MAX HERING DE QUEIROZ

CONTROLE SUPERVISÓRIO MODULAR E MULTITAREFA DE SISTEMAS COMPOSTOS

FLORIANÓPOLIS
2004

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

CONTROLE SUPERVISÓRIO MODULAR E MULTITAREFA DE SISTEMAS COMPOSTOS

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

MAX HERING DE QUEIROZ

Florianópolis, maio de 2004

CONTROLE SUPERVISÓRIO MODULAR E MULTITAREFA DE SISTEMAS COMPOSTOS

Max Hering de Queiroz

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

_____
José Eduardo Ribeiro Cury, Dr.– UFSC
Orientador

_____
Jefferson Luiz Brum Marques, Dr. – UFSC
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

_____
José Eduardo Ribeiro Cury, Dr.– UFSC
Presidente

_____
Rafael Santos Mendes, Dr. – UNICAMP

_____
Paulo Eigi Miyagi, Dr. – USP

_____
Victor Juliano De Negri, Dr. – UFSC

_____
Jean-Marie Farines, Dr. – UFSC

_____
Eduardo Camponogara, Dr. – UFSC

Agradecimentos

A presente Tese de Doutorado é fruto de um longo esforço coletivo. Em primeiro lugar, sou muito grato ao governo brasileiro que, por intermédio das agências CNPq e CAPES, financiou três anos e dois meses de pesquisa na UFSC, bem como o estágio de dez meses na Universidade de Toronto, Canadá. Agradeço ao Prof. José Cury pela dedicação, competência, paciência e amizade com que orientou este trabalho. Do mesmo modo, sou grato ao Prof. Murray Wonham por acolher-me tão gentilmente junto a seu grupo de pesquisa na Universidade de Toronto. Também agradeço a colaboração e amizade dos demais professores, colegas e funcionários do LCMI, na UFSC, e do SCG, na Universidade de Toronto. O apoio e carinho de meus queridos pais, irmãos e amigos ajudaram a superar com alegria os desafios desta empreitada. Agradeço em especial a minha esposa Mariana por ter nutrido com muito amor cada passo meu. Finalmente, devo graças a Deus pelo milagre da vida.

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

CONTROLE SUPERVISÓRIO MODULAR E MULTITAREFA DE SISTEMAS COMPOSTOS

Max Hering de Queiroz

Maio/2004

Orientador: Prof. José Eduardo Ribeiro Cury, Dr.

Área de Concentração: Controle e Automação.

Palavras-chave: controle supervisório; sistemas a eventos discretos; controle modular; controle multitarefa; sistemas compostos.

Número de Páginas: 150.

A presente Tese de Doutorado introduz uma nova metodologia para o tratamento eficiente de múltiplas especificações e múltiplas tarefas no controle supervisório de sistemas a eventos discretos (SEDs) compostos. A abordagem modular local desenvolvida no mestrado do autor permite explorar a arquitetura modular das especificações e da planta em sistemas compostos de forma a evitar a composição de modelos e, por conseguinte, a complexidade induzida pelo crescimento exponencial no tamanho do modelo global. No presente trabalho, essa abordagem é consolidada pela aplicação bem sucedida a um problema real envolvendo uma célula de manufatura. Para viabilizar a implementação do sistema de controle em controlador lógico programável, propõe-se uma estrutura genérica que preserva a característica modular dos supervisores e da planta. Além dessas contribuições, introduz-se o gerador com marcações coloridas, um caso particular de autômato de Moore, como modelo que distingue múltiplas classes de tarefas em SEDs. Os principais resultados da teoria de controle supervisório são estendidos para lidar com esse modelo, de forma a permitir a síntese automática de supervisores que, além de respeitar o comportamento especificado, garantem a vivacidade de múltiplos objetivos de controle. Investiga-se também a propriedade de reversibilidade como alternativa para evitar bloqueio de vária tarefas. A conveniência da metodologia multitarefa é ilustrada por três exemplos acadêmicos. Finalmente, os resultados de controle multitarefa são combinados com a abordagem modular local. A clareza das soluções e a eficiência computacional proporcionadas pela metodologia proposta são elucidadas na síntese de supervisores reduzidos para um exemplo de sistema flexível de manufatura.

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

MODULAR AND MULTITASKING SUPERVISORY CONTROL OF COMPOSITE SYSTEMS

Max Hering de Queiroz

May/2004

Advisor: Prof. José Eduardo Ribeiro Cury, Dr.

Area of Concentration: Control and Automation.

Keywords: discrete event systems; automata; multitasking; control systems; distributed control.

Number of Pages: 150.

This Thesis introduces a new methodology for efficiently dealing with multiple specifications and multiple tasks in supervisory control of composite discrete event systems (DES). The local modular approach presented in the author's Master Dissertation exploits the modular architecture of specifications and plant in composite systems for the purpose of avoiding composition of models and, by consequence, the complexity induced by the exponential growth of the size of the global model. In the present work, this approach is consolidated by the well-succeeded application to a real problem concerning a manufacturing cell. A generic structure that preserves the modular characteristic of supervisors and plant is proposed to implement the control system in a programmable logic controller. In addition to these contributions, the colored marking generator – a special type of Moore automaton – is introduced as a model that distinguishes multiple classes of tasks in DES. The main results of the supervisory control theory are extended to deal with this model in order to allow the automatic synthesis of supervisors that, beyond respecting the specified behavior, assure the liveness of multiple control objectives. The property of reversibility is also investigated as an alternative way of avoiding blocking of various tasks. The convenience of the multitasking methodology is illustrated by three academic examples. Finally, the results of multitasking control are linked with the local modular approach. The clarity of solutions and the computational efficiency afforded by the proposed methodology are elucidated in the synthesis of reduced supervisors for an example of flexible manufacturing system.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

$\exists$	existe
$\forall$	para todo
$\wedge$	operador “e”
$\vee$	operador “ou”
:	tal que
♦	indica final de demonstração
$\|\|$	operador de produto síncrono
$\|G\|$	número de estados do gerador G
$\chi\|Q$	função χ restrita ao domínio Q
$\delta(q_0, s)!$	a função δ é definida para o par (q_0, s)
$Pwr(A)$	conjunto potência de A (todos os subconjuntos de A)
$\bar{L}$	prefixo-fechamento de L
$P_i: \Sigma^* \rightarrow \Sigma_i^*$	projeção natural de Σ^* para Σ_i^*,
$G \subseteq H$	G é um subgerador de H
$\varnothing$	conjunto vazio
$\varnothing_{\Sigma,C}$	gerador com marcações coloridas vazio para Σ e C
Σ	alfabeto
Σ_c	conjunto de eventos controláveis
Σ_u	conjunto de eventos não-controláveis
Σ^*	conjunto de todas cadeias finitas de Σ, incluindo ε
ε	cadeia vazia
α, β, σ	eventos
s, u, v	palavras
a, **b**, **c**	cores

i	cor para tarefa de “alcançar o estado inicial”
v	cor vácuo (completada por qualquer palavra)
B, C, D, E	conjuntos de cores
C_{ac}	conjunto de tarefas acessíveis
K, L, M	linguagens
Λ_C, A_C, M_C, N_C	comportamentos coloridos
G, H	geradores
$L(G)$	linguagem gerada por G
$L_m(G)$	linguagem marcada por G
$L_c(\Lambda_C)$	linguagem marcada por $c \in C$
$L_B(\Lambda_C)$	linguagem marcada por $B \subseteq C$
$\Lambda_C(G)$	comportamento colorido de G
S	supervisor
$S_1 \wedge S_2$	conjunção dos supervisores S_1 e S_2
S/G	supervisor S controlando G
$\Re(S(s))$	eventos habilitados por S após a cadeia s
$\Im(S(s))$	novas cores marcadas por S após a cadeia s
$Ac(G)$	operação que elimina todos os estados não-acessíveis de G
$Tr(G)$	operação que elimina todos os estados não-acessíveis ou não-coacessíveis de G
$WTr(G, B)$	operação que elimina todos os estados de G que não são acessíveis e fracamente coacessíveis e.r.a B
$STr(G, B)$	operação que elimina todos os estados de G que não são acessíveis e fortemente coacessíveis e.r.a B
$SupC(\mathrm{A}_D, G)$	supremo subcomportamento de A_D controlável e.r.a G
$SupC(K, G)$	suprema sublinguagem de K controlável e.r.a G
$SupCSNB(\mathrm{A}_D,G,B)$	supremo subcomportamento de A_D controlável e.r.a G e fortemente não-bloqueante e.r.a B
$SupSNB(\mathrm{A}_D, B)$	supremo subcomportamento de A_D fortemente não-bloqueante e.r.a B

Lista de Abreviaturas

CLP	controlador lógico programável
e.r.a	em relação a
GMC	gerador com marcação colorida
MM	máquina de montagem
MP	máquina de pintura
RSP	representação por sistema-produto
RW	Ramadge-Wonham
SED	sistema a eventos discretos
SEDMT	sistema a eventos discretos multitarefa
SFM	sistema flexível de manufatura
TCS	teoria de controle supervisório

1. Introdução

A atual tendência de globalização da economia em países industrializados tem acirrado a concorrência entre as empresas, provocando uma busca incessante por maior qualidade e menor custo dos bens e serviços. Com isso, a eficiência e a flexibilidade dos meios produtivos e gerenciais têm sido fatores decisivos ao sucesso das empresas. A busca por competitividade, aliada à crescente escassez dos recursos naturais e à valorização da mão de obra, tem justificado grandes esforços na otimização e automação flexível dos processos (CASTELLS, 1999). Em paralelo, o desenvolvimento acelerado das tecnologias de computação, de comunicação e de sensores nas últimas décadas tem possibilitado um grande aumento da capacidade de aquisição, armazenagem e processamento de informações pelos sistemas.

Esse contexto tem favorecido o surgimento de sistemas dinâmicos cada vez mais complexos, envolvendo diversas aplicações, como redes de computadores e de comunicação, sistemas automatizados de manufatura, robótica, sistemas de software distribuídos, controle de tráfego, automação predial, entre outras. Tais aplicações são em grande parte governadas por regras operacionais – chamadas de lógica de controle – projetadas por humanos. Entre esses sistemas, destaca-se a classe de sistemas a eventos discretos (SEDs), que são caracterizados por uma dinâmica dirigida pela ocorrência de eventos (CASSANDRAS e LAFORTUNE, 1999). São exemplos de eventos discretos a chegada de uma mensagem num computador, a ativação de um sensor ótico e o início de operação de uma máquina. A ocorrência desses eventos não depende diretamente da passagem do tempo, mas sim de uma mudança discreta no estado do sistema.

Em geral, os sistemas a eventos discretos de maior complexidade podem ser modelados pela interação de múltiplos subsistemas concorrentes (SIMON, 1967) e, por isso, são denominados sistemas compostos. Cada subsistema de um sistema composto possui um comportamento característico bem definido, sendo responsável pela execução de tarefas particulares. Os sistemas flexíveis de manufatura, por exemplo, são caracteristicamente compostos pelo funcionamento paralelo de diversos sistemas de transformação, teste, transporte e armazenagem de materiais, como máquinas de usinagem, robôs e veículos autoguiados. A operação concorrente e a interação adequada dos subsistemas fazem com que o sistema composto atinja os resultados esperados, porém, na

ausência de controle, podem levá-lo a situações indesejáveis, como o choque entre dois robôs que compartilham um recurso ou o bloqueio do sistema. Assim, a lógica de controle é responsável pela coordenação dos subsistemas de forma que o sistema composto possa atingir seus objetivos evitando situações indesejáveis.

Em oposição aos sistemas dirigidos pelo tempo, os sistemas a eventos discretos não podem ser representados por modelos matemáticos baseados em equações diferenciais e a diferenças e, portanto, não são tratados pela Teoria de Controle clássica. Entretanto, em razão da crescente importância dessa classe de sistemas, a modelagem de SEDs tem sido objeto de diversas pesquisas recentes. Na literatura, encontram-se diversas abordagens formais para o desenvolvimento da lógica de controle, nas quais se incluem: Cadeias de Markov (ÇINLAIR, 1975; PUTERMAN, 1994), Redes de Filas (KLEINROCK, 1975), Simulação (LAW e KELTON, 2000), Lógica Temporal (MANNA e PNUELI, 1992), Redes de Petri (MURATA, 1989; KROGH e HOLLOWAY, 1991; ZHOU e VENKATESH, 1999) e Teoria de Controle Supervisório (RAMADGE e WONHAM, 1987; CASSANDRAS e LAFORTUNE, 1999; WONHAM, 2003).

A maior parte dessas abordagens limita-se à análise de soluções de controle propostas, que são geralmente geradas de forma empírica, com base na experiência e inspiração do projetista. Ao permitir a verificação de propriedades de modelos, tais ferramentas podem auxiliar o processo de depuração da lógica de controle proposta. Em contrapartida, a Teoria de Controle Supervisório (TCS) se destaca por fundamentar a síntese automática de uma lógica de controle ótima a partir de um modelo matemático do sistema em malha aberta (planta) e das especificações.

Na TCS, a planta e as especificações são respectivamente modeladas por geradores e linguagens (HOPCROFT e ULLMAN, 1979). Os estados do modelo são classificados como marcados se forem alcançados por cadeias de eventos que completam algum objetivo de controle (tarefa). Um evento é classificado como controlável se a sua ocorrência puder ser diretamente evitada pela entidade controladora, chamada de supervisor. A TCS apresenta uma série de ferramentas formais para a síntese automática de supervisores ótimos, isto é, que impedem a ocorrência de um evento controlável se e somente se ele puder desencadear uma seqüência de eventos não-controláveis que transgridam a linguagem especificada ou que levem o sistema para estados a partir dos quais não seja possível atingir estados marcados (situação de bloqueio). Assim, garante-se segurança na ocorrência de eventos e vivacidade dos objetivos de controle de forma minimamente restritiva.

Os algoritmos originalmente apresentados por (WONHAM e RAMADGE, 1987) para a síntese de supervisores ótimos têm complexidade polinomial em relação ao número de estados dos modelos da planta e da especificação. Isso viabiliza sua aplicação à solução de problemas envolvendo modelos com um número razoável de estados. Tipicamente, ferramentas como o CTCT (WONHAM, 2003) permitem operar geradores com até 100.000 estados.

Contudo, a aplicação da TCS à resolução de problemas envolvendo sistemas compostos costuma ser problemática. Duas são as razões principais: explosão do número de estados do modelo global da planta pela composição de subsistemas; e existência de múltiplas tarefas para as quais se deve evitar bloqueio.

1.1 Explosão do número de estados

É de conhecimento comum que a composição de geradores provoca crescimento exponencial do número de estados da planta. Esse fator inviabiliza a aplicação dos algoritmos originais de síntese de supervisores para sistemas compostos de maior porte. Por exemplo, o modelo de uma planta composta por dez subsistemas de dez estados pode ter até 10^{10} estados. Essa questão tem sido considerada por vários autores que procuram explorar diferentes aspectos do problema, como modularidade (WONHAM e RAMADGE, 1988) e simetria (EYZELL e CURY, 1998a e 1998b), no sentido de superar dificuldades computacionais. Também o uso de estruturas eficientes como "*Binary Decision Diagram*" (BRYANT, 1986), "*Integer Decision Diagram*" (ZHANG e WONHAM, 2001) e "*State Tree Structure*" (MA e WONHAM, 2003) tem permitido o tratamento de geradores com mais de 10^{24} estados.

No trabalho de mestrado, QUEIROZ (2000) e QUEIROZ e CURY (2000a, 2000b, 2000c e 2002a) propõem evitar o crescimento exponencial do tamanho dos modelos explorando, além da modularidade das especificações (WONHAM e RAMADGE, 1988), a própria modularidade natural da planta em sistemas compostos. Ao invés de se construir um supervisor monolítico (em um único bloco) para toda a planta, na abordagem modular proposta procura-se construir, sempre que possível, um supervisor local para cada especificação, modelando-o apenas em termos dos subsistemas afetados por sua ação. Neste caso, deseja-se que os supervisores resultantes sejam localmente modulares, isto é, que a ação conjunta dos supervisores tenha o mesmo desempenho que a do supervisor monolítico. Quando essa propriedade é verificada, a abordagem de controle modular é bastante vantajosa no sentido de promover maior flexibilidade, maior eficiência computacional e segurança na aplicação do controle. Além de diminuir a complexidade

computacional da síntese, a abordagem modular local favorece obtenção de supervisores de tamanhos reduzidos. Essa característica costuma deixar a lógica de controle mais clara, compreensível e, portanto, mais confiável ao projetista, além de facilitar a implementação e atualização do sistema de controle.

Apesar de a TCS ter sido bastante difundida e aceita no meio acadêmico nas últimas décadas, verifica-se uma grande escassez de aplicações reais (BALEMI *et al.*, 1993; BRANDIN, 1996; LEDUC, 1996). Essa carência de aplicações se deve em parte aos problemas provocados pelo enorme número de estados dos modelos de sistemas reais e em parte à falta de resultados práticos que orientem a implementação do sistema de controle de forma clara.

1.2 Múltiplas tarefas

Para sistema a eventos discretos as tarefas representam os objetivos de controle, ou seja, situações (estados) desejáveis de acontecer. Um exemplo típico de tarefa em sistemas de manufatura é o término de operação de uma máquina. Quando um SED atinge um estado a partir do qual não é mais possível completar tarefas, diz-se que o SED está em situação de bloqueio. Ao modelar um SED por um gerador, faz-se a identificação das tarefas completas através da marcação dos estados. Com isso, evita-se o bloqueio do sistema garantindo-se que no respectivo modelo qualquer estado alcançável possa ser levado a um estado marcado.

Em sistemas compostos, é natural definir objetivos de controle específicos para cada subsistema. A existência de múltiplas tarefas pode, também, decorrer da complexidade do problema. Por exemplo, num sistema flexível de manufatura o término de cada tipo de produto pode definir uma tarefa distinta. Para esses problemas, em geral deseja-se evitar o bloqueio para cada tipo de tarefa, de forma que, para cada estado que possa ser alcançado, seja possível completar todos objetivos de controle (possivelmente por caminhos distintos e em estados distintos).

No entanto, o uso de uma única marcação para representar todas as tarefas em sistemas compostos provoca uma perda de informação que pode comprometer a qualidade das soluções de controle. Como a marcação de um gerador é binária, na composição de modelos um estado global é marcado se o estado correspondente em todos os subsistemas for marcado. Assim, uma tarefa do sistema global reflete as situações em que todos os subsistemas estão simultaneamente em estados marcados. Portanto, ao evitar bloqueio na planta global, um supervisor acaba fazendo com que sempre seja possível atingir pelo

menos um estado em que todas as tarefas estão simultaneamente completadas. Tal supervisor pode ser mais restritivo do que o necessário.

Por outro lado, a marcação do modelo global pode ser definida para representar as situações em que pelo menos um objetivo de controle seja atingido. Nesse caso, um supervisor não-bloqueante garante que sempre seja possível alcançar estados em que pelo menos uma tarefa é completa. Visto que pode bloquear uma tarefa importante, tal supervisor é potencialmente menos restritivo do que o desejável.

Consequentemente, o uso de uma única marcação no modelo de SEDs com múltiplas tarefas pode comprometer a qualidade do sistema de controle sintetizado pela abordagem de RAMADGE e WONHAM (1987). Uma alternativa para o cálculo de supervisores mais refinados é o desenvolvimento de um modelo que permita fazer distinção de tarefas no processo de síntese.

1.3 Objetivos

O presente trabalho tem como objetivo geral o desenvolvimento de contribuições teóricas e práticas que fundamentem uma abordagem eficiente para a aplicação da Teoria de Controle Supervisório em problemas envolvendo sistemas compostos com múltiplas especificações e múltiplas tarefas. Para o tratamento eficiente de múltiplas especificações em problemas compostos, propõe-se o uso da arquitetura de controle modular local em conjunto com algoritmos para redução de supervisores e resolução de conflito. Nesse sentido, são apresentados os resultados práticos de uma aplicação bem sucedida da abordagem modular local na síntese de supervisores reduzidos para um problema real, envolvendo uma célula de manufatura. Além da solução matemática do problema de controle, apresenta-se a proposta de uma estrutura genérica para implementação física de sistemas de controle que, de forma hierárquica, permite a execução paralela dos supervisores modulares, comanda o funcionamento do sistema composto e faz interface entre o modelo abstrato da planta e o sistema real. Essa estrutura fundamenta a implementação clara dos supervisores modulares em Diagrama Escada no controlador lógico programável (CLP) da célula de manufatura.

Como contribuição teórica, introduz-se um novo modelo para SEDs que permite a diferenciação das classes de tarefas envolvidas no problema de controle. Faz-se então a extensão dos principais conceitos e algoritmos da TCS para esse novo modelo, com o objetivo de proporcionar uma metodologia conveniente para a síntese de supervisores que evitem o bloqueio de múltiplas tarefas de forma minimamente restritiva. Por fim, para

permitir o tratamento eficiente de sistemas compostos com múltiplos objetivos de controle e múltiplas especificações, a arquitetura de controle modular local é generalizada para a abordagem de controle multitarefa.

1.4 Estrutura

Este documento está organizado da seguinte forma: o Capítulo 2 introduz os conceitos da Teoria de Controle Supervisório no âmbito da solução do problema de controle para uma célula de manufatura real pela abordagem de controle modular local. O Capítulo 3 apresenta a proposta de uma estrutura de implementação de sistema de controle que é aplicada à célula de manufatura. No Capítulo 4 são apresentados os resultados formais que fundamentam uma abordagem para a síntese de supervisores multitarefa. Esses resultados são aplicados na resolução de três problemas envolvendo sistemas a eventos discretos multitarefa. Finalmente, o Capítulo 5 faz a extensão formal da arquitetura modular local para a abordagem multitarefa culminando na síntese de uma lógica de controle ótima e clara para um sistema flexível de manufatura. As conclusões e sugestões de pesquisas são resumidas ao final.

2. Sistemas Compostos e a Teoria de Controle Supervisório

Neste capítulo são introduzidos os principais conceitos da Teoria de Controle Supervisório (TCS), iniciada por RAMADGE e WONHAM (1987), tendo como enfoque as questões envolvendo sistemas compostos. A apresentação da TCS é feita de forma bastante objetiva e direcionada aos aspectos práticos de sistemas compostos. Um estudo mais detalhado do assunto pode ser feito a partir de WONHAM (2003), CASSANDRAS e LAFORTUNE (1999), KUMAR e GARG (1995) ou RAMADGE e WONHAM (1989).

Apresenta-se inicialmente um problema real de controle envolvendo uma célula de manufatura. Além de servir como motivação, a resolução desse problema é usada como exemplo para a aplicação dos conceitos e metodologias apresentadas nas seções seguintes. Essas seções incluem a modelagem, a síntese de supervisores pelas abordagens monolítica e modular local (QUEIROZ e CURY, 2000b), bem como a redução de supervisores.

É importante salientar que, embora o problema motivador deste trabalho seja um caso particular de sistema de manufatura, os resultados apresentados abrangem outras importantes áreas de aplicação, incluindo redes de comunicação e de computadores, sistemas de controle de tráfego e sistemas de *software* distribuídos.

2.1 Um Problema Real: Célula de Manufatura

Uma célula de manufatura (Figura 1) é composta por uma mesa circular de quatro posições (M_0), onde são efetuadas operações de furo e teste de peças metálicas, e de mais quatro dispositivos operacionais: a esteira de entrada (M_1), a furadeira (M_2), o aparelho de teste (M_3) e o manipulador robótico (M_4).

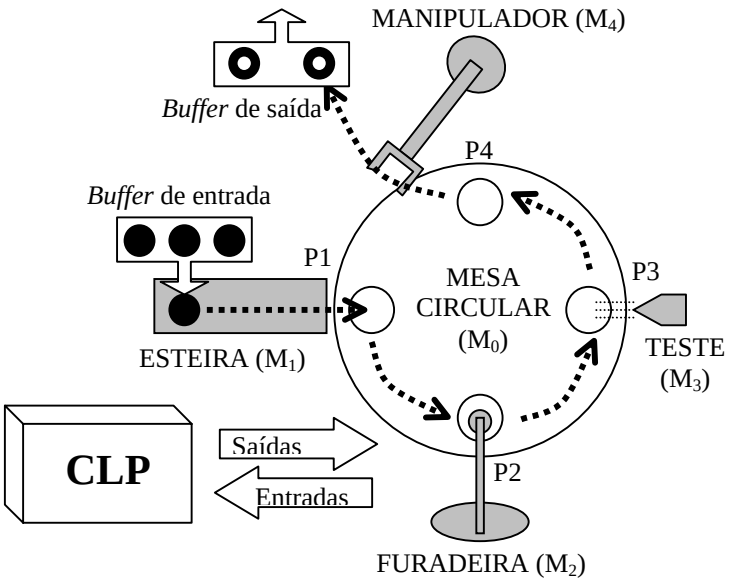


Figura 1: Célula de Manufatura

A mesa circular, comandada por um controlador lógico programável (CLP) FPC100 da Festo, foi projetada para operar peças conforme a seguinte seqüência:

1. a esteira gira até que uma peça seja posicionada em P1 (Figura 1);
2. a mesa gira 90^o;
3. a peça é furada;
4. a mesa gira 90^o;
5. a peça é testada;
6. a mesa gira 90^o;
7. manipulador robótico retira a peça da mesa.

O funcionamento isolado de cada dispositivo corresponde a uma seqüência de operações específicas, como acionamento de motores e atuadores pneumáticos e leitura de sensores. Por exemplo, inicia-se a operação da esteira ligando o motor da mesma e, quando um sensor indutivo indica a presença de peça na posição P1, termina-se o ciclo operacional desligando o motor. Essas seqüências operacionais podem ser implementadas no CLP sem maiores dificuldades. Porém, se forem executadas em paralelo sem sincronização, a célula de manufatura pode apresentar os seguintes comportamentos indesejáveis:

- operar a esteira, a furadeira, o teste ou o manipulador enquanto a mesa estiver girando;
- sobrepor peças na posição P1;

- girar a mesa sem que as peças em P2, P3 e P4 tenham sido furadas, testadas ou retiradas, respectivamente;
- furar, testar ou acionar o robô sem peças nas posições P2, P3 e P4, respectivamente;
- furar ou testar duas ou mais vezes a mesma peça;
- girar a mesa sem nenhuma peça.

O programa de controle original da mesa, fornecido pelo fabricante, permite operar em seqüência apenas uma peça por vez, ou seja, a esteira só pode ser acionada novamente depois que o manipulador retirar a peça da mesa. Esta restrição da lógica de controle evita os problemas que podem ocorrer na operação de múltiplas peças em paralelo. Entretanto, esse modo de funcionamento é muito pouco eficiente, visto que a esteira, a furadeira, o teste e o manipulador passam a maior parte do tempo parados enquanto poderiam estar operando em paralelo.

Outra possível solução de controle é levar o sistema a operar sempre com peças em todas as posições. Apesar de simples, tal lógica de controle também pode ser inconveniente em problemas nos quais a alimentação de peças no *buffer* de entrada não seja contínua. Como a mesa deve operar sempre cheia, as peças ficam em espera na mesa até que entre uma nova peça no sistema. Além da possibilidade de aumentar o tempo médio de produção, esse procedimento poderia prejudicar a qualidade de produtos perecíveis. O objetivo deste problema é, então, sintetizar um novo programa de controle para o CLP que garanta uma maior eficiência da célula de manufatura.

É importante ressaltar que nesta célula de manufatura não há sensores que indiquem a presença de peça nas posições P2, P3 e P4. Portanto, a informação sobre o estado da mesa deve ser obtida de forma indireta, conforme a evolução das operações sobre as peças a partir do estado inicial (mesa vazia).

Os problemas operacionais descritos não ocorrem nas seqüências operacionais particulares de cada subsistema, mas decorrem da descoordenação entre o início e o final das diversas seqüências. Com isso, pode-se considerar que o funcionamento particular de cada subsistema esteja implementado diretamente no CLP, de forma que o início de operação de cada aparelho possa ser comandado pelo programa de controle, que também é informado dos finais de operação. Assim, a lógica de controle pode ser expressa de forma abstrata em termos dos eventos descritos na Tabela 1.

Tabela 1: Descrição dos eventos da célula de manufatura

EQUIPAMENTO	EVENTO	DESCRIÇÃO
Mesa Giratória	α_0	Comando que inicia um giro de 90º da mesa.
	β_0	Sinal de final de operação da mesa giratória. (Uma vez iniciada a operação, não pode ser impedido.)
Esteira	α_1	Comando que inicia a operação da esteira para o depósito de uma peça no *pallet* da mesa giratória situado na posição P1.
	β_1	Sinal de final de operação da esteira automática. (Uma vez iniciada a operação, não pode ser impedido.)
Furadeira	α_2	Comando que inicia a furação da peça que estiver na posição P2.
	β_2	Sinal de final de operação da furadeira automática. (Uma vez iniciada a operação, não pode ser impedido.)
Teste	α_3	Comando que inicia o teste de uma peça situada na posição P3.
	β_3	Sinal de final de operação do teste automático. (Uma vez iniciada a operação, não pode ser impedido.)
Manipulador Robótico	α_4	Comando que inicia a retirada de uma peça do *pallet* da mesa giratória situado na posição P4.
	β_4	Sinal de final de operação do manipulador robótico. (Uma vez iniciada a operação, não pode ser impedido.)

O problema de controle a ser tratado pode então ser colocado como segue:

Problema 1: Seja a planta para a célula de manufatura definida pelo modelo que representa todas as possíveis interações no funcionamento da mesa circular, da esteira, da furadeira, do teste e do robô. Deve-se projetar uma lógica de controle a ser implementada no CLP que, restringindo a planta somente o necessário, permita a operação concorrente de 0 a 4 peças pelas 5 máquinas sem que ocorram os problemas especificados, de forma a garantir uma produção continuada, ou seja, evitando situações de bloqueio.

Apesar da complexidade do problema, com o auxílio de ferramentas de análise, como Redes de Petri (MURATA, 1989), é possível (embora não seja trivial) desenvolver de forma empírica uma lógica de controle não-bloqueante que atenda às especificações. No entanto, esse problema exige uma solução ótima, no sentido de ser minimamente restritiva, e essa propriedade só pode ser garantida se a solução for obtida por um processo formal de síntese a partir do modelo da planta e das especificações.

Assim, é fundamental que se possa obter um modelo formal para a planta e para as especificações de forma sistemática que seja adequado ao problema em questão. Cabe observar que a planta em geral, como no caso da célula de manufatura, é composta por diversos subsistemas que devem ser coordenados pelo sistema de controle. Portanto, o modelo global para a planta pode ser obtido pela composição de modelos para seus subsistemas. Da mesma forma, pode-se modelar cada especificação isoladamente em termos apenas dos subsistemas relevantes para a sua definição. Deste modo, o modelo para

o comportamento especificado também pode ser obtido pela composição de modelos elementares.

O objetivo da teoria a ser apresentada nas próximas seções é resolver problemas como o descrito acima. A colocação do problema justifica a metodologia a ser apresentada como sendo composta pelas seguintes fases:

1. obtenção de um modelo para a planta a ser controlada;
2. obtenção de um modelo de representação das especificações;
3. síntese de uma lógica de controle não-bloqueante e ótima.

2.2 Modelagem de Sistemas a Eventos Discretos

Sistemas a Eventos Discretos (SEDs) são sistemas dinâmicos cuja mudança de estado ocorre em pontos discretos do tempo, em decorrência de eventos isolados. Esses eventos ocorrem em geral em intervalos de tempo irregulares e desconhecidos. São exemplos de eventos discretos o início e final de operação dos equipamentos da célula de manufatura.

Em contrapartida aos sistemas dirigidos pelo tempo, que são classicamente modelados por equações diferenciais e a diferenças, o comportamento dessa classe de sistemas pode ser matematicamente modelado por linguagens, que são conjuntos de cadeias finitas de símbolos representando todas as seqüências de eventos admitidas pelo sistema (HOPCROFT e ULLMAN, 1979). Nesta seção, são introduzidos os conceitos preliminares sobre linguagens, bem como uma abordagem para a modelagem de sistemas compostos que fundamenta a metodologia de controle modular local apresentada neste capítulo.

É importante ressaltar que a representação matemática apresentada a seguir não admite a ocorrência simultânea de dois ou mais eventos distintos, de forma que é assumido que os sistemas tratados apresentem evolução seqüencial. Essa hipótese é bastante razoável visto que, em tempo contínuo, a probabilidade de dois eventos distintos ocorrerem exatamente no mesmo tempo é praticamente nula. Contudo, em sistemas digitais o processamento de informações é feito em intervalos de tempo, dentro dos quais múltiplos eventos podem ocorrer. Para a célula de manufatura em questão, a hipótese é garantida impondo-se ao CLP o processamento seqüencial dos sinais representando os eventos, conforme a estrutura de controle a ser apresentada no Capítulo 3.

