ-compatível, **senão**, fixar $i = i + 1$ **fim se**.

fim laço

Saída: C_I

5.5 Exemplo: Supervisão hierárquica de uma linha de transferência.

Esta abordagem será ilustrada pelo desenvolvimento de um supervisor hierárquico para uma linha de transferência apresentada por Wonham (1998) no controle hierárquico tradicional. A linha de transferência consiste de duas máquinas M_1 e M_2 mais uma unidade de teste TU ligadas por dois *buffers* B_1 e B_2 na seqüência M_1, B_1, M_2, B_2, TU como mostrado na Figura 5.13. A peça testada por TU pode ser aceita ou rejeitada; se for aceita, esta é liberada do sistema; senão volta para o buffer B_1 para re-processamento por M_2

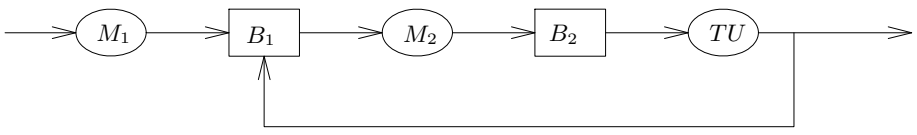


Figura 5.13: Linha de Transferência.

Os modelos dos componentes do sistema são mostrados a seguir.

Os eventos controláveis 1 e 3 indicam o início de operação das máquinas M_1 e M_2, respectivamente, e os eventos não controláveis 2 e 4 indicam o fim de operação das mesmas. O início da unidade de teste é representado pelo evento controlável 5 e os sinais de decisão de "aceito" ou "rejeitado" são representados pelos eventos não controláveis 60 e 80 respectivamente. Caso a peça seja aceita pela unidade de teste,

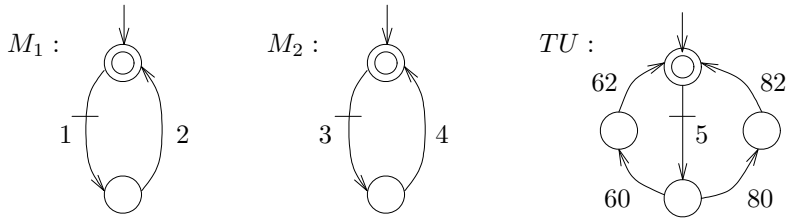


Figura 5.14: Modelo dos componentes do sistema.

esta é enviada para fora do sistema (evento 62), caso contrário retorna para o buffer B_1 (evento 82).

Consideram-se os buffers de capacidade unitária. Na Figura 5.15 mostram-se as especificações para não *overflow* e não *underflow* dos buffers.

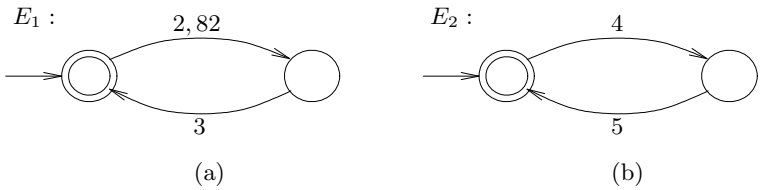


Figura 5.15: Especificações de não overflow e não underflow dos buffers: (a) B_1 e (b) B_2.

A Figura 5.16 apresenta o modelo da solução para este problema, neste autômato substitui-se o evento não controlável 80 pelo evento controlável 81 sob a suposição que se acontecem muitas falhas detectadas pela TU o "gerente" toma uma ação de desabilitar o re-processamento da peça para remediar o problema.

Adota-se como planta G para tratar do problema de controle hierárquico a solução do sistema para não *overflow* e não *underflow* dos *buffers*.

No alto nível, a preocupação do supervisor atribui-se ao controle da relação entre peças de entrada e peças aceitas ou recusadas. Portanto, considera-se como eventos relevantes os eventos relacionados com este fato $(1, 60, 81)$. Com estes dados constrói-se uma agregação e o resultado conjuntamente com os padrões de controle são mostrados na Figura 5.17. Observe-se que o bloco X_0 apresenta um não determinismo, motivo pelo qual o evento que produz não determinismo, 1 é instanciado em $1'$ e $1''$.

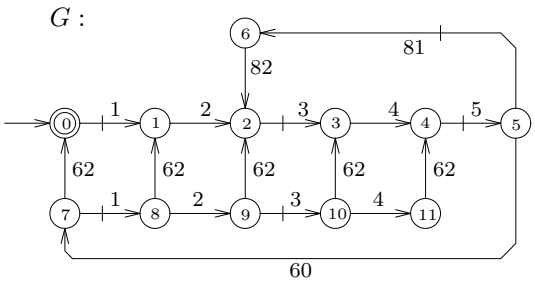
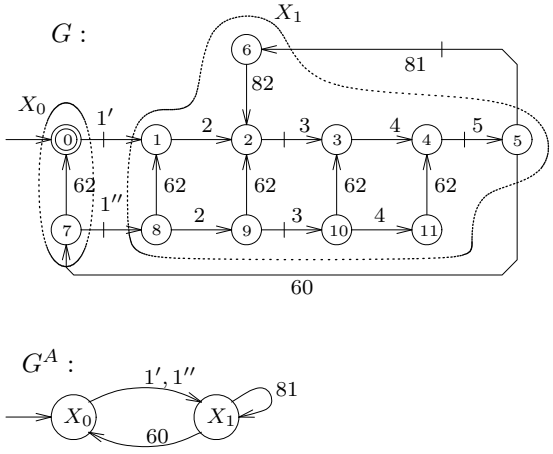


Figura 5.16: Planta.

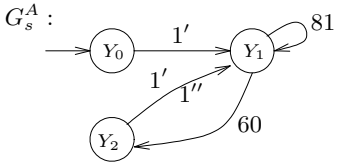


$$\Gamma_{0,0} = \{(\emptyset, M); (\{1'\}, M)\}$$
$$\Gamma_{0,7} = \{(\emptyset, M); (\{1'\}, M); (\{1''\}, M); (\{1', 1''\}, M)\}$$
$$\Gamma_{1,1} = \{(\{60\}, N); (\{60, 81\}, N)\}$$
$$\Gamma_{1,8} = \{(\{60\}, N); (\{60, 81\}, N)\}$$
$$\Gamma_{1,6} = \{(\{60\}, N); (\{60, 81\}, N)\}$$

Figura 5.17: Agregação do sistema.

5.5.1 Estruturas de controle dependentes do estado

Observa-se que o bloco X_0 do autômato agregado (Fig. 5.17) apresenta conjuntos de estruturas de controle diferentes para os dois estados de entrada 0 e 7 ($\Gamma_{0,0} \neq \Gamma_{0,7}$), motivo pelo qual o bloco X_0 é dividido em dois, um associado a $\Gamma_{0,0}$ e outro a $\Gamma_{0,7}$ como mostrado na Figura 5.18 representado pelos estados Y_0 e Y_2. Já o bloco X_1, para todas as entradas apresenta um mesmo conjunto de estruturas de controle ($\Gamma_{1,1} = \Gamma_{1,1} = \Gamma_{1,6}$), sendo mantido tal como está. Na Figura 5.18, X_1 está representado por Y_1.



$$\Gamma(Y_0) = \Gamma_{0,0} = \{(\emptyset, M); (\{1'\}, M)\}$$
$$\Gamma(Y_1) = \Gamma_{1,1} = \Gamma_{1,8} = \Gamma_{1,6} = \{(\{60\}, N); (\{60, 81\}, N)\}$$
$$\Gamma(Y_2) = \Gamma_{0,7} = \{(\emptyset, M); (\{1'\}, M); (\{1''\}, M); (\{1', 1''\}, M)\}$$

Figura 5.18: Planta hierárquica com estruturas de controle dependentes do estado.

A seguir define-se uma especificação para o sistema (Fig 5.19). Neste caso, toda vez que uma peça entra no sistema e se esta for recusada duas vezes seguidas pela unidade de teste, então é forçada uma finalização e o sistema pára.

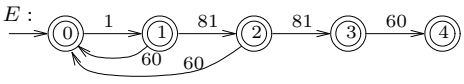


Figura 5.19: Especificação.

Continuando, obtém-se a linguagem alvo (Fig 5.20) e aplica-se o Algoritmo 4.1 para síntese da máxima linguagem Γ-*compatível* apresentado para sistemas com estruturas de controle dependentes do estado. O resultado é mostrado na Figura 5.21.

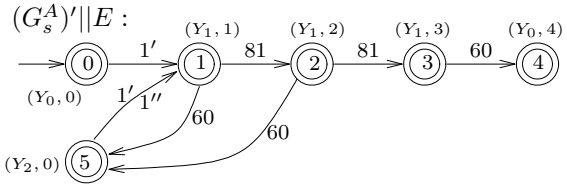


Figura 5.20: Linguagem alvo.

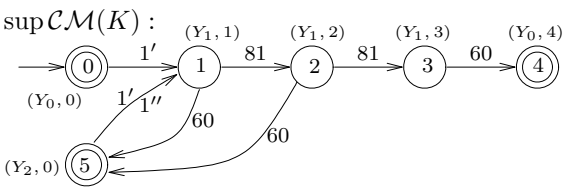
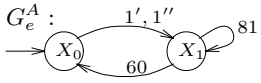


Figura 5.21: Máxima linguagem Γ-*compatível*.

5.5.2 Estruturas de controle (s, e, ps)-dependente

Neste caso, a fim de manter a estrutura de transição da planta de alto nível, utilizam-se estruturas de controle (s, e, ps)-dependente, tal como mostrado na Figura 5.22.



$$\Gamma(X_0, \epsilon, X_0) = \Gamma_{0,0} = \{(\emptyset, M); (\{1'\}, M)\}$$
$$\Gamma(X_0, 60, X_1) = \Gamma_{0,7} = \{(\emptyset, M); (\{1'\}, M); (\{1''\}, M); (\{1', 1''\}, M)\}$$
$$\Gamma(X_1, 1', X_0) = \Gamma_{1,1} = \{(\{60\}, N); (\{60, 81\}, N)\}$$
$$\Gamma(X_1, 1'', X_0) = \Gamma_{1,8} = \{(\{60\}, N); (\{60, 81\}, N)\}$$
$$\Gamma(X_1, 81, X_1) = \Gamma_{1,6} = \{(\{60\}, N); (\{60, 81\}, N)\}$$

Figura 5.22: Planta hierárquica com estruturas de controle (s, e, ps)-dependente.

A especificação definida para o caso anterior continua sendo válida. A seguir obtém-se a linguagem alvo (Fig. 5.23) e aplica-se o Algoritmo 5.2 para síntese da máxima linguagem Γ-*compatível* apresentado para sistemas com estruturas de controle (s, e, ps)-dependente. O resultado da máxima linguagem Γ-*compatível* é mostrado na Figura 5.24.

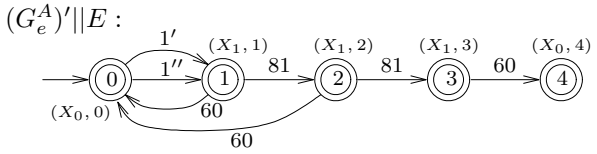


Figura 5.23: Linguagem alvo.

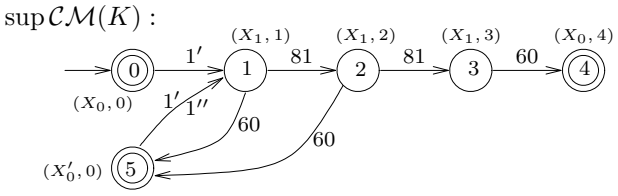


Figura 5.24: Máxima linguagem Γ-*compatível*.

5.6 Conclusão

Neste capítulo foi apresentado um novo modelo para controle supervisório hierárquico por agregação de estados. Esse modelo é construído a partir de uma planta a controlar e um conjunto de eventos considerados relevantes para especificação.

A forma de construção deste modelo já apresenta a vantagem de possuir diretamente a consistência hierárquica forte. Isso não acontece nas abordagens de (Zhong e Wonham

1990, Wong e Wonham 1996) que ainda precisam de um refinamento adequado do canal de informação. Na abordagem de Caines e Hubbard (1998a), se o modelo obtido não possuir consistência hierárquica forte (IBC-não bloqueante), não existe uma maneira sistemática de refinar a agregação de tal modo a satisfazer esta condição, e os autores somente apresentam dicas, as quais podem levar a diversas soluções.

O novo modelo, como foi visto, apresenta um custo computacional elevado na obtenção das estruturas de controle para o nível agregado (exponencial em relação ao número de eventos relevantes para especificação), apesar de o procedimento para obter Γ ser feito com sub-autômatos reduzidos. No entanto uma vez obtido este modelo do nível agregado, problemas de controle supervisório poderão ser resolvidos sobre um autômato reduzido e conseqüentemente com menor complexidade.

Uma alternativa para a solução aqui adotada para tratar o não determinismo do autômato agregado poderia considerar a opção pela determinização clássica. Nesse caso, seria necessário juntar dois ou mais blocos com estruturas de controle diferentes, levando à necessidade de refinamento da estrutura de controle dos blocos unidos. Esse refinamento pode levar a uma perda de informação da estrutura de controle, e conseqüentemente à perda de consistência hierárquica forte. Em (da Cunha 2001), um problema similar a este é tratado.

Capítulo 6

Redução do modelo agregado com estruturas de controle dependentes do estado

No capítulo 5, constrói-se o modelo agregado para controle hierárquico sem se preocupar se ele pode ainda ser representado por um modelo mais compacto.

O objetivo deste capítulo é propor formas sistemáticas para redução do modelo agregado com estruturas de controle dependentes do estado apresentado no capítulo 5 (Torrico e Cury 2001, Torrico e Cury 2002c) de tal forma que este mantenha as boas propriedades do controle hierárquico.

Para obter um autômato agregado com boas propriedades, previamente é necessário tratar o problema de não determinismo que pode aparecer pelo fato da agregação. No capítulo 5, este problema a princípio foi resolvido renomeando-se todas transições que produziam o não determinismo. Neste capítulo o que se pretende é apresentar um procedimento para diminuir o número de renomeações e de estados do modelo agregado. A próxima seção descreve detalhadamente o problema a ser resolvido.