Observa-se também que a modelagem apresentada não considera os instantes de tempo em que os eventos ocorrem, mas apenas a ordem em que acontecem. Todavia, a abordagem de Ramadge e Wonham pode ser estendida para suportar um modelo temporizado (BRANDIN e WONHAM, 1993).

2.2.1 Linguagens e Geradores

Para a modelagem matemática de SEDs, os eventos que podem ocorrer são associados a etiquetas (símbolos). Define-se o alfabeto Σ como o conjunto finito de etiquetas de eventos que ocorrem num SED a controlar, também chamado de planta. Seja Σ^* o conjunto de todas as cadeias finitas (palavras) de elementos do conjunto Σ, incluindo a cadeia vazia ε. Seja s uma palavra de Σ^*. Uma palavra $u \in \Sigma^*$ é um prefixo de s se houver $v \in \Sigma^*$ tal que $uv = s$.

Na Teoria de Controle Supervisório proposta por RAMADGE e WONHAM (1989), considera-se que todos os eventos são gerados espontaneamente pelo sistema em malha aberta, denominado planta, e que a ocorrência de alguns eventos pode ser evitada. Assim, o alfabeto de eventos Σ é particionado em eventos controláveis e não-controláveis. Eventos controláveis $\Sigma_c \subseteq \Sigma$ são aqueles cuja ocorrência pode ser desabilitada por agentes externos, como é o caso do início de operação das máquinas na célula de manufatura da Seção 2.1. Eventos não-controláveis $\Sigma_u \subseteq \Sigma$ são aqueles que não podem ser diretamente impedidos de ocorrer e, por isso, são considerados permanentemente habilitados. O final de operação de uma máquina da célula de manufatura (Seção 2.1) é um exemplo de evento não-controlável. Em geral, sinais de comando são modelados por eventos controláveis, visto que basta não enviar o sinal para que sua ocorrência seja desabilitada. Da mesma forma, sinais de resposta (sinais de sensores, por exemplo) correspondem a eventos que não podem ser diretamente desabilitados.

Uma linguagem sobre o alfabeto Σ é um subconjunto de Σ^*. Assim, o comportamento gerado por um SED de alfabeto Σ pode ser modelado por uma linguagem $L \in Pwr(\Sigma^*)$[1], representando todas as cadeias finitas de eventos que o SED pode gerar, ou seja, que são fisicamente realizáveis. Para algumas cadeias de L, o SED pode atingir certos objetivos de controle (tarefas), como por exemplo a geração de um produto num sistema de manufatura. Assim, define-se o comportamento marcado de um SED como a linguagem $L_m \subseteq L$ contendo todas as cadeias que completam tarefas do sistema.

[1] Seja A um conjunto. $Pwr(A)$ é o conjunto de todos subconjuntos de A (conjunto potência de A).

Essas linguagens podem ser representadas por geradores (HOPCROFT e ULLMAN, 1979). Um gerador é uma quíntupla $G = (Q, \Sigma, \delta, q_0, Q_m)$, onde:

- Q é um conjunto de estados;
- Σ é o alfabeto de eventos;
- $\delta: Q \times \Sigma \rightarrow Q$, a função de transição, é uma função parcial definida em cada estado de Q para um subconjunto de Σ;
- $q_0 \in Q$ é o estado inicial;
- $Q_m \subseteq Q$ é o conjunto de estados marcados.

Pode-se associar ao gerador G uma função de eventos ativos $\Gamma: Q \rightarrow Pwr(\Sigma)$, que, para cada estado $q \in Q$, indique o conjunto de eventos que possam ocorrer, ou seja, $\Gamma(q) = \{\sigma: \sigma \in \Sigma \text{ e } \delta(q, \sigma)!\ [2]\}$.

A função de transição δ pode ser estendida para cadeias de eventos como a função $\delta: Q \times \Sigma^* \rightarrow Q$ tal que, para $q \in Q$, $s \in \Sigma^*$ e $\sigma \in \Sigma$, $\delta(q, \varepsilon) = q$ e $\delta(q, s\sigma) = \delta(\delta(q, s), \sigma)$, sempre que $q' = \delta(q, s)$ e $\delta(q', \sigma)$ estiverem ambas definidas.

O comportamento de um gerador G é caracterizado por dois subconjuntos de Σ^* chamados de *linguagem gerada* de G (todas as seqüências de eventos que a planta pode gerar), denotado por $L(G)$, e de *linguagem marcada* de G (seqüências representando tarefas completas), denotado por $L_m(G)$. Formalmente, a linguagem gerada $L(G)$ é definida por $L(G) = \{s: \ s \in \Sigma^* \text{ e } \delta(q_0, s)!\}$ e a linguagem marcada $L_m(G)$ é definida por $L_m(G) = \{s: s \in L(G) \text{ e } \delta(q_0, s) \in Q_m\}$. Dois geradores G_1 e G_2 são equivalentes se gerarem e marcarem as mesmas linguagens, isto é, se $L(G_1) = L(G_2)$ e $L_m(G_1) = L_m(G_2)$. Quando uma linguagem puder ser marcada for um gerador com um número finito de estados, diz-se que a linguagem é regular (HOPCROFT e ULLMAN, 1979).

Os geradores podem ser ilustrados por diagramas de transição de estado, que são grafos direcionados cujos nós representam os estados e os ramos representam os eventos. Nesses diagramas, os estados marcados são caracterizados por nós desenhados com linhas duplas e o estado inicial é identificado por uma seta. Os eventos controláveis são representados por ramos interceptados. Por exemplo, a Figura 2 ilustra o gerador que marca a linguagem $\{\varepsilon, \alpha_1\beta_1, \alpha_1\beta_1\alpha_1\beta_1, ...\}$, onde o evento α_1 é controlável e β_1 é não-controlável.

[2] $\delta(q, \sigma)!$ é uma abreviação de "$\delta(q, \sigma)$ está definida".

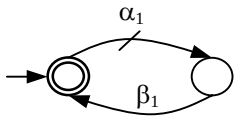


Figura 2: Exemplo de gerador

2.2.2 Operações sobre linguagens e autômatos

O prefixo-fechamento, ou simplesmente fechamento, de uma linguagem L é definido por $\overline{L} := \{u: \exists v \in \Sigma^* \wedge uv \in L\}$. Essa operação retorna o conjunto de todos os prefixos (cadeias incompletas) das palavras de L. Diz-se que L é prefixo-fechada se $L = \overline{L}$. Claramente, o comportamento gerado por um SED é prefixo-fechado, mas o comportamento marcado nem sempre o é.

A noção de bloqueio num gerador está relacionada com a idéia de se executar uma seqüência de eventos a partir da qual não seja possível completar uma tarefa. Diz-se que um gerador G é não-bloqueante se $\overline{L_m(G)} = L(G)$, isto é, se qualquer seqüência de eventos gerada for o prefixo de pelo menos uma tarefa completa.

Um estado $q \in Q$ é chamado de acessível se $\exists s \in \Sigma^*$ tal que $\delta(q_0, s) = q$, ou seja, se existir uma cadeia aceita por G que, partindo do estado inicial, alcance o estado q. É chamado de coacessível se $\exists s \in \Sigma^*$ tal que $\delta(q, s) \in Q_m$, isto é, se houver uma cadeia aceita por G que, partindo de q, alcance um estado marcado. Diz-se que um gerador é coacessível se todos seus estados forem coacessíveis e que é aparado (ou *trim*) se todos seus estados forem acessíveis e coacessíveis. Define-se como $Ac(G)$ a operação que elimina todos os estados não-acessíveis de G, bem como as transições que sairiam ou levariam a estes estados. Já a operação $Tr(G)$ elimina todos os estados não-acessíveis ou não-coacessíveis de G. Portanto, G é não-bloqueante se $Ac(G)$ for coacessível.

Sejam Σ e Σ_i conjuntos de eventos com $\Sigma_i \subset \Sigma$. P_i: $\Sigma^* \rightarrow \Sigma_i^*$, a projeção natural de Σ^* para Σ_i^*, é definida recursivamente por:

$$P_i(\varepsilon) = \varepsilon$$
$$P_i(e) = \begin{cases} \varepsilon & \text{se } e \notin \Sigma_i \\ e & \text{se } e \in \Sigma_i \end{cases}$$
$$P_i(ue) = P_i(u)P_i(e) \text{ onde } u \in \Sigma^*;\ e \in \Sigma$$

O conceito de projeção natural pode ser estendido para linguagens regulares como $P_i(L) = \{u_i \in \Sigma_i^*: u_i = P_i(u) \text{ para algum } u \in L\}$. A projeção inversa é, então, definida como $P_i^{-1}(L_i) = \{u \in \Sigma^* : P_i(u) \in L_i\}$.

Sejam $L_i \subseteq \Sigma_i^*$, $i=1,...,n$. Seja $\Sigma = \cup_{i=1}^{n}\Sigma_i$ e $P_i: \Sigma^* \rightarrow \Sigma_i^*$. Define-se o produto síncrono de linguagens $\|_{i=1}^{n}L_i \subseteq \Sigma^*$ como:

$$\|_{i=1}^{n}L_i = \cap_{i=1}^{n}P_i^{-1}(L_i) = \{u \in \Sigma^*: \wedge_{i=1}^{n}P_i(u) \in L_i\}.$$

Sejam geradores G_i, $i=1,...,n$. A composição síncrona $G = \|_{i=1}^{n}G_i$ é obtida fazendo-se a evolução em paralelo dos n geradores G_i, na qual um evento comum a múltiplos geradores só é executado se todos os geradores que contiverem este evento o executarem simultaneamente. Quando não houver eventos comuns, chama-se essa operação de composição assíncrona. As linguagens resultantes da composição síncrona são caracterizadas por:

$$L(G) = \|_{i=1}^{n}L(G_i);$$
$$L_m(G) = \|_{i=1}^{n}L_m(G_i).$$

Formalmente, o produto síncrono de dois geradores $G_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, Q_{m1})$ e $G_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, Q_{m2})$, com respectivas funções de eventos ativos Γ_1 e Γ_2, é definido como o gerador:

$$G_1 \| G_2 := Ac(Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta, (q_{01}, q_{02}), Q_{m1} \times Q_{m2}),$$

onde

$$\delta((q_1, q_2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)), \text{ se } \sigma \in \Gamma_1(q_1) \cap \Gamma_2(q_2) \\ (\delta_1(q_1, \sigma), q_2), \text{ se } \sigma \in \Gamma_1(q_1) - \Sigma_2 \\ (q_1, \delta_2(q_2, \sigma)), \text{ se } \sigma \in \Gamma_2(q_2) - \Sigma_1 \\ \text{indefinida, senão} \end{cases} ;$$

A função de eventos ativos Γ para $G_1 \| G_2$ pode ser obtida por:

$$\Gamma((q_1, q_2)) = [\Gamma_1(q_1) \cup (\Sigma_2 - \Sigma_1)] \cap [\Gamma_2(q_2) \cup (\Sigma_1 - \Sigma_2)]$$
$$= [\Gamma_1(q_1) \cap \Gamma_2(q_2)] \cup [\Gamma_1(q_1) - \Sigma_2] \cup [\Gamma_2(q_2) - \Sigma_1].$$

Pode-se mostrar que o produto síncrono é uma operação comutativa e associativa, respeitadas eventuais mudanças nos nomes dos estados do gerador resultante. Assim, o produto de múltiplos geradores pode ser obtido iterativamente pelo produto síncrono de geradores dois a dois.

2.2.3 Modelagem de Sistemas Compostos

O primeiro passo para a síntese de supervisores conforme a Teoria de Controle Supervisório é a obtenção de um modelo em termos de geradores que represente o funcionamento da planta. Em geral, os sistemas de maior complexidade são compostos por

diversos subsistemas concorrentes interagindo entre si (RAMADGE e WONHAM, 1989; WILLNER e HEYMANN, 1991). No projeto desses sistemas, chamados de sistemas compostos, a modelagem das diversas partes envolvidas, G'_i, $i = 1,...,n'$, é um passo intermediário na representação do comportamento conjunto do sistema. Como a composição de subsistemas assíncronos provoca a explosão do número de estados do sistema global, a representação do comportamento global do sistema por um único gerador $G = \|_{i=1}^{n'} G'_i$ acaba sendo uma operação indesejável no processo de síntese de supervisores, que tem complexidade computacional polinomial no número de estados da planta.

Este é o caso da célula de manufatura, que é composta pelo funcionamento concorrente da mesa giratória (M_0), da esteira (M_1), da furadeira (M_2), do teste (M_3) e do robô (M_4). Para a obtenção de um modelo global que represente o funcionamento em malha aberta da célula, faz-se inicialmente a modelagem independente de cada dispositivo, em termos dos eventos de início e final de operação. Assim, considera-se que os subsistemas M_i, $i=0,...,4$, têm suas operações iniciadas respectivamente pelos eventos controláveis α_i, $i=0,...,4$, e terminam de operar com o evento não-controlável β_i, $i=0,...,4$. As máquinas M_i, $i=0,...,4$, podem então ser modeladas respectivamente pelos geradores G_i, $i=0,...,4$, representados na Figura 3.

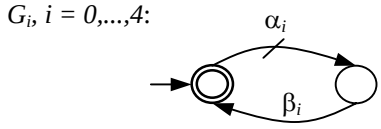


Figura 3: Gerador para M_i, i = 0,...,4

A planta da célula de manufatura pode então ser representada de forma monolítica por um gerador G de 32 estados e 160 transições obtido pela composição assíncrona dos 5 modelos dos dispositivos $G = \|_{i=0}^{4} G_i$, que pode ser realizada através da ferramenta computacional CTCT (WONHAM, 2003). Conforme será apresentado na Seção 2.3, a aplicação da abordagem de controle monolítico sobre esse modelo, além do alto esforço computacional, acaba gerando um supervisor com um grande número de estados e transições.

Uma alternativa à representação monolítica da planta composta é a representação por sistema-produto (RAMADGE, 1989; RAMADGE e WONHAM, 1989), que consiste na modelagem do sistema através de um conjunto de subsistemas completamente assíncronos entre si. Em muitos casos, é natural que cada subsistema possua um alfabeto exclusivo. Assim, a sincronização entre os diversos módulos é feita apenas pela ação de controle. Este modelo representa a estrutura descentralizada natural de operações

concorrentes em malha aberta para um sistema composto e serve de plataforma para a abordagem de controle modular local, a ser apresentada na Seção 2.4.

Nos casos em que a modelagem inicial da planta contém subsistemas que compartilham eventos, pode-se obter uma representação por sistema-produto (RSP) mais refinada possível, fazendo-se a composição dos subsistemas síncronos originais (que têm eventos em comum). Cria-se, assim, um conjunto com o maior número possível de subsistemas assíncronos distintos que modela o sistema global em malha aberta.

No caso da célula de manufatura o conjunto de modelos $\{G_i,\ i = 0,...,4\}$ é uma representação por sistema-produto, uma vez que os subsistemas não têm qualquer evento em comum.

2.2.4 Modelagem das especificações

O funcionamento de um sistema a eventos discretos em malha aberta inclui uma série de cadeias de eventos indesejáveis, resultantes da interação descoordenada dos diversos subsistemas na ausência de controle. Essas cadeias podem levar a estados proibidos da planta, representando a ocorrência de problemas inadmissíveis, como, por exemplo, a colisão entre um robô e um veículo autoguiado num sistema de manufatura. Pode ainda ser o caso dessas cadeias violarem o ordenamento desejado de eventos específicos, como no caso de justiça no acesso a recursos. Para expressar matematicamente o comportamento desejado ao sistema de forma sistemática, definem-se especificações genéricas de controle como linguagens sobre subconjuntos de eventos relevantes da planta. Pode-se mostrar (WONHAM, 2003) que especificações de estados proibidos (predicados sobre Q) podem ser equivalentemente representadas como linguagens.

Outro tipo de especificação é a imposição de não-bloqueio ao sistema controlado, que garante que sempre seja possível completar tarefas. A definição dessa especificação é feita pela marcação da planta, que delimita os objetivos de controle. Uma vez que as especificações genéricas podem adicionar novos estados (memória) ao modelo, pode ser bastante conveniente usar a marcação das especificações para refinar a definição de tarefa completa. Contudo, o uso desse artifício deve ser levado em consideração no processo de síntese de supervisores, conforme será discutido na próxima seção.

Por exemplo, o objetivo de controle marcado pela planta da célula de manufatura é que se alcance um estado em que todas as máquinas estejam simultaneamente paradas. Marcando apenas os estados iniciais das especificações genéricas apresentadas a seguir, evita-se que sejam consideradas tarefas completas as situações em que as máquinas estejam paradas com peças na mesa.

O autômato $E_{gen,a}$ descrito na Figura 4 modela uma especificação genérica para a célula de manufatura que garante que a mesa não vai girar à toa, isto é, sem ao menos uma peça bruta em P1 ou uma peça furada em P2 ou uma peça testada em P3. De acordo com essa especificação, pelo menos um dos eventos β_1, β_2 e β_3, representando respectivamente o depósito de peças em P1, P2 e P3, deve preceder a ocorrência do evento α_0 que inicia o giro da mesa.

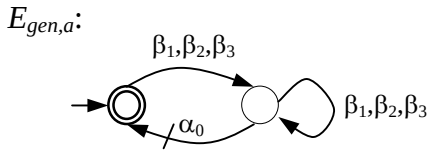


Figura 4: Gerador para a especificação $E_{gen,a}$

As quatro especificações de segurança que impedem a mesa de girar enquanto a esteira ($E_{gen,b1}$), a furadeira ($E_{gen,b2}$), o teste ($E_{gen,b3}$) ou/e o manipulador ($E_{gen,b4}$) estiverem operando podem ser descritas pelo gerador $E_{gen,bi}$ de índice i da Figura 5. De forma genérica, a especificação $E_{gen,bi}$ garante que, quando a mesa (M_0) ou a máquina M_i começar a operar, os eventos α_0 e α_i não poderão mais ocorrer até que seja sinalizado o fim de operação.

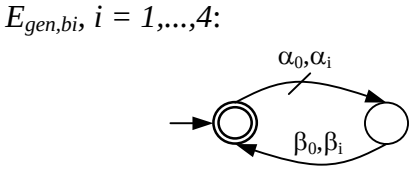


Figura 5: Gerador para as especificações $E_{gen,bi}$, $i = 1,...,4$

Os possíveis problemas decorrentes do fluxo de múltiplas peças na mesa podem ser evitados pelas especificações: $E_{gen,c1}$, relativa à movimentação de peças brutas entre as posições P1 e P2; $E_{gen,c2}$, para a manipulação de peças furadas entre P2 e P3; e $E_{gen,c3}$, correspondente ao fluxo de peças testadas entre P3 e P4. Para isso, a especificação $E_{gen,c1}$ evita sobrepor peças em P1, furar sem peça bruta em P2 e girar a mesa com peça bruta em P2. Já a especificação $E_{gen,c2}$ proíbe furar duas vezes a mesma peça, testar sem peça furada em P3 e girar a mesa com peça furada e não testada em P3, enquanto $E_{gen,c3}$ impede testar duas vezes a mesma peça, acionar o manipulador sem peça em P4 e girar a mesa com peça em P4. Como as especificações têm a mesma estrutura, pode-se ilustrá-las pelo modelo indexado em i, $E_{gen,ci}$, da Figura 6. Os pares de números dentro dos estados indicam a quantidade de peças brutas (se $i = 1$), furadas (se $i = 2$) ou testadas (se $i = 3$) nas posições P_i e P_{i+1}, respectivamente.

$E_{gen,ci}$, i = 1,2,3:

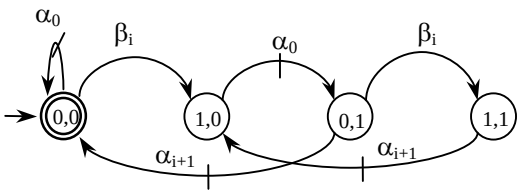


Figura 6: Gerador para as especificações $E_{gen,ci}$, i = 1,2,3

No caso geral, para $j=1,...,m$, sejam as especificações genéricas locais $E_{gen,j}$ definidas respectivamente em subconjuntos de eventos da planta G, $\Sigma_{gen,j} \subseteq \Sigma$. Pode-se obter um modelo monolítico para as especificações genéricas por $E_{gen} = ||_{j=1}^{m} E_{gen,j}$. Com isso, o comportamento desejado ao sistema pode ser expresso por uma única linguagem calculada por $K_{global} = E_{gen} \,||\, L_m(G)$, denominada especificação global.

Para a célula de manufatura da Seção 2.1, a especificação genérica E_{gen}, calculada através da ferramenta CTCT (WONHAM, 2003) pela composição das especificações $E_{gen,x}$, $x \in \{a, b1, b2, b3, b4, c1, c2, c3\}$, é marcada por um gerador aparado de 296 estados e 745 transições. Assim, o comportamento desejado pode ser expresso de forma monolítica pela linguagem $K_{global} \subset L_m(G)$, marcada por um gerador aparado de 151 estados e 350 transições.

2.3 Controle Monolítico

O objetivo do controle supervisório monolítico, segundo RAMADGE e WONHAM (1989), é projetar um único supervisor S cuja função é impedir a ocorrência de eventos controláveis, conforme a seqüência de eventos observados na planta G, de forma que o sistema em malha fechada não transgrida o comportamento desejado descrito pela especificação global K_{global} e seja não-bloqueante. A Figura 7 ilustra a estrutura de malha fechada representando a interação de um supervisor monolítico com a planta.

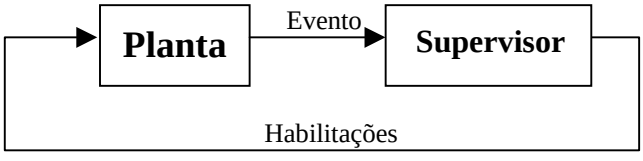


Figura 7: Esquema de controle monolítico

Como o supervisor S tem uma ação habilitadora (ou desabilitadora, dualmente) de eventos, diz-se que S é um controle de natureza permissiva. Assim, a entrada de controle na planta (habilitações) é definida como o conjunto de eventos habilitados pelo supervisor

num determinado instante. Para que a entrada de controle seja admissível, é necessário que contenha todos os eventos não-controláveis que possam ocorrer na planta. O conjunto de todas as entradas de controle válidas associado ao gerador G é denominado estrutura de controle. A função do supervisor, então, é selecionar entradas de controle para cada evento observado, de forma que o sistema não saia do comportamento especificado e tampouco entre em situações de bloqueio.

2.3.1 Supervisor

O supervisor pode ser definido formalmente como uma função S: $L(G) \rightarrow Pwr(\Sigma)$, que associa um conjunto de eventos habilitados a cada cadeia de eventos observada na planta. De forma dual, os eventos a serem desabilitados pela estrutura de controle podem ser calculados como os eventos que podem ocorrer na planta e não estão habilitados por S. Dessa forma, os eventos desabilitados após a ocorrência de $s \in L(G)$ são dados por $\Gamma(\delta(q_0, s)) - S(s)$, onde Γ é a função de eventos ativos da planta.

Diz-se que um supervisor S é admissível se não implicar a desabilitação de eventos não-controláveis, isto é, se

$$\forall s \in L(G), \Sigma_u \cap \Gamma(\delta(q_0, s)) \subseteq S(s).$$

Define-se por S/G o gerador que representa S controlando G. Com isso, a linguagem gerada por S/G é dada por:

1. $\varepsilon \in L(S/G)$;

2. $s\sigma \in L(S/G) \Leftrightarrow (s \in L(S/G)) \wedge (s\sigma \in L(G)) \wedge (\sigma \in S(s))$.

O comportamento marcado original que permanece após a supervisão é obtido por $L_m(S/G) = L(S/G) \cap L_m(G)$.

Diz-se que um supervisor S é próprio (ou não-bloqueante) para G se garantir o não-bloqueio do sistema em malha fechada, isto é, se $\overline{L_m(S/G)} = L(S/G)$.

Na prática, um supervisor S pode ser representado por um gerador H, cujo alfabeto esteja contido em Σ e cujas mudanças de estado sejam ditadas pela ocorrência de eventos na planta, de modo que a ação de controle sobre G esteja implícita na estrutura de transição de H. Para isso, o supervisor H é construído de forma que as transições não habilitadas por S não apareçam na estrutura de transição de estados de H e que as transições habilitadas por S apareçam na estrutura de H. Portanto, a ação de controle do supervisor H, definida para cada estado do gerador correspondente, é desabilitar os eventos ativos (fisicamente possíveis) da planta que não possam ocorrer em H após uma cadeia de eventos observada.

Caso S seja admissível, o funcionamento do sistema controlado H/G, equivalente a S/G, pode ser descrito pelo SED correspondente à composição síncrona de H e G, isto é, $H \parallel G$. Assim, os comportamentos gerado e marcado do sistema controlado podem ser obtidos diretamente do gerador $H \parallel G$, ou seja, $L(H/G) = L(H \parallel G)$ e $L_m(H/G) = L_m(H \parallel G)$. Neste caso, para ser coerente com a idéia de que todos os estados marcados da planta que "sobrevivem" com a supervisão permanecem marcados no sistema controlado, deve-se marcar todos os estados de H.

No entanto, observa-se na prática que, em muitos casos, a interpretação de tarefa cumprida associada à marcação da planta G pode ser refinada de acordo com as especificações. Por exemplo, na planta da célula de manufatura da Seção 2.1, qualquer estado em que todas as máquinas estiverem paradas é marcado. Porém, ao se especificar o fluxo de peças na mesa, definem-se como tarefas completas apenas as seqüências de eventos que levam o sistema a um estado em que todas as máquinas estejam paradas e a mesa esteja vazia. Com isso, um estado marcado do modelo da planta pode ou não representar o cumprimento de uma tarefa especificada, dependendo da seqüência de eventos pela qual for alcançado. Nesses casos, faz sentido que o supervisor possa indicar quais estados marcados pela planta realmente completam uma tarefa especificada. Para fazer isso, marcam-se no supervisor H apenas os estados marcados pela especificação (WONHAM, 2003). Desse modo, além de restringir o comportamento da planta, o supervisor assim definido tem a função de desmarcar estados, ou seja, uma tarefa do sistema em malha fechada H/G é considerada completa somente se for marcada pela planta e pelo supervisor – neste documento classificado como desmarcador.

2.3.2 Existência de Supervisores

Os conceitos apresentados na seqüência são fundamentais para definir as condições para a existência de um supervisor não-bloqueante que atenda a uma dada especificação.

Uma linguagem $K \subseteq L(G)$ é uma sublinguagem controlável de $L(G) \subseteq \Sigma^*$ (ou controlável e.r.a[3] G) se $\bar{K}\Sigma_u \cap L(G) \subseteq \bar{K}$. Isso quer dizer que a ocorrência de um evento não-controlável e fisicamente possível, após qualquer cadeia de $\bar{K}$, mantém a seqüência no conjunto $\bar{K}$.

Diz-se que uma linguagem K é $L_m(G)$-fechada se $K = \bar{K} \cap L_m(G)$, isto é, se todos os seus prefixos que são palavras de $L_m(G)$ forem também palavras de K.

[3] e.r.a é uma abreviação para a expressão "em relação a".

Conforme demonstrado por RAMADGE e WONHAM (1987), as condições necessárias e suficientes para a existência de um supervisor próprio S que controle a planta G de forma a atender uma dada especificação global $K_{global} \subseteq L_m(G)$, ou seja $L_m(S/G) = K_{global}$, são a controlabilidade e o $L_m(G)$-fechamento de K_{global}. No caso do supervisor poder desmarcar estados, apenas a controlabilidade de K_{global} é necessária e suficiente para que exista H não-bloqueante tal que $L_m(H/G) = K_{global}$.

No caso da célula de manufatura em questão, pode-se usar o CTCT para mostrar que a especificação global K_{global} é controlável e.r.a G, porém não é $L_m(G)$-fechada. A falta de $L_m(G)$-fechamento ocorre pela própria construção de K_{global} através da composição de especificações genéricas com a planta. Uma vez que as especificações genéricas não são prefixo-fechadas, alguns prefixos de K_{global} que levam a planta a um estado marcado não são considerados marcados pela especificação. Por exemplo, a seqüência de eventos $\alpha_1\beta_1$ leva a planta G a um estado marcado (todas as máquinas paradas), mas, pelo fato de haver uma peça em P1, essa tarefa não é considerada completa pela especificação genérica $E_{gen,c1}$. Neste caso, faz sentido considerar que o supervisor tem o poder de desmarcar estados, de forma que é possível obter um supervisor desmarcador monolítico H_{mon} tal que $L_m(H_{mon}/G) = K_{global}$. Assim, o próprio gerador aparado de 151 estados e 350 transições (Figura 8) que marca K_{global} pode ser tomado como supervisor H_{mon} para G.

Seja M uma linguagem contida na linguagem gerada por G. O conjunto de sublinguagens de M controláveis e.r.a G é denotada por $C(M, G) := \{K: K \subseteq M$ e K é controlável e.r.a $G\}$ e, por ser fechada sobre união e não-vazia ($\varnothing \in C(M, G)$), contém um (único) elemento supremo, chamado $SupC(M, G)$ (WONHAM e RAMADGE, 1987). É provado (WONHAM, 2003) que, se a linguagem M for $L_m(G)$-fechada, $SupC(M, G)$ também será.

Nem sempre é possível construir um supervisor que restrinja o comportamento do sistema a uma linguagem especificada de forma exata. Entretanto, quando um comportamento especificado não é controlável, é possível projetar um supervisor próprio que atenda às especificações de forma minimamente restritiva. Neste caso, o controle monolítico objetiva sintetizar um supervisor S para uma linguagem especificada $K \subseteq L(G)$, tal que $L_m(S/G) = SupC(K, G)$. Se a restrição da linguagem $SupC(K, G)$ não for aceitável, diz-se que o problema de controle não tem solução.

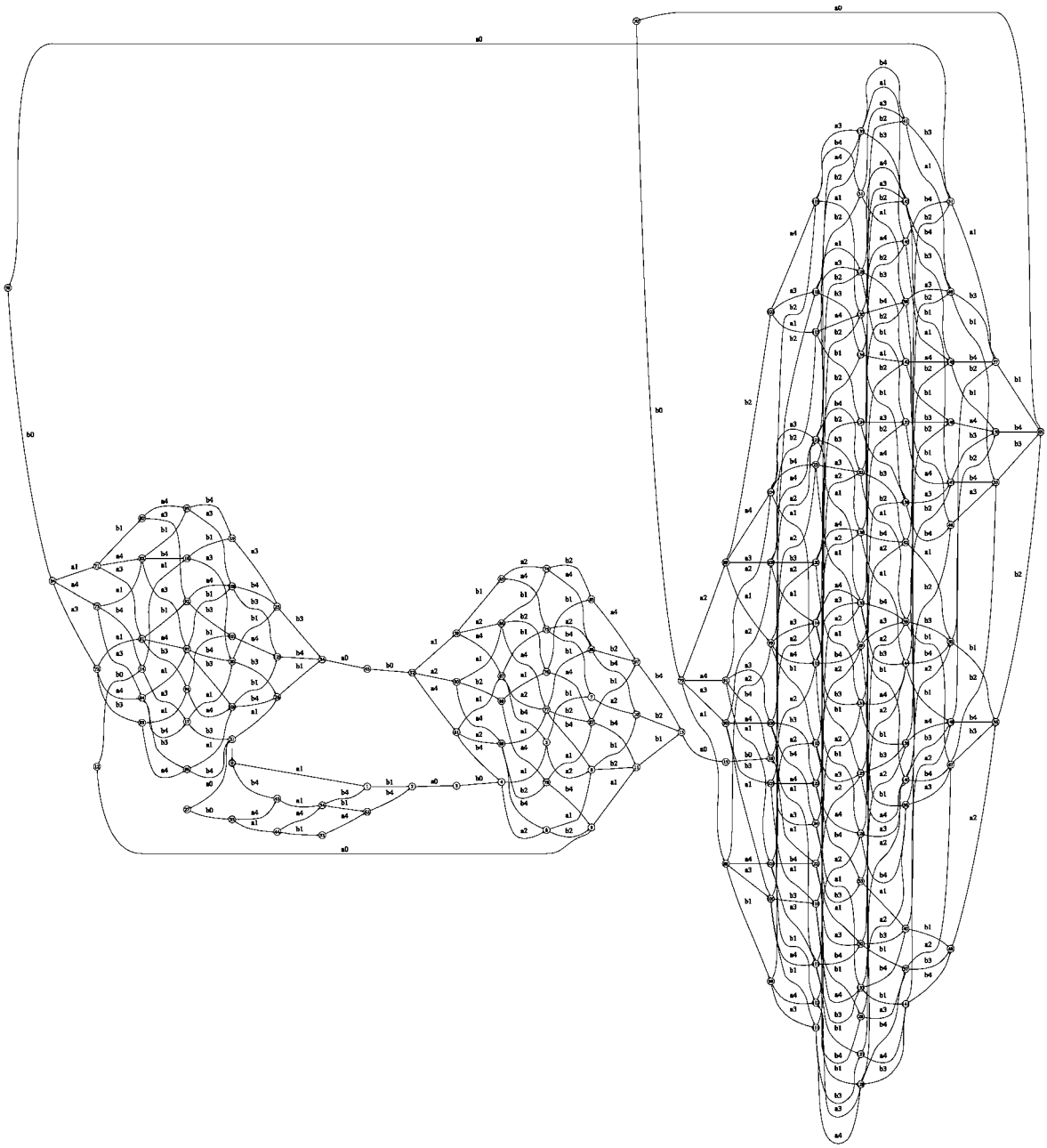


Figura 8: Supervisor monolítico

Portanto, na teoria de controle supervisório, um passo fundamental para a síntese de supervisores é o cálculo da máxima linguagem controlável contida em uma linguagem que representa o comportamento desejado. Um algoritmo formal é proposto por WONHAM e RAMADGE (1987). A complexidade desse cálculo, embora polinomial no número de estados do modelo da planta e da especificação, é um fator limitante em aplicações pois o número de estados que representa o sistema cresce exponencialmente com o número de elementos do sistema composto.