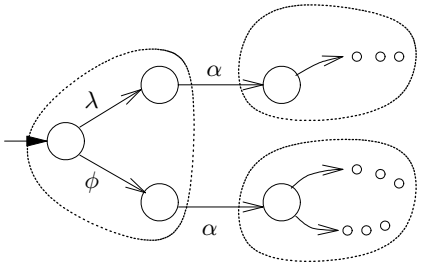
6.1 Descrição do Problema

Dentro da abordagem de controle hierárquico por agregação de estados um dos problemas que aparece pelo fato da agregação é o não determinismo. Considerando o autômato agregado $G^A = (\Sigma^A, \pi, \delta^A, q_0^A)$, a seguir ilustram-se as duas situações

possíveis de ocorrer:

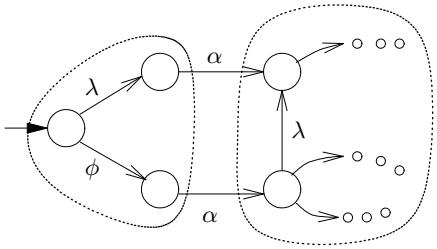
(a) G^A é propriamente não determinístico, isto é, existem $X_i, X_j, X_k \in \pi$ e $\sigma \in \Sigma^A$, tal que, $\delta^A(X_i, \sigma) = \{X_j, X_k\}$, com $X_j \neq X_k$. A Figura 6.1(a) ilustra um exemplo deste caso.

(b) Existe $X_i, X_j \in \pi$ e $\sigma \in \Sigma^A$ tal que $\delta(X_i, \sigma) = X_j$, e tal que, existem $x, y \in X_i$ e $z, w \in X_j$, $z \neq w$, com $\delta(x, \sigma) = z$ e $\delta(y, \sigma) = w$. A Figura 6.1(b) ilustra um exemplo deste caso.



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

(a) Caso de não determinismo

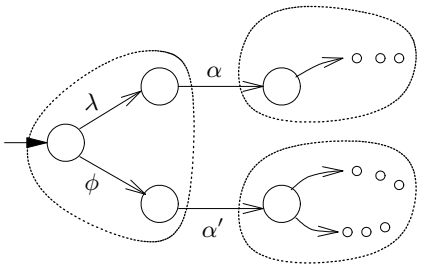


$\Sigma = \{\alpha, \beta, \lambda, \phi\}$

(b) Caso de não determinismo

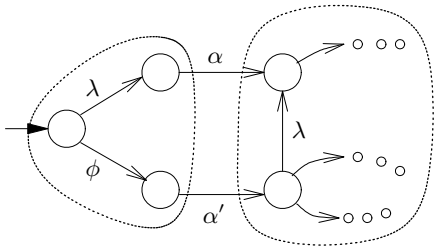
Figura 6.1: Possíveis situações de não determinismo.

Para as agregações ilustradas na Figura 6.1, qualquer um dos casos anteriores de não determinismo a princípio poderia ser eliminado simplesmente renomeando o evento envolvido $\alpha \in \Sigma$ que gera o não determinismo, em eventos $\alpha, \alpha', \ldots$ que irão aumentar o alfabeto Σ, conforme o procedimento adotado na seção 5.2. Na Figura 6.2 apresentam-se exemplos de renomeação das agregações apresentadas na Figura 6.1.



$\Sigma = \{\alpha, \alpha', \beta, \lambda, \phi\}$

(a) Renomeação



$\Sigma = \{\alpha, \alpha', \beta, \lambda, \phi\}$

(b) Renomeação

Figura 6.2: Exemplo de renomeação.

Considerando que o autômato agregado G^A possui estruturas de controle dependentes do estado $X_i \in \pi$, do tipo $\Gamma(X_i) \subseteq 2^{\Sigma^A} \times \{M, N\}$ (Cury et al. 2001), o número de padrões de controle $(\gamma, \#) \in \Gamma(X_i)$ por cada estado cresce exponencialmente com o conjunto de eventos ativos após X_i. Como conseqüência o processo de renomeação leva a um crescimento exponencial do número de padrões de controle.

No caso de não determinismo do tipo (b) a renomeação de algumas transições poderá ser evitada, se as possíveis continuações (padrões de controle) após aquelas transições forem iguais.

O objetivo deste trabalho é desenvolver métodos para renomeação e redução do autômato agregado de tal forma que este seja determinístico e mantenha as propriedades de consistência hierárquica.

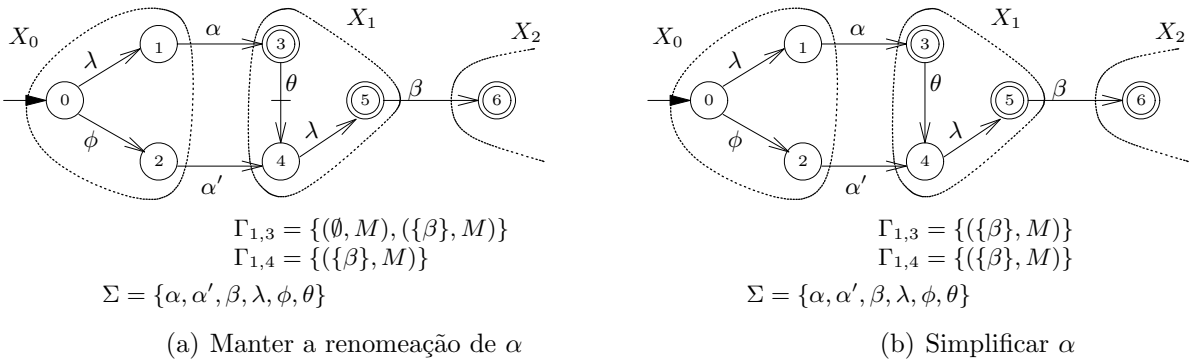
6.2 Resolução do problema

A proposta para resolução do problema consiste numa renomeação estática e dinâmica.

6.2.1 Renomeção Estática

Este método consiste inicialmente em renomear todos os eventos que geram o não determinismo para depois proceder a sua redução. Propõe-se o procedimento sistemático a seguir.

Inicialmente todos os eventos relevantes que geram não determinismo seja do tipo (a) ou (b) são renomeados. A seguir são calculadas as estruturas de controle Γ_{ij} para cada bloco X_i e por cada entrada no bloco $x_j \in I(X_i)$. Caso ocorra um não determinismo do tipo (b) serão verificadas se as possíveis continuações após aquelas transições que produzem o não determinismo são iguais. Para ilustrar esta idéia considere o caso particular das agregações mostradas na Figura 6.3. Na Figura 6.3(a) a transição do bloco X_1 com o evento β poderá ser desabilitada dependendo do estado de entrada em X_1, e assim as estruturas de controle para cada entrada são diferentes ($\Gamma_{1,3} \neq \Gamma_{1,4}$). Isto indica que é necessária a renomeação do evento de entrada α e uma divisão deste bloco para cada entrada. Para o exemplo da Figura 6.3(b), a continuação do bloco X_1 não depende do estado de entrada ($\Gamma_{1,3} = \Gamma_{1,4}$), desta forma o evento α não precisa ser distinguido, podendo assim ser simplificado.



(a) Manter a renomeação de α (b) Simplificar α

Figura 6.3: Análise da renomeação do evento α

O próximo passo é obter o autômato agregado com estruturas de controle dependentes do estado. Para redução deste autômato consideram-se dois ou mais estados como equivalentes se eles pertencem a uma mesma classe de equivalência de Nerode (Carroll e Long 1989) e as estruturas de controle para estes blocos são iguais.

Para o caso geral apresenta-se a seguir um algoritmo de redução do autômato agregado (redução de renomeações e estados).

Algoritmo 6.1 *Calcula o autômato agregado com número reduzido de estados e renomeações.*

Entradas: $G = (\Sigma, Q, \delta, q_0, Q_m)$, $G^A = (\Sigma^A, \pi, \delta^A, q_0^A)$, $\Sigma^A \subseteq \Sigma$.

para todo $X_i \in \pi$; $i = 1, |\pi|$ **faça**

- Renomear os eventos de saída que geram o não determinismo seja do tipo (a) ou (b), isto é, para qualquer $\sigma \in \Sigma^A$, tais que existe $x, y, \ldots \in X_i$ com $\delta(x, \sigma)$, $\delta(y, \sigma)$, $\delta(\ldots, \sigma)$ definidos, renomear por eventos $\sigma, \sigma', \sigma'' \ldots$. Então se tem $\delta(x, \sigma)$, $\delta(y, \sigma')$, $\delta(\ldots, \sigma'')$ definidos.
- Calcular as estruturas de controle $\Gamma_{X_i,z}$ para o bloco $X_i \in \pi$ por cada entrada no bloco $z \in I(X_i)$).

fim para

- Obter G_k^A, o autômato do nível agregado com estruturas de controle dependentes do estado.
- Obter a lista de eventos renomeados para cada bloco, isto é, $f(X_i, \sigma) = \{\sigma, \sigma', \ldots\}$.

laço

para todo $X_i \in \pi_k$; $i = 1, |\pi_k|$ **faça**

se $\exists \sigma, \sigma' \in \Sigma_k^A$ com $f(X_i, \sigma) = \{\sigma, \sigma', \ldots\}$ tal que $\delta_k^A(X_i, \sigma) = X_j$ e $\delta_k^A(X_i, \sigma') = X_j$ **então**

- Simplificar renomeação e atualizar Γ_i

fim se

fim para

- Obter G^A_{k+1} minimizando os estados equivalentes em G^A_k . Dois estados $X_1, X_2 \in \pi_k$ são equivalentes se e somente se X_1 e X_2 pertencem à mesma classe de equivalência de Nerode e $\Gamma_1 = \Gamma_2$

se $G^A_{k+1} = G^A_k$ **então**

- Fazer N $\leftarrow k$ e parar!

senão

$k \leftarrow k + 1$

fim se

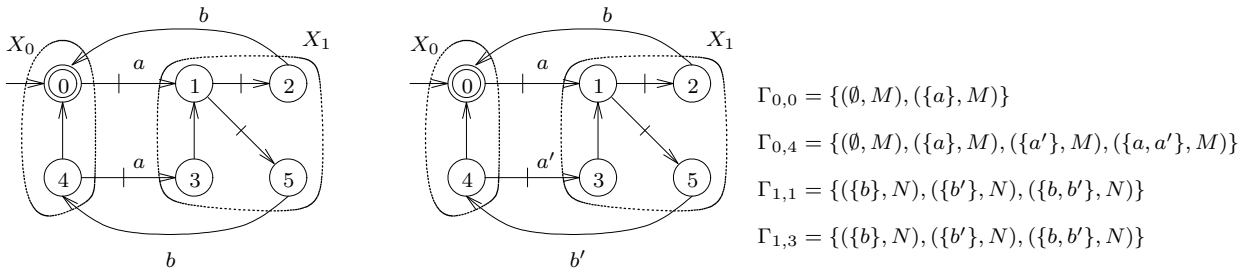
fim laço

Saída: $G^A_N = (\Sigma^A, \pi_N, \delta^A_N, q^A_{0N})$

Este algoritmo apresenta um procedimento recursivo e a cada ciclo diminui pelo menos um estado ou uma transição, o que garante que o algoritmo pára num número finito de passos.

Para ilustrar este algoritmo apresenta-se o exemplo a seguir:

Exemplo 6.1 (Exemplo de aplicação) *Considera-se a agregação inicial da Figura 6.4(a).*



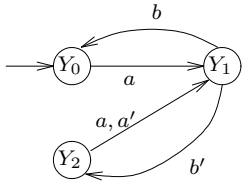
(a) Agregação inicial (b) Renomeação

Figura 6.4: Exemplo de minimização

Observa-se que os eventos "a" e "b" a princípio precisam ser renomeados. Na Figura 6.4(b) mostra-se a renomeação inicial e as estruturas de controle para cada bloco e cada entrada no bloco. Como próximo passo do algoritmo obtém-se o autômato agregado com estruturas de controle dependentes do estado (Fig. 6.5(a)). Dos blocos que saem transições renomeadas de uma mesma instância de evento que levam a um mesmo bloco, deverão ocorrer simplificações nas renomeações. Para o caso do exemplo, no autômato da Figura 6.5(a) tem-se os eventos de saída "a" e "a′" do bloco Y_2

chegando em Y_1 que serão representados os dois por "a"já que $\Gamma_{1,1} = \Gamma_{1,3}$.

Uma vez realizadas as renomeações requeridas, reduz-se o autômato. No caso do exemplo (Fig. 6.5(b)) os blocos Y_0 e Y_2 são equivalentes, isto porque pertencem à mesma classe de equivalência de Nerode e têm as mesmas estruturas de controle. Observa-se que depois da redução (Fig 6.5(c)) chega-se novamente ao caso da Figura 6.5(a). Seguindo este procedimento recursivo obtém-se o autômato reduzido com estruturas de controle dependentes do estado (Fig. 6.5(d)).

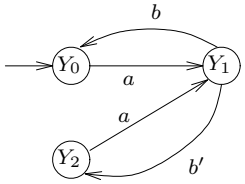


$\Gamma_{Y_0} = \Gamma_{0,0} = \{(\emptyset, M), (\{a\}, M)\}$

$\Gamma_{Y_2} = \Gamma_{0,4} = \{(\emptyset, M), (\{a\}, M), (\{a'\}, M), (\{a, a'\}, M)\}$

$\Gamma_{Y_1} = \Gamma_{1,1} = \Gamma_{1,3} = \{(\{b\}, N), (\{b'\}, N), (\{b, b'\}, N)\}$

(a) Autômato agregado Γ dependente do estado

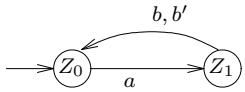


$\Gamma_{Y_0} = \{(\emptyset, M), (\{a\}, M)\}$

$\Gamma_{Y_2} = \{(\emptyset, M), (\{a\}, M)\}$

$\Gamma_{Y_1} = \{(\{b\}, N), (\{b'\}, N), (\{b, b'\}, N)\}$

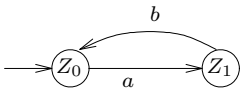
(b) Simplificando renomeação e atualizando Γ



$\Gamma_{Z_0} = \{(\emptyset, M), (\{a\}, M)\}$

$\Gamma_{Z_1} = \{(\{b\}, N), (\{b'\}, N), (\{b, b'\}, N)\}$

(c) Minimizando



$\Gamma_{Z_0} = \{(\emptyset, M), (\{a\}, M)\}$

$\Gamma_{Z_1} = \{(\{b\}, N)\}$

(d) Simplificando renomeação e atualizando Γ

Figura 6.5: Passos do algoritmo

O problema de renomeação resolvido anteriormente apresenta um método exaustivo, pois inicialmente todas as transições que produzem não determinismo serão renomeadas, levando a um crescimento inicial do número de padrões de controle por bloco. Na próxima seção propõe-se outro método, onde a estrutura inicial do autômato agregado deverá ser estudada para ver a possibilidade de uma renomeação dinâmica das transições que produzem o não determinismo na medida em que as estruturas de controle estejam sendo calculadas.

6.2.2 Renomeação dinâmica

Este método inicialmente analisa todas as transições que geram o não determinismo para depois verificar se é necessária uma renomeação.

Para motivar a renomeação dinâmica apresenta-se o exemplo a seguir.