Por outro lado, mesmo quando o equipamento computacional oferece condições suficientes para a síntese de um supervisor monolítico ótimo, como no caso da célula de manufatura, o grande tamanho das soluções pode representar um obstáculo à aplicação dos resultados. Em muitos casos, a plataforma do sistema de controle (geralmente CLP) pode ser desprovida de memória e capacidade de processamento suficientes para a implementação de supervisores com muitos estados e transições. Além disso, o programa de controle representando a ação de supervisores imensos costuma ser incompreensível ao operador humano, o que acaba inviabilizando a implementação, depuração e alteração manual do programa. Isso pode justificar o fato de que, duas décadas após a introdução da Teoria de Controle Supervisório, sejam poucas as aplicações existentes na indústria.

Entretanto, pesquisas recentes têm promovido o desenvolvimento de algoritmos para calcular supervisores de menor tamanho e abordagens para reduzir a complexidade computacional do processo de síntese. Nesse sentido, as próximas seções mostram que uma alternativa viável para sistemas compostos é o uso de uma arquitetura modular local de controle (QUEIROZ e CURY, 2000a) em conjunto com algoritmos de redução de supervisores (SU e WONHAM, 2004).

2.4 Redução de Supervisores

Diz-se que dois supervisores S_1 e S_2 são equivalentes, se suas ações de controle sobre a planta G produzirem o mesmo comportamento, ou seja, se S_1/G for equivalente a S_2/G. No caso de S_1 e S_2 serem representados por geradores H_1 e H_2, respectivamente, a equivalência é verificada quando $L(H_1 \parallel G) = L(H_2 \parallel G)$ e $L_m(H_1 \parallel G) = L_m(H_2 \parallel G)$. Diz-se, então, que H_2 é um supervisor reduzido para H_1 se ambos forem equivalentes e se o número de estados de H_1 (denotado por $|H_1|$) for menor ou igual a $|H_2|$.

Segundo VAZ e WONHAM (1986), dado um supervisor na forma de um gerador H, pode-se obter um supervisor reduzido H_r agregando-se os estados de $H \parallel G$ em blocos, não necessariamente disjuntos. No processo de agregação, deve-se garantir que as desabilitações dos estados de um bloco não entrem em conflito com os eventos habilitados em outros estados do mesmo bloco e que o supervisor reduzido, representando a estrutura de transição entre os blocos, seja determinístico. Caso H seja desmarcador, deve-se ter o cuidado adicional de não agrupar estados "marcadores" (que são marcados em H e nos respectivos estados de $H \parallel G$) com estados "desmarcadores" (que não são marcados em H e, para algum estado marcado de G, o respectivo estado de $H \parallel G$ não é marcado), de forma a preservar a igualdade das linguagens marcadas. VAZ e WONHAM (1986) provam que existe um tamanho mínimo de supervisor reduzido para um dado H, mas pode haver

múltiplos supervisores com o número mínimo de estados. Um algoritmo formal para minimização de supervisores é apresentado por VAZ e WONHAM (1986), que consiste em comparar todas as possíveis combinações de blocos. Claramente, esse algoritmo tem complexidade exponencial e, assim, é realizável apenas para geradores de pequeno porte.

Observa-se, no entanto, que a busca por um tamanho mínimo pode ser desnecessária em aplicações, quando se pode obter supervisores suficientemente reduzidos através de algoritmos de menor complexidade. SU e WONHAM (2001 e 2004) apresentam um algoritmo de complexidade polinomial para redução de supervisores que também provê um limite inferior (conservador) para o número mínimo de estados. Em alguns casos, o tamanho do supervisor reduzido é igual ao limite inferior calculado e, portanto, mínimo. Este algoritmo é implementado na ferramenta CTCT (WONHAM, 2003), que tipicamente converge em tempo razoável somente para supervisores com menos de 1000 estados. Outro algoritmo para redução de supervisores com complexidade polinomial é proposto por MINHAS (2002).

Usando o algoritmo de SU e WONHAM (2001), reduz-se o supervisor monolítico H_{mon} para a célula de manufatura de 151 estados (Figura 8) para o supervisor H_{red} com 62 estados apresentado na Figura 9. Observa-se que, apesar de proporcionar uma economia de memória maior que 50%, a lógica de controle implícita em H_{red} é de difícil compreensão. O algoritmo calcula que o número mínimo de estados é maior que 47. Portanto, não se pode concluir se H_{red} é mínimo, mas qualquer outro resultado só pode gerar uma economia adicional de no máximo 15 estados.

A seguinte tabela apresenta de forma reduzida o tamanho dos geradores envolvidos no processo de síntese do supervisor monolítico reduzido para a célula de manufatura.

Tabela 2: Número de estados dos geradores na síntese monolítica

E_{gen}	G	K_{global}	H_{mon}	H_{red}
296	32	151	151	62

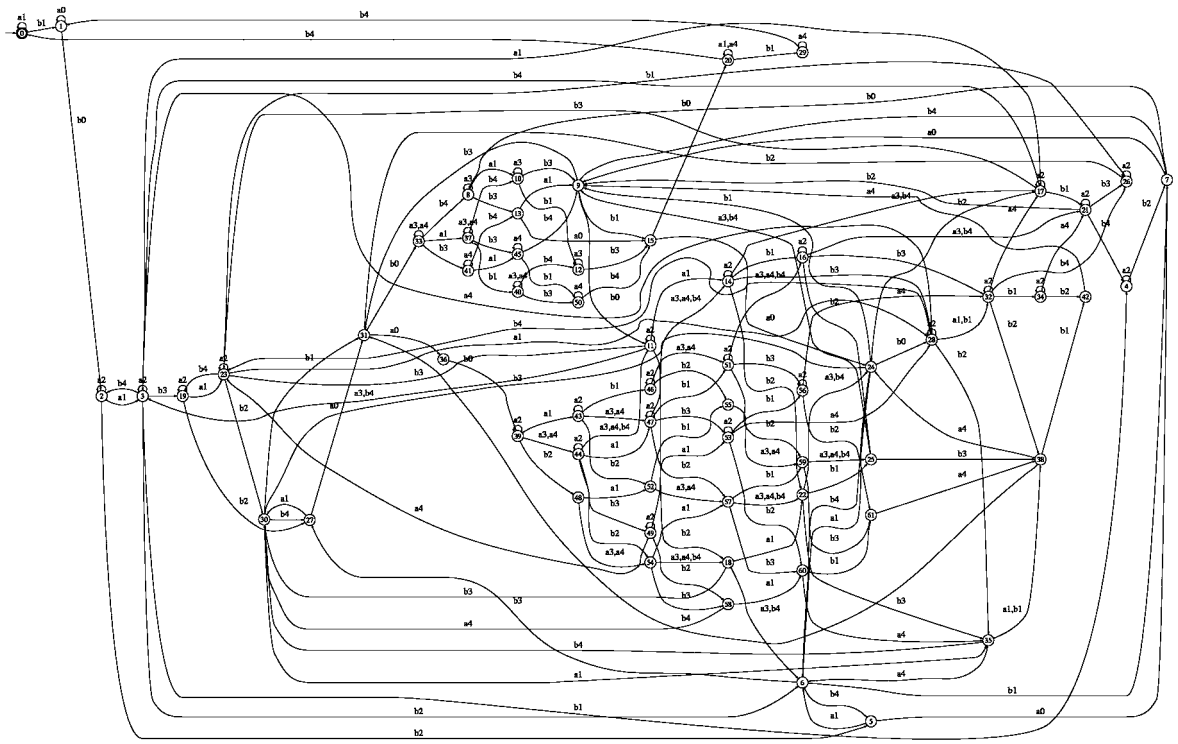


Figura 9: Supervisor monolítico reduzido

A quantidade de memória empregada pelo programa de controle depende do número de estados dos supervisores. Assim, a economia de memória proporcionada pela redução pode em muitos casos viabilizar a implementação do sistema de controle em dispositivos como controladores lógicos programáveis.

Entretanto, em sistemas compostos, o tamanho do supervisor monolítico acaba crescendo exponencialmente com o número de subsistemas e especificações genéricas. Bastam, por exemplo, seis subsistemas assíncronos de seis estados para que só a planta global possua mais de 45.000 estados. Esse fator, pode inviabilizar o cálculo da máxima linguagem controlável bem como a redução de supervisores monolíticos para problemas de maior escala.

Nesse sentido, o uso de uma abordagem mais eficiente que gere supervisores de menor tamanho faz-se necessária para o tratamento de sistemas compostos reais. A abordagem de controle supervisório modular e, em particular, a abordagem modular local, a serem apresentadas na próxima seção, são alternativas vantajosas ao controle monolítico.

2.5 Controle Modular

Como o projeto de controle para sistemas compostos costuma envolver uma grande quantidade de especificações, a abordagem de controle modular (WONHAM e RAMADGE, 1988) é uma alternativa natural para a síntese mais eficiente de controladores. Nessa abordagem, ao invés de se projetar um único supervisor monolítico que satisfaça todas as especificações, procura-se construir um supervisor para cada especificação, de forma que, atuando em conjunto, os supervisores satisfaçam a especificação global. A ação conjunta de supervisores modulares, ilustrada na Figura 10, desabilita um evento controlável da planta sempre que este for desabilitado por algum dos subcontroladores.

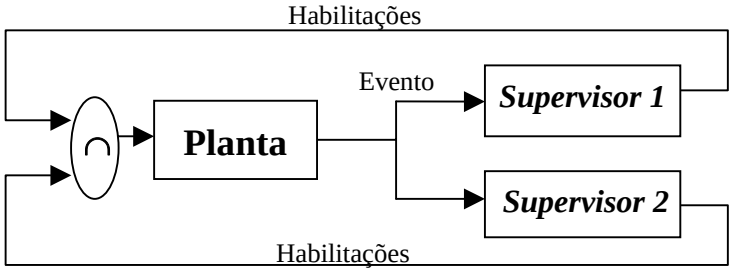


Figura 10: Esquema de controle modular

A síntese modular permite, assim, que problemas complexos possam ser decompostos em módulos mais simples, de forma a atribuir maior flexibilidade ao controlador resultante. Além de ser mais facilmente construído, um supervisor modular costuma ser mais facilmente modificado, atualizado e corrigido. Com isso, deseja-se que a ação conjunta dos supervisores modulares tenha o mesmo resultado que a do supervisor monolítico. Entretanto, os controladores modulares têm suas ações de controle baseadas numa versão parcial do estado de funcionamento do sistema global. Isso pode em muitos casos gerar conflito, isto é, bloqueio do sistema na ação conjunta de controle. Por conseguinte, a síntese modular é, em geral, degradada em relação à solução monolítica

Sejam as linguagens $L_i \subseteq \Sigma^*$, $i = 1,...,n$. Diz-se que o conjunto de linguagens $\{L_i, = 1,...,n\}$ é modular (ou não-conflitante) se $\cap_{i=1}^{n} \overline{L_i} = \overline{\cap_{i=1}^{n} L_i}$. Isso quer dizer que, sempre que um prefixo for aceito por todo o conjunto de linguagens, todo o conjunto deve aceitar uma palavra contendo esse prefixo, ou seja, as linguagens não geram conflito.

Conforme demonstrado por WONHAM e RAMADGE (1988), a condição necessária e suficiente para que o resultado do controle modular seja equivalente ao monolítico – portanto, não-bloqueante e ótimo – é a modularidade das linguagens marcadas pelas ações dos supervisores. Quando essa propriedade é verificada, a

abordagem de controle modular é bastante vantajosa no sentido de promover maior flexibilidade, maior eficiência computacional e segurança na aplicação do controle.

Contudo, a abordagem de controle modular clássica se baseia em especificações expressas em termos do modelo global da planta, cujo tamanho cresce exponencialmente com o número de subsistemas do sistema composto. Assim, apesar de (quando possível) poder ser vantajoso em relação ao controle monolítico, o controle modular clássico não é muito eficiente para lidar com sistemas compostos.

2.5.1 Abordagem de controle modular local

Em QUEIROZ (2000) e em QUEIROZ e CURY (2000a, 2000b, 2000c e 2002a), a abordagem de controle modular proposta por WONHAM e RAMADGE (1988) é estendida de forma a permitir implementar os supervisores a partir de modelos de menor tamanho, tirando proveito da característica descentralizada do modelo em malha aberta de sistemas compostos. Assim, essa abordagem permite explorar, além da modularidade das especificações, a modularidade da planta, de forma a diminuir a complexidade computacional da síntese de supervisores e o tamanho das soluções.

Seja a Representação por Sistema Produto de um sistema G formada por subsistemas $G_i = (Q_i, \Sigma_i, \delta_i, q_{0i}, Q_{mi})$, $i \in I = \{1,...,n\}$. Para $j=1,...,m$, sejam agora as especificações genéricas locais $E_{gen,j}$ definidas respectivamente em subconjuntos de eventos $\Sigma_{gen,j} \subseteq \Sigma$. De forma alternativa à especificação global, pode-se representar o comportamento desejado ao sistema como um conjunto de especificações expressas apenas em termos dos subsistemas que elas restringirem, chamados de planta local. Para $j=1,...,m$, a planta local $G_{loc,j}$ associada à especificação $E_{gen,j}$ é definida por:

$$G_{loc,j} = ||_{i \in I_{loc,j}} G_i, \text{ com } I_{loc,j} = \{k \in I \mid \Sigma_k \cap \Sigma_{gen,j} \neq \varnothing\}.$$

Assim, a planta local $G_{loc,j}$ é composta apenas pelos subsistemas da modelagem original que estão diretamente (e indiretamente) afetados por $E_{gen,j}$. Desta forma, o comportamento desejado para o sistema-produto pode ser expresso através de um conjunto de especificações locais calculadas, para $j=1,...,m,$, por:

$$K_{loc,j} = E_{gen,j} \,||\, L_m(G_{loc,j}).$$

A planta enxuta, definida como $G_e = ||_{j=1}^{m} G_{loc,j}$, engloba apenas os subsistemas que compõem ao menos uma planta local, ou seja, que são relevantes ao problema de controle. Para simplificar este documento, assume-se que sejam modelados no sistema-produto apenas subsistemas restringidos por alguma especificação, ou seja, que $G = G_e$.

No caso da célula de manufatura da Seção 2.1, para $x \in \{a, b1, b2, b3, b4, c1, c2, c3\}$, pode-se obter as plantas locais $G_{loc,x}$ respectivas às especificações $E_{gen,x}$ fazendo-se a composição dos modelos de $\{G_i, i = 0,...,4\}$ que tenham eventos comuns a elas. Formam-se desta forma os seguintes geradores:

- $G_{loc,a} = G_0 \parallel G_1 \parallel G_2 \parallel G_3$ (16 estados e 64 transições);
- $(G_{loc,bi} = G_0 \parallel G_i)$, $i = 1,...,4$, (4 estados e 8 transições);
- $(G_{loc,ci} = G_0 \parallel G_i \parallel G_{i+1})$, $i = 1,2,3$, (8 estados e 24 transições).

Calculam-se, então, as especificações locais $K_{loc,x}$, $x \in \{a, b1, b2, b3, b4, c1, c2, c3\}$, pela composição síncrona das especificações $E_{gen,x}$ com suas respectivas plantas locais. Com o auxílio do CTCT, obtêm-se as seguintes linguagens, que são marcadas por geradores aparados cujos tamanhos são indicados entre parênteses:

- $K_{loc,a} = E_{gen,a} \parallel L_m(G_{loc,a})$ (32 estados e 120 transições);
- $(K_{loc,bi} = E_{gen,bi} \parallel L_m(G_{loc,bi}))$, $i = 1,...,4$, (3 estados e 4 transições);
- $(K_{loc,ci} = E_{gen,ci} \parallel L_m(G_{loc,ci}))$, $i = 1,2,3$, (32 estados e 72 transições).

O conceito apresentado a seguir, extensão da definição de modularidade para linguagens sobre alfabetos distintos, é fundamental para a aplicação dessa abordagem.

Sejam linguagens $L_i \subseteq \Sigma_i^*$, $i = 1,...,n$. Diz-se que o conjunto de linguagens $\{L_i, = 1,...,n\}$ é localmente modular se $\|_{i=1}^{n} \overline{L_i} = \overline{\|_{i=1}^{n} L_i}$. Se, para $i = 1,...,n$, as linguagens L_i forem marcadas por geradores aparados H_i, pode-se provar que a modularidade local é verificada se e somente se $H = \|_{i=1}^{n} H_i$ for um gerador aparado.

É importante observar que a modularidade (local) de todos os subconjuntos de linguagens não garante a modularidade (local) do conjunto completo. Por exemplo, sejam as linguagens $L_1 = \{\alpha\beta, \alpha\gamma, \mu\}$, $L_2 = \{\alpha\gamma, \alpha\mu, \beta\}$ e $L_3 = \{\alpha\mu, \alpha\beta, \gamma\}$ definidas em $\Sigma_1 = \Sigma_2 = \Sigma_3 = \{\alpha, \beta, \gamma, \mu\}$. Verifica-se que, embora $\{L_1, L_2\}$, $\{L_2, L_3\}$ e $\{L_1, L_3\}$ sejam localmente modulares, o conjunto $\{L_1, L_2, L_3\}$ não o é. Essa observação implica que a verificação da modularidade nem sempre possa ser decomposta e que requeira portanto o cálculo da composição síncrona do conjunto completo de linguagens.

O seguinte resultado mostra que a síntese da máxima linguagem controlável de múltiplas especificações pode ser executada diretamente a partir das especificações locais sem perda de performance (aumento de restrição) em relação à solução monolítica – e, por conseguinte, à solução modular clássica –, desde que a modularidade local seja válida.

Teorema 1: (QUEIROZ e CURY, 2000b) Sejam $E_{gen,j}$, $j = 1,...,m$, especificações genéricas sobre um sistema a eventos discretos em malha aberta representado na forma de um sistema-produto $\{G_i, i = 1,...,n\}$. Sejam K_{global} e G definidos como na Seção 2.2 e, para $j = 1,...,m$, sejam $K_{loc,j}$ e $G_{loc,j}$ definidos como no início desta seção. Se o conjunto $\{SupC(K_{loc,j}, G_{loc,j}), \quad j = 1,...,m\}$ for localmente modular, então, $SupC(K_{global}, G) = \|_{j=1}^{m} SupC(K_{loc,j}, G_{loc,j})$.

O Teorema 1 fundamenta a abordagem para a síntese de controladores modulares proposta por Queiroz e Cury (2000b). Dadas m especificações genéricas $E_{gen,j}$, $j = 1,...,m$, sobre um sistema composto, é necessário apenas expressá-las em termos das respectivas plantas locais $G_{loc,j}$, $j = 1,...,m$, representando os subsistemas afetados por elas. A partir daí, calculam-se localmente as máximas linguagens controláveis $SupC(K_{loc,j}, G_{loc,j})$, $j = 1,...,m$, contidas nas mesmas. Para cada especificação, pode-se então construir um supervisor não-bloqueante que, desabilitando eventos controláveis da respectiva planta local, gere em malha fechada a máxima linguagem controlável obtida. A condição de modularidade local, pelo teorema anterior, garante que este procedimento não resulte em perda de performance em relação ao controle monolítico.

Dessa forma, a síntese de supervisores modulares locais tem complexidade computacional polinomial no tamanho dos modelos da planta local e da especificação local, que são limitados aos subsistemas afetados por cada especificação genérica. Portanto, a complexidade da síntese não cresce exponencialmente com a complexidade do sistema composto. Também o tamanho dos supervisores modulares locais depende apenas do tamanho da planta local e, portanto, pode ser pequeno mesmo que o sistema-produto seja de grande porte. Com isso, o uso de algoritmos de redução de supervisores acaba sendo viável na abordagem modular local.

Porém, é importante ressaltar que o teste de modularidade local, feito após a síntese de supervisores para garantir a validade da abordagem, exige a composição de todos os supervisores locais. Assim, a complexidade da verificação da modularidade local, embora vantajosa em relação a outras abordagens encontradas na literatura, acaba crescendo exponencialmente com o número de especificações e subsistemas envolvidos. De fato, tal complexidade é natural, visto que o problema de conflito, ao contrário da controlabilidade, é em geral um problema global que ocorre pela interação de todos os supervisores com toda a planta. Isso aponta para a necessidade do desenvolvimento de métodos mais eficientes para a verificação da modularidade como, por exemplo, a verificação de condições suficientes sobre a estrutura da planta que garantam a modularidade local. Outra

alternativa interessante nesse sentido é a implementação de interfaces entre os supervisores que evitem a propagação de bloqueio (LEDUC, 2002).

O funcionamento conjunto dos supervisores localmente modulares é ilustrado pela Figura 11. Pode-se observar que, apesar de nenhuma restrição na estrutura de informações ter sido previamente imposta (LIN e WONHAM, 1990; RUDIE e WONHAM, 1992), os controladores locais observam e controlam apenas o comportamento dos subsistemas afetados. Assim, pode-se afirmar que a abordagem de controle modular local induz uma estrutura de controle decentralizada que surge naturalmente do processo de síntese.

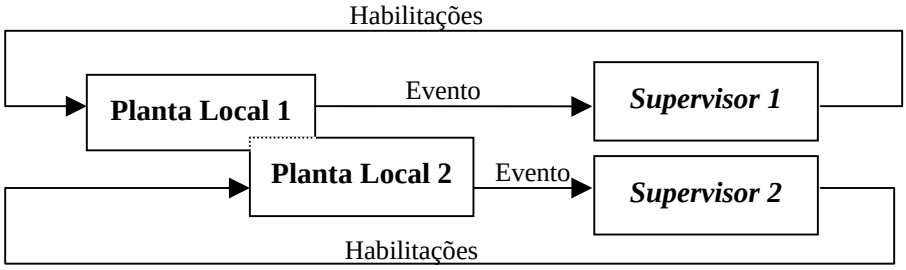


Figura 11: Esquema de controle modular local

Para a célula de manufatura da Seção 2.1, o cálculo da solução modular local pode ser feito computacionalmente com o auxílio da ferramenta CTCT (WONHAM, 2003). Para $x \in \{a, b1, b2, b3, b4, c1, c2, c3\}$, verifica-se que as especificações locais $K_{loc,x}$ não são $L_m(G_{loc,x})$-fechadas. Porém, da mesma forma que no controle monolítico, não há qualquer problema em permitir que os supervisores desmarquem estados da planta. Assim, ao usar supervisores desmarcadores para o controle da célula, pode-se abdicar da propriedade de $L_m(G_{loc,x})$-fechamento.

As especificações locais $K_{loc,a}$, $K_{loc,b1}$, $K_{loc,b2}$, $K_{loc,b3}$ e $K_{loc,b4}$ são controláveis em relação às suas plantas locais. Assim, os geradores aparados que marcam essas especificações podem ser tomados como supervisores não-bloqueantes $S_{loc,a}$, $S_{loc,b1}$, $S_{loc,b2}$, $S_{loc,b3}$ e $S_{loc,b4}$. Para as especificações não-controláveis $K_{loc,c1}$, $K_{loc,c2}$ e $K_{loc,c3}$, calculam-se as máximas linguagens controláveis $SupC(K_{loc,ci}, G_{loc,c1})$, $i = 1,2,3$, de forma que os geradores aparados para essas linguagens, com 24 estados e 52 transições, podem ser tomados como os supervisores não-bloqueantes ótimos $S_{loc,c1}$, $S_{loc,c2}$ e $S_{loc,c3}$. Para verificar a condição de modularidade local, calcula-se o gerador $S = S_{loc,a} \parallel S_{loc,b1} \parallel S_{loc,b2} \parallel S_{loc,b3} \parallel S_{loc,b4} \parallel S_{loc,c1} \parallel S_{loc,c2} \parallel S_{loc,c3}$[4], formado por 151 estados e 350 transições. Como este supervisor é aparado

[4] Observa-se que, para testar a modularidade local a partir de supervisores reduzidos, essa composição síncrona deve incluir o modelo da planta G.

– e, consequentemente, equivalente ao supervisor monolítico –, garante-se que a abordagem modular local é ótima.

Para cada um dos oito supervisores modulares calculados, a ação de controle pode ser identificada para cada estado como um conjunto de eventos a serem desabilitados. Esse conjunto contém os eventos que não estão previstos na estrutura de transição do supervisor e, no estado correspondente, podem ocorrer na planta.

A partir da solução modular local da célula de manufatura, alguns supervisores mínimos podem ser obtidos de forma direta. Como $S_{loc,a}/G_{loc,a} = H_{gen,a} \parallel G_{loc,a}$, onde $H_{gen,a}$ é um gerador aparado para a especificação $E_{gen,a}$, pode-se tomar o gerador $H_{gen,a}$ como supervisor reduzido $S_{red,a}$. Da mesma forma, os geradores aparados para as especificações $E_{gen,bi}$, $i = 1,...,4$, podem representar os supervisores reduzidos $S_{red,bi}$, $i = 1,...,4$, para os supervisores $S_{loc,bi}$, $i = 1,...,4$. Já os supervisores $S_{red,ci}$, $i = 1,2,3$, são calculados computacionalmente a partir dos supervisores $S_{loc,ci}$, $i = 1,2,3$, pela implementação do algoritmo de SU e WONHAM (2001) no CTCT. Como o limite inferior calculado para o valor mínimo confere com o tamanho obtido, a redução é ótima. A Figura 12 apresenta o conjunto de todos supervisores modulares minimizados que controlam a célula de manufatura (Seção 2.1) de maneira minimamente restritiva de forma a respeitar o comportamento especificado. As linhas tracejadas representam a ação de controle, indicando para cada estado os eventos a serem desabilitados na planta.

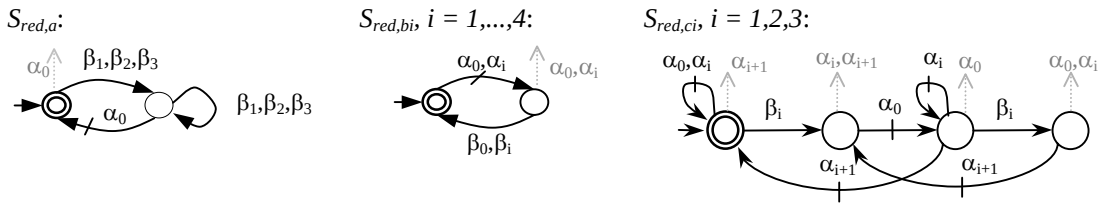


Figura 12: Supervisores modulares reduzidos para a célula de manufatura

A seguinte tabela apresenta de forma resumida o tamanho dos geradores envolvidos na síntese de supervisores modulares locais reduzidos para a célula de manufatura. Observa-se que são obtidos supervisores ótimos do tamanho dos modelos das especificações genéricas sem produzir geradores de tamanho elevado, a não ser no teste de modularidade local. Como cada supervisor concerne apenas uma especificação, a lógica de controle implícita nos supervisores minimizados acaba sendo compreensível (neste caso, similar à lógica das especificações genéricas).

Tabela 3: Número de estados dos geradores na síntese modular local

x	$E_{gen,x}$	$G_{loc,x}$	$K_{loc,x}$	$S_{loc,x}$	$S_{red,x}$
a	2	16	32	32	2
$bi,\ i = 1,...,4$	2	4	3	3	2
$ci,\ i = 1,2,3$	4	8	32	24	4

Pela abordagem modular (WONHAM e RAMADGE, 1988), os supervisores modulares $S_{mod,x}$, $x \in \{a, b1, b2, b3, b4, c1, c2, c3\}$, são calculados a partir de especificações modulares $K_{mod,x}$ que contém informações sobre a planta global ($K_{mod,x} = E_{gen,x} \parallel G$). No caso da célula de manufatura da Seção 2.1, os resultados obtidos pela abordagem modular são mostrados na Tabela 4.

Tabela 4: Número de estados dos geradores na síntese modular

x	$E_{gen,x}$	G	$K_{mod,x}$	$S_{mod,x}$	$S_{red,x}$
a	2	32	64	64	2
$bi,\ i = 1,...,4$	2	32	24	24	2
$ci,\ i = 1,2,3$	4	32	128	96	4

Apesar de depender do tamanho da planta global, os supervisores modulares são suficientemente pequenos para o uso do algoritmo de redução, que leva ao mesmo resultado apresentado na Figura 12. Observa-se, porém, que tanto o cálculo da máxima linguagem controlável quanto a redução de supervisores são executados a partir de geradores maiores que implicam maior esforço computacional. Na verificação de não-conflito, obtém-se o mesmo gerador aparado de 151 estados pela composição de geradores maiores do que na abordagem modular local.

2.5.2 Resolução de conflitos

É importante observar que, quando a modularidade das especificações controláveis não é verificada, existe um conjunto de supervisores conflitantes que impedem o controle modular de ser diretamente aplicado. Nesse caso, é possível utilizar diferentes abordagens para a resolução de conflitos.

Uma alternativa evidente é a aplicação de controle monolítico, descartando-se os supervisores modulares. Quando o problema de conflito está restrito a uma parte específica da planta, pode-se calcular um supervisor monolítico para o subconjunto de especificações conflitantes. Observa-se que o teste de modularidade pode ser aproveitado para o cálculo do supervisor monolítico. Seja H_{conf} o gerador bloqueante obtido pela composição síncrona dos supervisores modulares (sem redução). Esse gerador modela o comportamento

conflitante do sistema em malha fechada. Pode-se provar que $SupC(K_{global}, G) = SupC(L_m(H_{conf}), H_{conf})$. Assim, é possível obter um gerador aparado H_{mon} para $SupC(K_{global}, G)$ a partir de H_{conf} pelo processo de eliminação de maus estados conforme o algoritmo para o cálculo da máxima linguagem controlável proposto por WONHAM e RAMADGE (1987).

Se H_{conf} for tomado como planta para H_{mon}, a ação de controle de H_{mon} consistirá apenas na desabilitação de eventos controláveis que possam levar o sistema global a bloqueio. Tal supervisor, denominado coordenador, possui a vantagem de preservar a arquitetura modular de controle. A Figura 13 ilustra o funcionamento do controle modular sob a ação de um coordenador. Da mesma forma que para um supervisor modular, a redução do número de estados de um coordenador pode ser em geral bem maior que a redução do supervisor monolítico. Dois exemplos de cálculo de coordenador são apresentados no Capítulo 5.

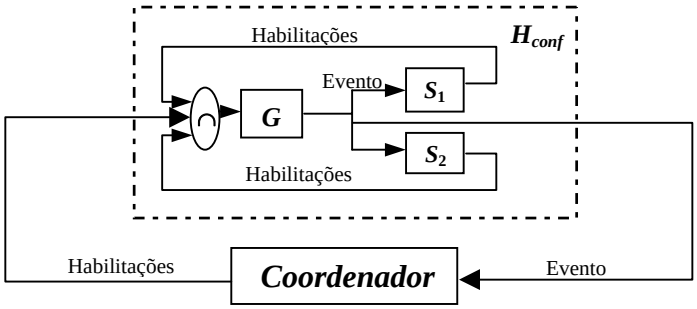


Figura 13: Esquema de controle modular com coordenação

Como conflito é em geral um problema global do sistema controlado, a síntese e redução de coordenador pode ser de difícil computação para problemas de grande porte. Entretanto, essas operações podem ser viabilizadas pelo uso de técnicas mais elaboradas para o tratamento eficiente de geradores com muitos estados, como é o caso do trabalho de ZHANG e WONHAM (2001).