Exemplo 6.2 *Considera-se a agregação inicial mostrada na Figura. 6.6.*

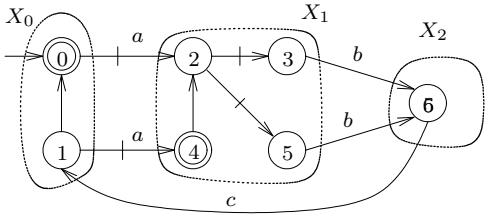


Figura 6.6: Exemplo de agregação.

Adota-se o bloco X_2 como ponto de partida para a análise, uma vez que este não apresenta não determinismo, e conseqüentemente as estruturas de controle deste bloco não serão alteradas.

Obtêm-se as estruturas de controle para este bloco ($\Gamma_{2,6} = \{(\{c\}, N)\}$).

No bloco X_1 observa-se que as transições com o evento "b" não precisarão ser renomeadas, uma vez que as suas continuações serão iguais. Considerando esta hipótese, as estruturas de controle para o bloco X_1 são: $\Gamma_{1,2} = \{(\{b\}, N)\}$; $\Gamma_{1,4} = \{(\{b\}, M)\}$. As continuações das transições com o evento "a" que saem do bloco X_0 já não serão iguais, uma vez que $\Gamma_{1,2} \neq \Gamma_{1,4}$, motivo pelo qual este evento é renomeado em "a" e "a′" (Fig. 6.7).

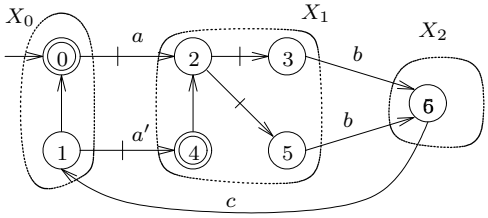


Figura 6.7: Renomeação

Posteriormente, são calculadas as estruturas de controle para o bloco X_0 ($\Gamma_{0,0} = \{(\emptyset, M), (\{a\}, M)\}$, $\Gamma_{0,1} = \{(\emptyset, M), (\{a\}, M), (\{a'\}, M), (\{a, a'\}, M)\}$). O autômato

resultante com estruturas de controle dependentes do estado é apresentado na Figura 6.8.

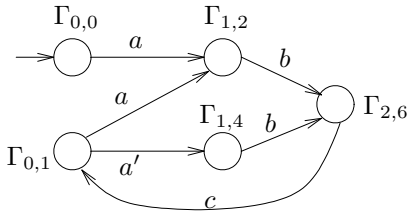


Figura 6.8: Autômato reduzido com estruturas de controle dependentes do estado

A partir do exemplo anterior chega-se às seguintes conclusões:

1. Podem existir blocos sem não determinismo. Estes terão preferência de serem analisados inicialmente. No exemplo tem-se unicamente o bloco X_2.

2. Os blocos dependentes são aqueles que apresentam não determinismo. As transições que produzem o não determinismo serão renomeadas dependendo das estruturas de controle dos blocos sucessores. No exemplo renomeações em X_1 dependem de seu sucessor X_2, e em X_0 dependem de seu sucessor X_1.

Para o caso geral apresenta-se o algoritmo 6.2 a seguir.

Algoritmo 6.2 *Calcula o autômato agregado com mínimo número de renomeações de uma forma direta.*

Entradas: $G = (\Sigma, Q, \delta, q_0, Q_m)$, $G^A = (\Sigma^A, \pi, \delta^A, q_0^A)$, $\Sigma^A \subseteq \Sigma$.
para todo $X_i \in \pi$; i=1,$|\pi|$ **faça**
  se X_i tem eventos de saída que produzem não determinismo **então**
    - Considerando σ como o evento que produz não determinismo em X_i
    se $\exists x, y \in X_i$ com $\delta(x, \sigma) \in X_j$ e $\delta(y, \sigma) \in X_j$ **então**
      se $j < i$ (Quer dizer que X_j já foi analisado) **então**
        - Seja $x' = \delta(x, \sigma)$ e $y' = \delta(y, \sigma)$, com $x', y' \in I(X_j)$
        se $\Gamma_{X_j,x'} = \Gamma_{X_j,y'}$ **então**
          - Manter o evento σ e calcular as estruturas de controle para X_i por cada entrada $z \in I(X_i)$
        senão
          - Renomear σ

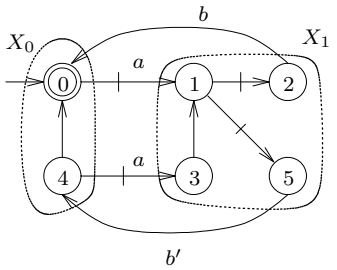
fim se
senão
- Renomear σ
fim se
senão
- Renomear σ
fim se
senão
- Calcular as estruturas de controle para X_i por cada entrada $z \in I(X_i)$
fim se
fim para
- Obter G_k^A, o autômato do nível agregado com estruturas de controle dependentes do estado.
- Obter a lista de eventos renomeados para cada bloco, isto é, $f(X_i, \sigma) = \{\sigma, \sigma', \ldots\}$.
laço
para todo $X_i \in \pi_k$; $i = 1, |\pi_k|$ **faça**
se $\exists \sigma, \sigma' \in \Sigma_k^A$ com $f(X_i, \sigma) = \{\sigma, \sigma', \ldots\}$ tal que $\delta_k^A(X_i, \sigma) = X_j$ e $\delta_k^A(X_i, \sigma') = X_j$ **então**
- Simplificar renomeação e atualizar Γ_i
fim se
fim para
- Obter G_{k+1}^A minimizando G_k^A os estados equivalentes. Dois estados $X_1, X_2 \in \pi_k$ são equivalentes se e somente se X_1 e X_2 pertencem à mesma classe de equivalência de Nerode e $\Gamma_1 = \Gamma_2$
se $G_{k+1}^A = G_k^A$ **então**
- Fazer N $\leftarrow k$ e parar!
senão
$k \leftarrow k + 1$
fim se
fim laço
Saída: $G_N^A = (\Sigma^A, \pi_N, \delta_N^A, q_{0N}^A)$

O exemplo a seguir ilustra a aplicação do algoritmo. Verifica-se que a renomeação de eventos será mais simplificada e direta evitando-se obter padrões de controle desnecessários.

Exemplo 6.3 (Forma simplificada de renomeação) *Considerando novamente a agregação inicial da Figura 6.4(a), percebe-se que esta não possui blocos independentes, conseqüentemente não há preferência por qual bloco começar a análise.*

Caso A *Iniciando a análise pelo bloco* X_1.

No bloco X_1 *renomeiam-se as transições com eventos de saída que produzem não determinismo e obtêm-se as estruturas de controle.*



$\Gamma_{1,1} = \{(\{b\}, N), (\{b'\}, N), (\{b, b'\}, N)\}$

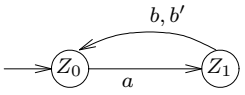
$\Gamma_{1,3} = \{(\{b\}, N), (\{b'\}, N), (\{b, b'\}, N)\}$

Figura 6.9: Renomeação dos eventos de saída do bloco X_1

Observa-se então que após X_0 *as continuações das transições com o evento "a" são iguais, uma vez que* $\Gamma_{1,1} = \Gamma_{1,3}$*, e portanto não é necessária a renomeação deste evento.*

As estruturas de controle para o bloco X_0 *serão* $\Gamma_{0,0} = \Gamma_{0,4} = \{(\emptyset, M), (\{a\}, M)\}$

O autômato agregado com estruturas de controle dependentes do estado é mostrado na Figura 6.10



$\Gamma_{Z_0} = \{(\emptyset, M), (\{a\}, M)\}$

$\Gamma_{Z_1} = \{(\{b\}, N), (\{b'\}, N), (\{b, b'\}, N)\}$

Figura 6.10: Agregação com estruturas de controle dependentes do estado

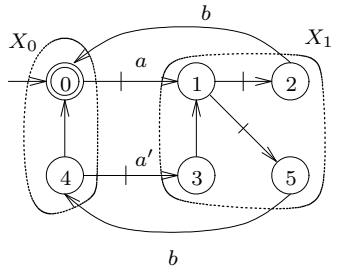
Observa-se que a renomeação do evento "b" pode ser simplificada da mesma forma que no caso da Figura 6.5(c), obtendo-se assim o mesmo resultado da Figura 6.5(d).

Caso B *Iniciando a análise pelo bloco* X_0.

No bloco X_0 *renomeiam-se as transições com eventos que produzem não determinismo e obtêm-se as estruturas de controle.*

Observa-se que após o bloco X_1 *as continuações das transições com o evento "b" não são iguais, pois* $\Gamma_{0,0} \neq \Gamma_{0,1}$*, e portanto este evento precisa de renomeação.*

Calculam-se as estruturas de controle para X_1 *(*$\Gamma_{1,1} = \Gamma_{1,3} = \{(\{b\}, N), (\{b'\}, N), (\{b, b'\}, N)\}$*).*

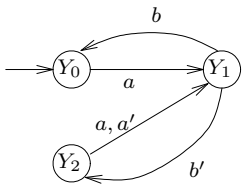


$\Gamma_{0,0} = \{(\emptyset, M), (\{a\}, M)\}$

$\Gamma_{0,4} = \{(\emptyset, M), (\{a\}, M), (\{a'\}, M), (\{a, a'\}, M)\}$

Figura 6.11: Renomeação dos eventos de saída do bloco X_0

Posteriormente obtém-se o autômato agregado com estruturas de controle dependentes do estado (Fig. 6.12).



$\Gamma_{Y_0} = \Gamma_{0,0} = \{(\emptyset, M), (\{a\}, M)\}$

$\Gamma_{Y_2} = \Gamma_{0,4} = \{(\emptyset, M), (\{a\}, M), (\{a'\}, M), (\{a, a'\}, M)\}$

$\Gamma_{Y_1} = \Gamma_{1,1} = \Gamma_{1,3} = \{(\{b\}, N), (\{b'\}, N), (\{b, b'\}, N)\}$

Figura 6.12: Agregação com estruturas de controle dependentes do estado

Observa-se que esta figura corresponde ao caso da Figura 6.5(a). O procedimento deve ser repetido obtendo-se desta forma o autômato reduzido conforme Figura 6.5(d).

Observa-se que dependendo da escolha do bloco inicial, este último método possibilita a diminuição do trabalho de cálculo das estruturas de controle, como também a diminuição do número de padrões de controle. Na pior das hipóteses este método leva a um resultado igual ao do algoritmo 6.2, em que todos os eventos que produzem não determinismo são renomeados.

Recomenda-se inicialmente identificar os blocos independentes de renomeação (aqueles que não tem como saída eventos que produzem não determinismo). Assim poderão ser evitados vários passos do algoritmo.

6.3 Considerações sobre os métodos apresentados

Uma das questões que poderia surgir em relação aos métodos apresentados é se o modelo agregado reduzido mantém a propriedade de consistência hierárquica forte.

Considerando que o autômato agregado antes da sua redução leva à consistência hierárquica forte, o teorema a seguir garante que o modelo agregado e reduzido mantém esta propriedade.

Teorema 6.1 *O autômato agregado e reduzido mantém a propriedade de consistência hierárquica forte.*

Prova: Na hipótese de que o autômato agregado antes da sua redução leva à consistência hierárquica forte, o procedimento de redução resume-se a duas etapas: a simplificação de estados e a simplificação de multiplicidade de transições com eventos renomeados.

- A simplificação de estados não altera a linguagem em malha aberta do autômato agregado, uma vez que para esta simplificação é utilizada a equivalência de Nerode. Por outro lado, não altera as estruturas de controle de cada estado, porque dois estados do nível agregado são considerados equivalentes se além da equivalência de Nerode têm estruturas de controle iguais.

- A simplificação de multiplicidade de transições com eventos renomeados simplesmente elimina transições redundantes de uma mesma instância de evento que têm continuações iguais.

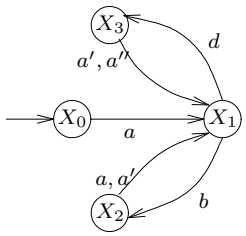
♢

Outras questões que poderiam surgir em relação aos dois algoritmos apresentados anteriormente são quanto à convergência e ao fato de o resultado obtido ser mínimo.

A convergência nos dois métodos pode ser verificada facilmente, uma vez que em cada ciclo do algoritmo é reduzido pelo menos um estado ou uma transição. No pior dos casos, os algoritmos convergem num número de passos igual ao número de transições do autômato inicial agregado com estruturas de controle dependentes do estado.

O resultado obtido com os algoritmos anteriores não necessariamente será o mínimo, uma vez que para eliminar a multiplicidade dos eventos renomeados aparece um conflito a ser resolvido. Para ilustração deste problema considera-se o autômato agregado com estruturas de controle dependentes do estado apresentado na Figura 6.13.

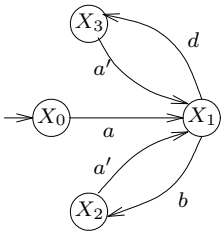
Sejam os eventos a, a', a'' instâncias renomeadas do evento a. As transições do estado X_2 com a, a' para o estado X_1 deverão ser simplificadas a uma única transição para evitar redundância. Esta simplificação implica chamar o evento a de a' ou vice-versa.



$\Gamma_{X_0} = \{(\emptyset, M), (\{a\}, M)\}$
$\Gamma_{X_1} = \{(\{b, d\}, N)\}$
$\Gamma_{X_2} = \{(\{a\}, M), (\{a'\}, M), (\{a, a'\}, M)\}$
$\Gamma_{X_3} = \{(\{a', a''\}, M)\}$

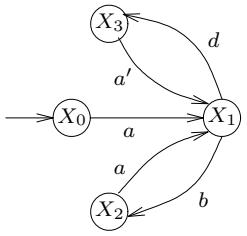
Figura 6.13: Autômato agregado Γ dependente do estado.

A mesma situação acontece para as transições do estado X_3 com a', a'' para o estado X_1. Dependendo da escolha da nomeação, o autômato agregado poderá ser reduzido ou não. Na Figura 6.14 apresentam-se duas situações de nomeação. No primeiro caso (Fig. 6.14(a)) o autômato ainda pode ser reduzido, uma vez que os estados X_2 e X_3 chegam a ser equivalentes, pois pertencem a uma mesma classe de equivalência de Nerode e têm estruturas de controle iguais. No segundo caso (Fig. 6.14(b)) o autômato não poderá ser mais reduzido, pois não existem estados equivalentes. Este tipo de conflito a princípio é resolvido escolhendo a nomeação de maneira aleatória.



$\Gamma_{X_0} = \{(\emptyset, M), (\{a\}, M)\}$
$\Gamma_{X_1} = \{(\{b, d\}, N)\}$
$\Gamma_{X_2} = \{(\{a'\}, M)\}$
$\Gamma_{X_3} = \{(\{a'\}, M)\}$

(a) Autômato redutível

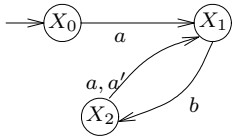


$\Gamma_{X_0} = \{(\emptyset, M), (\{a\}, M)\}$
$\Gamma_{X_1} = \{(\{b, d\}, N)\}$
$\Gamma_{X_2} = \{(\{a\}, M)\}$
$\Gamma_{X_3} = \{(\{a'\}, M)\}$

(b) Autômato não redutível

Figura 6.14: Simplificação de multiplicidade do evento "a"

Em algumas situações a nomeação é mais direta. Considerando o caso apresentado na Figura 6.15, as transições com a, a' de X_2 para X_1 serão reduzidas simplesmente para a, uma vez que o objetivo é buscar a equivalência entre os estados X_0 e X_2.



$\Gamma_{X_0} = \{(\emptyset, M), (\{a\}, M)\}$
$\Gamma_{X_1} = \{(\{b, d\}, N)\}$
$\Gamma_{X_2} = \{(\{a\}, M), (\{a'\}, M), (\{a, a'\}, M)\}$

Figura 6.15: Simplificação do evento "a"

Caso numa agregação não existam estes tipos de conflitos a serem resolvidos, a

redução do autômato chegará ao mínimo possível.

6.4 Conclusão

Neste capítulo desenvolveu-se uma extensão do modelo agregado com estruturas de controle dependentes do estado apresentado no capítulo 5. Foram desenvolvidos dois métodos para redução do autômato agregado tal que o resultado tenha um número reduzido de estados e renomeações. Um ponto relevante é o fato de que o modelo agregado reduzido mantém a propriedade de consistência hierárquica forte.

A simplificação da renomeação de alguns eventos, implicitamente diminui exponencialmente o número de padrões de controle dentro da estrutura de controle.

Para consolidar os resultados obtidos, implementou-se uma ferramenta computacional com os algoritmos propostos. A ferramenta foi desenvolvida em linguagem C++ tendo como base a ferramenta *Grail* (Raymond e Wood 1996), a qual apresenta uma biblioteca de funções para manipular autômatos.

Capítulo 7

Controle Supervisório Hierárquico Modular por Agregação de Estados

Os controles hierárquico e modular são opções para manipulação de problemas complexos através, respectivamente, da decomposição vertical e horizontal do problema em sub-problemas menores, para posteriormente montar as suas soluções numa estrutura hierárquica ou modular.

As considerações sobre controle hierárquico foram introduzidas detalhadamente nos capítulos 3 e 5. O controle modular de SEDs foi inicialmente proposto por Wonham e Ramadge (1988). A síntese modular permite que problemas complexos possam ser decompostos em módulos mais simples. Enquanto supervisores são projetados para serem não bloqueantes (em relação à planta) quando operam individualmente, não está garantida que as suas combinações também o serão. Por isto é necessário verificar que a síntese dos componentes sejam não conflitantes.

Neste capítulo propõe-se uma combinação da arquitetura de controle hierárquico e modular envolvendo as abordagens desenvolvidas no capítulo 5 (Torrico e Cury 2002c) e (Wonham e Ramadge 1988). À decomposição hierárquica (decomposição vertical) proposta no capítulo 5, adiciona-se a decomposição modular (decomposição horizontal) de Wonham e Ramadge (1988). O objetivo deste capítulo é propor uma arquitetura hierárquica modular baseada na agregação de estados, de forma que esta arquitetura permita um comportamento consistente entre os níveis de hierarquia e que a ação conjunta de vários controladores gerenciais implementados no baixo nível sejam não bloqueantes.

7.1 O problema de controle hierárquico modular

Considera-se uma planta G, um modelo abstrato G^A, dois controladores f_i^A, $i = 1, 2$ para G^A, um controlador f para G, e um *operador de conjunção* ($\textcircled{\cap}$), conectados tal como mostrado na Figura 7.1. G é a planta a ser controlada no mundo real por um operador f, enquanto que G^A é uma abstração de G obtida pela agregação de seus estados que será controlada pela ação conjunta de dois gerentes f_i^A, $i = 1, 2$. A planta G informa ao nível agregado somente alguns eventos considerados relevantes ($\sigma \in \Sigma^A$), atualizando o modelo G^A, mas também informa ao operador f a ocorrência de qualquer evento $\lambda \in \Sigma$. A cada ocorrência de um evento $\sigma \in \Sigma^A$, cada gerente f_i^A, $i = 1, 2$ envia um comando com_i ao *operador de conjunção* que processa os dois comandos com_i, $i = 1, 2$ e os sintetiza num único comando com a ser implementado pelo operador f. O operador f recebe duas informações, o comando com do operador de conjunção e a ocorrência de eventos $\lambda \in \Sigma$ gerados pela planta G, e com estes dados envia uma entrada de controle Con para a planta G. A entrada de controle Con_i^A resulta ser uma entrada de controle virtual, uma vez que o comportamento de G^A é determinado totalmente pelo comportamento de G.

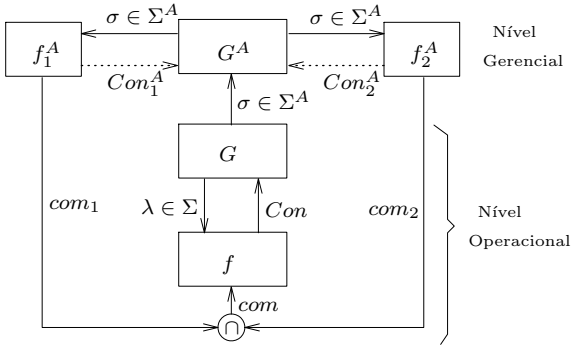


Figura 7.1: Arquitetura do controle hierárquico modular.

Sejam L_m e L linguagens definidas sobre um alfabeto Σ com $L = \overline{L_m}$, onde L e L_m representam respectivamente as linguagens gerada e marcada de uma planta G. G^A representa o modelo abstraído de G obtido pela agregação dos seus estados como no capítulo 5. Σ^A é o alfabeto de eventos para G^A, onde $\Sigma^A \subseteq \Sigma$.

Deseja-se que para duas especificações realizáveis de alto nível a arquitetura proposta possua um comportamento consistente. Assim para $i = 1, 2$, sejam $E_i^A \subseteq L(G^A)$ especificações sobre G^A. A máxima linguagem Γ-compatível sobre $L(G^A)$ para a especificação E_i^A será $\sup \mathcal{CM}(E_i^A, L(G^A)) \subseteq E_i^A \subseteq L(G^A)$ (Cury et al. 2001). Define-se a *modularidade hierárquica*, como a propriedade verificada quando para dois superviso-

res do nível gerencial f_1^A e f_2^A, tais que, $L_m(f_i^A/G^A) = \sup\mathcal{CM}(E_i^A, L(G^A))$, $i = 1, 2$, $\Theta(L_m(f/G)) = \sup\mathcal{CM}(E_1^A \cap E_2^A, L(G^A))$ com $\overline{L_m(f/G)} = L(f/G)$. Esta condição indica que a imagem da ação do supervisor f sobre a planta G será igual à implementação monolítica no alto nível, ou seja, aquela que seria obtida resolvendo-se o problema de síntese de controle para a especificação $E_1^A \cap E_2^A$. O problema de controle é obter condições necessárias e suficientes para modularidade hierárquica.

7.2 Modelo para controle hierárquico modular

Nesta seção serão detalhados o papel do operador de conjunção e o supervisor f.

O operador de conjunção é um elemento que se encarrega de processar os comandos com_1 e com_2 vindos de f_1^A e f_2^A respectivamente e traduzir num comando com.

Funcionamento do *operador de conjunção.-*

Na implementação on-line o *operador de conjunção* terá a seguinte função:

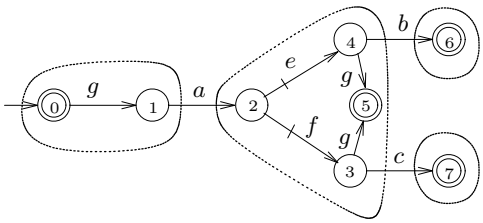
Para cada bloco atingido $X_i \in \pi$ de G^A.

- Se $com_1 = (\gamma_1, N) \in \Gamma_i$ e $com_2 = (\gamma_2, \#) \in \Gamma_i$, para $\# = M, N$, então, $com = ([\gamma_1 \cap \gamma_2], N)$ se $com \in \Gamma_i$, senão com =indefinido!
- Se $com_1 = (\gamma_1, M) \in \Gamma_i$ e $com_2 = (\gamma_2, M) \in \Gamma_i$, então, $com = ([\gamma_1 \cap \gamma_2], M)$ se $com \in \Gamma_i$, senão com =indefinido!

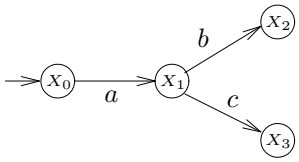
A implementação conjunta de f_1^A e f_2^A no baixo nível é realizada pelo supervisor f que interpreta o comando com como no caso do controle hierárquico por agregação de estados apresentado no capítulo 5 (Torrico e Cury 2002c) no qual se tem um único supervisor no nível agregado.

Observa-se que para um funcionamento adequado do supervisor f, o comando com sempre terá que existir dentro da estrutura de controle ativa. Caso não exista então não existe um supervisor f consistente, no sentido de processar o comando de entrada com. O exemplo a seguir ilustra esta situação.

Exemplo 7.1 *Considera-se a planta mostrada na Figura 7.2(a); na Figura 7.2(b) apresenta-se o correspondente modelo agregado com estruturas de controle dependentes do estado.*



(a) Planta



$\Gamma_{X_0} = \{(\{a\}, M)\}$
$\Gamma_{X_1} = \{(\{b\}, M), (\{c\}, M), (\{b, c\}, M)\}$
$\Gamma_{X_2} = \Gamma_{X_3} = \{(\emptyset, M)\}$

(b) Autômato agregado com Γ dependente do estado

Figura 7.2: Caso de estudo

Para as especificações E_1^A e E_2^A do alto nível apresentadas na Figura 7.3 observa-se que as mesmas são Γ-compatíveis. Portanto estes autômatos podem ser usados como supervisores f_1^A, f_2^A, tais que $L_m(f_1^A/G^A) = E_1^A$ e $L_m(f_2^A/G^A) = E_2^A$. Observa-se que as linguagens realizáveis no nível agregado são modulares, isto é, $\overline{E_1^A} \cap \overline{E_2^A} = \overline{E_1^A \cap E_2^A}$.

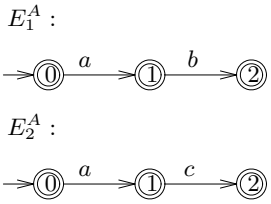


Figura 7.3: Especificações

Neste caso, quando o sistema alcança o bloco X_1, f_1^A envia a entrada de controle $com_1 = (\{b\}, M)$ e f_2^A a entrada de controle $com_2 = (\{c\}, M)$. Para um comportamento consistente teria que existir a entrada de controle $(\emptyset, M)$, permitindo desabilitar os eventos b e c simultaneamente.

No exemplo anterior também é mostrado que a condição de modularidade de linguagens realizáveis no nível agregado, não é suficiente para garantir uma realização não bloqueante.

Para garantir a existência do supervisor f, o teorema a seguir apresenta as condições para que isto seja possível.

Teorema 7.1 *Dados dois supervisores* f_1^A *e* f_2^A *tais que* $L_m(f_1^A/G^A) = K_1^A$, $L_m(f_2^A/G^A) = K_2^A$, *com* $\overline{K_1^A} \cap \overline{K_2^A} = \overline{K_1^A \cap K_2^A}$. *A* Γ*-compatibilidade de* $K_1^A \cap K_2^A$ *é condição necessária e suficiente para a modularidade hierárquica.*

Prova:

(Somente se)

Sejam K_1^A e K_2^A duas linguagens tais que $L_m(f_1^A/G^A) = K_1^A$ e $L_m(f_2^A/G^A) = K_2^A$. Por hipótese $\overline{K_1^A} \cap \overline{K_2^A} = \overline{K_1^A \cap K_2^A}$

Para qualquer $s \in L(f_1^A \circledcap f_2^A/G^A)$ tal que $\delta^A(q_0^A, s) = X_i \in \pi$, sejam $com_1 = (\gamma_1, \#_1) \in \Gamma_i$ e $com_2 = (\gamma_2, \#_2) \in \Gamma_i$, entradas de controle aplicadas pelos supervisores f_1^A e f_2^A respectivamente.

Por Γ-compatibilidade de $K_1^A \cap K_2^A$, para qualquer bloco $X_i \in \pi$ atingido em G^A, existe uma entrada de controle no alto nível $(\gamma, \#) \in \Gamma_i$, esta entrada de controle pode ser interpretada como a ação de um supervisor equivalente f_e^A tal que $L_m(f_e^A/G^A) = K_1^A \cap K_2^A$, que garante uma realização controlável e não bloqueante no baixo nível, seja $L_m(f/G)$ a implementação de $K_1^A \cap K_2^A$ no baixo nível, segundo o Teorema 5.2 $\theta(L_m(f/G)) = K_1^A \cap K_2^A$

(Se)

Pelo Teorema 5.3, $\theta(L_m(f/G))$ é Γ-compatível.

$\diamondsuit$

O teorema anterior garante que para cada cadeia de eventos da intersecção no alto nível existe uma entrada de controle que permite uma realização não bloqueante no baixo nível.

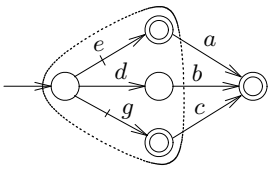
Existem condições suficientes para modularidade hierárquica em que o teste da Γ compatibilidade não será necessário. Desta forma são introduzidas algumas definições apresentadas a seguir.

Definição 7.1 Fechamento de conjunção das estruturas de controle de G^A

A estrutura de controle de G^A *é dita ser fechada para a conjunção se*

1. $(\gamma_1, N), (\gamma_2, \#) \in \Gamma_i \rightarrow \exists (\gamma_3, N) \in \Gamma_i$ *tal que* $\gamma_3 = \gamma_1 \cap \gamma_2$ *para* $\# = M, N$
2. $(\gamma_1, M), (\gamma_2, M) \in \Gamma_i \rightarrow \exists (\gamma_3, M) \in \Gamma_i$ *tal que* $\gamma_3 = \gamma_1 \cap \gamma_2$

Exemplo 7.2 *Análise de um bloco* X_i.