Outras abordagens para resolução de conflitos são encontradas na literatura. WONG *et al.* (1995) propõem evitar conflitos introduzindo uma interface entre um dos supervisores conflitantes e a planta. Essa interface suspende o supervisor modular quando o conflito aparece e o reativa no momento apropriado. O uso de interfaces entre supervisores modulares também é proposto por LEDUC (2002) como forma de evitar a propagação de bloqueio. CHEN *et al.* (1995) apresentam um esquema para resolução de conflitos baseado na atribuição de prioridades a supervisores individuais. Já o trabalho de

WONG e WONHAM (1998) propõe a resolução de conflitos através de coordenação hierárquica.

2.6 Conclusão do capítulo

Os resultados obtidos com o exemplo indicam que a metodologia de controle supervisório modular local apresentada é vantajosa na síntese de supervisores para sistemas compostos. Especialmente no que diz respeito à complexidade computacional da síntese, o controle modular local se mostrou mais eficiente do que a abordagem monolítica e do que a abordagem modular, uma vez que não há explosão do número de estados do modelo com o aumento do porte do sistema.

Além disso, a solução do problema da célula de manufatura evidenciou um caráter deficiente da Teoria de Controle Supervisório, no que concerne a noção de tarefas completas associadas à marcação dos estados. A modelagem da planta e da especificação por geradores e linguagens não permite distinguir de forma clara o significado das diferentes marcações no sistema controlado. Assim, quando múltiplas tarefas são representadas pela marcação de um gerador, não é fácil distinguir quais delas permanecem vivas (coacessíveis) pelo não-bloqueio do sistema controlado. Da mesma forma, nem sempre é evidente saber em que situações um supervisor deve "desmarcar" um estado da planta. Para o problema da célula de manufatura em questão, o uso de supervisores desmarcadores foi satisfatório. Porém, uma abordagem mais clara e eficiente para o tratamento das marcações em problemas mais gerais sobre sistemas compostos se faz necessária. Esse assunto é desenvolvido nos capítulos 4 e 5.

Finalmente, uma das grandes vantagens do controle modular local está na característica distribuída dos supervisores em sistemas compostos. Garantidas as condições de modularidade, tem-se maior flexibilidade, maior segurança e economia computacional na aplicação do controle supervisório. Essas vantagens são exploradas no próximo capítulo, onde são apresentados os principais resultados da implementação física da solução de controle obtida para a célula de manufatura.

3. Implementação do Sistema de Controle

Este capítulo apresenta os resultados de uma aplicação bem sucedida da TCS para a síntese de um sistema de controle real para a célula de manufatura da Seção 2.1. Os problemas práticos que aparecem na implementação física do sistema de controle são discutidos e levam à proposta de uma estrutura genérica de implementação do sistema de controle baseado em supervisores modulares locais. O detalhamento da programação dessa estrutura de controle no CLP ilustra o potencial da metodologia de síntese e implementação de supervisores para sistemas a eventos discretos. Esses resultados são resumidos por QUEIROZ *et al.* (2001) e por QUEIROZ e CURY (2002b).

3.1 Aspectos práticos da implementação

Do processo de síntese apresentado no capítulo anterior é gerado um conjunto de supervisores modulares locais representados por máquinas de estados finitos. Para cada estado dos supervisores, a ação de controle é definida como a desabilitação de um conjunto de eventos controláveis calculado no processo de síntese. Assim, teoricamente, a implementação física do sistema de controle consiste em "jogar" os autômatos dos supervisores em paralelo, de acordo com os eventos sinalizados pela planta real, e de enviar sinais de desabilitação de eventos controláveis para a planta, de acordo com o estado ativo dos autômatos.

Contudo, na prática, normalmente o comportamento real da planta física não corresponde exatamente ao comportamento modelado, em função de algumas simplificações no processo de modelagem. As principais simplificações são a suposição de ocorrência espontânea de eventos controláveis e a abstração de certos comportamentos internos dos subsistemas.

A Teoria de Controle Supervisório pressupõe que os eventos são gerados exclusivamente pela planta de forma que o seu comportamento é direcionado pelo supervisor apenas pela desabilitação de certos eventos. Entretanto, em grande parte dos problemas reais, incluindo a maioria dos sistemas de manufatura, os eventos modelados como controláveis correspondem a comandos que, na verdade, devem ser gerados pelo sistema de controle. Para a célula de manufatura apresentada no Capítulo 2, por exemplo,

os eventos controláveis α_i (início de operação do aparelho M_i) representam comandos que devem ser enviados às respectivas máquinas pelo CLP, pois não ocorreriam espontaneamente.

Para lidar com esse problema, BALEMI (1992) propõe o uso de uma abordagem entrada/saída para a síntese de supervisores geradores de comandos. A estrutura de controle proposta por BALEMI (1992) tem a característica de gerar os comandos (entradas) que dirigem o sistema físico a partir do modelo do sistema controlado, que é atualizado de acordo com os sinais de resposta (saídas). Para isso é necessário embutir o comportamento da planta nos supervisores, fazendo-se a composição do modelo global da planta com cada supervisor modular. Desta forma, acabam-se perdendo as propriedades de redução e descentralização dos supervisores modulares locais em sistemas compostos. Com isso, tal abordagem entrada/saída acaba levando à composição de supervisores com um grande número de estados. Para lidar com o problema da complexidade computacional, BALEMI (1992) propõe o uso de "*Binary Decision Diagrams*" (BRYANT, 1986), que permitem realizar de forma eficiente operações sobre geradores com muitos estados. Entretanto, a implementação prática de sistemas de controle a partir de máquinas de estados imensas é muitas vezes inviável por duas principais razões: a capacidade de memória e de processamento em equipamentos de controle (geralmente CLPs) é limitada e o operador humano precisa compreender a lógica de controle para ter confiança no sistema de controle e para poder fazer sua manutenção.

Um outro aspecto importante da modelagem a ser considerado na implementação do sistema de controle é a simplificação que normalmente se faz no modelo da planta. Para fins de síntese da lógica de controle, na modelagem de cada subsistema, costuma-se abstrair de seus eventos internos que não têm relação com as especificações. Esse procedimento resulta em modelos menores e, por conseqüência, em economia computacional. É claro que essas abstrações devem garantir uma consistência entre o comportamento real e o modelo, no sentido de não prejudicar a qualidade dos supervisores obtidos. Em casos mais complexos, essa consistência deve ser assegurada usando-se resultados de abordagens de controle hierárquico (ZHONG e WONHAM, 1990; WONG e WONHAM, 1996; CUNHA e CURY, 2002; TORRICO e CURY, 2002), porém em casos mais simples sua verificação é imediata.

Por exemplo, para a célula de manufatura da Seção 2.1, todos os problemas especificados podem ser evitados levando-se em conta o início e o fim de operação de cada máquina. No entanto, cada início de operação de uma máquina desencadeia uma seqüência de comandos e respostas, a última das quais deve ser interpretada como o fim de operação.

Apesar de não estarem embutidas na estrutura de transição dos supervisores modulares, essas seqüências operacionais também devem ser comandadas pelo sistema de controle.

3.2 Estrutura de Implementação

Para resolver esses problemas práticos mantendo a estrutura modular da solução de controle, propõe-se implementar uma interface entre o sistema real (planta física) e os supervisores modulares obtidos a partir de um modelo abstrato do sistema. Essa interface tem a função de, respeitando as desabilitações dos supervisores, comandar o funcionamento da planta conforme o modelo do sistema-produto (planta abstrata) e de traduzir o comportamento abstrato em sinais reais de entrada e saída, especificados pelas seqüências operacionais (planta operacional) de cada subsistema. Com isso, essa interface faz o sistema real se comportar de acordo com o modelo assumido por RAMADGE e WONHAM (1987) (planta RW), que gera espontaneamente e seqüencialmente eventos que não estão desabilitados.

Com o objetivo de executar os supervisores modulares, o sistema-produto e as seqüências operacionais de forma a preservar suas características modulares, propõe-se implementar o sistema de controle conforme a hierarquia de três níveis apresentada na Figura 14.

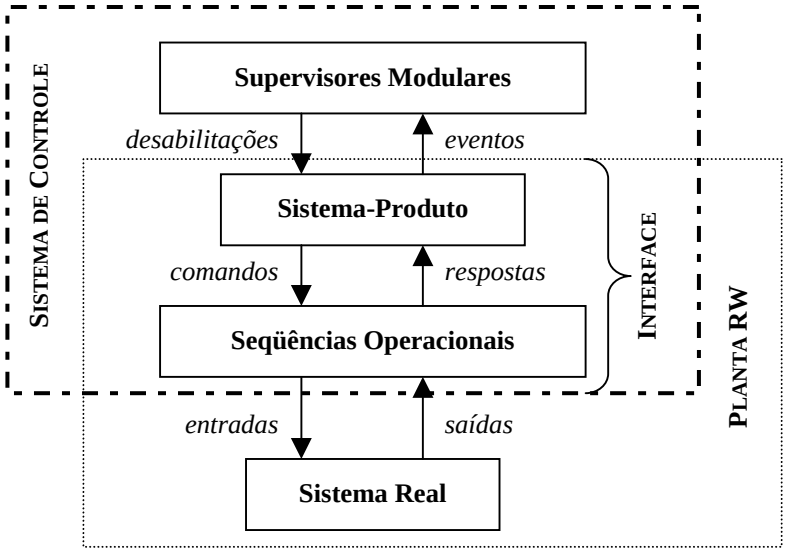


Figura 14: Estrutura Básica do Sistema de Controle

Essa estrutura, cuja dinâmica é explicada a seguir, pode ser programada em linguagens de programação de CLPs (Diagrama Escada, *Sequential Function Charts*, etc.), em linguagens de programação de PCs (C++, Java, etc.), bem como diretamente em *hardware* (circuitos digitais, elétricos, pneumáticos ou hidráulicos). Tal estrutura de

controle difere do esquema de implementação proposto em outros trabalhos (BALEMI *et al.*, 1993; BRANDIN, 1996; LEDUC, 1996; FABIAN e HELLGREN, 1998; CHARBONNIER *et al.*, 1999) por executar os supervisores exatamente como previsto pela TCS, enquanto uma interface resolve as diferenças entre o modelo abstrato e o sistema real. Essa característica permite uma implementação modular da planta e dos supervisores, o que torna o programa de controle claro e flexível.

3.2.1 Supervisores Modulares

O conjunto de supervisores modulares locais reduzidos é implementado no nível mais alto do sistema de controle exatamente como concebido teoricamente por RAMADGE e WONHAM (1989). O programa de controle atualiza os estados ativos dos supervisores de acordo com a estrutura de seus geradores e com a ocorrência de eventos sinalizada pelo nível do Sistema-Produto. Um mapa de retroalimentação associa os estados ativos a um conjunto de sinais de desabilitação de eventos controláveis que controlam o Sistema-Produto. Pode-se observar que não há necessidade de se programar as transições dos supervisores que saem e entram no mesmo estado, uma vez que essas transições não alteram os estados ativos e, por conseqüência, a ação de controle.

3.2.2 Sistema-Produto

Em certos casos, os sinais de desabilitação gerados pelos supervisores podem ser diretamente associados a ações de controle que de fato impeçam a ocorrência do respectivo evento controlável. Por exemplo, o evento de passagem por uma porta pode ser desabilitado enviando um sinal para fechar a porta. Assim, a desabilitação da passagem pode ser associada ao fechamento da porta. Naturalmente, para que o evento possa ocorrer quando habilitado, a porta deve abrir automaticamente na ausência de desabilitação. Para esses casos o nível do Sistema-Produto corresponde ao Sistema Real em conjunto com uma eventual interface para condicionamento dos sinais de desabilitação e de ocorrência de eventos.

Entretanto, verifica-se que em muitos problemas, especialmente em sistemas de manufatura, há eventos cuja ocorrência não é espontânea e, portanto, devem ser executados por comandos. Claramente, tais eventos são controláveis (basta não executá-los para desabilitar sua ocorrência). Para simplificar a descrição da estrutura proposta, assume-se que todos eventos controláveis sejam associados a comandos. Nesse caso, ao invés de gerar sinais de desabilitação, o sistema de controle deve comandar a ocorrência dos

eventos controláveis conforme o comportamento calculado para a planta sob ação de controle dos supervisores.

Contudo, os geradores dos supervisores não contêm necessariamente a informação completa do comportamento da planta. Sendo assim, para comandar a execução dos subsistemas modelados, a planta é implementada na forma de máquinas de estado concorrentes no nível do Sistema-Produto. As transições controláveis e não-controláveis funcionam de forma distinta. Enquanto as transições não-controláveis são condicionadas a eventos representando respostas das Seqüências Operacionais, as transições controláveis, condicionadas à habilitação dos supervisores, geram os eventos representando comandos para as Seqüências Operacionais. Observa-se que, assim como no grafo de um gerador, um mesmo evento pode estar associado a distintas transições.

A evolução paralela dos geradores assíncronos do Sistema-Produto segue os comandos executados (transições controláveis) e as respostas do nível Seqüências Operacionais (transições não-controláveis), sinalizando a ocorrência de eventos aos supervisores. Para evitar a execução de transições controláveis indesejáveis, é necessário que o sistema de controle assegure que a ação dos Supervisores Modulares esteja sempre atualizada antes que uma nova transição controlável ocorra no Sistema-Produto. Esse cuidado também evita que múltiplos eventos controláveis sejam sinalizados simultaneamente aos supervisores.

Diz-se que uma transição controlável é aceita se puder ser executada, isto é, se seu estado predecessor estiver ativo e se não estiver sendo desabilitada pelos supervisores. Já uma transição não-controlável é considerada aceita se seu estado predecessor estiver ativo e se seu evento correspondente estiver sendo sinalizado pelas seqüências operacionais. Assim, a principal função deste nível do sistema de controle é executar comandos associados a transições controláveis aceitas.

Quando múltiplas transições no Sistema-Produto são aceitas (e puderem ser executadas) simultaneamente, o programa precisa escolher qual transição executar, baseado num esquema de prioridades, ou passar esse grau de liberdade a um nível de decisão superior. Em qualquer caso, para consistência do sistema de controle, deve-se garantir que as transições não-controláveis aceitas (que já ocorreram no sistema real) sejam executadas antes das controláveis (que podem ser adiadas), pelo fato de que os estados da planta devem estar atualizados antes de se executar um novo comando.

Quando a execução de cada parte do programa ocorre em intervalos discretos de tempo (ciclos), é possível que mais do que uma resposta das Seqüências Operacionais seja

sinalizada ao Sistema-Produto no mesmo ciclo. Com isso, múltiplas transições não-controláveis podem ficar aceitas no mesmo ciclo, o que implica a ocorrência simultânea de múltiplos eventos. Quando isso ocorre, deve-se assegurar que as transições dos supervisores modulares ocorram de forma seqüencial. Uma alternativa é alterar a estrutura dos supervisores para tratar cada possível ocorrência simultânea de transições não-controláveis como uma nova transição. Cada possível combinação de eventos não-controláveis corresponde assim a um novo evento. Evidentemente, esse método pode levar ao crescimento combinatório do número de transições. Outra abordagem possível é impor a ocorrência de apenas uma transição no Sistema-Produto a cada ciclo. Para isso deve-se garantir que as transições não-controláveis aceitas e não executadas num ciclo permaneçam aceitas nos ciclos seguintes até que sejam executadas. Neste caso, qualquer ordem no tratamento dos eventos correspondentes deve levar os Supervisores Modulares aos mesmos estados. Do contrário, o comportamento do sistema em malha fechada dependerá fortemente do esquema de prioridades.

Em suma, os principais cuidados que devem ser tomados para garantir a correta operação da estrutura de controle proposta são:

1. a ação dos supervisores (desabilitações) deve estar atualizada para a ocorrência de uma transição controlável no Sistema-Produto;
2. as transições não-controláveis do Sistema-Produto devem ser tratadas com maior prioridade que as transições controláveis;
3. a sinalização da ocorrência de eventos para os Supervisores Modulares deve ser seqüencial.

3.2.3 Seqüências Operacionais

O nível das Seqüências Operacionais funciona como uma interface entre o Sistema-Produto modelado e o Sistema Real. Neste nível, o programa interpreta os comandos abstratos do Sistema-Produto como seqüências lógicas que guiam a operação de cada subsistema particular. Esses procedimentos de baixo nível geram os sinais de saída do sistema de controle (entradas do sistema real) e lêem os sinais de entrada (saídas do sistema real), fornecendo ao Sistema-Produto respostas lógicas que refletem a ocorrência de eventos não-controláveis.

O esquema de controle proposto é ilustrado na seção seguinte, que apresenta os resultados de sua implementação em Diagrama Escada para o problema real da célula de manufatura.

3.3 Aplicação à Célula de Manufatura

Como resultado da aplicação da abordagem de controle supervisório modular local à célula de manufatura apresentada na Seção 2.1, foram obtidos os oito supervisores reduzidos $S_{red,x}$, $x \in \{a, b1, b2, b3, b4, c1, c2, c3\}$, (Figura 12) cuja ação conjunta gera a lógica de controle minimamente restritiva. Esses supervisores e a interface, formada pelo sistema-produto, composto pelas plantas G_0, G_1, G_2, G_3 e G_4, e suas respectivas seqüências operacionais, formam o sistema de controle. De acordo com a estrutura proposta na seção anterior, o sistema de controle da célula de manufatura é implementado hierarquicamente num controlador lógico programável FPC100 da Festo (FESTO, 1999) conforme o esquema apresentado na Figura 15.

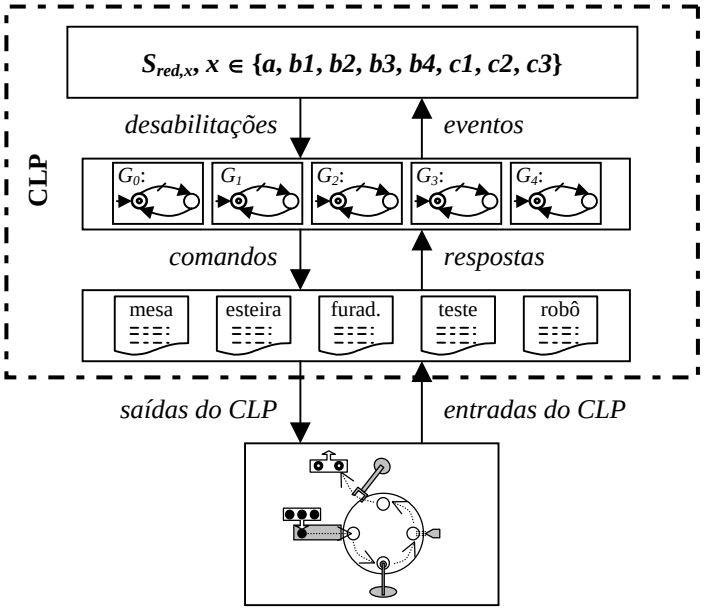


Figura 15: Sistema de Controle para a Célula de Manufatura

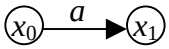
O controlador da Festo em questão, assim como a maioria dos controladores, tem sua memória dividida em duas partes principais: a primeira parte contendo o programa do usuário e a segunda, a tabela de dados. O programa do usuário, muitas vezes implementado em Diagrama Escada (ISO/IEC, 2003), consiste na lógica que relaciona as entradas com variáveis internas ("*flags*") e saídas. A tabela de dados contém as informações necessárias para a execução do programa, como o estado das entradas, saídas e variáveis e o valor de temporizadores e contadores. Durante cada ciclo de varredura, a CPU seqüencialmente lê o *status* das entradas do CLP, executa a lógica de controle embutida no programa do usuário e, por fim, atualiza as saídas do CLP. O tempo de varredura varia tipicamente entre 1 ms e 100ms.

A programação do CLP é feita através do *software* FST (PLAGEMANN, 1991). As principais funções básicas disponíveis para programação em Diagrama Escada estão descritas na Tabela 5.

Tabela 5: Operadores do Diagrama Escada para o CLP da Festo

Contato normalmente aberto	A --| |--	O estado da entrada é copiado para a saída se a variável *A* for verdadeira. Senão, o estado da saída é falso.
Contato normalmente fechado	A --|/|--	O estado da entrada é copiado para a saída se a variável *A* for falsa. Senão, o estado da saída é falso.
Bobina	Q --( )--	O estado da entrada é copiado para a variável booleana *Q* e para a saída.
Bobina Liga (SET)	Q --(S)--	Quando o estado da entrada é verdadeiro, a variável booleana *Q* é ativada, senão *Q* não é afetada.
Bobina Desliga (RESET)	Q --(R)--	Quando o estado da entrada é verdadeiro, a variável booleana *Q* é desativada, senão *Q* não é afetada.
Identificador	pos --|L|--	Identifica a posição do diagrama com a etiqueta *pos*.
Salto	pos ---->>	Quando a entrada for verdadeira, o interpretador do diagrama salta para a posição identificada pela etiqueta *pos*.

Para programação de máquinas de estado em Diagrama Escada, assim como em BRANDIN (1996) e em FABIAN e HELLGREN (1998), representa-se cada estado por uma variável booleana interna. Para cada transição do autômato é implementado um "degrau" (ou passo) do Diagrama Escada, segundo o qual, sempre que o contato representando o estado anterior e o contato relativo a um determinado evento interno ou externo ao CLP estiverem ativos, atribuem-se os valores 1 ao estado seguinte e 0 ao estado anterior através de bobinas sustentáveis. A Figura 16 exemplifica a implementação em Diagrama Escada de uma máquina com uma transição (a) e dois estados (x_0 e x_1).




```
   x0    a      x1
|--][-+-][--+--(S)-|
|           |   x0 |
|           +--(R)-|
```

Figura 16: Implementação em Diagrama Escada (dir.) de uma máquina de 2 estados (esq.)

Assim, no nível mais alto do sistema de controle para a célula de manufatura, os oito supervisores $S_{red,x}$, $x \in \{a, b1, b2, b3, b4, c1, c2, c3\}$, são implementados como máquinas de estados concorrentes, de forma que as desabilitações, associadas a variáveis booleanas internas (d-αi, $i = 0,...,4$), são sinalizadas de acordo com os estados ativos. A Figura 17 ilustra a implementação do supervisor $S_{red,c1}$ em Diagrama Escada.

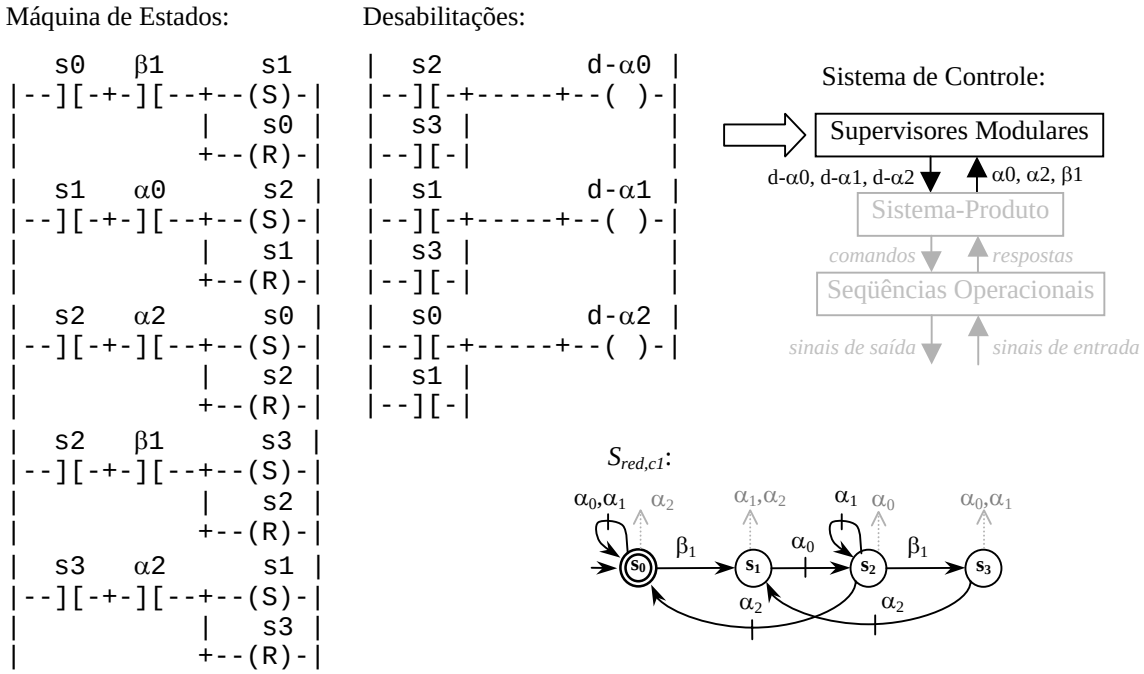


Figura 17: Implementação em Diagrama Escada do Supervisor $S_{red,c1}$

Os cinco subsistemas G_0, G_1, G_2, G_3 e G_4 são programados no nível do Sistema-Produto como máquinas de estado assíncronas. As transições não-controláveis são disparadas por respostas das Seqüências Operacionais sinalizando o final de operação da respectiva máquina. Já as transições controláveis são executadas automaticamente sempre que não forem desabilitadas pelos supervisores, ativando-se as variáveis que iniciam as seqüências de operação. Cada transição ativa também o sinal que representa a ocorrência do respectivo evento para os supervisores.

Para garantir o correto funcionamento do sistema de controle conforme discutido na Seção 3.2.2, é importante cuidar para que não ocorram duas transições seguidas no Sistema-Produto sem que os supervisores tenham sido atualizados. No caso particular do CLP em questão, isso pode ser realizado fazendo-se um salto do final de cada transição para o primeiro degrau dos supervisores. A ordem de execução dos degraus define uma hierarquia de prioridades para as transições.

Como exemplo, a Figura 18 mostra a implementação em Diagrama Escada da planta para a esteira (G_1). A variável interna “e-ini” dispara a seqüência operacional da esteira, enquanto a variável “e-fim”, sinalizando o final do ciclo de operação da esteira, permanece ativada até que a variável “e3” seja desativada pela ocorrência da respectiva transição no Sistema-Produto. Para garantir maior prioridade na execução das transições não-controláveis, os degraus associados a estas devem anteceder as transições controláveis.

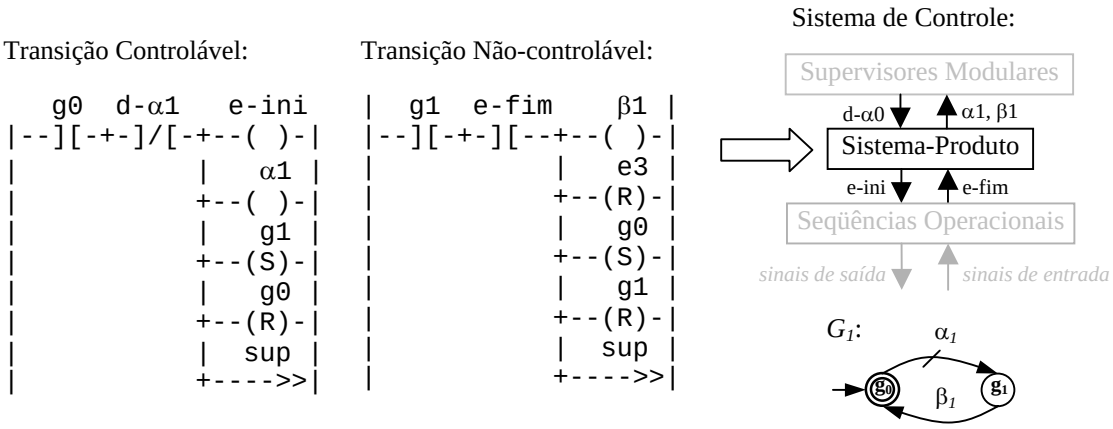


Figura 18: Implementação em Diagrama Escada da Planta G_1

Observa-se que o operador de salto pode ser substituído pela adição de uma variável interna do CLP, representando o fato de os supervisores estarem atualizados. Essa variável deve ser ativada (SET) logo após a atualização de todas as transições e desabilitações dos Supervisores Modulares e desativada (RESET) pela ocorrência de qualquer transição no Sistema-Produto. Tal variável deve ser condição para a ocorrência de cada transição no Sistema-Produto.

As Seqüências Operacionais para a mesa circular, a esteira, a furadeira, o teste e o manipulador robótico também podem ser programadas como máquinas de estados, cujas transições iniciais são disparadas pelos sinais do Sistema-Produto e as seguintes pelos sinais de entrada do CLP. Assim, os sinais de saída do CLP são ativados seqüencialmente conforme a lógica de funcionamento interna de cada dispositivo. O final de cada ciclo de operação é sinalizado ao Sistema-Produto ativando-se a respectiva variável booleana. Por exemplo, o programa para a seqüência operacional da esteira, mostrado na Figura 19, inicia a operação da esteira ligando o motor da mesma (saída O0.1) e, quando um sensor indutivo indicar a presença de peça na posição P1 (entrada I0.1), termina o ciclo operacional após desligar o motor.

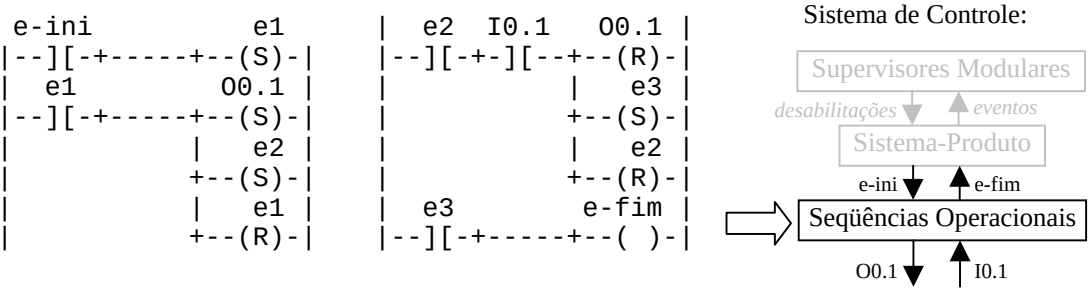


Figura 19: Implementação da Seqüência Operacional para a Esteira

As outras estruturas periféricas do programa, como inicialização do sistema físico, ativação dos estados iniciais dos autômatos e procedimentos de parada, podem ser implementadas sem grandes dificuldades. O Diagrama Escada final para o controle do CLP, exibido no Anexo 1, tem 87 degraus (5233 Bytes) e utiliza 82 dos 256 "*flags*" disponíveis no CLP. Portanto, a memória do CLP (12 Kbytes) não é um obstáculo para a implementação prática do controlador.

Após a aplicação do novo sistema de controle, a célula de manufatura passou a funcionar com maior produtividade e flexibilidade, pois a lógica de controle permite operar de 0 a 4 peças em paralelo, conforme a mesa for sendo alimentada. Além disso, a estrutura do programa se mostrou bastante flexível e compreensível, possibilitando uma depuração rápida e mudanças no programa com bastante facilidade.

3.4 Conclusão do capítulo

Esse capítulo contribui à Teoria de Controle Supervisório com dois resultados práticos para o tratamento de sistemas compostos: a proposta de uma estrutura genérica de implementação baseada na abordagem de controle modular local e o exemplo de uma aplicação bem sucedida da TCS na solução completa de um problema real. A modularidade e a localidade dos supervisores permitiram o desenvolvimento de um sistema de controle estruturado programada em Diagrama Escada, uma linguagem de programação bastante difundida no meio industrial. O funcionamento flexível e produtivo da célula de manufatura e a clareza e flexibilidade do programa final do CLP são indicadores positivos da qualidade da metodologia usada.

A estrutura de controle apresentada neste capítulo abre espaço para novas pesquisas na direção de promover a aplicação da Teoria de Controle Supervisório na resolução de problemas reais. Para a aceitação da metodologia no meio industrial, é fundamental que sejam realizados novos estudos de caso envolvendo sistemas de maior escala. Também, é preciso que sejam desenvolvidas técnicas para garantir a consistência na dinâmica do sistema de controle quando implementado numa linguagem e plataforma distinta do Diagrama Escada para CLP. A implementação em *Sequential Function Charts* (ISO/IEC, 2003), por exemplo, é bastante apropriada para preservar a estrutura das máquinas de estado nos três níveis da arquitetura de controle apresentada na Seção 3.2.

Outra linha de pesquisa de interesse prático é a adaptação da estrutura de controle proposta para implementação distribuída (em múltiplos CLPs, por exemplo). A

distribuição do sistema de controle pode ser feita no nível das seqüências operacionais, bem como no nível dos supervisores modulares e sistema-produto.

Por fim, a formalidade da metodologia de síntese modular local e a implementação estruturada do controle, permite que o programa de controle seja calculado automaticamente a partir do modelo da planta, das especificações genéricas e das seqüências operacionais. Viabiliza-se portanto o desenvolvimento de *software* para a geração automática de código estruturado para controle supervisório de SEDs.