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

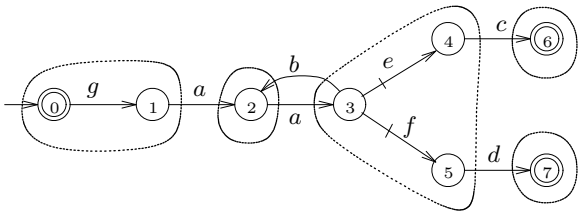
Figura 7.4: Análise de um bloco

Para a agregação mostrada na Figura 7.4, observe que as estruturas de controle não satisfazem a condição de fechamento de conjunção, pois:

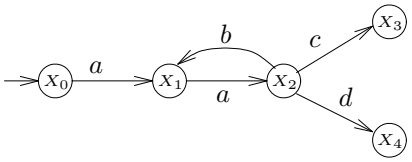
$(\{a, b\}, M), (\{b, c\}, M) \in \Gamma_i$, *entretanto* $(\{b\}, M) \notin \Gamma_i$

Dados dois supervisores f_1^A e f_2^A tal que $L_m(f_1^A/G^A) = K_1^A$ e $L_m(f_2^A/G^A) = K_2^A$, o fechamento para conjunção não garante o não bloqueio da ação conjunta de ambos supervisores, conforme mostra o contra-exemplo a seguir:

Exemplo 7.3 *Seja a planta mostrada na Figura 7.5(a); na Figura 7.5(b) apresenta-se o correspondente modelo agregado com estruturas de controle dependentes do estado .*



(a) Planta

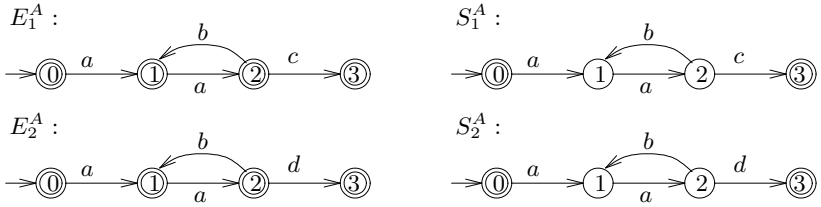


$\Gamma_{X_0} = \{(\{a\}, M)\}$
$\Gamma_{X_1} = \{(\{a\}, N)\}$
$\Gamma_{X_2} = \{(\{b\}, N), (\{b, c\}, N), (\{b, d\}, N), (\{b, c, d\}, N)\}$
$\Gamma_{X_3} = \Gamma_{X_4} = \{(\emptyset, M)\}$

(b) Autômato agregado com Γ dependente do estado

Figura 7.5: Caso de estudo

Para as especificações E_1^A e E_2^A do alto nível apresentadas na Figura 7.6(a), na Figura 7.6(b) apresentam-se os autômatos das respectivas máximas linguagens Γ-compatíveis. Observa-se que as estruturas de controle são fechadas para conjunção, então para cada cadeia no nível agregado observada por f_1^A e f_2^A existe um comando "com" resultante da conjunção entre "com$_1$" e "com$_2$" que permite a evolução do sistema.



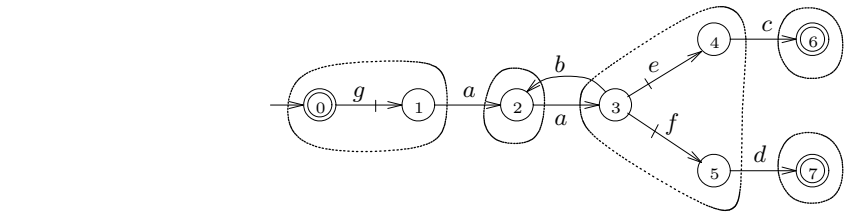
(a) Especificações (b) Máxima linguagem realizável

Figura 7.6: Especificações e máximas realizações

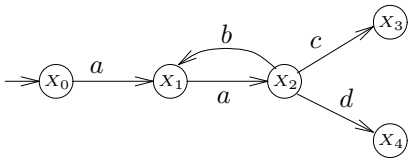
No caso, quando o sistema alcançar o bloco X_2, f_1^A envia a entrada de controle $com_1 = (\{b, c\}, N)$ e f_2^A a entrada de controle $com_2 = (\{b, d\}, N)$, como resultado da conjunção tem-se $com = (\{b\}, N)$, que desabilita os eventos c e d; então observa-se que o sistema no nível agregado ficará bloqueado entre os estados X_1 e X_2 sem poder completar nenhuma tarefa, correspondendo no baixo nível a um bloqueio nos estados 2 e 3.

O fechamento para conjunção das estruturas de controle não garante o não bloqueio mesmo que a intersecção entre as linguagens realizáveis do nível agregado seja Γ-compatível. O exemplo a seguir ilustra este caso.

Exemplo 7.4 *Considera-se a agregação da Figura 7.7.*



(a) Planta



$\Gamma_{X_0} = \{(\emptyset, M), (\{a\}, M)\}$
$\Gamma_{X_1} = \{(\{a\}, N)\}$
$\Gamma_{X_2} = \{(\{b\}, N), (\{b, c\}, N), (\{b, d\}, N), (\{b, c, d\}, N)\}$
$\Gamma_{X_3} = \Gamma_{X_4} = \{(\emptyset, M)\}$

(b) Autômato agregado com Γ dependente do estado

Figura 7.7: Caso de estudo

Para as mesmas especificações E_1 e E_2 do alto nível apresentadas na Figura 7.6(a), neste caso os autômatos das respectivas máximas linguagens Γ-compatíveis resultam

iguais aos da Figura 7.6(b). Ao contrário do exemplo anterior (Exemplo 7.3), verifica-se que a intersecção entre as máximas linguagens Γ*-compatíveis é* Γ*-compatível, mas isto não implica que estas linguagens serão modulares, isto é,* $\overline{L_m(S_1)} \cap \overline{L_m(S_2)} \neq \overline{L_m(S_1) \cap L_m(S_2)}$*. Portanto acontece o mesmo problema que no exemplo anterior.*

O teorema a seguir garante que, dado um modelo abstrato com estruturas de controle fechada para conjunção, a intersecção de duas especificações realizáveis e modulares no alto nível é Γ-compatível

Teorema 7.2 *Dados dois supervisores* f_1^A *e* f_2^A *tais que* $L_m(f_1^A/G^A) = K_1^A$, $L_m(f_2^A/G^A) = K_2^A$, *com* $\overline{K_1^A} \cap \overline{K_2^A} = \overline{K_1^A \cap K_2^A}$. *O fechamento de conjunção das estruturas de controle do nível gerencial implica que* $K_1^A \cap K_2^A$ *é* Γ*-compatível.*

Prova:

Sejam $E_1^A, E_2^A \subseteq L(G^A)$ duas linguagens quaisquer de alto nível, tais que $\sup\mathcal{CM}(E_1^A, L(G^A)) = K_1^A$ e $\sup\mathcal{CM}(E_2^A, L(G^A)) = K_2^A$. Por hipótese $\overline{K_1^A} \cap \overline{K_2^A} = \overline{K_1^A \cap K_2^A}$

Denota-se $L(f_1^A \conj f_2^A/G^A)$ como a implementação conjunta dos supervisores do alto nível. Para qualquer $s \in L(f_1^A \conj f_2^A/G^A)$ tal que $\delta^A(q_0^A, s) = X_i \in \pi$, sejam, $com_1 = (\gamma_1, \#_1) \in \Gamma_i$ e $com_2 = (\gamma_2, \#_2) \in \Gamma_i$, entradas de controle aplicadas pelos supervisores de alto nível. Por fechamento para conjunção das estruturas de controle $\exists\ com = (\gamma_3, \#_3) \in \Gamma_i$, tal que $\gamma_3 = \gamma_1 \cap \gamma_2$ e $\#_3 = M$ se $\#_1 = M$ e $\#_2 = M$, outro caso $\#_3 = N$.

Então o comando *com* pode ser interpretado como a ação de um supervisor equivalente f_e^A tal que $L_m(f_e^A/G^A) = L_m(f_1^A \conj f_2^A/G^A) = \sup\mathcal{CM}(E_1^A, L(G^A)) \cap \sup\mathcal{CM}(E_2^A, L(G^A))$.

Seja $s \in K_1^A \cap K_2^A$; obviamente $s \in K_1^A$ e $s \in K_2^A$, portanto $\exists(\gamma_1, M), (\gamma_2, M)$ tais que $\gamma_j \cap \Sigma_{L(G^A)}(s) = \Sigma_{K_j^A}(s)$ para $j = 1, 2$. Tem-se $\gamma_1 \cap \gamma_2 \cap \Sigma_{L(G^A)}(s) = \Sigma_{K_1^A}(s) \cap \Sigma_{K_2^A}(s) = \Sigma_{\overline{K_1^A} \cap \overline{K_2^A}}(s) = \Sigma_{\overline{K_1^A \cap K_2^A}}(s) = \Sigma_{K_1^A \cap K_2^A}(s)$. Pelo fechamento para conjunção $(\gamma_1 \cap \gamma_2, M) \in \Gamma(s)$.

Seja agora $s \in \overline{K_1^A} - K_1^A$ e $s \in \overline{K_2^A}$; pode-se facilmente deduzir que $s \in \overline{K_1^A} \cap \overline{K_2^A}$ e $s \notin K_1^A \cap K_2^A$, ou seja, $s \in \overline{K_1^A} \cap \overline{K_2^A} - K_1^A \cap K_2^A = \overline{K_1^A \cap K_2^A} - K_1^A \cap K_2^A$. Nesse caso $\exists(\gamma_1, N), (\gamma_2, \#)$ tais que $\gamma_j \cap \Sigma_{L(G^A)}(s) = \Sigma_{K_j^A}(s)$ para $j = 1, 2$. Pelo fechamento para conjunção $(\gamma_1 \cap \gamma_2, N) \in \Gamma(s)$. Raciocínio similar para $s \in \overline{K_2^A} - K_2^A$ e $s \in \overline{K_1^A}$ permite concluir que $K_1^A \cap K_2^A$ é Γ-compatível. Portanto, $K_1^A \cap K_2^A \subseteq \sup\mathcal{CM}(E_1^A \cap E_2^A, L(G^A))$.

Para a inclusão reversa usamos o fato que $\sup\mathcal{CM}$ é monótono i.e., se $E \subseteq E'$, $\sup\mathcal{CM}(E, L)) \subseteq \sup\mathcal{CM}(E', L)$. Como $E_1^A \cap E_2^A \subseteq E_1^A$, isto implica que $\sup\mathcal{CM}(E_1^A \cap E_2^A, L(G^A)) \subseteq K_1^A$, similarmente $\sup\mathcal{CM}(E_1^A \cap E_2^A, L(G^A)) \subseteq K_2^A)$. Assim, $K_1^A \cap K_2^A \supseteq \sup\mathcal{CM}(E_1^A \cap E_2^A, L(G^A))$

Portanto $K_1^A \cap K_2^A = \sup \mathcal{CM}(E_1^A \cap E_2^A, L(G^A))$

$\diamondsuit$

Decorrente deste teorema, o corolário a seguir garante que, dado um modelo abstrato com estruturas de controle fechada para conjunção, duas especificações realizáveis e modulares no alto nível poderão ser implementadas no baixo nível sem nenhum conflito ou perda de otimalidade.

Corolário 7.1 *Dados dois supervisores* f_1^A *e* f_2^A *tais que* $L_m(f_1^A/G^A) = K_1^A$, $L_m(f_2^A/G^A) = K_2^A$, *com* $\overline{K_1^A} \cap \overline{K_2^A} = \overline{K_1^A \cap K_2^A}$. *O fechamento de conjunção das estruturas de controle do nível gerencial é condição suficiente para que a modularidade hierárquica seja satisfeita.*

Prova:

Pelo teorema 7.2 sabe-se que $K_1^A \cap K_2^A$ é Γ-compatível, então é possível uma implementação $L_m(f/G)$ realizável e não bloqueante no baixo nível, segundo o Teorema 5.2 $\theta(L_m(f/G)) = \sup \mathcal{CM}(E_1^A \cap E_2^A, L(G^A))$

$\diamondsuit$

Os resultados obtidos serão exemplificados na seção a seguir.

7.3 Exemplo de aplicação

Esta abordagem será ilustrada pelo desenvolvimento de vários supervisores hierárquicos para a linha de transferência (Fig. 7.8) apresentada na seção 5.5.

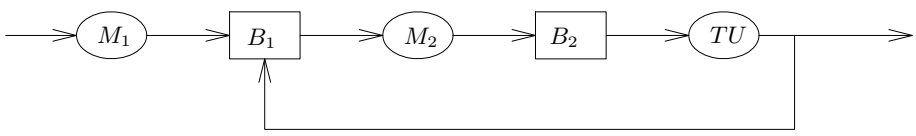
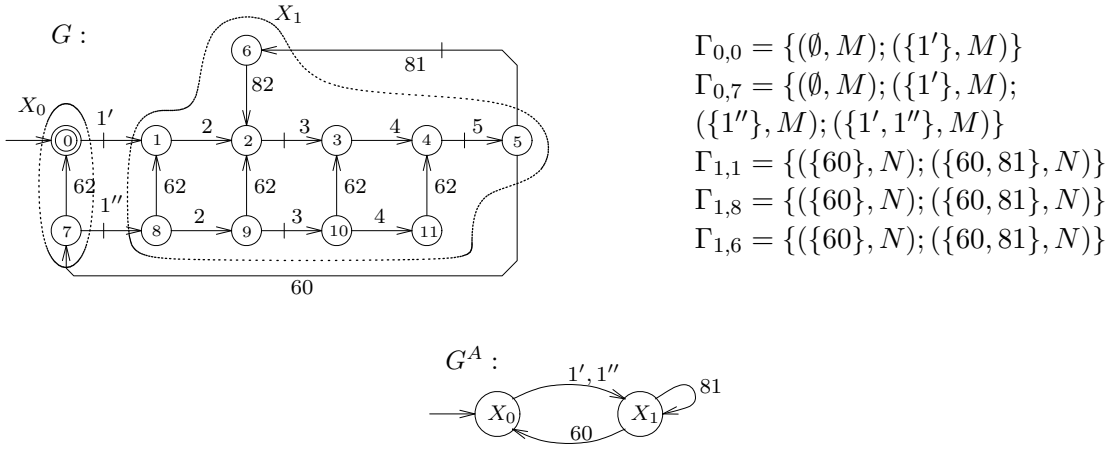


Figura 7.8: Linha de Transferência.

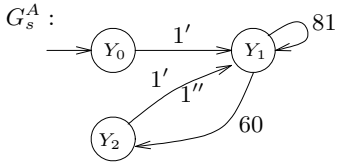
Para ilustrar o problema de controle hierárquico modular parte-se do mesmo modelo agregado obtido na seção 5.5.1(Figuras 7.9 e 7.10).

No alto nível, a preocupação dos gerentes continua sendo o controle da relação entre peças de entrada e peças aceitas ou recusadas. Portanto, os eventos $1, 60$ e 81 foram



$\Gamma_{0,0} = \{(\emptyset, M); (\{1'\}, M)\}$
$\Gamma_{0,7} = \{(\emptyset, M); (\{1'\}, M);$
$(\{1''\}, M); (\{1', 1''\}, M)\}$
$\Gamma_{1,1} = \{(\{60\}, N); (\{60, 81\}, N)\}$
$\Gamma_{1,8} = \{(\{60\}, N); (\{60, 81\}, N)\}$
$\Gamma_{1,6} = \{(\{60\}, N); (\{60, 81\}, N)\}$

Figura 7.9: Planta e sua agregação.



$\Gamma(Y_0) = \Gamma_{0,0}$
$\Gamma(Y_1) = \Gamma_{1,1} = \Gamma_{1,8} = \Gamma_{1,6}$
$\Gamma(Y_2) = \Gamma_{0,7}$

Figura 7.10: Planta hierárquica com estruturas de controle dependentes do estado.

considerados como relevantes para especificação. Cabe lembrar que os eventos $1'$ e $1''$ são instâncias do evento 1, os quais foram divididos por causa do não determinismo.

Pode-se observar que as estruturas de controle de G_s^A são fechadas para conjunção.

De modo a ilustrar a condição de modularidade hierárquica consideram-se duas especificações para o sistema (Fig 7.11). A primeira especificação (E_1) indica que, toda vez que uma mesma peça for recusada duas vezes seguidas pela unidade de teste, o sistema deve parar. A segunda especificação (E_2) indica que se duas peças contíguas que entraram no sistema forem recusadas pela unidade de teste mesmo que seja só por uma vez, então uma parada do sistema é também forçada.

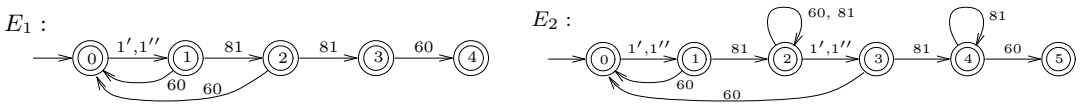


Figura 7.11: Especificações.

Continuando, obtêm-se as linguagens alvo e aplica-se o Algoritmo para síntese da máxima linguagem Γ-*compatível* apresentado para sistemas com estruturas de controle dependentes do estado (Algoritmo 4.1). Os resultado são mostrados na Figura 7.12.

Sejam $\sup \mathcal{CM}(E_1||L(G_s^A)) = K_1^A$ e $\sup \mathcal{CM}(E_2||L(G_s^A)) = K_2^A$, pode ser verificada

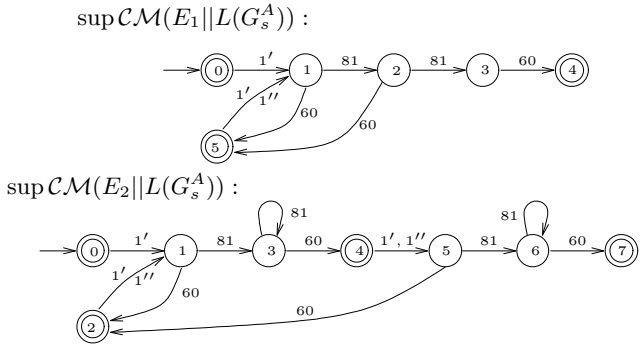


Figura 7.12: Máximas linguagens Γ-*compatíveis*.

a modularidade de alto nível, i.e. $\overline{K_1^A} \cap \overline{K_2^A} = \overline{K_1^A \cap K_2^A}$, conseqüentemente é possível uma implementação realizável e não bloqueante no baixo nível. A seguir mostra-se uma parte da implementação no baixo nível resultante da ação conjunta de ambos supervisores de alto nível.

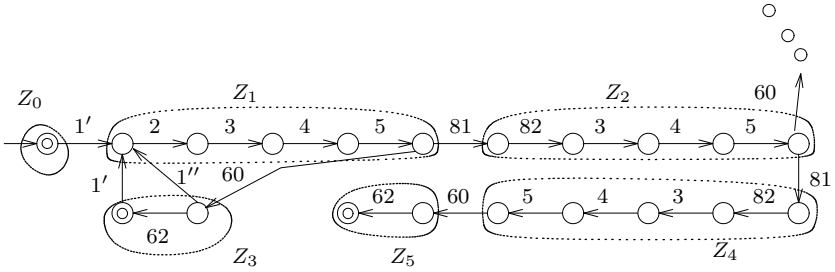


Figura 7.13: Implementação no baixo nível

A função do operador de conjunção será interpretar as entradas de controle vindos de ambos supervisores f_1^A e f_2^A e traduzir num comando único. Para o caso do exemplo, no bloco Z_5 (Bloco Y_2 atingido em G_s^A) da implementação, o supervisor f_1^A aplica a entrada de controle $(\emptyset, M)$ e o supervisor f_2^A aplica a entrada de controle $(\{1', 1"\}, M)$, finalmente o operador de conjunção traduz as duas entradas e manda como resposta a entrada de controle $(\emptyset, M)$.

7.4 Conclusão

O controle hierárquico modular foi aplicado com sucesso ao modelo proposto. Neste modelo a condição de fechamento para conjunção das estruturas de controle e a modularidade de alto nível garantem a existência de um supervisor equivalente à ação conjunta de dois supervisores no alto nível, permitindo assim um comportamento consistente entre os níveis de hierarquia e uma implementação não bloqueante no baixo nível. A abordagem proposta combina os ganhos obtidos pela agregação e

pela modularidade.

Capítulo 8

Exemplo de aplicação

Neste capítulo apresenta-se uma aplicação das abordagens desenvolvidas nesta tese a uma célula de manufatura. O exemplo foi desenvolvido utilizando-se as funções implementadas como extensão da ferramenta computacional *Grail* para manipulação de máquinas de estado finito.

8.1 Descrição do problema

Para descrição do problema considera-se a célula de manufatura apresentada na Figura 8.1. Esta célula é composta por uma mesa giratória de três posições, uma esteira de entrada, uma máquina que fura e testa as peças, e finalmente um manipulador robótico que classifica as peças boas e ruins. O funcionamento da célula é comandado por um controlador lógico programável conforme a seguinte seqüência de passos:

- a esteira gira até que uma peça seja posicionada em P_1;
- a mesa gira 120°;
- a peça é furada e testada;
- a mesa gira 120°;
- o robô retira a peça da mesa para um dos *buffers* de saída.

A célula foi inicialmente projetada para operar em seqüência apenas uma peça por vez, ou seja, a esteira só pode ser acionada para alimentação de nova peça depois que

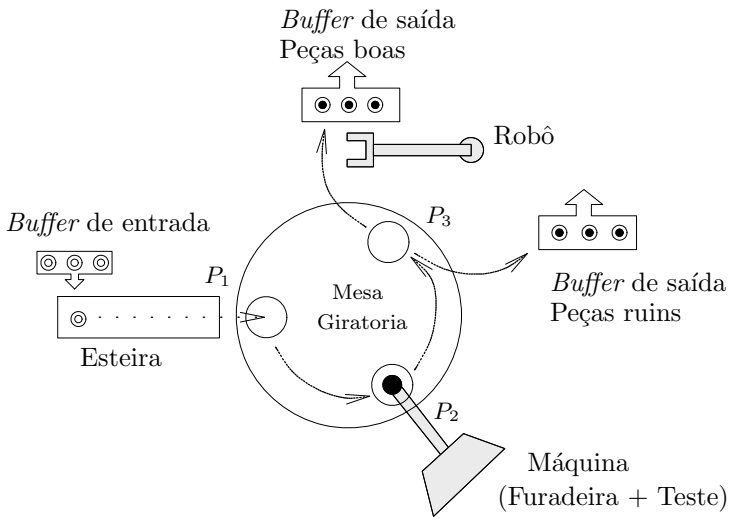


Figura 8.1: Célula de manufatura

o robô retirar a peça anterior da mesa. Esta restrição da lógica de controle evita os problemas que podem ocorrer na operação de múltiplas peças em paralelo, entre os quais se destacam: acionar a esteira, a máquina ou o robô enquanto a mesa girar; sobrepor peças na posição P_1; girar a mesa sem que as peças em P_2 e P_3 tenham sido trabalhadas ou retiradas respectivamente; acionar a máquina ou o robô sem peças nas posições P_2 e P_3 respectivamente; trabalhar duas vezes a mesma peça; girar a mesa sem nenhuma peça.

Entretanto, esse modo de funcionamento é pouco eficiente, visto que a esteira, a máquina e o robô passam a maior parte do tempo parados enquanto poderiam estar operando em paralelo. Em (de Queiroz e Cury 2001), este problema foi resolvido de uma forma minimamente restritiva para uma mesa similar, utilizando-se uma abordagem de controle modular. O objetivo deste capítulo é resolver o mesmo problema em dois níveis de abstração, nível operacional e agregado: no nível operacional serão resolvidos os problemas operacionais descritos anteriormente; no nível agregado resolve-se o problema da classificação das peças boas e ruins. O critério utilizado para separação das tarefas para o alto nível está associado com as possíveis tarefas sujeitas a mudanças. Neste caso, se a especificação implementada no alto nível fosse parar o sistema quando uma peça ruim for detectada, para maior continuidade na produção, a especificação poderia ser alterada para alguma outra estratégia de parada, por exemplo, parar o sistema quando duas peças contíguas forem detectados como ruins. Assim, os eventos relacionados com a classificação das peças boas e ruins foram escolhidas para estarem no alto nível. Por outro lado, as especificações para o baixo nível são especificações de

segurança que não poderiam ser alteradas continuamente.

8.2 Modelo do nível operacional

A planta real no nível operacional será construída a partir da composição dos subsistemas envolvidos. Assim, consideram-se o modelo para a mesa, esteira, máquina e robô, conforme apresentado na Figura 8.2. Os eventos α_x são eventos controláveis e representam o início de operação de cada sub-sistema, e os eventos β_x são eventos não controláveis que representam o final de operação de cada subsistema.

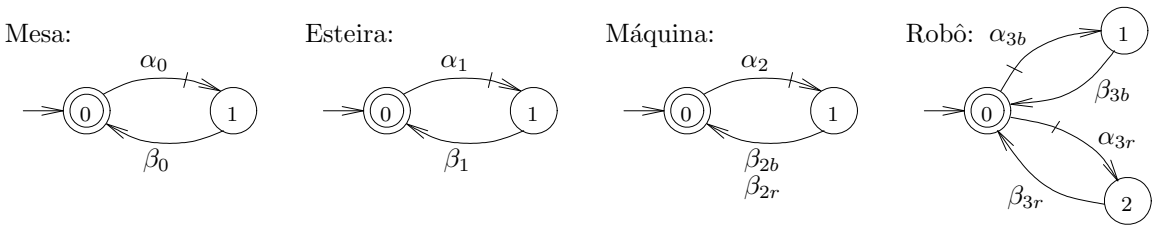


Figura 8.2: Modelagem dos sub-sistemas

Especificamente, β_{2b} e β_{2r}, representam o final de operação do teste de uma peça, tendo-se como resultado peça boa (β_{2b}) ou ruim (β_{2r}). Similarmente α_{3b} e β_{3b} representam a manipulação de peças boas, e α_{3r}, β_{3r} a manipulação de peças ruins.

No *Grail*, aplicando o filtro de composição síncrona da ferramenta `fmsync` entre todos os subsistemas obtém-se um modelo de 24 estados, 116 transições e 11 símbolos. Para isto segue-se a linha de comando a seguir:

```
fmsync Mesa Esteira | fmsync Máquina |fmsync Robô > Planta_real
```

Como próximo passo modelam-se as especificações do nível operacional.

O autômato A descrito na Figura 8.3 modela uma especificação que garante que a mesa não vai girar inutilmente, isto é, sem ao menos uma peça bruta em P_1 ou uma peça trabalhada em P_2.

Os autômatos B_i apresentados na Figura 8.4 são especificações de segurança que impedem que a mesa gire enquanto a esteira, a máquina e/ou o robô estiverem operando.

Os problemas que podem ocorrer por causa do fluxo de múltiplas peças são evitados

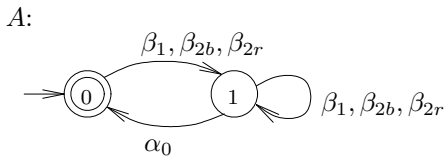


Figura 8.3: Especificação A

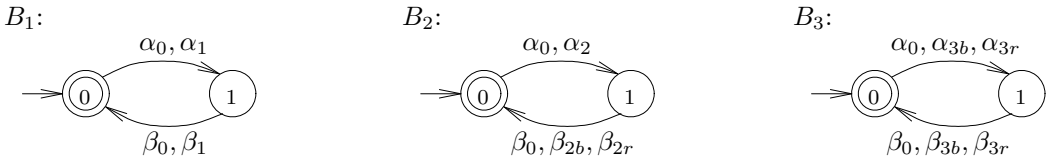


Figura 8.4: Especificação B

com as especificações C_i: na Figura 8.5, C_1, cuida da movimentação de peças brutas entre P_1 e P_2, e C_2 da movimentação de peças trabalhadas entre P_2 e P_3. A especificação C_1 evita sobrepor peças em P_1, operar sem peça bruta em P_2 e girar a mesa com peça não trabalhada em P_2, enquanto C_2 impede trabalhar duas vezes a mesma peça, acionar o robô sem peça em P_3.

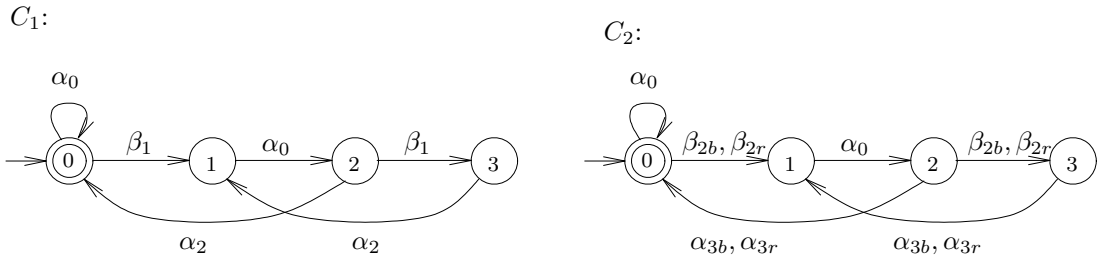


Figura 8.5: Especificação C

Calcula-se, então, o comportamento alvo no nível operacional pela composição síncrona das especificações A, B_1, B_2, B_3, C_1, C_2 e a planta do nível operacional. O resultado foi um autômato de 67 estados e 166 transições.

O passo seguinte é o cálculo da máxima linguagem controlável contida na linguagem alvo (Ramadge e Wonham 1989). No Grail aplica-se a linha de comando a seguir:

```
fmsupc Planta_real Alvo No_controlaveis > Sup_de_baixo_nivel
```

onde o autômato `Alvo` representa a linguagem alvo, o autômato `No_controlaveis` representa um autômato que tem como alfabeto o conjunto de eventos não controláveis do sistema. O autômato do supervisor obtido `Sup_de_baixo_nivel` o qual tem como linguagem a máxima linguagem controlável possui 51 estados, 122 transições e 11 símbolos.