4. Controle Supervisório Multitarefa

Este capítulo propõe uma abordagem para o tratamento de múltiplas classes de tarefas no controle supervisório de SEDs. Conforme discutido no Capítulo 2, o uso de uma marcação única na modelagem de sistemas compostos pode dificultar o correto tratamento de múltiplos objetivos de controle na síntese de supervisores. Introduz-se aqui o gerador com marcação colorida (GMC), um tipo especial de autômato de Moore (MOORE, 1964), como um modelo que distingue classes de tarefas em SEDs. A seguir, generalizam-se algumas propriedades (como não-bloqueio) e algumas operações (como composição de geradores) para GMCs. A partir daí, os principais resultados da Teoria de Controle Supervisório são estendidos para este modelo, o que permite a síntese de supervisores mais refinados em problemas de controle nos quais a distinção de classes de tarefas é necessária. Também é investigada a verificação da propriedade de reversibilidade como uma forma alternativa para garantir a vivacidade de múltiplas tarefas. Três exemplos de problemas multitarefa ilustram a conveniência da abordagem. Os principais resultados deste capítulo são apresentados de modo resumido por QUEIROZ *et al.* (2004).

4.1 Motivação

Na TCS, o comportamento em malha aberta de SEDs é modelado por geradores, cujos estados marcados representam a realização de alguma tarefa. Com isso, uma cadeia de eventos é considerada uma tarefa completa sempre que o estado alcançado for marcado. O comportamento especificado para o sistema é expresso através de uma linguagem admissível, representando um conjunto de seqüências de eventos que não pode ser transgredido pela planta. Enquanto a linguagem admissível pode ser vista como uma especificação de segurança (garantindo que “nada de ruim” aconteça), a exigência de não-bloqueio pode ser compreendida como uma especificação de vivacidade que assegura que o supervisor não impeça a planta de completar tarefas (que “algo de bom” aconteça). Outras classes de especificações de vivacidade, como estabilidade (OZVEREN e WILLSKY, 1991) e justiça (GOHARI, 2002), também têm sido estudadas no contexto da TCS.

A abordagem de Ramadge e Wonham provê algoritmos computacionais para a síntese de supervisores que, através da desabilitação de eventos controláveis, restringe o

comportamento da planta de forma a respeitar a linguagem admissível, evitando bloqueio. Nota-se que tal lógica de controle não garante que uma tarefa seja eventualmente completada, mas ela sempre habilita um caminho para completar uma tarefa. Dessa forma, esses supervisores não podem forçar a realização de tarefas, em conformidade com o fato que eles não podem nem mesmo diretamente forçar a ocorrência de eventos.

Em muitos problemas reais, diversos tipos de tarefas são executadas concomitantemente. Particularmente, vários problemas interessantes sobre SEDs envolvendo múltiplas tarefas podem ser encontrados na área de sistemas de manufatura e de comunicação (FABIAN e KUMAR, 1997; THISTLE *et al.*, 1997). Nessas situações, a modelagem da planta por um gerador com marcação única pode ser problemática, já que a realização de qualquer tipo de tarefa seria assim identificada pela mesma marcação. Essa perda de informação pode restringir a qualidade dos supervisores.

Especialmente em sistemas compostos, a marcação de cada subsistema costuma ter um significado próprio. Nesse caso, o modelo global, representando a composição de todos os subsistemas, identifica uma cadeia de eventos como completa se e somente se ela levar todos os subsistemas a um estado marcado. Assim, pela abordagem original da TCS, os supervisores são levados a considerar uma tarefa como completa somente se todos os subsistemas envolvidos estiverem em um estado marcado. Dessa forma, pela abordagem original da TCS, um supervisor não-bloqueante garante que todos os subsistemas possam completar uma tarefa no mesmo estado global. Essa restrição pode ser muito conservadora para alguns problemas.

Por exemplo, considera-se a seguinte alteração do problema clássico de controle da movimentação de um gato e um rato dentro de um labirinto, apresentado por RAMADGE e WONHAM (1989), que pode representar, por exemplo, a movimentação de veículos autoguiados no chão de fábrica. O labirinto contém cinco recintos, num dos quais (cozinha) há comida disponível, ligados por portas exclusivas e direcionais[5]. Os eventos para o modelo do gato e do rato representam suas passagens através das portas. Ambos os modelos consideram completa qualquer seqüência de eventos que leve o respectivo animal à cozinha. Seria desejável que os supervisores garantissem ao gato e ao rato o acesso à comida, sem que ambos pudessem ocupar instantaneamente o mesmo compartimento do labirinto. Porém, por não distinguir os dois tipos de tarefas, o sistema de controle considera marcado apenas o estado em que os dois subsistemas completam uma tarefa, ou seja, em que o gato e rato estão na cozinha ao mesmo tempo. Como esse estado não é permitido, a abordagem tradicional de síntese não admite solução. Por outro lado, mesmo que o modelo

[5] Mais detalhes são encontrados na Figura 26, na página 92.

global marcasse todos os estados em que o rato ou o gato estivessem na cozinha, a solução ótima não permitiria necessariamente aos dois animais comer nem distinguiria qual animal está comendo, já que a propriedade de não-bloqueio garante apenas que pelo menos um estado marcado seja sempre alcançável.

O objetivo do trabalho a seguir é proporcionar um mecanismo conveniente para a síntese de supervisores minimamente restritivos que assegurem a vivacidade de múltiplas tarefas. Tais supervisores devem desabilitar um evento se e somente se ele puder iniciar uma seqüência não-controlável que leve o sistema para fora do comportamento seguro ou que torne alguma tarefa inacessível.

FABIAN e KUMAR (1997) também apresentam uma abordagem para sintetizar supervisores mutuamente não-bloqueantes para sistemas restringidos por especificações disjuntivas, que poderiam ser interpretadas como diferentes tarefas. Os resultados apresentados neste capítulo têm uma característica diferente uma vez que neste trabalho as tarefas são definidas principalmente pela planta e a especificação é obtida pela composição de todas as regras.

THISTLE e MALHAMÉ (1997) e THISTLE *et al*. (1997) estudam questões similares que surgem na interação de serviços em redes de telefonia, onde a especificações para um novo serviço define uma nova linguagem marcada. Eles assumem que a especificação seja modelada por um autômato com múltiplos conjuntos de estados marcados (associados a serviços diferentes) e estendem os resultados básicos da TCS para assegurar o não-bloqueio de múltiplas linguagens marcadas.

Na abordagem para controle supervisório multitarefa proposta neste capítulo, os SEDs são modelados por autômatos de Moore, cujas saídas indicam a realização de tarefas. Como cada tarefa pode ser associada a uma linguagem marcada, alguns dos resultados aqui apresentados coincidem em muitos aspectos com aqueles de THISTLE e MALHAMÉ (1997). Contudo, o presente trabalho é desenvolvido num contexto mais geral, onde a ocorrência de tarefas pode estar implícita na planta ou definida pelas especificações. Além do mais, promovem-se várias contribuições adicionais para a teoria, que incluem a composição de SEDs multitarefa, a generalização do conceito de $L_m(G)$–fechamento e o estudo de reversibilidade.

A seguir, é introduzida a abordagem para modelagem de SEDs com múltiplas tarefas e são discutidas algumas propriedades e operações sobre esse modelo. Na seção seguinte, estendem-se os principais resultados da TCS para tratar geradores com múltiplas marcações. Na seqüência, são resolvidos três problemas ilustrativos de controle

supervisório multitarefa. Finalmente, investiga-se o uso da reversibilidade como um modo alternativo de assegurar a vivacidade de múltiplas tarefas.

4.2 Sistemas a Eventos Discretos Multitarefa

Diz-se que um SED completa uma tarefa quando ele executa uma seqüência de eventos que atinge um objetivo do problema de controle. Por exemplo, uma máquina pode completar uma tarefa chamada "produzir uma peça" sempre que atingir o estado inicial. Duas tarefas pertencem à mesma classe quando ambas estão associadas a objetivos equivalentes, isto é, quando têm o mesmo significado no problema de controle. Quando um SED inclui múltiplas classes de tarefas, é chamado de sistema a eventos discretos multitarefa (SEDMT).

4.2.1 Comportamento colorido

Com o intuito de diferenciar as múltiplas classes de tarefas de um SEDMT, são associadas cores (etiquetas) a estas. Seja Σ o conjunto de todos os eventos que podem ocorrer no sistema e C o conjunto de todas as cores. Para cada cor $c \in C$ pode-se associar uma linguagem $L_c \in Pwr(\Sigma^*)$ que representa o conjunto de todas as seqüências de eventos de Σ que completam tarefas da respectiva classe. Desse modo, o comportamento colorido de um SEDMT pode ser modelado pelo conjunto $\{(L_c, c), c \in C\}$. De forma genérica, pode-se definir um *comportamento colorido* $\Lambda_C \in Pwr(Pwr(\Sigma^*) \times C)$ como um conjunto de pares (linguagem, cor), com a restrição de que quaisquer dois pares distintos tenham cores distintas.

Para um comportamento colorido Λ_C , a *linguagem marcada por* $c \in C$ é definida por $L_c(\Lambda_C) := L$ tal que $(L, c) \in \Lambda_C$. A *linguagem marcada por* $B \subseteq C$, com $B \neq \varnothing$, é definida por $L_B(\Lambda_C) := \cup_{b \in B}\{L$ tal que $(L, b) \in \Lambda_C\}$. A *linguagem gerada* por Λ_C é o conjunto de todas cadeias que podem completar qualquer tarefa de C, isto é, $L(\Lambda_C) := \overline{L_C(\Lambda_C)}$. Por conveniência, define-se $L_\varnothing(\Lambda_C) := L(\Lambda_C)$.

Em alguns problemas, pode acontecer que um SED não inclua qualquer classe de tarefas e, portanto, o conjunto de cores seja vazio. Para esse caso particular, o comportamento pode ser descrito simplesmente por uma linguagem prefixo-fechada L contendo qualquer seqüência de eventos que possa acontecer no sistema. Para representar tal comportamento como um comportamento colorido, pode-se reservar a cor **v** (de vácuo) para uma tarefa sem significado que é completada por qualquer cadeia de L. Então, o comportamento colorido de tal SED seria dado por $\Lambda_C := \{(L, \mathbf{v})\}$, com $C = \{\mathbf{v}\}$.

Seja $\Lambda_C \in Pwr(Pwr(\Sigma^*) \times C)$ um comportamento colorido. O prefixo-fechamento de Λ_C é dado por

$$\overline{\Lambda_C} := \{(\overline{L_c(\Lambda_C)}, c), \forall c \in C\}.$$

Sejam $M_B \in Pwr(Pwr(\Sigma^*) \times B)$ e $N_C \in Pwr(Pwr(\Sigma^*) \times C)$ comportamentos coloridos. Diz-se que $M_B \subseteq N_C$ se $B \subseteq C$ e $\forall b \in B$, $L_b(M_B) \subseteq L_b(N_C)$. A união de M_B e N_C é definida por:

$$M_B \cup N_C := \{(L_b(M_B), b), \forall b \in B - C\} \cup \{(L_b(N_C), b), \forall b \in C - B\}$$
$$\cup \{(L_b(M_B) \cup L_b(N_C), b), \forall b \in B \cap C\}.$$

Proposição 1: Seja E um conjunto não-vazio de cores em $B \cap C$. Então $L_E(M_B \cup N_C) = L_E(M_B) \cup L_E(N_C)$.

Prova:
$$\begin{aligned}
L_E(M_B \cup N_C) &= \cup_{c\in E}\{L\colon (L, c) \in M_B \cup N_C\} \\
&= \cup_{c\in E}\{L\colon (L, c) \in [(L_b(M_B), b), \forall b \in B - C] \\
&\qquad \cup [(L_b(N_C), b), \forall b \in C - B] \\
&\qquad \cup [(L_b(M_B) \cup L_b(N_C), b), \forall b \in B \cap C]\} \\
&= \cup_{c\in E}\{L\colon (L, c) \in [(L_b(M_B) \cup L_b(N_C), b), \forall b \in B \cap C]\} \\
&= \cup_{c\in E}\{L\colon (L, c) \in [(L_b(M_B) \cup L_b(N_C), b), \forall b \in E]\} \\
&= \cup_{c\in E}\{L_c(M_B) \cup L_c(N_C)\} \\
&= [\cup_{c\in E}\{L_c(M_B)\}] \cup [\cup_{c\in E}\{L_c(N_C)\}] \\
&= L_E(M_B) \cup L_E(N_C). \qquad \blacklozenge
\end{aligned}$$

O resultado anterior pode ser reformulado para um conjunto arbitrário de comportamentos coloridos, todos com o mesmo conjunto não-vazio de cores E, da seguinte forma:

Proposição 2: Seja X um conjunto de índices e sejam M_{Ex}, para $x \in X$, comportamentos coloridos com conjuntos de cores E. Então, a linguagem marcada por E na união dos comportamentos coloridos é igual à união das linguagens marcadas por E em cada comportamento colorido, ou seja, $L_E(\cup_{x\in X} M_{Ex}) = \cup_{x\in X} L_E(M_{Ex})$.

Prova:
$$\begin{aligned}
L_E(\cup_{x\in X} M_{Ex}) &= \cup_{c\in E}\{L\colon (L, c) \in \cup_{x\in X} M_{Ex}\} \\
&= \cup_{c\in E}\{L\colon (L, c) \in [(\cup_{x\in X} L_e(M_{Ex}), e), \forall e \in E]\} \\
&= \cup_{c\in E}\{\cup_{x\in X} L_c(M_{Ex})\} \\
&= \cup_{x\in X}\{\cup_{c\in E} L_c(M_{Ex})\} \\
&= \cup_{x\in X} L_E(M_{Ex}). \qquad \blacklozenge
\end{aligned}$$

A interseção de comportamentos coloridos M_B e N_C é definida por:

$$M_B \cap N_C := \{(L_b(M_B) \cap L_b(N_C), b), \forall b \in B \cap C\}.$$

Proposição 3: Seja E um conjunto não-vazio de cores em $B \cap C$. É sempre verdade que $\forall b \in E$, $L_b(\mathrm{M}_B \cap \mathrm{N}_C) = L_b(\mathrm{M}_B) \cap L_b(\mathrm{N}_C)$, mas, em geral, $L_E(\mathrm{M}_B \cap \mathrm{N}_C) \subseteq L_E(\mathrm{M}_B) \cap L_E(\mathrm{N}_C)$.

Prova: A primeira afirmação decorre diretamente da definição de interseção.
Pela definição de linguagem marcada por E, é claro que $L_E(\mathrm{M}_B \cap \mathrm{N}_C) \subseteq L_E(\mathrm{M}_B)$ e que $L_E(\mathrm{M}_B \cap \mathrm{N}_C) \subseteq L_E(\mathrm{N}_C)$. Portanto, $L_E(\mathrm{M}_B \cap \mathrm{N}_C) \subseteq L_E(\mathrm{M}_B) \cap L_E(\mathrm{N}_C)$.
Agora, considerando por exemplo $E = B = C = \{\mathbf{a}, \mathbf{b}\}$, $\Sigma = \{\alpha, \beta\}$, $\mathrm{M}_B = \{(\{\alpha\}, \mathbf{a}), (\{\beta\}, \mathbf{b})\}$ e $\mathrm{N}_C = \{(\{\beta\}, \mathbf{a}), (\{\alpha\}, \mathbf{b})\}$. Têm-se $L_E(\mathrm{M}_B) = L_E(\mathrm{N}_C) = \{\alpha, \beta\}$ e $\mathrm{M}_B \cap \mathrm{N}_C = \{(\{\}, \mathbf{a}), (\{\}, \mathbf{b})\}$.
Nesse caso, $L_E(\mathrm{M}_B) \cap L_E(\mathrm{N}_C) = \{\alpha, \beta\} \supset \{\} = L_E(\mathrm{M}_B \cap \mathrm{N}_C)$. ♦

Sejam os comportamentos coloridos $\mathrm{M}_{B1} \subseteq Pwr(Pwr(\Sigma_1^*) \times B_1)$ e $\mathrm{N}_{B2} \subseteq Pwr(Pwr(\Sigma_2^*) \times B_2)$. A composição síncrona de M_{B1} e N_{B2} é dada por

$$\mathrm{M}_{B1} \,||\, \mathrm{N}_{B2} := \{(L_b(\mathrm{M}_{B1}) \,||\, L_b(\mathrm{N}_{B2}), b), \forall b \in B_1 \cap B_2\}$$

$$\cup \{(L_b(\mathrm{M}_{B1}) \,||\, \overline{L_{B2}(\mathrm{N}_{B2})}, b), \forall b \in B_1 - B_2\}$$

$$\cup \{(\overline{L_{B1}(\mathrm{M}_{B1})} \,||\, L_b(\mathrm{N}_{B2}), b), \forall b \in B_2 - B_1\}.$$

4.2.2 Gerador com Marcação Colorida

Uma forma direta de se representar um comportamento colorido é modelar cada linguagem por um gerador, de modo que um SEDMT seja modelado por um conjunto de pares (gerador, cor). Por conveniência, um SEDMT pode ser modelado por um gerador especial, cujos estados são marcados por subconjuntos de cores de acordo com as classes de tarefas completadas. Essa particular máquina de estados, denominada gerador com marcação colorida (GMC), é definida formalmente como uma sêxtupla:

$$G := (Q, \Sigma, C, \delta, \chi, q_0),$$

onde:

- Q: conjunto de estados;
- Σ: conjunto de eventos;
- C: conjunto de cores;
- δ: $Q \times \Sigma \rightarrow Q$: função de transição de estados (estendida para cadeias como usual);
- χ: $Q \rightarrow Pwr(C)$: função de marcação;
- q_0: estado inicial.

Pode-se associar ao gerador com marcação colorida G uma função de eventos ativos Γ: $Q \rightarrow Pwr(\Sigma)$, que associa cada estado $q \in Q$ a um subconjunto de Σ com todos eventos que possam ocorrer em q, ou seja, $\Gamma(q) := \{\sigma: \sigma \in \Sigma \text{ e } \delta(q, \sigma)!\}$.

Esse modelo acrescenta ao autômato usual um conjunto de cores C, representando todas as classes de tarefas que um sistema pode executar, e uma função de marcação χ, que atribui a cada estado de Q um subconjunto (vazio ou não) de cores. Consequentemente, um GMC é basicamente um autômato de Moore (MOORE, 1964), cujas saídas, representadas por subconjuntos de cores, definem as classes de tarefas que são completadas após uma seqüência de eventos. Alternativamente, um GMC pode ser representado por um autômato com um conjunto indexado de estados marcados, como proposto por THISTLE *et al.* (1997).

O uso de máquinas de Moore em controle supervisório não é novidade na literatura. Na abordagem de controle hierárquico proposta por ZHONG e WONHAM (1990), as saídas dos modelos operacionais sinalizam a ocorrência de eventos relevantes para o nível gerencial. RAMIREZ *et al.* (1999) atribuem um vetor com sinais de controle às saídas do autômato de Moore modelando o sistema. ZAD *et al.* (1998) também fazem uso de autômatos de Moore no processo de diagnóstico de falhas em sistemas a eventos discretos.

É importante observar que o uso de cores em Redes de Petri Coloridas (JENSEN, 1992) tem um sentido completamente diferente da abordagem aqui proposta. Enquanto num Gerador com Marcação Colorida as cores são relacionadas apenas às saídas da função de marcação, identificando e classificando as tarefas completas, as cores de uma Rede de Petri Colorida são associadas às fichas e são variáveis da função de incidência, permitindo diferenciar entidades.

4.2.3 Linguagens associadas a um GMC

A linguagem gerada por um gerador com marcação colorida G, denotada por $L(G)$, representa todas as possíveis cadeias finitas de eventos que são alcançadas a partir do estado inicial q_0. Como esse conceito independe da marcação, $L(G)$ é formalmente definida da mesma forma que a linguagem gerada por um gerador usual por:

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

A linguagem marcada por um gerador representa o conjunto de cadeias de eventos geradas que completam uma tarefa. Como um GMC costuma envolver múltiplas classes de tarefas, pode-se definir uma linguagem marcada para cada classe de tarefa como o conjunto de cadeias que levem a estados cujas funções de marcação contenham a cor

relativa àquela classe. Assim, $L_c(G)$, a linguagem marcada por $c \in C$, é formalmente definida por:

$$L_c(G) := \{s: s \in L(G) \wedge c \in \chi(\delta(q_0, s))\}.$$

O conceito de linguagem marcada por uma cor pode ser estendido para um subconjunto não-vazio de cores como o conjunto de cadeias de eventos que completem qualquer tarefa representada por alguma das cores em questão. Então, para um conjunto de cores $\varnothing \subset B \subseteq C$, define-se a linguagem marcada por B como:

$$L_B(G) := \{s: s \in L(G) \wedge B \cap \chi(\delta(q_0, s)) \neq \varnothing\}.$$

Quando não se especifica o conjunto de cores de uma linguagem marcada, subentende-se que se trata da linguagem $L_C(G)$ que contém todas as cadeias de eventos de G que completem qualquer tarefa. Pode-se provar que a linguagem marcada por um conjunto de cores é a união de todas as linguagens marcadas pelas cores desse conjunto. Essa idéia é representada pela seguinte proposição:

Proposição 4: Seja B um conjunto não-vazio de cores do GMC G. É verdade que $L_B(G) = \cup_{b \in B} L_b(G)$.

Prova: $L_B(G) = \{s: s \in L(G) \wedge [B \cap \chi(\delta(q_0, s)) \neq \varnothing]\}$
$= \{s: s \in L(G) \wedge [\exists b \in B: b \in \chi(\delta(q_0, s))]\}$
$= \cup_{b \in B}\{s: s \in L(G) \wedge b \in \chi(\delta(q_0, s))\}$
$= \cup_{b \in B} L_c(G)$. ♦

O comportamento colorido de um GMC G, denotado por $\Lambda_C(G)$, é definido pelo conjunto de todas linguagens marcadas de G associadas com suas respectivas cores, isto é,

$$\Lambda_C(G) := \{(L_c(G), c), c \in C\}.$$

4.2.4 Relações entre GMCs

Há várias formas de se construir um gerador que marque (ou gere) um determinado conjunto de linguagens associadas a cores definidas. Diz-se que dois geradores com marcação colorida são equivalentes quando seus conjuntos de cores forem iguais, gerarem a mesma linguagem e marcarem por cada cor a mesma linguagem, ou seja, quando possuírem a mesma linguagem gerada e o mesmo comportamento colorido (e, consequentemente, o mesmo conjunto de cores). Formalmente, os GMCs $G_1 = (Q_1, \Sigma_1, C_1, \delta_1, \chi_1, q_{01})$ e $G_2 = (Q_2, \Sigma_2, C_2, \delta_2, \chi_2, q_{02})$ são equivalentes se:

$$L(G_1) = L(G_2);$$

$$\Lambda_{C1}(G_1) = \Lambda_{C2}(G_2).$$

Define-se a relação $\subseteq$ entre GMCs (lida como "subgerador de") por $G_1 \subseteq G_2$ sempre que:

- $Q_1 \subseteq Q_2$;
- $\Sigma_1 = \Sigma_2$;
- $C_1 = C_2$;
- $\delta_1(q, s) = q' \Rightarrow \delta_2(q, s) = q'$;
- $\chi_1 = \chi_2|Q_1$;
- $q_{01} = \begin{cases} q_{02} & \text{se } q_{02} \in Q_1, \\ \varnothing & \text{senão.} \end{cases}$

Intuitivamente, G_1 é um subgerador de G_2 sempre que G_1 possa ser obtido de G_2 pela extração de alguns estados e/ou transições. Claramente, $G_1 \subseteq G_2$ implica $L(G_1) \subseteq L(G_2)$ e $\Lambda_{C1}(G_1) \subseteq \Lambda_{C2}(G_2)$, mas a implicação reversa nem sempre é verdadeira.

4.2.5 Propriedades de GMCs

Diz-se que um estado $q \in Q$ é acessível se puder ser alcançado por uma seqüência de transições a partir do estado inicial q_0, ou seja, se

$$\exists s \in \Sigma^* \text{ tal que } \delta(q_0, s) = q.$$

Denota-se por Q_{ac} o conjunto de todos estados acessíveis de Q. G é acessível se q for acessível para todo $q \in Q$ ($Q = Q_{ac}$).

Diz-se que um estado $q \in Q$ é fracamente coacessível e.r.a B se houver uma seqüência de transições que, partindo de q, leve a um estado que marque pelo menos uma cor de B, isto é, se

$$\exists b \in B, \exists s \in \Sigma^* \text{ tal que } b \in \chi(\delta(q, s)).$$

Denota-se por $Q_{wco,B}$ o conjunto de todos estados de Q que são fracamente coacessíveis e.r.a B. Assim, G é fracamente coacessível e.r.a B quando qualquer estado $q \in Q$ for fracamente coacessível e.r.a B ($Q = Q_{wco,B}$).

Diz-se que um estado $q \in Q$ é fortemente coacessível e.r.a B se, para qualquer cor b de B, houver uma seqüência de transições que, partindo de q, leve a um estado que marque b, isto é, se

$$\forall b \in B, \exists s \in \Sigma^* \text{ tal que } b \in \chi(\delta(q, s)).$$

Denota-se por $Q_{sco,B}$ o conjunto de todos estados de Q que são fortemente coacessíveis e.r.a B. Claramente, $Q_{sco,B} = \cap_{b \in B} Q_{sco,\{b\}}$. Nota-se que é possível que de alguns estados $q \in Q_{sco,B}$ só se possa completar uma tarefa $b \in B$ em estados $q' \in Q - Q_{sco,B}$. G é fortemente coacessível e.r.a B se q for fortemente coacessível e.r.a B para todo $q \in Q$, isto é, se $Q = Q_{sco,B}$.

Diz-se que um gerador com marcação colorida G é fracamente aparado (*trim*) e.r.a B se G for acessível e fracamente coacessível e.r.a B, isto é, se $Q_{ac} = Q_{wco,B}$. Da mesma maneira, G é fortemente aparado e.r.a B se G for acessível e fortemente coacessível e.r.a B, isto é, se $Q_{ac} = Q_{sco,B}$.

4.2.6 Operações sobre GMCs

Para a definição formal de algumas operações sobre GMCs a seguir, define-se o gerador com marcações coloridas vazio para Σ e C como

$$\varnothing_{\Sigma,C} := (\varnothing, \Sigma, C, \varnothing, \varnothing, \varnothing).$$

A operação $Ac(G)$, que elimina todos estados não-acessíveis de G, é definida por

$$Ac(G) := \begin{cases} (Q_{ac}, \Sigma, C, \delta|(\Sigma \times Q_{ac}), \chi|(\Sigma \times Q_{ac}), q_0) & \text{se } Q_{ac} \neq \varnothing \\ \varnothing_{\Sigma,C} & \text{senão.} \end{cases}$$

Define-se $WTr(G, B)$ como uma operação sobre G que elimina todos estados que não são acessíveis e fracamente coacessíveis e.r.a B. Seja $Q_{wtr,B} := Q_{ac} \cap Q_{wco,B}$. Então,

$$WTr(G, B) := \begin{cases} (Q_{wtr,B}, \Sigma, C, \delta|(\Sigma \times Q_{wtr,B}), \chi|(\Sigma \times Q_{wtr,B}), q_0) & \text{se } Q_{wtr,B} \neq \varnothing \\ \varnothing_{\Sigma,C} & \text{senão.} \end{cases}$$

Além disso, define-se $STr(G, B)$ como uma operação sobre G que, em múltiplas iterações, elimina todos estados que não são acessíveis e fortemente coacessíveis e.r.a B. Para isso, define-se $Q_{str,B} := Q_{ac} \cap Q_{sco,B}$ e define-se a função $PSTr(G, B)$ como

$$PSTr(G, B) := \begin{cases} (Q_{str,B}, \Sigma, C, \delta|(\Sigma \times Q_{str,B}), \chi|(\Sigma \times Q_{str,B}), q_0) & \text{se } Q_{str,B} \neq \varnothing \\ \varnothing_{\Sigma,C} & \text{senão.} \end{cases}$$

É natural que, se G for fortemente coacessível e.r.a B, $PSTr(G, B)$ também o será. Porém, se não for, $PSTr(G, B)$ pode ainda não ser fortemente coacessível e.r.a B, uma vez que essa operação pode apagar estados de $Q - Q_{sco,B}$ que são necessários para a coacessibilidade de alguns estados em $Q_{sco,B}$. Nesse sentido, define-se a seqüência

$$G_0 = G,$$
$$G_{j+1} = PSTr(G_j, B), \quad j = 0, 1, \ldots .$$

Então, define-se *STr*(*G*, *B*) como o limite

$$STr(G, B) := \lim G_j \quad (j \to \infty).$$

Claramente, se *Q* for finito, o algoritmo para *STr*(*G*, *B*) converge num número finito de iterações.

Exemplo 1: Na Figura 20, apresenta-se o cálculo iterativo de *STr*(*G*, *B*) para um gerador com marcação colorida *G* com $B = \{\mathbf{a}, \mathbf{b}\}$.

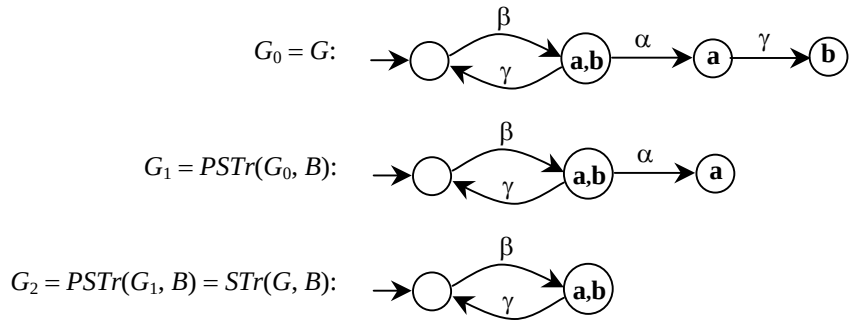


Figura 20: Exemplo de aparo forte

A composição síncrona de autômatos de Moore, não encontrada na literatura, implica a definição de uma função de saída que represente a composição das saídas dos sistemas envolvidos. Para isso, considera-se que, quando uma mesma cor estiver associada a estados de geradores modelando subsistemas SEDs distintos, a tarefa representada pela cor seja marcada pelo sistema composto apenas quando todos os subsistemas estiverem em estados que marquem a mesma cor. Assim, da mesma forma que a função de transição composta sincroniza os eventos em comum, a função de marcação composta sincroniza as cores compartilhadas.

Nesse sentido, define-se a composição síncrona de GMCs $G_1 = (Q_1, \Sigma_1, C_1, \delta_1, \chi_1, q_{01})$ e $G_2 = (Q_2, \Sigma_2, C_2, \delta_2, \chi_2, q_{02})$, com respectivas funções de eventos ativos Γ_1 e Γ_2, como o GMC

$$G_1 \parallel G_2 := Ac(Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, C_1 \cup C_2, \delta, \chi, (q_{01}, q_{02})),$$

onde:

- $$\delta((q_1, q_2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)), & \text{if } \sigma \in \Gamma_1(q_1) \cap \Gamma_2(q_2) \\ (\delta_1(q_1, \sigma), q_2), & \text{if } \sigma \in \Gamma_1(q_1) - \Sigma_2 \\ (q_1, \delta_2(q_2, \sigma)), & \text{if } \sigma \in \Gamma_2(q_2) - \Sigma_1 \\ \text{indefinida}, & \text{senão} \end{cases} ;$$

- $\chi((q_1, q_2)) = [\chi_1(q_1) \cup (C_2 - C_1)] \cap [\chi_2(q_2) \cup (C_1 - C_2)]$
  $= [\chi_1(q_1) \cap \chi_2(q_2)] \cup [\chi_1(q_1) - C_2] \cup [\chi_2(q_2) - C_1]$.

A função de eventos ativos Γ associada a $G_1 \parallel G_2$ é calculada por:

$$\Gamma((q_1, q_2)) = [\Gamma_1(q_1) \cup (\Sigma_2 - \Sigma_1)] \cap [\Gamma_2(q_2) \cup (\Sigma_1 - \Sigma_2)]$$
$$= [\Gamma_1(q_1) \cap \Gamma_2(q_2)] \cup [\Gamma_1(q_1) - \Sigma_2] \cup [\Gamma_2(q_2) - \Sigma_1].$$

Pode-se verificar que:

- $L(G_1 \parallel G_2) = L(G_1) \parallel L(G_2)$;
- $L_c(G_1 \parallel G_2) = \begin{cases} L_c(G_1) \parallel L_c(G_2), & \text{se } c \in C_1 \cap C_2 \\ L_c(G_1) \parallel L(G_2), & \text{se } c \in C_1 - C_2 \\ L(G_1) \parallel L_c(G_2), & \text{se } c \in C_2 - C_1 \end{cases}$.

Se $L(G_1) = \overline{L_{C1}(G_1)}$ e $L(G_2) = \overline{L_{C2}(G_2)}$, tem-se:

- $\Lambda_C(G_1 \parallel G_2) = \Lambda_{C1}(G_1) \parallel \Lambda_{C2}(G_2)$.

No caso particular em que $\Sigma_1 = \Sigma_2$ e $C_1 = C_2 = C$, tem-se:

- $L(G_1 \parallel G_2) = L(G_1) \cap L(G_2)$;
- $\Lambda_C(G_1 \parallel G_2) = \Lambda_C(G_1) \cap \Lambda_C(G_2)$.

Exemplo 2: Na Figura 21, apresenta-se a composição síncrona de G_1 e G_2, cujos alfabetos são respectivamente $\{\alpha, \gamma\}$ e $\{\beta, \gamma\}$ e os conjuntos de cores são respectivamente {**a**, **b**} e {**a**, **c**}.

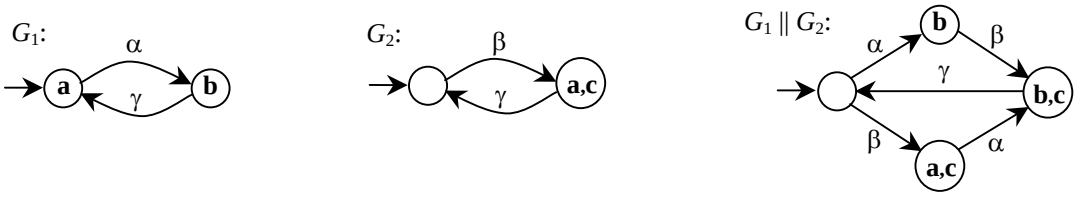


Figura 21: Exemplo de composição síncrona

Proposição 5: Dado um comportamento colorido $\Lambda_C = \{(L_c, c), c \in C\}$, existe um gerador com marcação colorida $G = (Q, \Sigma, C, \delta, \chi, q_0)$ tal que:

- $\forall c \in C, L_c(G) = L_c$;
- $L(G) = \cup_{c \in C} \overline{L_c}$.

Prova: Nota-se que $\cup_{c \in C} \overline{L_c}$ é prefixo-fexhada e, $\forall c \in C$, contém L_c. Então, para cada cor $c \in C$, pode-se obter um gerador usual $H_c = (Q_c, \Sigma, \delta_c, q_{0c}, Q_{mc})$ tal que $L_m(H_c) = L_c$ e $L(H_c) = \cup_{c \in C} \overline{L_c}$. Portanto, para cada cor $c \in C$, há um CMG $G_c = (Q_c, \Sigma, \{c\}, \delta_c, \chi_c, q_{0c})$

tal que $L_c(G_c) = L_c$ e $L(G_c) = \cup_{c\in C}\overline{L_c}$. Seja $G = ||_{c\in C}G_c$. Então:

$L(G) = L(||_{c\in C}G_c) = ||_{c\in C}L(G_c) = ||_{c\in C}(\cup_{c\in C}\overline{L_c}) = \cup_{c\in C}\overline{L_c}$;

$\forall c \in C, L_c(G) = L_c(||_{c\in C}G_c)$

$= L_c(G_c) \,||\, [||_{c\in C-\{c\}}L(G_c)]$

$= L_c \,||\, [||_{c\in C-\{c\}}(\cup_{c\in C}\overline{L_c})]$

$= L_c \,||\, [\cup_{c\in C}\overline{L_c}]$

$= L_c.$ ♦

Se o comportamento colorido Λ_C puder ser modelado por um gerador com marcação colorida G com um número finito de estados, diz-se que Λ_C é regular.

4.2.7 Bloqueio

A noção de bloqueio num gerador está relacionada com a idéia de se executar uma seqüência de eventos a partir da qual não seja possível completar uma tarefa. Quando o gerador compreende múltiplas cores, a idéia de bloqueio permite duas interpretações, dependendo da exigência de se poder completar tarefas de todas as classes (bloqueio forte) ou de pelo menos uma das classes de tarefas em questão (bloqueio fraco).

Dado um subconjunto não-vazio de cores B, diz-se que um gerador G é fracamente não-bloqueante e.r.a B se

$$\overline{L_B(G)} = L(G),$$

isto é, se qualquer seqüência de eventos gerada for o prefixo de pelo menos uma tarefa completa representada por uma das cores de B.

Dado um subconjunto não-vazio de cores B, diz-se que um gerador G é fortemente não-bloqueante e.r.a B se

$$\forall b \in B, \overline{L_b(G)} = L(G),$$

isto é, se qualquer seqüência de eventos gerada puder ser levada (não necessariamente pelo mesmo caminho nem no mesmo estado) a completar tarefas de todas as classes representadas por cores de B.

Quando o subconjunto de cores B não é especificado, subentende-se que $B = C$.

De acordo com a próxima proposição, o não-bloqueio forte ou fraco de a GMC $G = (Q, \Sigma, C, \delta, \chi, q_0)$ também pode ser analisado pelas propriedades de seus estados.

Proposição 6: Seja $\varnothing \subset B \subseteq C$, onde C é o conjunto de cores de um gerador com marcação colorida G. As seguintes implicações são verdadeiras:

a) G é fortemente coacessível e.r.a $B \Leftrightarrow \forall b \in B$, G for fracamente (ou fortemente) coacessível e.r.a $\{b\}$;

b) G é fracamente não-bloqueante e.r.a $B \Leftrightarrow Ac(G)$ for fracamente coacessível e.r.a B;

c) G é fortemente não-bloqueante e.r.a $B \Leftrightarrow Ac(G)$ for fortemente coacessível e.r.a B.

Provas:

a) $\forall q \in Q, \forall b \in B, \exists s \in \Sigma^*, b \in \chi(\delta(q, s))$
$\Leftrightarrow \forall b \in B, \{\forall q \in Q, [\exists c \in \{b\}]$ ou $[\forall c \in \{b\}], \exists s \in \Sigma^*, c \in \chi(\delta(q, s))\}$.

b) Nota-se que $\overline{L_B(G)} = \{u: \exists v \in \Sigma^*, uv \in L_B(G)\}$
$= \{u: \exists v \in \Sigma^*, B \cap \chi(\delta(q_0, uv)) \neq \varnothing\}$
$= \{u: \exists v \in \Sigma^*, \exists b \in B, b \in \chi(\delta(q_0, uv))\}$
$= \{u: \exists v \in \Sigma^*, \exists b \in B, \delta(q_0, u) = q, b \in \chi(\delta(q, v))\}$
$= \{u: u \in \Sigma^*, \delta(q_0, u) = q, \exists b \in B, \exists v \in \Sigma^*, b \in \chi(\delta(q, v))\}$.

Então, $\overline{L_B(G)} = L(G)$
$\Leftrightarrow \{u: u \in \Sigma^*, \delta(q_0, u) = q, \exists b \in B, \exists v \in \Sigma^*, b \in \chi(\delta(q, v))\}$
$= \{u: u \in \Sigma^*, \delta(q_0, u) = q\}$
$\Leftrightarrow \forall q \in Q_{ac}, \exists b \in B, \exists v \in \Sigma^*, b \in \chi(\delta(q, v))$.

c) Nota-se que $\overline{L_b(G)} = \{u: \exists v \in \Sigma^*, uv \in L_b(G)\}$
$= \{u: \exists v \in \Sigma^*, b \in \chi(\delta(q_0, uv))\}$
$= \{u: \exists v \in \Sigma^*, \delta(q_0, u) = q, b \in \chi(\delta(q, v))\}$
$= \{u: u \in \Sigma^*, \delta(q_0, u) = q, \exists v \in \Sigma^*, b \in \chi(\delta(q, v))\}$.

Então, $\forall b \in B, \overline{L_b(G)} = L(G)$
$\Leftrightarrow \forall b \in B, \{u: u \in \Sigma^*, \delta(q_0,u) = q, \exists v \in \Sigma^*, b \in \chi(\delta(q,v))\} = \{u: u \in \Sigma^*, \delta(q_0,u) = q\}$
$\Leftrightarrow \forall q \in Q_{ac}, \forall b \in B, \exists v \in \Sigma^*, b \in \chi(\delta(q, v))$. ♦

Exemplo 3: Seja o gerador com marcação colorida G apresentado na Figura 22. Pode-se observar que G é fracamente não-bloqueante e.r.a {**a**, **b**, **c**}, mas fortemente não-bloqueante apenas e.r.a {**b**, **c**}, pois a tarefa **a** só pode ser completada a partir do estado inicial.

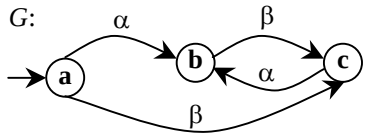


Figura 22: Exemplo de gerador com marcação colorida

Proposição 7: Seja $\varnothing \subset B_1 \subseteq B_2 \subseteq C$, onde C é o conjunto de cores de um gerador com marcação colorida G. As seguintes implicações são verdadeiras:

1. G fracamente não-bloqueante e.r.a $B_1 \Rightarrow G$ fracamente não-bloqueante e.r.a B_2;
2. G fortemente não-bloqueante e.r.a $B_2 \Rightarrow G$ fortemente não-bloqueante e.r.a B_1;
3. G fortemente não-bloqueante e.r.a $B_1 \Rightarrow G$ fracamente não-bloqueante e.r.a B_1.

Provas:

1. $B_1 \subseteq B_2 \Rightarrow L_{B_1}(G) = \cup_{b\in B_1} L_b(G) \subseteq \cup_{b\in B_2} L_b(G) = L_{B_2}(G) \Rightarrow \overline{L_{B_1}(G)} \subseteq \overline{L_{B_2}(G)}$.
   Então, $\overline{L_{B_1}(G)} = L(G) \Rightarrow L(G) = \overline{L_{B_1}(G)} \subseteq \overline{L_{B_2}(G)} \subseteq L(G) \Rightarrow \overline{L_{B_2}(G)} = L(G)$.
2. $(\forall b \in B_2, \overline{L_b(G)} = L(G)$ e $B_1 \subseteq B_2) \Rightarrow \forall b \in B_1, \overline{L_b(G)} = L(G)$.
3. $\forall b \in B_1, \overline{L_b(G)} = L(G) \Rightarrow \overline{L_{B_1}(G)} = \overline{\cup_{b\in B_1} L_b(G)} = \cup_{b\in B_1} \overline{L_b(G)} = \cup_{b\in B_1} L(G) = L(G)$. ♦

A noção de não-bloqueio forte é estendida para comportamento coloridos como segue. Um comportamento colorido $\Lambda_C \in Pwr(Pwr(\Sigma^*) \times C)$ é fortemente não-bloqueante e.r.a $B \subseteq C$ sempre que

$$\forall b \in B, \overline{L_b(\Lambda_C)} = \overline{L_C(\Lambda_C)},$$

ou seja, quando qualquer cadeia incompleta possa completar tarefas de todas as cores de B. Da mesma forma, Λ_C é fracamente não-bloqueante e.r.a B sempre que

$$\overline{L_B(\Lambda_C)} = \overline{L_C(\Lambda_C)}.$$

Proposição 8: Existe um GMC fortemente aparado e.r.a B que modela um comportamento colorido Λ_C se e somente se Λ_C for fortemente não-bloqueante e.r.a B.

Prova:
(se) Seja Λ_C tal que $\forall b \in B, \overline{L_b(\Lambda_C)} = \overline{L_C(\Lambda_C)}$. Pela Proposição 5,existe um GMC G tal que $\forall c \in C, L_c(G) = L_c(\Lambda_C)$ e $L(G) = \cup_{c\in C} \overline{L_c(\Lambda_C)} = \overline{L_C(\Lambda_C)}$.
Como $\forall b \in B, \overline{L_b(G)} = \overline{L_C(\Lambda_C)} = L(G)$, G é fortemente não-bloqueante e.r.a B. Daí, $Ac(G)$ é um GMC fortemente aparado e.r.a B que modela Λ_C.
(somente se) Seja G um GMC fortemente aparado tal que $\forall c \in C, L_c(G) = L_c(\Lambda_C)$ e $L(G) = \cup_{c\in C} \overline{L_c(\Lambda_C)}$. Então, $\forall b \in B, \overline{L_b(G)} = L(G)$ implica que
$\forall b \in B, \overline{L_b(\Lambda_C)} = \overline{L_b(G)} = L(G) = \cup_{c\in C} \overline{L_c(\Lambda_C)} = \overline{L_C(\Lambda_C)}$. ♦

Teorema 2: Seja o conjunto de comportamentos fortemente não-bloqueantes e.r.a *B* contidos em Λ_C definidos por $SNB(\Lambda_C, B) := \{M_C \subseteq \Lambda_C: \forall b \in B, \overline{L_b(M_C)} = \overline{L_C(M_C)}\}$. O conjunto $SNB(\Lambda_C, B)$ tem um elemento supremo $SupSNB(\Lambda_C, B)$ que representa o máximo comportamento fortemente não-bloqueante contido em Λ_C.

Prova:

(i) Define-se o comportamento $Z_C = \{(\varnothing, c), \forall c \in C\}$, tal que $L_C(Z_C) = \cup_{c \in C} L_c(Z_C) = \varnothing$. Então, Z_C é fortemente não-bloqueante e contido em Λ_C. Daí, $SNB(\Lambda_C, B)$ é não-vazio.

(ii) Seja X um conjunto de índices e sejam $M_{Cx} \in SNB(\Lambda_C, B)$, $x \in X$, comportamentos coloridos fortemente não-bloqueantes e.r.a *B*. Seja $M_C = \cup\{M_{Cx}, x \in X\}$. Então, $M_C \subseteq \Lambda_C$ e, pela Proposição 2, $L_C(M_C) = L_C(\cup\{M_{Cx}, x \in X\}) = \cup\{L_C(M_{Cx}), x \in X\}$.

Daí, $\forall b \in B$,

$$\begin{aligned}
\overline{L_b(M_C)} &= \overline{L_b(\cup\{M_{Cx}, x \in X\})} \\
&= \overline{\cup\{L_b(M_{Cx}), x \in X\}} \\
&= \cup\{\overline{L_b(M_{Cx})}, x \in X\} \\
&= \cup\{\overline{L_C(M_{Cx})}, x \in X\} \\
&= \overline{\cup\{L_C(M_{Cx}), x \in X\}} \\
&= \overline{L_C(M_C)}.
\end{aligned}$$

Portanto, $M_C \in SNB(\Lambda_C, B)$, ou seja, o não-bloqueio forte de comportamento coloridos é fechado sob uniões arbitrárias.

(iii) Como $SNB(\Lambda_C, B)$ é não-vazio e fechado sob uniões arbitrárias, ele tem um elemento supremo único definido por $SupSNB(\Lambda_C, B) := \cup\{M_C, M_C \in CSNB(\Lambda_C, B)\}$. ♦

Observa-se que $L_C(SupSNB(\Lambda_C, B)) \subseteq \cup_{c \in C}\overline{L_c(SupSNB(\Lambda_C, B))} = \cap_{b \in B}\overline{L_b(SupSNB(\Lambda_C, B))} \subseteq \cap_{b \in B}\overline{L_b(\Lambda_C)}$. Em palavras, o supremo comportamento fortemente não-bloqueante está contido nos prefixos comuns às linguagens marcadas por $b \in B$.

Quando o conjunto *B* de tarefas relevantes está subentendido, pode-se representar $SupSNB(\Lambda_C, B)$ por $SupSNB(\Lambda_C)$.

Proposição 9: Seja $G = (Q, \Sigma, C, \delta, \chi, q_0)$ um GMC finito que modela um comportamento colorido regular Λ_C. Então, $SupSNB(\Lambda_C, B) = \Lambda_C(STr(G, B))$.

Prova:

(⊇) Pela Proposição 8, sabe-se que $\Lambda_C(STr(G, B))$ é fortemente não-bloqueante e.r.a *B* e, portanto, $\Lambda_C(STr(G, B)) \subseteq SupSNB(\Lambda_C, B)$.

(⊆) Como *G* é finito, pode-se assumir que, para algum natural *k*,

$STr(G, B) = G_k \subseteq G_{k-1} \subseteq \ldots \subseteq G_0 = G$, onde $G_{j+1} = PSTr(G_j, B)$.

Supõe-se agora que $SupSNB(\Lambda_C, B) \not\subseteq \Lambda_C(STr(G, B))$.

Então, $\exists c \in C$, $L_c(SupSNB(\Lambda_C)) \not\subseteq L_c(G_k)$,
o que implica que $\exists c \in C$, $\exists s \in \Sigma^*$, $s \in L_c(SupSNB(\Lambda_C, B))$, $s \notin L_c(G_k)$.
Naturalmente, $s \in L_c(SupSNB(\Lambda_C, B)) \subseteq L_c(\Lambda_C) = L_c(G)$ de modo que $s \in L(G)$ e $c \in \chi(\delta(q_0, s))$. Por conseqüência, se $s \notin L_c(G_k)$, tem-se que $\delta(q_0, s) \notin Q_k$ o que implica que, para algum $0 \leq l_1 < k$, $\delta(q_0, s)$ é um estado de G_{l1} que não é fortemente coacessível e.r.a B. Assim, $\exists b_1 \in B$ tal que para $\forall w \in \Sigma^*$, $\delta(q_0, sw) \notin Q_{l1}$ ou $b_1 \notin \chi(\delta(q_0, sw))$. Como $s \in L_c(SupSNB(\Lambda_C, B))$ também é não-bloqueante e.r.a $\{b_1\}$, $\exists w_1 \in \Sigma^*$, $b_1 \in \chi(\delta(q_0, sw_1))$ e $\delta(q_0, sw_1) \notin Q_{l1}$, mas $sw_1 \in L(G)$.
Pelo mesmo argumento, para algum $0 \leq l_2 < l_1$, $\delta(q_0, s)$ é um estado de G_{l2} que não é fortemente coacessível . Daí, $\exists b_2 \in B$ tal que for $\forall w \in \Sigma^*$, $\delta(q_0, sw) \notin Q_{l2}$ ou $b_2 \notin \chi(\delta(q_0, sw))$.)). Como $sw_1 \in L_b(SupSNB(\Lambda_C, B))$ também é não-bloqueante e.r.a $\{b_2\}$, $\exists w_2 \in \Sigma^*$, $b_2 \in \chi(\delta(q_0, sw_1w_2))$ e $\delta(q_0, sw_1w_2) \notin Q_{l1}$, mas $sw_1w_2 \in L(G)$.
Continuando-se desse modo, chega-se indutivamente a $\exists b_n \in B$, $\exists w_n \in \Sigma^*$, $b_n \in \chi(\delta(q_0, sw_1w_2\ldots w_n))$ e $\delta(q_0, sw_1w_2\ldots w_n) \notin Q_0$, mas $sw_1w_2\ldots w_n \in L(G)$. Como $Q_0 = Q$, esta última afirmação é absurda e, portanto, $SupSNB(\Lambda_C, B) \subseteq \Lambda_C(STr(G, B))$. ♦

4.3 Controle Supervisório Multitarefa

Seja um sistema a eventos discretos em malha aberta modelado por um gerador com marcação colorida $G = (Q, \Sigma, C, \delta, \chi, q_0)$, com função de eventos ativos Γ, cujo alfabeto seja particionado entre eventos controláveis $\sigma \in \Sigma_c$, eventos que podem ser desabilitados, e eventos não-controláveis $\sigma \in \Sigma_u$. O objetivo do controle supervisório é gerar uma entidade (denominada supervisor) que, desabilitando eventos controláveis, evite que o sistema controlado viole um conjunto de condições impostas (chamado de especificação). Essa entidade deve também garantir que o sistema controlado seja sempre capaz de completar um determinado conjunto de tarefas relevantes, explicitadas pelas cores da especificação.

4.3.1 Especificações

Pode-se considerar que há basicamente duas classes principais de especificações para um SED: segurança e vivacidade. Especificações de segurança procuram garantir que "nada de ruim aconteça". O comportamento seguro de um SED pode ser expresso através de linguagens admissíveis ou através de estados proibidos, usualmente definidos por um predicado sobre a planta (WONHAM, 2003). Pode-se mostrar (WONHAM, 2003) que ambas abordagens são duais, desde que alguma memória adicional (estados) possa ser introduzida no modelo. Especificações de vivacidade asseguram que "algo de bom

aconteça". No caso da condição de não-bloqueio forte, "algo de bom" significa que há sempre um jeito – não necessariamente justo – de se completar todas as tarefas. Dessa maneira, a especificação de vivacidade para um SEDMT pode ser expressa através de um conjunto D de tarefas "importantes", para o qual se requer não-bloqueio forte.

Para um SEDMT, o conjunto de especificações impostas à planta define um comportamento colorido admissível. Como as especificações podem trazer novas informações (memória) ao modelo, é muito conveniente permitir que elas possam definir novas classes de tarefas. Por exemplo, a especificação para evitar *underflow* num depósito entre duas máquinas pode destacar o estado em que o depósito está vazio, o que poderia definir uma nova tarefa. Então, para um conjunto de cores relevantes D, o grupo de condições impostas à planta, chamado especificação, é representado pelas linguagens admissíveis marcadas por $d \in D$:

$$K_d \subset L_d(G), \text{ se } d \in C, \text{ ou}$$
$$K_d \subset L(G), \text{ se } d \notin C.$$

Assim, uma especificação pode ser representada por um conjunto de linguagens associadas a cores (comportamento colorido admissível)

$$\mathrm{A}_D := \{(K_d, d), d \in D\}.$$

Quando $D = \varnothing$, não há tarefa relevante definida para o problema de controle e, portanto, nenhuma questão de bloqueio está envolvida. Nesse caso particular, o comportamento colorido seria vazio por definição. A especificação poderia então ser dada por uma linguagem gerada admissível $K = \overline{K} \subset L(G)$ e o problema resolvido pela teoria clássica. Contudo, esse tipo de especificação ainda pode ser representado como um comportamento colorido se for introduzida a cor vácuo **v** ($D = \{\mathbf{v}\}$) e fazendo-se $\mathrm{A}_D = \{(K, \mathbf{v})\}$.

As cores de uma especificação não estão necessariamente contidas nas cores da planta. Assim, dependendo da relação entre as cores da planta e as cores relevantes, o supervisor pode ter ou não a necessidade de definir a marcação de novas tarefas. Essa característica diferencia os supervisores incolores e pintores apresentados nas próximas subseções.

4.3.2 Supervisor Incolor

Quando o conjunto de cores de uma especificação está contido nas cores da planta ($D \subseteq C$), as cores do sistema em malha fechada são definidas apenas pela marcação de

tarefas na planta. Assim, um supervisor *S*, neste caso chamado de incolor, consiste simplesmente numa função

$$S\colon L(G) \to Pwr(\Sigma)$$

que associa um conjunto de eventos a cada seqüência de eventos gerada pela planta, de forma que os eventos habilitados após a ocorrência de $s \in L(G)$ são dados por

$$S(s) \cap \Gamma(\delta(q_0, s)).$$

A Figura 23 ilustra a ação de um supervisor incolor sobre uma planta.

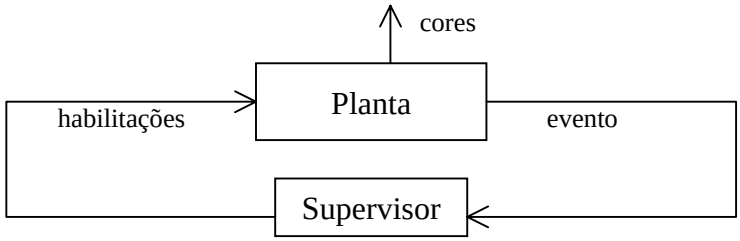


Figura 23: Supervisor incolor

Diz-se que um supervisor *S* é admissível se não implicar a desabilitação de eventos não-controláveis, isto é, se

$$\forall s \in L(G),\ \Sigma_u \cap \Gamma(\delta(q_0, s)) \subseteq S(s).$$

Define-se por *S*/*G* o SED (com múltiplas tarefas) que representa *S* controlando *G*. Com isso, a linguagem gerada por *S*/*G* é dada por:

1. $\varepsilon \in L(S/G)$;

2. $s\sigma \in L(S/G) \Leftrightarrow (s \in L(S/G)) \wedge (s\sigma \in L(G)) \wedge (\sigma \in S(s))$.

Já as linguagens marcadas por *S*/*G*, representando as cadeias marcadas pelo GMC da planta que permanecem após a ação de *S*, são definidas por:

$$L_C(S/G) := L(S/G) \cap L_C(G);$$

$$\forall c \in C,\ L_c(S/G) := L(S/G) \cap L_c(G).$$

Um supervisor incolor pode ser representado de modo mais conveniente na forma de um autômato sem marcação. Seja então o autômato

$$H = (\Sigma, X, \lambda, x_0),$$

onde:

- Σ: alfabeto de *G*;

- X: conjunto de estados;
- $\lambda: \Sigma \times X \rightarrow X$: função de transição de estados;
- x_0: estado inicial.

A função de eventos ativos associada a H é denotada por $\Phi: X \rightarrow Pwr(\Sigma)$. Diz-se que H implementa S sobre G se, para toda cadeia $s \in L(S/G)$ e qualquer evento $\sigma \in \Sigma$ tal que $s\sigma \in L(G)$, $\sigma \in \Phi(\lambda(x_0, s)) \Leftrightarrow \sigma \in S(s)$. Pode-se observar que um gerador sem cores que represente $L(S/G)$ também implementa S sobre G.

Assim, se S for admissível, o comportamento do sistema em malha fechada H/G, equivalente a S/G, pode ser obtido pela composição síncrona $H \parallel G$, definida por:

$$H/G := H \parallel G = Ac(Q \times X, \Sigma, C, \delta_{H/G}, \chi_{H/G}, (q_0, x_0))$$

onde:

- $\delta_{H/G}((q, x), \sigma) = \begin{cases} (\delta(q, \sigma), \lambda(x, \sigma)) & \text{se } \sigma \in \Gamma(q) \cap \Phi(x) \\ \text{indefinido} & \text{senão} \end{cases}$;
- $\chi_{H/G}((q, x)) = \chi(q)$.

A função de eventos ativos $\Gamma_{H/G}$ associada a H/G é calculada por:

$$\Gamma_{H/G}((q, x)) = \Gamma(q) \cap \Phi(x).$$

4.3.3 Supervisor Pintor

Muitas vezes as especificações incluem tarefas que não estão explícitas no comportamento em malha aberta da planta ($D - C = E \neq \varnothing$). Nesse caso, a função de marcar as novas cores no sistema controlado pertence ao supervisor. Um supervisor pintor é então definido como um mapeamento que associa a cada seqüência de eventos da planta um conjunto de eventos habilitados e um conjunto de novas cores (de E) representando tarefas completadas. Assim, um supervisor pintor S consiste numa função

$$S: L(G) \rightarrow Pwr(\Sigma) \times Pwr(E).$$

A ação de um supervisor pintor sobre uma planta é ilustrada na Figura 24.

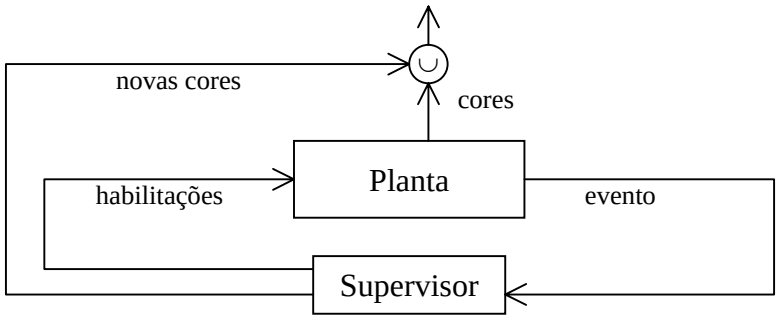


Figura 24: Supervisor pintor

Seja $S(s) = (\gamma, \mu)$, denota-se $\Re(S(s)) = \gamma$ e $\Im(S(s)) = \mu$. Os eventos que podem ocorrer após a ocorrência de uma cadeia de eventos $s \in L(G)$ são dados por

$$\Re(S(s)) \cap \Gamma(\delta(q_0, s)).$$

Diz-se que um supervisor pintor S é admissível se não implicar a desabilitação de eventos não-controláveis, isto é, se

$$\forall s \in L(G), \Sigma_u \cap \Gamma(\delta(q_0, s)) \subseteq \Re(S(s)).$$

A linguagem gerada pelo sistema controlado S/G é definida por:

1. $\varepsilon \in L(S/G)$;

2. $s\sigma \in L(S/G) \Leftrightarrow (s \in L(S/G)) \wedge (s\sigma \in L(G)) \wedge (\sigma \in \Re(S(s)))$.

Além de marcar com as cores de C as tarefas completas da planta que permanecem com a supervisão, o sistema controlado S/G marca com cores de E as tarefas definidas pelo supervisor pintor S. Assim as linguagens marcadas por S/G podem ser descritas por:

$$\forall c \in C, L_c(S/G) := L(S/G) \cap L_c(G);$$

$$L_C(S/G) := L(S/G) \cap L_C(G);$$

$$\forall e \in E, L_e(S/G) := \{s \in L(S/G): e \in \Im(S(s))\};$$

$$L_E(S/G) := \{s \in L(S/G): \Im(S(s)) \neq \varnothing\}.$$

Um supervisor pintor pode ser representado de forma mais conveniente por um gerador com marcação colorida. Seja o GMC

$$H = (X, \Sigma, E, \lambda, \kappa, x_0),$$

onde:

- X: conjunto de estados;

- Σ: alfabeto de G;
- E: conjunto de cores novas;
- $\lambda: \Sigma \times X \rightarrow X$: função de transição;
- $\kappa: X \rightarrow Pwr(E)$: função de marcação;
- x_0: estado inicial.

A função de eventos ativos associada a H é denotada por $\Phi: X \rightarrow Pwr(\Sigma)$. Diz-se que H implementa S sobre G se:

1. $\forall s \in L(S/G)$ e $\forall \sigma \in \Sigma$ tal que $s\sigma \in L(G)$, $\sigma \in \Phi(\lambda(x_0, s)) \Leftrightarrow \sigma \in \Re(S(s))$;
2. $\forall s \in L(S/G)$ e $\forall e \in E$, $e \in \kappa(\lambda(x_0, s))\} \Leftrightarrow e \in \Im(S(s))$.

Observa-se que basta apagar as cores de C de um GMC que modele S/G para obter um GMC que implementa S sobre G. Com isso, caso S seja admissível, o comportamento do sistema em malha fechada H/G, equivalente a S/G, pode ser computado pela composição síncrona $H \,||\, G$, definida por:

$$H/G := H \,||\, G = Ac(Q \times X, \Sigma, C \cup E, \delta_{H/G}, \chi_{H/G}, (q_0, x_0))$$

onde:

- $\delta_{H/G}((q, x), \sigma) = \begin{cases} (\delta(q, \sigma), \lambda(x, \sigma)) & \text{se } \sigma \in \Gamma(q) \cap \Phi(x) \\ \text{indefinida} & \text{senão} \end{cases}$;
- $\chi_{H/G}((q, x)) = \chi(q) \cup \kappa(x)$.

A função de eventos ativos $\Gamma_{H/G}$ associada a H/G é calculada por:

$$\Gamma_{H/G}((q, x)) = \Gamma(q) \cap \Phi(x).$$

4.3.4 Existência de Supervisores

Diz-se que um supervisor (incolor ou pintor) S é fortemente não-bloqueante e.r.a D se

$$\forall d \in D, \overline{L_d(S/G)} = L(S/G).$$

Da mesma forma, S é fracamente não-bloqueante e.r.a D se

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

Diz-se que uma especificação A_D é *D*–fechada e.r.a *G* se, para qualquer cor $d \in D \cap C$, toda cadeia de eventos, que seja prefixo de $L_d(A_D)$ e seja marcada por *d* pela planta, for marcada por *d* também pela especificação, ou seja, se

$$\forall d \in (D \cap C),\ L_d(A_D) = \overline{L_d(A_D)} \cap L_d(G).$$

Diz-se que um comportamento colorido A_D é controlável em relação a *G* se a união de todas as suas linguagens $L_d(A_D)$ for controlável, ou seja, se

$$\overline{L_D(A_D)}\Sigma_u \cap L(G) \subseteq \overline{L_D(A_D)}.$$

O resultado a seguir é uma extensão do Teorema 3.4.1 de WONHAM (2003).

Teorema 3: Seja uma especificação A_D colorida por *D* ($D - C = E$) , tal que, $\forall d \in D \cap C,\ \varnothing \subset L_d(A_D) \subseteq L_d(G)$, e $\forall d \in E,\ \varnothing \subset L_d(A_D) \subseteq L(G)$. As condições necessárias e suficientes para a existência de um supervisor <u>pintor</u> fortemente não-bloqueante e.r.a *D* tal que $\Lambda_D(S/G) = A_D$ e $L(S/G) = \overline{L_D(A_D)}$ são:

1. controlabilidade de A_D e.r.a *G*;

2. não-bloqueio forte de A_D e.r.a *D*;

3. *D*–fechamento de A_D e.r.a *G*.