O comportamento em malha fechada da planta real sob supervisão de `Sup_de_baixo_nivel` é então assumido como a nova planta de baixo nível para tratar o problema de controle hierárquico modular. Denota-se este autômato como `Planta_simplificada` .

8.3 Modelo do nível agregado

No nível agregado, a preocupação do supervisor restringe-se ao controle da classificação de peças boas e ruins que o sistema pode gerar. Portanto, consideram-se como eventos relevantes para controle do alto nível os eventos β_0, α_1, β_{2b}, β_{2r}, β_{3b} e β_{3r}.

Seguindo a metodologia apresentada no capítulo 5, obtém-se o modelo agregado com estruturas de controle dependentes do estado. Para automatizar o cálculo foi incluído na ferramenta *Grail* um filtro denominado de `fmagreg`, o qual permite obter o modelo agregado determinístico com estruturas de controle dependentes do estado. Para obtenção deste modelo no *Grail*, aplica-se a linha de comando a seguir:

```
fmagreg Planta_simplificada No_controláveis Relevantes
```

onde `Relevantes` é um autômato que tem como alfabeto o conjunto de eventos relevantes para controle no alto nível. O resultado desta linha de comando retorna três autômatos: o primeiro denominado de `agreg.des` representa o autômato agregado com estruturas de controle dependentes do estado (29 estados; 136 transições; 27 símbolos); o segundo denominado de `control.des` representa as estruturas de controle do modelo agregado (624 padrões de controle); finalmente o último autômato denominado de `renamed_plant.des` (89 estados; 216 transições; 32 símbolos) modela a `planta_ simplificada` com as respectivas renomeações feitas para eliminar o não determinismo. Para visualização amigável das estruturas de controle pode-se aplicar a linha de comando a seguir (da Cunha 2001):

```
cdeswrite agreg.des control.des > control.txt
```

Como apresentado no capítulo 6, o número de estados e renomeações do modelo

agregado pode ser reduzido de tal forma que o modelo agregado resultante mantenha as propriedades para controle hierárquico modular. Na ferramenta *Grail* foi introduzido o filtro `fmag_red` que permite obter o modelo agregado e reduzido. A sintaxe é a seguinte:

```
fmag_red Planta_simplificada No_controláveis Relevantes
```

Como resultado obtém-se também três autômatos, `agreg_red.des` (24 estados; 79 transições; 21 símbolos), `control_red.des` (509 padrões de controle) e `renamed_plant_red.des` (80 estados; 202 transições; 26 símbolos).

Até este ponto tem-se o modelo agregado e reduzido com estruturas de controle dependentes do estado (`agreg_red.des`, `control_red.des`). Na próxima seção introduzem-se as especificações requeridas e projetam-se supervisores para o nível agregado.

8.4 Controle do modelo agregado

Inicialmente definem-se as especificações para o nível agregado.

Os autômatos D_i apresentados na Figura 8.6 modelam as especificações do nível agregado, isto é, toda vez que o teste recusar ou aceitar uma peça, o robô deve respectivamente levar a peça para o "buffer" de peças ruins ou peças boas.

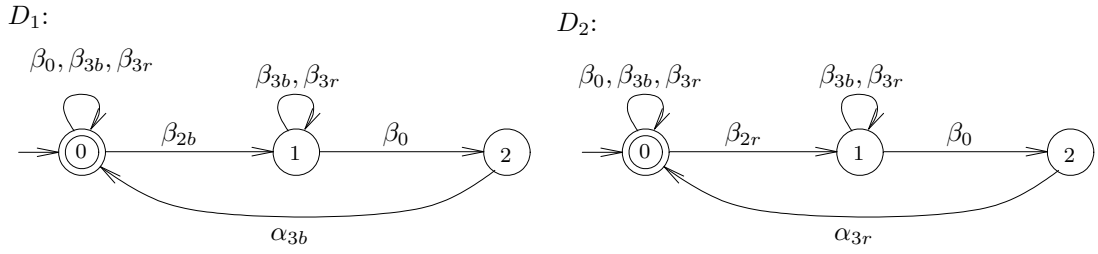


Figura 8.6: Especificações do nível agregado

No modelo agregado, para eliminar o não determinismo foram criadas algumas instâncias de eventos como resultado da renomeação dos eventos relevantes originais. Para não alterar o significado da especificação, por cada evento renomeado, todas as instâncias renomeadas deverão aparecer, como mostra a Figura 8.7.

Sejam `especif_d1` e `especif_d2` as especificações adaptadas ao novo alfabeto.

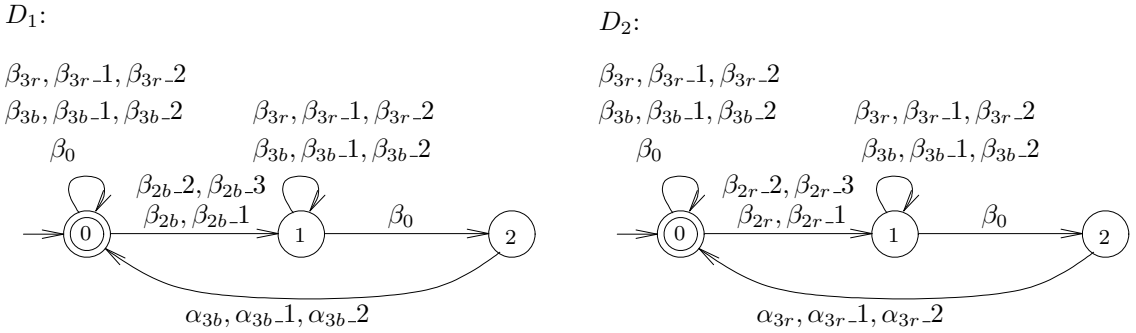


Figura 8.7: Especificações adaptadas ao novo alfabeto do nível agregado

Como próximo passo obtém-se a linguagem alvo para cada especificação. Para não perder a marcação da linguagem alvo, inicialmente marcam-se todos estados do modelo agregado para depois realizar o produto síncrono. No *Grail* tem-se:

`fmmark agreg_red.des | fmsync especif_d1 > alvo1` ; (49 estados , 135 transições)

`fmmark agreg_red.des | fmsync especif_d2 > alvo2` ; (49 estados , 135 transições)

Dado o modelo agregado com estruturas de controle dependentes do estado e dada uma especificação qualquer, no capítulo 4 apresenta-se o algoritmo para calcular a máxima linguagem Γ-compatível. Um filtro denominado de `cdessupg` foi introduzido no *Grail* por Antônio da Cunha para este cálculo.

`cdessupc agreg_red.des control_red.des alvo1 > sup_gamma1`; (38 estados, 105 transições)

`cdessupc agreg_red.des control_red.des alvo2 > sup_gamma2`; (38 estados, 105 transições)

Neste exemplo específico foram verificadas que estas linguagens são modulares, isto é, $\overline{L_m(\texttt{sup\_gamma1})} \cap \overline{L_m(\texttt{sup\_gamma2})} = \overline{L_m(\texttt{sup\_gamma1}) \cap L_m(\texttt{sup\_gamma2})}$, e ainda que $L_m(\texttt{sup\_gamma1}) \cap L_m(\texttt{sup\_gamma2})$ é Γ-compatível. No capítulo 7 foi visto que estas duas condições são necessárias e suficientes para a modularidade hierárquica. Portanto existem dois supervisores no nível agregado f_1^A e f_2^A não conflitantes tal que $L(f_1^A/\texttt{agreg\_red.des}) = L_m(\texttt{sup\_gamma1})$, $L(f_2^A/\texttt{agreg\_red.des}) = L_m(\texttt{sup\_gamma2})$ que permitem uma implementação não bloqueante. O resultado da

implementação no nível agregado teve como resultado um autômato com 20 estados 33 transições.

8.5 Conclusão

Neste capítulo foi apresentado um exemplo onde foram aplicadas todas as abordagens desenvolvidas nesta tese. As principais observações foram as seguintes:

- O exemplo apresentava o problema de não determinismo que foi resolvido renomeando os eventos envolvidos. Este fato trouxe um crescimento exponencial do número de padrões de controle para o modelo agregado.
- O processo de redução do autômato agregado permitiu diminuir: o número de estados de 29 para 24, as transições de 139 para 79, os símbolos de 27 para 21 e o número de padrões de controle de 624 para 509.
- O tempo de processamento para obter o modelo agregado num computador pentium III 700MHz de velocidade foi de 2 horas e 52 minutos. O grande "gargalo" neste procedimento é o cálculo das estruturas de controle do alto nível.
- Por outro lado uma vez obtido o modelo agregado, as demais operações no *Grail* não superavam os 3 segundos, ou seja, a solução de problemas de controle no alto nível tiram proveito do fato de que estes problemas são resolvidos sobre autômatos reduzidos. Portanto a metodologia se justifica em aplicações onde o modelo agregado obtido é utilizado para resolução de múltiplos problemas de controle.

Em (da Cunha 2001) apresenta-se um modelo simplificado mais restritivo, mas com um ganho em tempo de processamento muito maior (Tempo de processamento aprox 3 minutos).

Capítulo 9

Conclusão e Perspectivas

Neste trabalho inicialmente foram estudados os diversos modelos para controle hierárquico apresentados na literatura. Foram analisadas as vantagens e desvantagens tanto do controle hierárquico por agregação de estados (Caines e Hubbard 1997, Caines e Hubbard 1998a, Caines e Hubbard 1998b), quanto do controle hierárquico convencional (Zhong e Wonham 1990, Wong e Wonham 1996, Pu 2000). Em função disso, este trabalho contribui à teoria de Sistemas a Eventos Discretos com uma nova abordagem para controle hierárquico baseada na agregação de estados e estruturas de controle avançadas, permitindo diminuir a complexidade no projeto de supervisores em sistemas de grande porte.

As principais contribuições deste trabalho relacionam-se à modelagem e controle de sistemas a eventos discretos identificadas a seguir:

- Inicialmente desenvolveu-se um modelo formal para controle supervisório de sistemas a eventos discretos onde o sistema é representado por uma linguagem prefixo fechada e uma estrutura de controle avançada. A linguagem descreve o conjunto de todas as cadeias de eventos que o sistema pode gerar. A estrutura de controle é uma função que associa a cada cadeia um conjunto de padrões de controle, cada padrão de controle define o conjunto de eventos habilitados após a cadeia e um atributo de marcação que define se a cadeia é uma tarefa completa ou não.

- Baseado no modelo anterior foi introduzido um novo formalismo para o controle hierárquico por agregação de estados. Foram considerados dois níveis de hierarquia, nível operacional e agregado: para o nível operacional recorreu-se ao modelo tradicional de Ramadge e Wonham, no qual o sistema é definido sobre um alfabeto particionado em eventos controláveis e não controláveis; no nível

agregado, foi utilizado o modelo descrito no ítem anterior, baseado em estruturas de controle avançadas e marcação dinâmica associada.

- Desenvolveram-se formas sistemáticas para redução do modelo agregado com estruturas de controle dependentes do estado de tal forma que esta redução mantenha as propriedades do controle hierárquico.

- Apresentou-se uma combinação da arquitetura de controle hierárquico e modular envolvendo respectivamente as abordagens desenvolvidas no capítulo 5 e aquela de Wonham e Ramadge (1988). À decomposição hierárquica (decomposição vertical) adicionou-se a decomposição modular (decomposição horizontal). Esta nova arquitetura permite um comportamento consistente entre os níveis de hierarquia e a ação conjunta de vários controladores gerenciais implementados no baixo nível é não bloqueante e ótima.

- Finalmente, como complemento das abordagens apresentadas, foi estendida a ferramenta computacional *Grail* para tratar estes problemas.

As abordagens aqui desenvolvidas mostram-se apropriadas para modelagem e controle de sistemas num alto nível de abstração, como visto em (da Cunha e Cury 2002, Torrico e Cury 2002c). A utilização das estruturas de controle avançadas proporciona uma representação compacta, resultando numa economia do espaço de estados para descrever o sistema e permite modelar o sistema no seu nível de abstração.

A forma de construção da arquitetura hierárquica já apresenta a vantagem de possuir diretamente as boas propriedades para controle hierárquico.

Uma limitação do controle hierárquico aqui apresentado está na obtenção do modelo agregado, devido ao cálculo das estruturas de controle, o que no entanto é feito uma única vez. Uma vez obtido o autômato agregado com estruturas de controle, para qualquer especificação no nível agregado, o cálculo do supervisor é feito sobre o mesmo autômato agregado (reduzido). Portanto a metodologia se justifica em aplicações onde o modelo agregado obtido é utilizado para resolução de múltiplos problemas de controle.

Por outro lado o controle hierárquico modular permite ainda maiores ganhos na resolução de múltiplos problemas de controle no alto nível, tendo-se as vantagens de flexibilidade e ganho computacional da abordagem modular clássica de Wonham e Ramadge (1988).

Quanto a perspectivas para futuros trabalhos indica-se:

- Otimizar o cálculo das estruturas de controle usando o fato de que Γ é fechada para união.

- Buscar alternativas para solução do problema de não determinismo do nível agregado de tal forma a evitar o crescimento exponencial do número de padrões de controle.

- Explorar métodos para minimização do autômato agregado. Nesta tese apresentam-se apenas métodos de redução.

- Para sistemas compostos com estruturas de controle avançadas, explorar a abordagem modular local proposta em (de Queiroz e Cury 2000).

Apêndice A

Manual da Ferramenta *Grail*

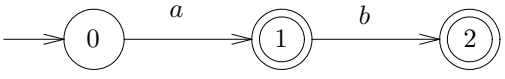
A.1 Introdução

O *Grail* é uma biblioteca de funções C++ para manipulação de autômatos finitos, expressões regulares e linguagens finitas. A ferramenta foi elaborada com a intenção de facilitar a pesquisa, auxiliar no ensino da matemática discreta e servir de base em aplicações que usam máquinas de estado.

Neste documento serão apresentadas unicamente as funções para manipulação de autômatos finitos.

A.2 Máquinas de estado finitas (FM)

No *Grail* o formato de especificação de uma FM consiste de uma lista de instruções armazenada em um arquivo ASCII, não necessariamente ordenadas e/ou não repetidas. Na Figura A.1 apresenta-se um autômato e sua correspondente especificação no *Grail*:




```
(START) |- 0
0  a  1
1  b  2
1  -| (FINAL)
2  -| (FINAL)
```

Figura A.1: Autômato e sua especificação de um autômato no *Grail*

O *Grail* oferece alguns predicados e vários filtros para trabalhar com FM. A Tabela A.1 mostra os predicados e a Tabela A.2 mostra os filtros.