Prova:

(se) Define-se um supervisor pintor *S* tal que, $\forall s \in \overline{L_D(A_D)}$,
$S(s) = (\Sigma_u \cup \{\sigma \in \Sigma_c: s\sigma \in \overline{L_D(A_D)}\}, \{d \in E: s \in L_d(A_D)\})$.
Afirma-se que $L(S/G) = \overline{L_D(A_D)}$.
Inicialmente, mostra-se que $L(S/G) \subseteq \overline{L_D(A_D)}$. Supõe-se que $s\sigma \in L(S/G)$, isto é,
$s \in L(S/G)$, $\sigma \in \Re(S(s))$ e $s\sigma \in L(G)$. Assumindo indutivamente que $s \in \overline{L_D(A_D)}$, tem-se que $\sigma \in \Sigma_u$ implica que $s\sigma \in \overline{L_D(A_D)}\Sigma_u \cap L(G) \subseteq \overline{L_D(A_D)}$;
sendo que $\sigma \in \Sigma_c$ implica que $s\sigma \in \overline{L_D(A_D)}$ pela definição de $S(s)$.
Para mostrar que $\overline{L_D(A_D)} \subseteq L(S/G)$, supõe-se que $s\sigma \in \overline{L_D(A_D)}$. Como
$\forall d \in D$, $L_d(A_D) \subseteq L_d(G) \subseteq L(G)$, tem-se $L_D(A_D) = \cup_{d \in D} L_d(A_D) \subseteq L(G)$ e, portanto,
$\overline{L_D(A_D)} \subseteq L(G)$. Daí $s\sigma \in L(G)$.
Assumindo indutivamente que $s \in L(S/G)$, tem-se que $\sigma \in \Sigma_u$ implica que $\sigma \in \Re(S(s))$, de modo que $s\sigma \in L(S/G)$; enquanto $\sigma \in \Sigma_c$ e $s\sigma \in \overline{L_D(A_D)}$ implica $\sigma \in \Re(S(s))$ e, assim, $s\sigma \in L(S/G)$. A afirmação está provada.
Ainda, tem-se que

$$\begin{aligned} \forall d \in D \cap C,\ L_d(S/G) &= L(S/G) \cap L_d(G) \quad \text{(por definição)} \\ &= \overline{L_D(A_D)} \cap L_d(G) \end{aligned}$$

$= \overline{L_d(\mathrm{A}_D)} \cap L_d(G)$ (como A_D é fortemente não-bloqueante e.r.a D)
$= L_d(\mathrm{A}_D)$ (como A_D é D–fechada e.r.a G).

e $\forall d \in E$, $L_d(S/G) = \{s \in L(S/G): d \in \Im(S(s))\}$ (por definição)
$= \{s \in L(S/G): d \in \{e \in E: s \in L_e(\mathrm{A}_D)\}\}$
$= \{s \in L(S/G): s \in L_d(\mathrm{A}_D)\}$
$= L(S/G) \cap L_d(\mathrm{A}_D)$
$= \overline{L_D(\mathrm{A}_D)} \cap L_d(\mathrm{A}_D)$
$= \overline{L_d(\mathrm{A}_D)} \cap L_d(\mathrm{A}_D)$ (como A_D é fortemente não-bloqueante e.r.a D)
$= L_d(\mathrm{A}_D)$.

Assim, $\forall d \in D$, $L_d(S/G) = L_d(\mathrm{A}_D)$. Portanto, $\Lambda_D(S/G) = \{(L_d(S/G), d), d \in D\} = \{(L_d(\mathrm{A}_D), d), d \in D\} = \mathrm{A}_D$ e, $\forall d \in D$, $\overline{L_d(S/G)} = \overline{L_d(\mathrm{A}_D)} = \overline{L_D(\mathrm{A}_D)} = L(S/G)$. Consequentemente, S é fortemente não-bloqueante e.r.a D.

(somente se) Seja S um supervisor admissível para G com $\Lambda_D(S/G) = \mathrm{A}_D$ e $L(S/G) = \overline{L_D(\mathrm{A}_D)}$. Assumindo que S é fortemente não-bloqueante e.r.a D, tem-se que, $\forall d \in D$, $L(S/G) = \overline{L_d(S/G)} = \overline{L_d(\mathrm{A}_D)}$. Portanto, $\forall d \in D \cap C$, $L_d(\mathrm{A}_D) = L_d(S/G) = L(S/G) \cap L_d(G) = \overline{L_d(\mathrm{A}_D)} \cap L_d(G)$, isto é, A_D é D–fechado e.r.a G. Ainda, $\forall d \in D$, $\overline{L_d(\mathrm{A}_D)} = L(S/G) = \overline{L_D(\mathrm{A}_D)}$, assim A_D é fortemente não-bloqueante e.r.a D. Finalmente, mostra-se que A_D é controlável e.r.a G. Sejam $s \in \overline{L_D(\mathrm{A}_D)}$ e $\sigma \in \Sigma_u$ tais que $s\sigma \in L(G)$. Então, s $\in L(S/G)$ e $\sigma \in \Re(S(s))$. Assim $s\sigma \in L(S/G) = \overline{L_D(\mathrm{A}_D)}$, isto é, $\overline{L_D(\mathrm{A}_D)}\Sigma_u \cap L(G) \subseteq \overline{L_D(\mathrm{A}_D)}$. ♦

Corolário 1: Seja uma especificação A_D, colorida por $D \subseteq C$, tal que, $\forall d \in D$, $\varnothing \subset L_d(\mathrm{A}_D) \subseteq L_d(G)$. As condições necessárias e suficientes para a existência de um supervisor incolor fortemente não-bloqueante e.r.a D tal que $\Lambda_D(S/G) = \mathrm{A}_D$ e $L(S/G) = \overline{L_D(\mathrm{A}_D)}$ são:

1. controlabilidade de A_D e.r.a G;

2. não-bloqueio forte de A_D e.r.a D;

3. D–fechamento de A_D e.r.a G.

Nos resultados anteriores, a condição de D–fechamento não é satisfeita somente quando $\overline{L_d(\mathrm{A}_D)} \cap L_d(G) \not\subset L_d(\mathrm{A}_D)$, para alguma cor d, ou seja, quando a especificação implica que um tarefa completada por algumas seqüências na planta não deva ser completada pelas mesmas seqüências no comportamento especificado. Em certos problemas, esse fato pode significar que a especificação A_D esteja refinando a definição da classe de tarefas associada a d, isto é, que a tarefa d é realizada apenas nas seqüências de $L_d(\mathrm{A}_D)$. Para esses casos, seria razoável assumir que o supervisor fosse capaz de

"desmarcar" algumas tarefas de G e, portanto, a condição de D–fechamento poderia ser desconsiderada para as respectivas cores. No entanto, isso é equivalente a assumir que a tarefa d seja completamente definida pela especificação A_D (e não mais por G) e que o supervisor seja pintor para a respectiva cor, desconsiderando as marcações de d pela planta.

A seguinte proposição mostra a existência de um comportamento controlável contido em A_D minimamente restritivo.

Proposição 10: Seja $C(A_D, G) := \{M_D \subseteq A_D: (\overline{L_D(M_D)}\Sigma_u \cap L(G) \subseteq \overline{L_D(M_D)})\}$ o conjunto de comportamentos controláveis contidos em A_D. O conjunto $C(A_D, G)$ tem um elemento supremo $SupC(A_D, G)$.

Prova:

(i) Define-se um comportamento $Z_D = \{(\varnothing, d), \forall d \in D\}$, tal que $L_D(Z_D) = \cup_{d\in D} L_d(Z_D) = \varnothing$. Então, Z_D é controlável e contido em A_D. Assim, $C(A_D, G)$ é não-vazio.

(ii) Seja X um conjunto de índices e sejam $M_{Dx} \in C(A_D, G)$, $x \in X$, comportamentos coloridos controláveis. Seja $M_D = \cup\{M_{Dx}, x \in X\}$. Então, $M_D \subseteq A_D$ e, pela Proposição 2, $L_D(M_D) = L_D(\cup\{M_{Dx}, x \in X\}) = \cup\{L_D(M_{Dx}), x \in X\}$.

Daí,
$$\begin{aligned}
\overline{L_D(M_D)}\Sigma_u \cap L(G) &= [\cup\{\overline{L_D(M_{Dx})}, x \in X\}]\Sigma_u \cap L(G) \\
&= [\cup\{\overline{L_D(M_{Dx})}\Sigma_u, x \in X\}] \cap L(G) \\
&= \cup\{\overline{L_D(M_{Dx})}\Sigma_u \cap L(G), x \in X\} \\
&\subseteq \cup\{\overline{L_D(M_{Dx})}, x \in X\} \\
&= \overline{L_D(M_D)}.
\end{aligned}$$

Portanto, $M_D \in C(A_D, G)$, isto é, a controlabilidade de comportamento coloridos é fechada para uniões arbitrárias.

(iii) Como $C(A_D, G)$ é não-vazio e fechado para uniões arbitrárias, ele tem um elemento supremo único definido por $SupC(A_D, G) := \cup\{M_D, M_D \in C(A_D, G)\}$. ♦

Na seqüência, apresenta-se um algoritmo computável para $SupC(A_D, G)$ nos casos em que A_D for regular e G for finito. Seja $G = (Q, \Sigma, C, \delta, \chi, q_0)$ com função de eventos ativos Γ. Seja $H_a = (X, \Sigma, D, \lambda_{um}, \kappa_{um}, x_0)$ um gerador com marcação colorida fracamente aparado que modele a especificação $A_D \subseteq \Lambda_C(G) \cup \{(L(G), e), e \in E\}$. Seja $G' := (Q, \Sigma, \varnothing, \delta, \chi', q_0)$, onde $\chi'(q) = \varnothing$, $\forall q \in Q$, um gerador sem cor que gere $L(G)$. Define-se $H = H_a \,||\, G'$ com $H = (X \times Q, \Sigma, D, \lambda, \kappa, (x_0, q_0))$ e função de eventos ativos Φ.

Diz-se que um estado (x, q) de H é controlável se $\Gamma(q) \cap \Sigma_u \subseteq \Phi((x, q))$, isto é, se não houver evento não-controlável que não esteja habilitado em (x, q) mas possa ocorrer no respectivo estado q de G. O algoritmo para calcular o supremo comportamento colorido

controlável contido em A_D consiste na eliminação iterativa de estados de H_0 que não sejam controláveis e fracamente aparados e.r.a D até convergir para o maior ponto-fixo.

Inicialmente, define-se $C(H)$ como uma operação sobre H que elimina todos os estados que não são controláveis. Para isso, considera-se

$$(X \times Q)_{con} := \{(x, q): (x, q) \in (X \times Q) \text{ e } \Gamma(q) \cap \Sigma_u \subseteq \Phi((x, q))\}$$

e define-se a função controlável como

$$C(H) := \begin{cases} WTr(((X\times Q)_{con},\Sigma,D,\delta|(\Sigma\times(X\times Q)_{con}),\chi|(\Sigma\times(X\times Q)_{con}),(x_0,q_0)),D) \text{ se } (x_0,q_0)\in(X\times Q)_{con} \\ \varnothing_{\Sigma,D} \text{ senão.} \end{cases}$$

A seguir, define-se a operação $C_\infty(H)$ como o limite da seqüência

$$H_0 = H,$$
$$H_{j+1} = C(H_j), \quad j = 0, 1, \ldots .$$

Claramente essa seqüência preserva regularidade e converge num número finito de passos para um H com um número finito de estados. Tem-se então

$$C_\infty(H) := \lim H_j \quad (j \to \infty).$$

Proposição 11: Para a seqüência acima, tem-se $\Lambda_D(C_\infty(H)) = SupC(A_D, G)$.

Prova: Segue diretamente por analogia aos resultados de WONHAM e RAMADGE (1987). ♦

É razoável esperar que as propriedades de não-bloqueio forte e.r.a D e de D–fechamento sejam garantidas pela própria construção da especificação A_D. No entanto, a controlabilidade de A_D nem sempre é verificada. Neste caso, seria desejável que o supervisor S, restringindo A_D a $SupC(A_D, G)$, fosse fortemente não-bloqueante. Contudo, nem sempre isso é verdade, pois $SupC(A_D, G)$ não é necessariamente fortemente não-bloqueante e.r.a D, mesmo se A_D o for.

Exemplo 4: Seja um sistema modelado pelo GMC da Figura 25, cujo conjunto de cores é $D = \{\mathbf{a}, \mathbf{b}\}$. Seja $A_D = \{(\{\varepsilon + \alpha + \alpha\alpha\}, \mathbf{a}), (\{\varepsilon + \alpha\alpha\}, \mathbf{b})\}$ uma especificação fortemente não-bloqueante e D–fechada. O supremo comportamento controlável contido em A_D, dado por $SupC(A_D, G) = \{(\{\varepsilon + \alpha\}, \mathbf{a}), (\{\varepsilon\}, \mathbf{b})\}$, é bloqueante e.r.a $\{\mathbf{b}\}$.

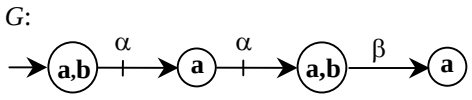


Figura 25: Exemplo de planta

O próximo teorema prova a existência de um máximo comportamento controlável e fortemente não-bloqueante contido em A_D. Pelo Corolário 1, esse comportamento, se não-vazio, pode ser gerado por um supervisor fortemente não-bloqueante minimamente restritivo.

Teorema 4: Seja $CSNB(A_D, G, B) := C(A_D, G) \cap SNB(A_D, B)\}$ o conjunto de comportamentos coloridos controláveis e fortemente não-bloqueantes contidos em A_D. O conjunto $CSNB(A_D, G, B)$ tem um elemento máximo $SupCSNB(A_D, G, B)$.

Prova: Pela Proposição 10 e Teorema 2, pode-se dizer que $CSNB(A_D, G, B)$ é não-vazio e fechado para uniões arbitrárias. Portanto, contém um elemento supremo único definido por $SupCSNB(A_D, G, B) := \cup\{M_D, M_D \in CSNB(A_D, G, B)\}$. ♦

Quando o conjunto B de tarefas relevantes é subentendido, pode-se representar $SupCSNB(\Lambda_D, G, B)$ como $SupCSNB(\Lambda_D, G)$. O resultado a seguir mostra que um supervisor para o máximo comportamento controlável e fortemente não-bloqueante pode ser mais restritivo do que um supervisor para a máxima linguagem controlável.

Proposição 12: Em geral, $L_D(SupCSNB(A_D, G, D)) \subseteq SupC(L_D(A_D), G)$.

Prova: Pode-se mostrar que $L_D(SupCSNB(A_D, G, D))$ é controlável e contido em $L_D(A_D)$. Portanto, $L_D(SupCSNB(A_D, G, D)) \subseteq SupC(L_D(A_D), G)$. O Exemplo 4, apresentado acima, é um caso em que $L_D(SupCSNB(A_D, G, D)) = \{(\{\varepsilon\}, \mathbf{a}), (\{\varepsilon\}, \mathbf{b})\}$

$$\neq \{(\{\varepsilon + \alpha\}, \mathbf{a}), (\{\varepsilon\}, \mathbf{b})\}$$
$$= SupC(L_D(A_D), G).$$ ♦

A próxima proposição – uma extensão da Proposição 3.5.4 de WONHAM (2003) – indica que, garantido o D–fechamento de uma especificação, essa propriedade estará garantida ao máximo comportamento controlável e fortemente não-bloqueante. Assim, quando A_D for D–fechado, $SupCSNB(A_D, G, D)$ será o comportamento controlável, fortemente não-bloqueante e D–fechado menos restritivo contido em A_D.

Proposição 13 : Se A_D for D–fechado e.r.a G, $SupCSNB(A_D, G, D)$ também o será.

Prova:

Seja $M_D = SupCSNB(A_D, G, D)$.

Supõe-se que M_D não seja D–fechado. Então, $\exists e \in (D \cap C)$ e $\exists\, s \in \overline{L_e(M_D)} \cap L_e(G)$ tal que $s \notin L_e(M_D)$. Seja $K_e = L_e(M_D) \cup \{s\}$. Então, $\overline{K_e} = \overline{L_e(M_D)}$ e $K_e \supset L_e(M_D)$.

Seja agora $N_D = \{(L_d(M_D), d), d \in D - \{d\}\} \cup \{(K_e, e)\}$.

Então, $N_D \supset M_D$ e $\overline{L_D(N_D)} = \overline{L_D(M_D)}$.

Portanto, $\overline{L_D(N_D)}\Sigma_u \cap L(G) = \overline{L_D(M_D)}\Sigma_u \cap L(G) \subseteq \overline{L_D(M_D)} = \overline{L_D(N_D)}$ e

$\forall d \in D, \overline{L_d(N_D)} = \overline{L_d(M_D)} = \overline{L_D(M_D)} = \overline{L_D(M_D)}$, isto é, N_D é controlável e fortemente não-bloqueante e.r.a D.
Também, $M_D \subseteq A_D$ implica que $\forall d \in D, \overline{L_d(M_D)} \subseteq \overline{L_d(A_D)}$ e, portanto,
$\forall d \in (D \cap C), \overline{L_d(M_D)} \cap L_d(G) \subseteq \overline{L_d(A_D)} \cap L_d(G) = L_d(A_D)$.
Assim, $\forall d \in (D \cap C)$, $s \in L_d(A_D)$ e daí $N_D \subseteq A_D$. Com isso, $N_D \in CSNB(A_D, G, D)$ e $N_D \supset M_D$, o que contradiz o fato que M_D é supremo. Consequentemente, $SupCSNB(A_D, G, D)$ é D–fechado e.r.a G. ♦

A próxima proposição sugere um algoritmo para a síntese de supervisor ótimo que restrinja uma planta G a um comportamento colorido especificado A_D de forma maximamente permissiva.

Proposição 14: $SupCSNB(A_D, G, D)$ é o maior ponto-fixo do operador Ω sobre comportamentos $M_D \subseteq A_D$ definido por $\Omega(M_D) = SupSNB(SupC(M_D, G), D)$.

Prova: Pelas definições de *SupC* e *SupSNB*, sabe-se que Ω é monotônico, isto é, $M_D \subseteq N_C \Rightarrow \Omega(M_D) \subseteq \Omega(N_C)$. Assim, de acordo com o Teorema de Knaster-Tarski (TARSKI, 1955), o maior ponto-fixo de Ω sobre subcomportamentos de A_D existe.
Seja Ω_D o maior ponto-fixo de Ω. Pela definição de Ω, $\Omega_D \subseteq A_D$ deve ser controlável e fortemente não-bloqueante e.r.a D. Portanto, $\Omega_D \subseteq SupCSNB(A_D, G, D)$.
Claramente, $SupCSNB(A_D, G, D) = \Omega(SupCSNB(A_D, G, D))$.
Por conseguinte, $SupCSNB(A_D, G, D) \subseteq \Omega_D$. ♦

A proposição anterior leva a crer que seja possível obter $SupCSNB(A_D, G, D)$ através de operações consecutivas de Ω sobre A_D até chegar a um ponto-fixo. Considera-se a seguinte seqüência:

$$M_D{}^0 = A_D,$$
$$M_D{}^{j+1} = \Omega(M_D{}^j), \quad j = 0, 1, \ldots .$$

Proposição 15: O limite da seqüência acima, definido por $M_D{}^\infty := \lim M_D{}^j \ (j \to \infty)$, existe e $SupCSNB(A_D, G, D) \subseteq M_D{}^\infty$.

Prova: Tem-se $M_D{}^1 = \Omega(M_D{}^0) \subseteq A_D = M_D{}^0$. Então, pela monotonicidade de Ω, $M_D{}^0 \supseteq M_D{}^1 \supseteq M_D{}^2 \supseteq \ldots$ Daí $M_D{}^\infty = \lim M_D{}^j = \cap_{j=0,\infty} M_D{}^j$ existe. Tem-se também que $SupCSNB(A_D, G, D) = \Omega(SupCSNB(A_D, G, D)) \subseteq A_D = M_D{}^0$ e, pela monotonicidade de Ω, $SupCSNB(A_D, G, D) \subseteq M_D{}^j$ implica $SupCSNB(A_D, G, D) = \Omega(SupCSNB(A_D, G, D)) \subseteq \Omega(M_D{}^j) = M_D{}^{j+1}$. Portanto, $SupCSNB(A_D, G, D) \subseteq M_D{}^\infty$. ♦

A seguir, define-se um algoritmo computável para *SupCSNB*(A_D, *G*, *D*) nos casos em que A_D for regular e *G* for finito. Novamente, seja $G = (Q, \Sigma, C, \delta, \chi, q_0)$ com função de eventos ativos Γ. Seja $H_a = (X, \Sigma, D, \lambda_{um}, \kappa_{um}, x_0)$ um gerador com marcação colorida fracamente aparado que modele uma especificação $A_D \subseteq \Lambda_C(G) \cup \{(L(G), e), e \in E\}$. Seja *G*' um gerador sem marcação que gere $L(G)$ e define-se $H = H_a \parallel G'$ com $H = (X \times Q, \Sigma, D, \lambda, \kappa, (x_0, q_0))$ e função de eventos ativos Φ. O algoritmo proposto para calcular o supremo comportamento colorido controlável e fortemente não-bloqueante contido em A_D consiste em uma eliminação iterativa de estados de *H* que não sejam controláveis e fortemente aparado e.r.a *D* até se convergir para o maior ponto-fixo. Para isso, considera-se a seguinte seqüência:

$$H_0 = H,$$
$$H_{2j-1} = C_\infty(H_{2j-2}),$$
$$H_{2j} = STr(H_{2j-1}), \quad j = 1, 2, \ldots .$$

A seqüência decrescente acima evidentemente preserva regularidade e converge num número finito de passos para um *H* com um número finito de estados. Então, pode-se definir

$$STrC_\infty(H) := \lim H_{2j} \quad (j \to \infty).$$

Proposição 16: Para a seqüência acima, $\Lambda_D(STrC_\infty(H)) = SupCSNB(A_D, G, D)$.

Prova: Pela Proposição 9 e pela Proposição 11, tem-se
$\Lambda_D(H_{2j}) = SupSNB(SupC(\Lambda_D(H_{2j-2}), G), D) = \Omega(\Lambda_D(H_{2j-2}))$.
Como *H* tem um número finito de estados de estados, sabe-se que existe um inteiro *n* tal que, para $j > n$, $H_{2j-1} = H_{2j} = STrC_\infty(H)$. Portanto, $\Lambda_D(STrC_\infty(H)) \subseteq A_D$ é um ponto-fixo do operador Ω, o que implica, pela Proposição 14, que $\Lambda_D(STrC_\infty(H)) \subseteq SupCSNB(A_D, G, D)$. Então, usando a Proposição 15, pode-se concluir que
$SupCSNB(A_D, G, D) = \Lambda_D(STrC_\infty(H))$. ♦

Apagando-se as marcações de $STrC_\infty(H)$ nas cores da planta (*C*), obtém-se um supervisor ótimo *H*', tal que $\Lambda_D(H'/G) = SupCSNB(A_D, G, D)$.

Observa-se que, quando *D* contém apenas um elemento, o algoritmo acima eqüivale à abordagem clássica para síntese de supervisor não-bloqueante. Quando *D* possui múltiplos elementos, nem sempre é possível definir uma marcação única para os geradores (comuns) que modelam a especificação e a planta, de forma a poder obter uma solução equivalente a *SupCSNB*(A_D, *G*, *D*) pelos algoritmos clássicos. Entretanto, o GMC *H* definido no algoritmo acima para representar A_D, pode ser modelado na forma de um

conjunto de geradores comuns associados a cores $\{(H_d, d), d \in D\}$, tais que $L_m(H_d) = L_d(H)$. Esses geradores podem ser operados em conjunto, de forma análoga ao algoritmo proposto para GMCs, fazendo-se:

$$H_d{}^0 = H_d$$
$$L^{i+1} = \cap_{d \in D} \overline{SupC(L_m(H_d{}^i), G)},$$
$$L_m(H_d{}^{i+1}) = L^{i+1} \cap L_m(H_d{}^i),\ i = 0, 1, ...$$

Dessa forma, pode-se iterativamente chegar a um conjunto de geradores comuns não-bloqueantes $\{(H_d{}^\infty, d), d \in D\}$, todos gerando a mesma linguagem e cada qual marcando uma linguagem controlável ótima. Por analogia com a Proposição 16, deduz-se que $L_m(H_d{}^\infty) = L_d(SupCSNB(\Lambda_D(H), G, D))$, ou seja, o conjunto de geradores obtidos representam o supremo comportamento controlável e fortemente não-bloqueante e.r.a D contido em $\Lambda_D(H) = A_D$. Ressalta-se, no entanto, que é mais conveniente proceder o processo de síntese num único GMC ao invés de utilizar múltiplos geradores comuns, todos com o mesmo número de estados do GMC equivalente. Além de facilitar a compreensão do modelo, a representação compacta do GMC pode promover economia de memória e eficiência à implementação computacional do algoritmo.

4.4 Exemplos

Para elucidar as idéias apresentadas nas seções anteriores, são apresentados três exemplos de problemas de controle em que a diferenciação das tarefas se faz necessária para a síntese de supervisores admissíveis que, controlando a planta de forma minimamente restritiva, garantam que todos os tipos de tarefas possam sempre ser cumpridos. No primeiro exemplo apresenta-se como solução um supervisor incolor e, nos dois outros, são obtidos supervisores pintores.

Problema 2: (RAMADGE e WONHAM, 1989) Um gato e um rato compartilham o labirinto apresentado na Figura 26, sendo que as setas g_i sinalizam as passagens em mão única exclusivas para o gato e as setas r_i sinalizam as passagens em mão única exclusivas para o rato. Todas as passagens, menos a g_7, podem ser fechadas pelo controlador. Na cozinha, identificada na Figura 26 pela comida, há alimento disponível para o rato e gato. O gato e o rato encontram-se inicialmente nos recintos indicados pelos respectivos ícones. Espera-se obter um controlador que garanta a maior liberdade de movimento evitando que ambos animais ocupem o mesmo recinto ao mesmo tempo de forma que o gato e o rato possam sempre comer (tarefas **g** e **r**, respectivamente) e o sistema possa voltar ao estado inicial (tarefa **i**).

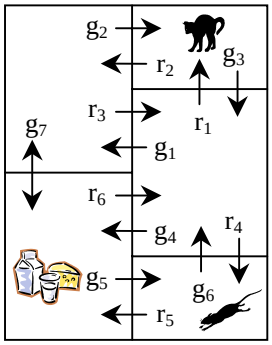


Figura 26: Labirinto para gato e rato

Sejam o alfabeto $\Sigma = \{g_i, r_j: 1 \leq i \leq 7, 1 \leq j \leq 6\}$ e o conjunto de cores $C = \{\mathbf{i}, \mathbf{g}, \mathbf{r}\}$. Modela-se o movimento do gato e do rato pelo sistema composto pelos geradores G_g e G_r sobre Σ e C, apresentados na Figura 27, de forma que a planta global G seja obtida por $G = G_g \parallel G_r$.

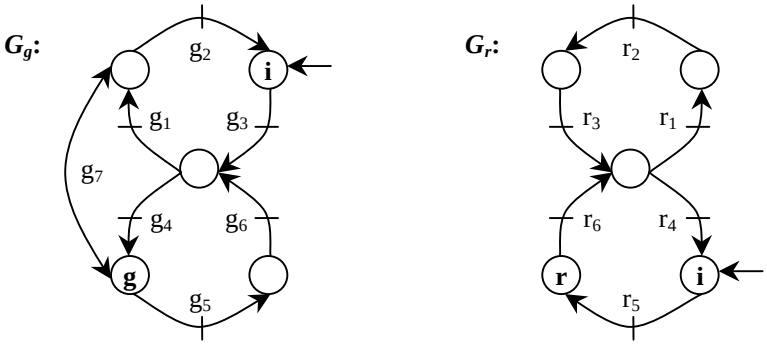


Figura 27: Modelos com marcação colorida para gato (G_g) e rato (G_r)

Um gerador para o máximo comportamento A_C que respeite as especificações é obtido de G, apagando-se os estados em que os dois animais ocupem o mesmo recinto. O máximo comportamento controlável e fortemente não-bloqueante e.r.a {**i**, **g**, **r**}contido em A_C, *SupCSNB*(A_C, G, C)**,** é então computado pelo método iterativo anteriormente descrito. Esse comportamento é modelado pelo GMC da Figura 28. Eliminando-se todas as cores desse modelo, obtém-se um supervisor incolor H que serve de solução ótima para esse problema de controle.

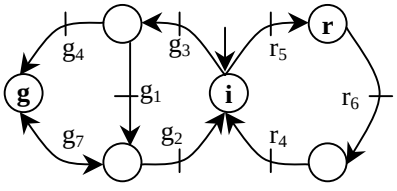


Figura 28: GMC para *SupCSNB*(A_C, G, C)

Esse comportamento coincide com a solução usual obtida por RAMADGE e WONHAM (1989), que considera como marcados apenas os estados iniciais. Portanto, os requerimentos extras de vivacidade do gato e do rato na verdade não introduzem uma restrição adicional a esse problema particular. Mesmo assim, observa-se que o uso de marcações coloridas favorece a verificação das especificações de vivacidade.

Problema 3: Uma Linha de Manufatura (WONHAM, 2003) é composta por três máquinas M_1, M_2 e M_3, dispostas em linha de acordo com a Figura 29. Dois depósitos unitários fazem a interface entre as máquinas. A máquina M_1 começa a operar com o evento a_1 e termina de operar, colocando uma peça no depósito consecutivo, pelo evento a_2. A máquina M_2 começa a operar com o evento b_1, retirando uma peça do depósito anterior, e termina de operar, colocando uma peça no depósito consecutivo, pelo evento b_2. Finalmente a máquina M_3 começa a operar com o evento c_1, retirando uma peça do depósito anterior, e, conforme a qualidade da peça, retira a peça da linha pelo evento c_2 ou a devolve para o primeiro depósito pelo evento c_3. Apenas o início de operação das máquinas pode ser controlado. O objetivo do controle supervisório neste problema é evitar, de forma minimamente restritiva, *overflow* ou *underflow* de peças nos depósitos temporários e garantir que o sistema possa sempre retornar ao estado inicial (tarefa **i**), que os depósitos possam sempre ser simultaneamente esvaziados (tarefa **e**) e que cada máquina sempre possa operar (tarefas **a**, **b** e **c**).

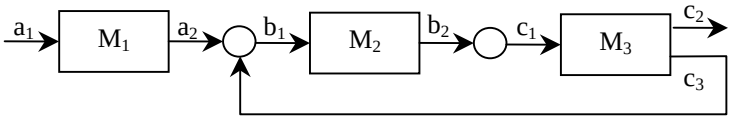


Figura 29: Linha de manufatura

A linha de manufatura sem ação de controle é modelada pelo sistema composto pelos geradores com marcação colorida G_1, G_2 e G_3, respectivos às máquinas M_1, M_2 e M_3, apresentados na Figura 30, sendo que $G = G_1 \parallel G_2 \parallel G_3$.

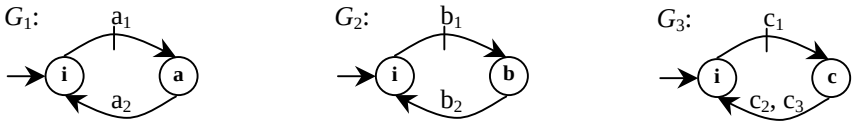


Figura 30: Modelo da planta

Observa-se que o modelo da planta não fornece informações sobre o estado dos depósitos, de forma que a marcação da cor **e**, sinalizando o esvaziamento dos depósitos, é atribuída à especificação e, por conseguinte, ao supervisor. As especificações genéricas $E_{1,\text{gen}}$ e $E_{2,\text{gen}}$ que evitam os problemas nos depósitos e definem a tarefa **e** são apresentadas

na Figura 31. Um gerador que modele o comportamento especificado A_D, com $D = \{\mathbf{i}, \mathbf{e}, \mathbf{a}, \mathbf{b}, \mathbf{c}\}$, é obtido por $E = E_{gen,1} \parallel E_{gen,2} \parallel G$.

Figura 31: Especificações genéricas

Pelo método iterativo de cálculo do máximo comportamento controlável e fortemente não-bloqueante contido em A_D obtém-se o comportamento *SupCSNB*(A_D, *G*, *D*) modelado pelo GMC da Figura 32. Um supervisor pintor ótimo *H* pode ser obtido a partir desse gerador apagando-se as cores {**i**, **a**, **b**, **c**} de todos os seus estados.