`iscomp`	testa se FM é completa
`isdeterm`	testa se FM é determinística
`isomorph`	testa se FM é isomorfa
`isuniv`	testa se FM é universal

Tabela A.1: Predicados do *Grail*

`fmalpha`	tira o alfabeto de uma FM
`fmcat`	concatena duas FMs
`fmcment`	complementa uma FM
`fmcomp`	completa uma FM
`fmcondat`	informa dados de controle sobre FM
`fmcross`	intersecciona duas FMs
`fmdeterm`	torna FM determinística
`fmenum`	enumera palavras reconhecidas pela FM
`fmexec`	dada uma cadeia executa a FM
`fmloop`	faz o self-loop de eventos da primeira FM na segunda FM
`fmmark`	marca todos os estados da FM
`fmmin`	minimiza a FM
`fmminrev`	minimiza a FM (outro método)
`fmplus`	faz o plus de uma FM
`fmproj`	faz a projeção de uma FM
`fmreach`	retira a componente acessível de uma FM
`fmremove`	elimina estados de uma FM
`fmrenum`	renumera os estados de uma FM
`fmreverse`	encontra o reverso de uma FM
`fmsort`	sorteia as instruções para os estados
`fmstar`	faz o fechamento Kleene de uma FM
`fmstats`	obtêm informações sobre a FM
`fmsupc`	encontra a máxima linguagem controlável
`fmsync`	faz o produto síncrono de duas FMs
`fmtodot`	converte uma FM para o formato .dot
`fmtovcg`	converte uma FM para o formato .vcg
`fmtrim`	encontra a componente trim de uma FM
`fmunion`	encontra a união de duas FMs

Tabela A.2: Filtros do *Grail*

Para controle hierárquico por agregação de estados foram implementadas algumas funções, as quais são descritas na tabela a seguir.

Filtro	Descrição	Autor
`fmagreg`	obtém o modelo agregado G_s	César Torrico
`fmag_red`	obtém o modelo agregado e reduzido G'_s	César Torrico
`cdeswrite`	escreve as estruturas de controle de G_s	Antônio da Cunha
`cdessupc`	econtra a máxima linguagem Γ-compatível	Antônio da Cunha

Tabela A.3: Filtros do *Grail* para controle hierárquico por agregação de estados

Na próxima seção serão descritos os filtros implementados para o controle hierárquico (Cap. 8).

A.3 Descrição dos filtros mais relevantes para controle hierárquico por agregação de estados

`fmmark` : *Marca todos os estados de um autômato.*

Sintaxe: `fmsync FM1`, onde

- `FM1` Representa um autômato qualquer.

`fmtodot` : *Converte um autômato para formato* `.dot`.

Sintaxe: `fmtodot FM1`, onde

- `FM1` Representa um autômato qualquer.

`fmsync` : *Produto síncrono de dois autômato*s.

Sintaxe: `fmsync FM1 FM2`, onde

- `FM1` e `FM2` Representam autômatos finitos determinísticos.

`fmsupc` : *Máxima linguagem controlável.*

Sintaxe: `fmsupc FM1 FM2 FM3`, onde

- `FM1` Representa o autômato da planta.
- `FM2` Representa o autômato que reconhece a linguagem alvo.
- `FM3` Representa um autômato que tenha como alfabeto os eventos não controláveis da planta.

`fmagreg` : *Obtém o modelo agregado.*

Sintaxe: `fmagreg FM1 FM2 FM3`, onde

- `FM1` Representa o autômato da planta.
- `FM2` Representa um autômato que tenha como alfabeto os eventos não controláveis da planta.
- `FM3` Representa um autômato que tenha como alfabeto os eventos relevantes para especificação.

O resultado da aplicação deste filtro são três autômatos: `agreg.des` ; `control.des` ; e `renamed_plant.des`.

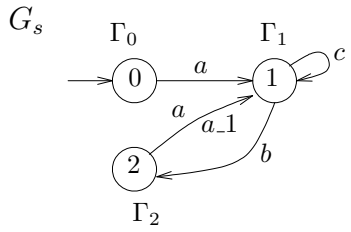
O modelo agregado está caracterizado pelos autômatos `agreg.des` e `control.des`. O autômato `agreg.des` representa apenas a estrutura de transição do autômato agregado, sem considerar as estruturas de controle nem a marcação do modelo. O autômato `control.des` representa as estruturas de controle do autômato agregado.

Por exemplo, o modelo com estruturas de controle dependentes do estado apresentado na Figura A.2, no *Grail* são dados pelos autômatos `agreg.des` e `control.des` apresentados na Figura A.3.

Observa-se que o autômato `control.des` resulta ser um autômato desconectado, formado por vários sub-autômatos isolados. Cada sub-autômato está associado a um estado do modelo agregado, representando as estruturas de controle para aquele estado. Ainda no autômato `control.des` as transições M e N representam os atributos de marcação. Um padrão de controle é representado pelas transições que saem de um mesmo estado e levam a um mesmo estado.

O autômato `renamed_plant.des` representa o autômato de baixo nível com as respectivas renomeações que podem ocorrer por causa do não determinismo.

`fmag_red` : *Obtém o modelo agregado e reduzido.*



$\Gamma_0 = \{(\emptyset, M); (\{a\}, M)\}$
$\Gamma_1 = \{(\{b\}, N); (\{b, c\}, N)\}$
$\Gamma_2 = \{(\emptyset, M); (\{a\}, M); (\{a\_1\}, M); (\{a, a\_1\}, M)\}$

Figura A.2: Modelo com estruturas de controle dependentes do estado.

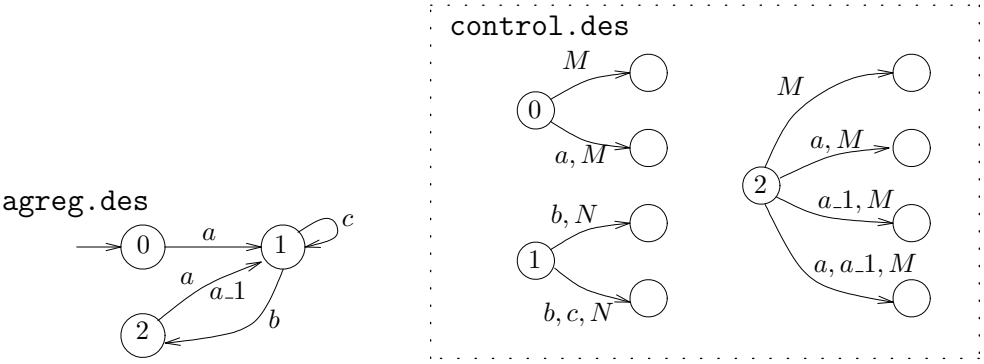


Figura A.3: Control.des

Sintaxe: `fmag_red FM1 FM2 FM3`

Os argumentos de `fmag_red` são os mesmos que de `fmagreg`.

O resultado da aplicação deste filtro são três autômatos: `agreg_red.des`, `control_red.des` e `renamed_plant_red.des`.

Os autômatos `agreg_red.des` e `control_red.des` representam o modelo agregado e reduzido com estruturas de controle dependentes do estado, já o autômato `renamed_plant_red.des` representa o modelo de baixo nível considerando as possíveis renomeações que poderiam ter sido feitas no autômato original. Estes autômatos são análogos com os autômatos `agreg.des`, `control.des` e `renamed_plant.des` respectivamente.

`cdeswrite` : *Escreve as estruturas de controle do modelo agregado.*

Sintaxe: `fmag_red FM1 FM2`, onde

- `FM1` Representa o modelo agregado (`agreg.des` ou `agreg_red.des`).

- `FM2` Representa as estruturas de controle do modelo agregado (`control.des` ou `control_red.des` respectivamente).

`cdessupc` : *Máxima linguagem* Γ *compatível.*

Sintaxe: `cdessupc FM1 FM2 FM3`, onde

- `FM1` Representa o modelo agregado (`agreg.des` ou `agreg_red.des`).
- `FM2` Representa as estruturas de controle do modelo agregado (`control.des` ou `control_red.des` respectivamente).
- `FM3` Representa o autômato que reconhece a linguagem alvo do nível agregado.

No capítulo 8 apresenta-se um exemplo de aplicação utilizando esta ferramenta computacional.

A.4 Utilização do Grail

As ferramentas Grail e Graphviz são utilizadas através de uma linha de comando. Para utilizá-las, é necessário ajustar o PATH:

```
set path="%path%";c:\sed\grail\bin;c:\sed\graphviz\bin;
```

Após acertar o PATH pode-se chamar todas as funções do Grail através da linha de comando. Para utilizar o Graphviz basta chamar a função `dotty`, como mostra o exemplo:

```
dotty strim.dot
```

As ferramentas podem ser obtidas nos seguintes endereços:

`ftp://ftp.lcmi.ufsc.br/pub/Windows/programacao/sed/`

`http://www.research.att.com/sw/tools/graphviz/`

Referências Bibliográficas

Brandin, B. e Wonham, W. (1994). Supervisory control of timed discrete-event system, *IEEE Trans. on Automatic Control* **39**(2): 329–342.

Caines, P., Hubbard, P. e Shen, G. (1997). A state aggregation and hierarchical supervisory control, *Proc. of 36th IEEE CDC* p. 3590–3591.

Caines, P. e Hubbard, P. (1997). A state aggregation approach to hierarchical supervisory control with applications to a transfer-line and comunication networks, *Technical Report*, Department of Electrical and Computer Enggineering, McGill University.

Caines, P. e Hubbard, P. (1998a). A state aggregation approach to hierarchical supervisory control with applications to a transfer-line example, *Proceedings of the WODES 98*, Cagliary, Italy, p. 2–7.

Caines, P. e Hubbard, P. (1998b). Trace-dc hierarchical supervisory control with applications to a transfer-line example, *Proceedings of the 37nd IEEE Conference on Decision and Control*, Tampa. FL, p. 3293–3298.

Carroll, J. e Long, D. (1989). *Theory of Finite Automata*, Prentice-Hall International Editions.

Claure, C. (1999). *Implementação de controle supervisório de SEDs aplicado a processos de manufatura*, Dissertação(mestrado), Pós-Graduação em Engenharia Elétrica - Universidade Federal de Santa Catarina, Fpolis - SC.

Cury, J., Torrico, C. e da Cunha, A. (2001). A new approach for supervisory control of discrete event sistems, *Proceedings of the European Control Conference 2001.*

Cury, J., Torrico, C. e da Cunha, A. (2002). Supervisory control of discrete event systems with dynamical marking attribution, *Article submitted to the European Journal of Control* .

da Cunha, A. e Cury, J. (2002). Hierarchically consistent controlled discrete event systems, *Proceedings of the 15th IFAC World Congress* .

da Cunha, A. (2001). *Controle Hierárquico de Sistemas a Eventos Discretos e Sistemas Híbridos*, Proposta de Tese submetida à Universidade Federal de Santa Catarina.

de Queiroz, M. H. e Cury, J. E. R. (2000). Modular supervisory control of large scale discrete event systems, *Discrete Event Systems. Analysis and Control (WODES)* p. 103–110.

de Queiroz, M. H. e Cury, J. E. R. (2001). Síntese modular do controle supervisório em diagrama em escada para uma célula de manufatura., *V SBAI(Simpósio Brasileiro de Automação Inteligente)* .

de Queiroz, M. (2000). *Controle supervisório modular de sistemas de grande porte*, Dissertação(mestrado), Pós-Graduação em Engenharia Elétrica - Universidade Federal de Santa Catarina, Fpolis - SC.

Eyzell, J. e Cury, J. (1998). Exploiting symmetry in the synthesis of supervisors for discrete event systems, *Proceedings of the American Control Conference*, Philadelphia, USA, June.

Eyzell, J. (2000). *Aspectos de Síntese de Supervisores para Sistemas a Eventos Discretos e Sistemas Híbridos*, Tese (doutorado), Programa de Pós Graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, Fpolis - SC - Brasil.

Golaszewski, C. e Ramadge, P. (1987). Control of discrete event processes with forced events, *IEEE Proceedings of the 26th Conference on Decision and Control*, Los Angeles, CA, August, p. 247–252.

Hopcroft, J. e Ullman, J. (1979). *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley USA.

Kumar, R. e Garg, V. (1995). *Modeling and Control of Logical Discrete Event Systems*, Kluwer Academic Publishers.

Lin, F., Li, Y. e Lin, W. (1998). A generalized framework for supervisory control of discrete event systems, *International Journal of Intelligent Control and Systems*, V. 2, World Scientific Publishing Company, p. 139–159.

Lin, F. e Wonham, W. (1990). Descentralized control and coordination of discrete-event systems with partial observation, *IEEE Transactions on Automatic Control* **35**(12): 1330–1337.

Pu, K. (2000). *Modeling and control of discrete event systems with hierarchical abstraction*, Master's thesis, Department of Electrical and Computer Engineering University of Toronto.

Ramadge, P. e Wonham, W. (1987). Supervisory control of a class of discrete event processes, *SIAM J. Control and Optimization* **25**(1): 206–230.

Ramadge, P. e Wonham, W. (1989). The control of discrete event systems, *Proceeding of the IEEE* **77**(1): 81–98.

Raymond, D. e Wood, D. (1996). Grail. *www.csd.uwo.ca/research/grail* or *www.lcmi.ufsc.br/∼cury/ensino-5202.html* .

Rudie, K. e Wonham, W. (1992). Think globaly, act locally: Decentralized supevisory control, *IEEE Transactions on Automatic Control* **37**(11): 1692–1708.

Torrico, C. e Cury, J. (2001). Controle supervisório hierárquico de sistemas a eventos discretos: Uma abordagem baseada na agregação de estados, *V SBAI(Simpósio Brasileiro de Automação Inteligente)* .

Torrico, C. e Cury, J. (2002a). Controle supervisório hierárquico modular por agregação de estados, *XIV Congresso Brasileiro de Automática (CBA2002)* .

Torrico, C. e Cury, J. (2002b). Controle supervisório hierárquico modular por agregação de estados, *Artigo submetido à revista da Sociedade Brasileira de Automática(SBA).*

Torrico, C. e Cury, J. (2002c). Hierarchical supervisory control of discrete event systems based on state aggregation, *Proceedings of the 15th IFAC World Congress* .

Wong, K. e Wonham, W. (1996). Hierarquical control of discrete-event systems, *Discrete Event Dynamical Systems* **6**: 241–273.

Wonham, W. e Ramadge, P. (1987). On the supremal controllable sublanguage of a given language, *SIAM J. Control and Optimization* **25**(3): 637–659.

Wonham, W. e Ramadge, P. (1988). Modular supervisory control of discrete event systems, *Mathematics of control, Signals and Systems* **1**(1): 13–30.

Wonham, W. (1998). *Notes on Control of Discrete-Event Systems. Course Notes for ECE 1636F/1637S*, Revision 98.09.01.

Zhong, H. e Wonham, W. (1990). On the consistency of hierarchical supervision in discrete-event systems, *IEEE Transactions on Automatic Control* **35**(10): 1125–1134.

Ziller, R. (1993). *A abordagem ramadge-wonham no controle de sistemas a eventos discretos: Contribuições à teoria*, Dissertação(mestrado), Pós-Graduação em Engenharia Elétrica - Universidade Federal de Santa Catarina, Fpolis - SC.