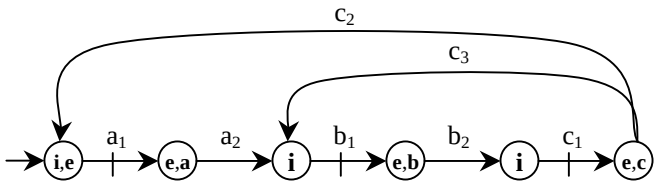


Figura 32: GMC para *SupCSNB*(A_D, *G*, *D*)

Pode-se observar que *SupCSNB*(A_D, *G*, *D*) está contido no máximo comportamento controlável e fracamente não-bloqueante e.r.a {**i**, **e**, **a**, **b**, **c**}, que é apresentado na Figura 33. Este comportamento permite a ocorrência de um bloqueio forte em que todas as máquinas acabam ficando trancadas no estado inicial com ambos os depósitos cheios.

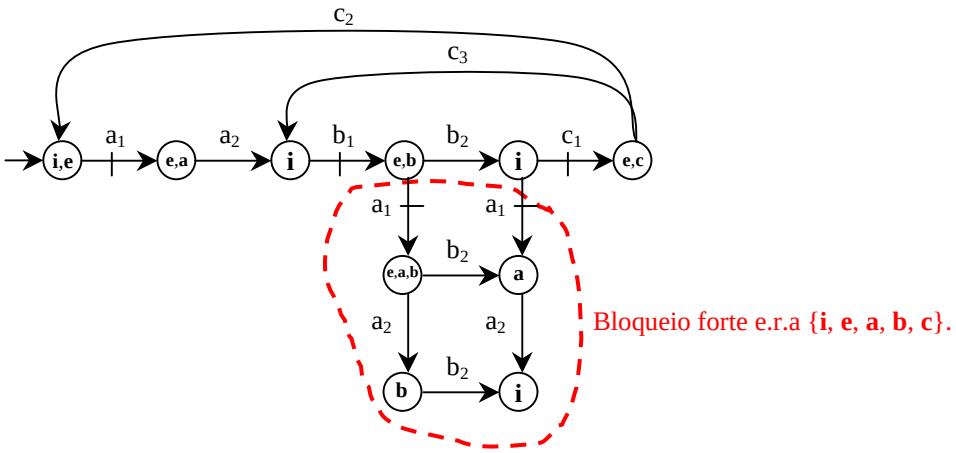


Figura 33: Máximo comportamento controlável e fracamente não-bloqueante e.r.a {**i**, **e**, **a**, **b**, **c**}

Essa solução bloqueante é obtida pela abordagem usual de RAMADGE e WONHAM (1989) se forem considerados marcados os estados iniciais das máquinas e todos os estados das especificações genéricas. Entretanto, conforme sugerido por WONHAM (2003), uma solução fortemente não-bloqueante similar pode ser obtida pela abordagem usual se forem marcados apenas os estados iniciais das máquinas e das especificações genéricas. Apesar de que a máxima solução controlável e não-bloqueante não seja $L_m(G)$-fechada, pode-se obter um supervisor desmarcador que implemente tal comportamento.

Problema 4: Considera-se uma equipamento para mistura de tintas composto de dois injetores de tinta (P_1 e P_2) e um misturador (M), arranjados de acordo com a Figura 34. Cada injetor pode fornecer doses de volume fixo com uma das três cores primárias (vermelho, amarelo e azul) ou com duas dessas cores na mesma proporção. Com o objetivo de se evitar manchar a tinta, uma vez que um injetor começa a fornecer uma cor, ele só pode continuar fornecendo a mesma cor ou passar a suprir uma mistura de duas cores primárias contendo a anterior. O misturador faz a homogeneização da tinta e prepara o equipamento para uma nova batelada.

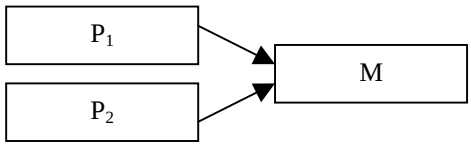


Figura 34: Disposição do misturador de tintas

Os eventos para cada injetor de tinta P_i, $i = 1, 2$, são r_i (vermelho – ou *red* em inglês), b_i (azul – *blue*), y_i (amarelo – *yellow*) , br_i (azul e vermelho), yb_i (amarelo e azul) e ry_i (vermelho e amarelo). Todos eventos são controláveis e indicam o suprimento de uma dose da respectiva tinta. O misturador M é modelado simplesmente por um estado com um auto-laço do evento controlável *mix*. Os autômatos mostrados na Figura 35 modelam o comportamento de P_1, P_2 e M. O modelo da planta G é dado por $G = P_1 \parallel P_2 \parallel M$.

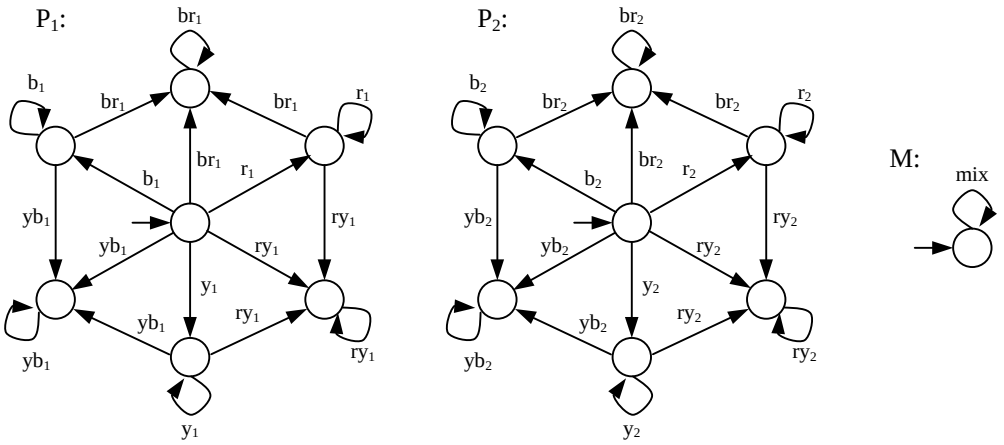


Figura 35: Modelos para P_1, P_2 e M

A especificação *T*, representada pelo autômato na Figura 36, expressa que cada batelada de tinta processada pelo equipamento deve conter duas doses de tinta. As múltiplas marcações indicam a inicial dos nomes em inglês das cores primárias (***r**ed*, ***y**ellow* e ***b**lue*) ou secundárias (***o**range*, ***g**reen* e ***p**urple* – laranja, verde e roxo, respectivamente) que estão prontas para serem misturadas.

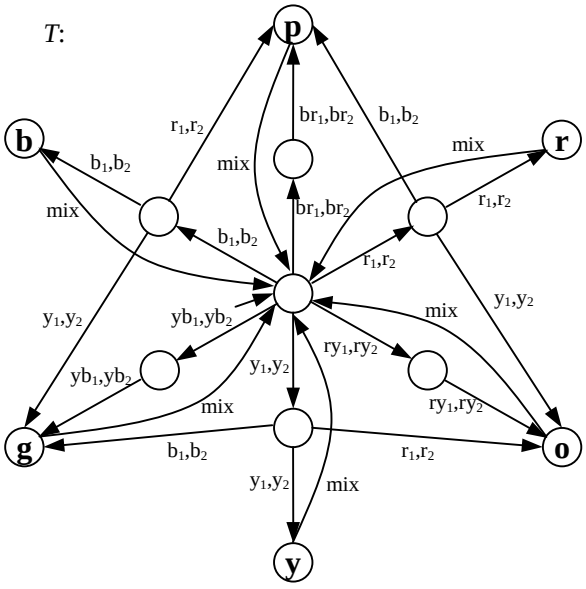


Figura 36: Triângulo de cores *T*

O comportamento colorido marcado por $H = T \parallel G$, representado por um autômato de Moore com 211 estados e 468 transições, é obviamente controlável, já que todos eventos são controláveis. Ele também é fracamente não-bloqueante e.r.a $D := \{\mathbf{r}, \mathbf{y}, \mathbf{b}, \mathbf{o}, \mathbf{g}, \mathbf{p}\}$. Contudo, tal comportamento é bloqueante em relação a qualquer de suas cores. Isso significa que o sistema sempre permite produzir pelo menos uma cor, mas

a evolução dos eventos pode fazer com que algumas cores não possam mais ser produzidas. O supremo comportamento controlável e fortemente não-bloqueante e.r.a *D* é o conjunto vazio, como se poderia naturalmente esperar pela formulação do problema. No entanto, se tomar-se como especificação de vivacidade o não-bloqueio forte em relação às cores **r**, **y** e **o** (garantindo que o sistema possa sempre produzir tinta vermelha, laranja e amarela), o supremo comportamento controlável e fortemente não-bloqueante contido em $\Lambda_D(H)$, *SupCSNB*($\Lambda_D(H)$, *G*, {**r**, **y**, **o**}), é marcado pelo gerador com marcação colorida com 13 estados, ilustrado na Figura 37. A solução obtida permite duas configurações do sistema: o injetor P_1 fornece apenas tinta vermelha e P_2 tinta amarela ou vice-versa.

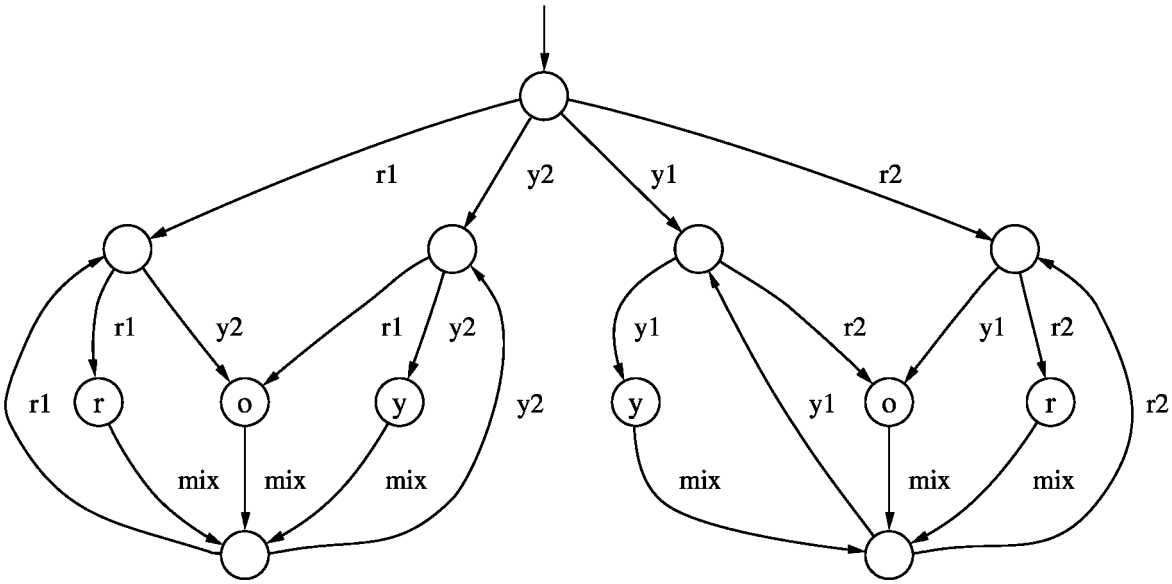


Figura 37: GMC para *SupCSNB*($\Lambda_D(H)$, *G*, {**r**, **y**, **o**})

Essa solução não pode ser obtida através da abordagem usual, para qualquer que seja o conjunto de estados escolhidos como marcados na especificação ou na planta. Uma solução similar pode ser obtida se forem usados múltiplos geradores comuns não-bloqueantes, um para cada tarefa de {**r**, **y**, **o**}. Conforme discussão anterior (na página 90), esses geradores podem ser operados iterativamente em conjunto, de forma a chegar a um conjunto de geradores que representam o supremo comportamento controlável e fortemente não-bloqueante e.r.a {**r**, **y**, **o**} contido em $\Lambda_D(H)$. Observa-se, no entanto, que é mais conveniente proceder o processo de síntese num único GMC de 211 estados ao invés de utilizar três geradores comuns, todos com os mesmos 211 estados e apenas marcações distintas.

4.5 Reversibilidade

Um sistema a eventos discretos é chamado é reversível quando sempre for possível retornar ao seu estado inicial de qualquer estado alcançável (MURATA, 1989). Essa propriedade é um requisito natural em muitas aplicações práticas, particularmente em sistemas de manufatura. Para um gerador usual, a reversibilidade pode ser testada marcando-se apenas o estado inicial e verificando-se não-bloqueio. Do mesmo modo, num gerador com marcação colorida, pode-se definir como uma tarefa o fato de atingir o estado inicial. Ao se impor não-bloqueio forte e.r.a qualquer conjunto contendo essa tarefa, garante-se a reversibilidade do SED. Nesta seção, estuda-se a relação entre reversibilidade e não-bloqueio forte em relação a múltiplas tarefas.

Seja um gerador com marcação colorida $G = (Q, \Sigma, C, \delta, \chi, q_0)$. Formalmente, G é dito reversível se

$$\forall q \in Q_{ac}, \exists s \in \Sigma^* \text{ tal que } \delta(q, s) = q_0.$$

Uma tarefa $c \in C$ é acessível se para algum estado acessível $q \in Q_{ac}$, $c \in \chi(q)$. O conjunto de tarefas acessíveis $C_{ac} \subseteq C$ é dado por $C_{ac} := \{c \in C: c \text{ é acessível}\}$. Para um GMC não-vazio, define-se a tarefa $\mathbf{i} \in C$ como "alcançar o estado inicial", isto é,

$$\mathbf{i} \in \chi(q) \Leftrightarrow q = q_0.$$

Assim, G é reversível se G for não-bloqueante e.r.a **i**.

Proposição 17: G é reversível se e somente se G for fortemente não-bloqueante e.r.a C_{ac}.

Prova:
(se) Claramente, **i** é acessível e $\mathbf{i} \in C_{ac}$. Portanto, se G for fortemente não-bloqueante e.r.a C_{ac}, G é não-bloqueante e.r.a **i** e, assim, reversível.
(somente se) Supõe-se que G seja reversível e fortemente bloqueante e.r.a C_{ac}. Então, existe $c \in C_{ac}$ e $q \in Q_{ac}$ tal que $\forall s \in \Sigma^*$, $c \notin \chi(\delta(q, s))$. Como $c \in C_{ac}$, sabe-se que $\exists v \in \Sigma^*$, $c \in \chi(\delta(q_0, v))$. Por conseqüência, $\forall u \in \Sigma^*$, $\delta(q, u) \neq q_0$ – do contrário, c estaria em $\chi(\delta(q, uv))$. Portanto, q_0 (e **i**) não pode ser alcançado a partir de q e daí G não é reversível, o que contradiz a suposição inicial. ♦

A proposição anterior indica que se pode testar não-bloqueio forte em relação a um conjunto de tarefas relevantes (incluindo **i**) através da verificação da reversibilidade do GMC e acessibilidade das respectivas tarefas. Em problemas em que reversibilidade é um requisito importante, essa abordagem pode ser mais eficiente do que testar coacessibilidade em relação a todas as tarefas para cada estado acessível do GMC. De acordo com a

próxima proposição, essa idéia pode ser explorada para se calcular economicamente o supremo subcomportamento de um comportamento colorido Λ_D, gerado por um GMC H_a, que seja controlável em relação a G e fortemente não-bloqueante em relação a qualquer conjunto de tarefas que inclua **i**, sendo que a tarefa **i** significa aqui alcançar o estado inicial de H_a. Observa-se que, em certos problemas, a especificação pode adicionar memória (novos estados) ao modelo e, portanto, o fato de atingir o estado inicial de G não implica necessariamente chegar ao estado inicial de H_a.

Proposição 18: Seja $H_a = (X, \Sigma, D, \lambda, \kappa, x_0)$ um gerador com marcação colorida fracamente aparado que modele uma especificação Λ_D. Seja $\mathbf{i} \in D \subseteq C_{ac}$ tal que, para qualquer $x \in X$, $\mathbf{i} \in \kappa(x) \Leftrightarrow x = x_0$. Então, $SupCSNB(\Lambda_D, G, D) = SupCSNB(\Lambda_D, G, \{\mathbf{i}\})$ sempre que $SupCSNB(\Lambda_D, G, D) \neq \{(\varnothing, d), \forall d \in D\}$.

Prova:
(⊆) Claramente, $SupCSNB(\Lambda_D, G, D)$ é (fortemente) não-bloqueante e.r.a $\{\mathbf{i}\}$ e, portanto, $SupCSNB(\Lambda_D, G, D) \subseteq SupCSNB(\Lambda_D, G, \{\mathbf{i}\})$.
(⊇) De acordo com o algoritmo indicado na Proposição 16, existe um CMG $H_{D\infty} \subseteq H_a$ que modela $SupCSNB(\Lambda_D, G, D)$ e um CMG $H_{\{\mathbf{i}\}\infty} \subseteq H_a$ que modela $SupCSNB(\Lambda_D, G, \{\mathbf{i}\})$. Portanto, se $SupCSNB(\Lambda_D, G, D) \neq \{(\varnothing, d), \forall d \in D\}$, para qualquer $d \in D$, existe $s \in L(SupCSNB(\Lambda_D, G, D)) \subseteq L(SupCSNB(\Lambda_D, G, \{\mathbf{i}\})) = L(H_{\{\mathbf{i}\}\infty})$ tal que $d \in \kappa(\lambda(x_0, s))$. Daí, qualquer $d \in D$ é acessível em $H_{\{\mathbf{i}\}\infty}$. Como **i** é marcada apenas em x_0 e $H_{\{\mathbf{i}\}\infty} \subseteq H_a$, $H_{\{\mathbf{i}\}\infty}$ é reversível. Assim, pela Proposição 17 e Proposição 8, pode-se dizer que $SupCSNB(\Lambda_D, G, \{\mathbf{i}\})$ é fortemente não-bloqueante e.r.a D.
Então, $SupCSNB(\Lambda_D, G, D) \supseteq SupCSNB(\Lambda_D, G, \{\mathbf{i}\})$. ♦

Pela proposição anterior, um modo natural de se calcular *SupCSNB* é obter a máxima sublinguagem controlável e reversível e testar acessibilidade de todas tarefas relevantes. Esse resultado poderia ter sido usado diretamente na resolução do Problema 2, já que **i** é uma das tarefas relevantes e todas as outras permanecessem acessíveis na solução. Portanto, o mesmo resultado poderia ser obtido calculando-se um supervisor controlável ótimo que garantisse que o gato e o rato pudessem sempre estar de volta em seus respectivos quartos iniciais num mesmo estado global (tarefa **i**). No comportamento assim obtido, pode-se verificar que o gato e o rato podem alcançar a comida a partir do estado inicial. Se as tarefas **g** e **r** não fossem acessíveis do estado inicial do comportamento reversível e controlável ótimo, o problema não admitiria qualquer solução (não-vazia) fortemente não-bloqueante e.r.a $\{\mathbf{i}, \mathbf{g}, \mathbf{r}\}$.

Na resolução do Problema 3, definiu-se como tarefa **i** apenas os estados iniciais das três máquinas da linha de transferência, sem levar em conta o estado dos depósitos. Nesse

caso o comportamento controlável não-bloqueante e.r.a {**i**} ótimo é equivalente ao supremo comportamento controlável e fracamente não-bloqueante contido na planta (Figura 33), o qual permite que o sistema fique travado com três máquinas no estado inicial e os depósitos cheios. Assim, a Proposição 18 não se aplica diretamente a esse problema, uma vez que as especificação sobre os depósitos adicionam nova memória ao sistema e, consequentemente, a reversibilidade do sistema restrito pelas especificações não é garantida pela coacessibilidade da tarefa **i**. Contudo, no intuito de se usar essa abordagem para calcular o comportamento ótimo, pode-se definir a reversibilidade através de uma tarefa **i'** dada pela conjunção das tarefas **i** e **e**, isto é, todas as máquinas e depósitos simultaneamente no estado inicial. Observa-se que a especificação de vivacidade para **i'** é potencialmente mais restritiva do que a vivacidade de **i** e **e** separadamente. No caso particular da linha de manufatura, não é mais restritiva. Dessa maneira, o comportamento reversível e controlável ótimo coincide com a solução fortemente não-bloqueante apresentada na Figura 32.

Finalmente, o misturador de tintas do Problema 4 é um caso em que reversibilidade seria um requerimento forte demais. Nesse problema, a abordagem sugerida pela Proposição 18 levaria à “solução” vazia, que é obviamente mais restritiva do que o máximo comportamento controlável e fortemente não-bloqueante e.r.a {**r**, **y**, **o**}.

4.6 Conclusão do capítulo

Neste capítulo, os geradores com marcação colorida foram introduzidos como uma alternativa viável para a diferenciação de classes de tarefas nos modelos de sistemas a eventos discretos. Baseado nesse modelo, foi possível definir o não-bloqueio forte como um refinamento da propriedade de não-bloqueio, que indica quando todo um conjunto de tarefas pode sempre ser executado. Com isso, foram desenvolvidos resultados e algoritmos que fundamentam uma metodologia para síntese de supervisores controláveis e fortemente não-bloqueantes ótimos. Essa metodologia permite tratar de forma diferenciada as tarefas que são definidas na planta e as tarefas que são introduzidas pela especificação de controle e, por conseguinte, pela ação do supervisor. A solução de três problemas em que foi possível lidar de forma clara e eficiente com as diferentes classes de tarefas da planta e das especificações realçou as vantagens dessa metodologia tanto para a modelagem quanto para a síntese de supervisores mais refinados, ou seja, menos restritivos do que um supervisor que assegure que todas as tarefas possam estar simultaneamente completas e mais restritivos do que um supervisor que garanta a vivacidade de pelo menos uma tarefa. Na seqüência, a propriedade reversibilidade foi introduzida como uma forma alternativa de

se assegurar vivacidade de múltiplas tarefas para alguns casos de SEDMT. Tal abordagem pôde ser aplicada com sucesso nos dois primeiros problemas apresentados, mas não no terceiro. Conclui-se que o uso de geradores com marcação colorida surge como uma forma natural e viável para o tratamento de múltiplas tarefas na obtenção de soluções potencialmente mais refinadas, o que pode justificar um conseqüente aumento da complexidade dos algoritmos e do modelo para SEDs.

Por fim, para o tratamento eficiente de sistemas compostos, é fundamental que se faça a extensão dos resultados de controle supervisório com diferenciação de classes de tarefas para outras abordagens de controle, como o controle hierárquico (ZHONG e WONHAM, 1990; CUNHA e CURY; 2002; TORRICO e CURY; 2002) e controle modular (WONHAM e RAMADGE, 1988; QUEIROZ e CURY, 2000). A extensão para controle modular é o tema do próximo capítulo. A implementação de rotinas computacionais para geradores com marcação colorida também é um trabalho importante para a aplicação dos resultados de controle supervisório multitarefa a problemas reais.

5. Controle Modular de SEDMTs

Os resultados do Capítulo 4 proporcionam os elementos de uma metodologia para o cálculo de supervisores ótimos para SEDMT com especificações expressas na forma de um comportamento colorido admissível composto de sublinguagens da planta. Em geral, uma especificação de controle pode ser apresentada como uma conjunção de múltiplos comportamentos admissíveis, cada qual abrangendo um requisito particular do problema. Neste caso, uma abordagem possível é fazer a composição de todas especificações num único comportamento colorido admissível e sintetizar um supervisor monolítico. Porém, como a composição de autômatos pode levar a um crescimento exponencial no tamanho do modelo global, a síntese monolítica pode ser computacionalmente problemática para casos de maior porte. Uma abordagem mais eficiente consiste em calcular um supervisor modular ótimo para cada especificação e então combinar a ação de todos supervisores sobre a planta. Como visto no Capítulo 3, além da eficiência no cálculo de supervisores, a arquitetura de controle modular pode prover flexibilidade, segurança e legibilidade para a implementação. Em contrapartida, como cada supervisor modular é concebido baseado numa visão parcial do problema global, é possível que suas ações em conjunto levem à ocorrência de conflito, o que pode ser resolvido de acordo com a discussão na Seção 2.5.2.

Neste capítulo a abordagem de controle supervisório multitarefa é estendida para uma arquitetura modular. Apresenta-se, assim, uma metodologia eficiente para a síntese de supervisores ótimos para SEDs compostos com múltiplas tarefas e múltiplas especificações. Na primeira seção, os resultados de controle supervisório modular desenvolvidos por WONHAM e RAMADGE (1988) são estendidos para SEDMTs. Apresentam-se as condições sob as quais não-bloqueio forte e otimidade são preservados pela conjunção de supervisores. Esses resultados são apresentados de forma resumida em QUEIROZ e CURY (2004). Na sessão seguinte, os resultados de QUEIROZ e CURY (2000) também são generalizados de forma a proporcionar uma metodologia de síntese que permite tratar a complexidade de sistemas de grande porte explorando a estrutura modular tanto das especificações quanto da planta em sistemas multitarefa compostos. Ao final, a solução de um problema de controle multitarefa para um exemplo de sistema flexível de manufatura ilustra as vantagens da metodologia proposta.

5.1 Controle Modular

Seja o comportamento em malha aberta de um SEDMT modelado por um gerador com marcação colorida $G = (Q, \Sigma, C, \delta, \chi, q_0)$, com função de eventos ativos Γ. Sejam S_1: $L(G) \rightarrow Pwr(\Sigma) \times Pwr(E)$ e S_2: $L(G) \rightarrow Pwr(\Sigma) \times Pwr(E)$ supervisores pintores para G, onde E é o conjunto de novas cores acrescentadas a G. Para uma cadeia $s \in L(G)$, com $S_1(s) = (\gamma_1, \mu_1)$ e $S_2(s) = (\gamma_2, \mu_2)$, a ação de controle do supervisor $S_1 \wedge S_2$: $L(G) \rightarrow Pwr(\Sigma) \times Pwr(E)$, representando a conjunção de S_1 e S_2, é dada por

$$S_1 \wedge S_2(s) := (\gamma_1 \cap \gamma_2, \mu_1 \cap \mu_2).$$

A conjunção de dois supervisores pintores é ilustrada na Figura 38. Um evento controlável é habilitado na planta se e somente se for habilitado por ambos supervisores para a cadeia correspondente à seqüência de eventos ocorridos planta. Em outras palavras, é suficiente que um supervisor desabilite um evento para que o mesmo seja desabilitado na planta. Da mesma forma, as novas cores definidas pelos supervisores são marcadas no sistema em malha fechada se e somente se forem marcadas por ambos supervisores. Pode-se verificar que o comportamento em malha fechada sob controle modular é dado por:

$$L(S_1 \wedge S_2/G) = L(S_1/G) \cap L(S_2/G);$$

$$\forall d \in C \cup E,\ L_d(S_1 \wedge S_2/G) = L_d(S_1/G) \cap L_d(S_2/G).$$

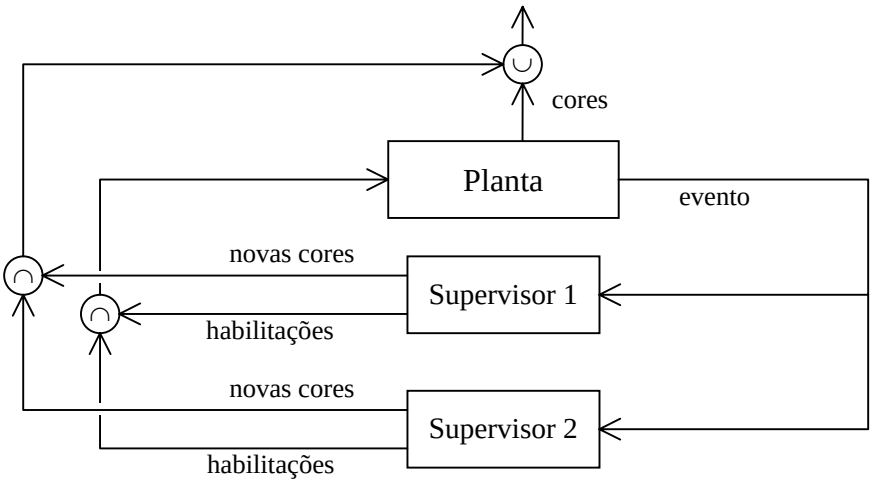


Figura 38: Arquitetura de controle modular multitarefa

Em alguns problemas, pode acontecer que o conjunto de novas cores de cada supervisor pintor modular seja diferente. Na ação conjunta, esse fato significa que cada supervisor se preocupa apenas com um subconjunto das novas cores e não impõe qualquer restrição sobre as demais novas cores. Nesse caso, pode-se considerar que ambos

supervisores sejam definidos para o mesmo conjunto de novas cores assumindo-se que as cores adicionadas para cada supervisor sejam sempre marcadas.

De acordo com a Seção 4.3.3, se S_1 e S_2 forem admissíveis, eles podem ser respectivamente representados por geradores com marcação colorida H_1 e H_2, tais que $S_1/G = H_1 \parallel G$ e $S_2/G = H_2 \parallel G$. Então, pela definição de conjunção de supervisores, pode-se verificar que:

$$\begin{aligned} S_1 \wedge S_2/G &= (S_1/G) \parallel (S_2/G) \\ &= (H_1 \parallel G) \parallel (H_2 \parallel G) \\ &= H_1 \parallel H_2 \parallel G. \end{aligned}$$

Para que se possa inferir as propriedades da arquitetura de controle modular a partir das propriedades de cada supervisor, é preciso que se investigue se características como não-bloqueio forte, controlabilidade e *D*–fechamento são preservadas pela interseção de comportamentos coloridos. Conforme apresentado a seguir, as condições para isso são não-conflito forte e fraco.

Sejam os comportamento coloridos M_D, $\mathrm{N}_D \subseteq Pwr(Pwr(\Sigma^*) \times D)$. Diz-se que M_D e N_D são fracamente não-conflitantes e.r.a $B \subseteq D$ sempre que

$$\overline{L_B(\mathrm{M}_D)} \cap \overline{L_B(\mathrm{N}_D)} = \overline{L_B(\mathrm{M}_D \cap \mathrm{N}_D)}.$$

Em palavras, M_D e N_D são fracamente não-conflitantes se cada prefixo comum puder ser estendido para uma cadeia que complete uma tarefa em M_D e complete a mesma tarefa em N_D. Além disso, diz-se que M_D e N_D são fortemente não-conflitantes quando

$$\forall b \in B, \overline{L_b(\mathrm{M}_D)} \cap \overline{L_b(\mathrm{N}_D)} = \overline{L_b(\mathrm{M}_D \cap \mathrm{N}_D)}.$$

Não-conflito forte de M_D e N_D significa que, para cada cor de *D*, as linguagens marcadas por essa cor em M_D e em N_D são não-conflitantes ou, em outras palavras, se uma cadeia puder ser estendida para completar uma tarefa em M_D e também puder ser estendida para completar a mesma tarefa em N_D, existe um modo comum de se completar essa tarefa. Naturalmente, não-conflito forte e.r.a *B* implica não-conflito fraco e.r.a *B*. As proposições seguintes indicam a relação entre não-conflito fraco/forte de comportamentos coloridos e não-bloqueio fraco/forte de geradores com marcação colorida.

Proposição 19: Sejam, respectivamente, H_M e H_N geradores com marcação colorida para M_D e $\mathrm{N}_D \subseteq Pwr(Pwr(\Sigma^*) \times D)$ fracamente não-bloqueantes e.r.a $B \subseteq D$. $H_\mathrm{M} \parallel H_\mathrm{N}$ é fracamente não-bloqueante e.r.a *B* se e somente se M_D e N_D forem fracamente não-conflitantes e.r.a *B*.

Prova: $\overline{L_B(H_M \parallel H_N)} = L(H_M \parallel H_N) \Leftrightarrow \overline{L_B(\Lambda_D(H_M \parallel H_N))} = L(H_M) \cap L(H_N)$

$\Leftrightarrow \overline{L_B(\Lambda_D(M_D) \cap \Lambda_D(N_D))} = \overline{L_B(H_M)} \cap \overline{L_B(H_N)}$

$\Leftrightarrow \overline{L_B(M_D \cap N_D)} = \overline{L_B(M_D)} \cap \overline{L_B(N_D)}$. ♦

Proposição 20: Sejam M_D, $N_D \subseteq Pwr(Pwr(\Sigma^*) \times D)$ fortemente não-bloqueantes e.r.a $B \subseteq D$. Sejam, respectivamente, H_M e H_N geradores com marcação colorida fortemente não-bloqueantes para M_D e N_D . $H_M \parallel H_N$ é fortemente não-bloqueante e.r.a B se e somente se M_D e N_D forem fortemente não-conflitantes e.r.a B.

Prova:

$\forall d \in B, \overline{L_d(H_M \parallel H_N)} = L(H_M \parallel H_N) \Leftrightarrow \forall d \in B, \overline{L_d(H_M) \cap L_d(H_N)} = L(H_M) \cap L(H_N)$

$\Leftrightarrow \forall d \in B, \overline{L_d(M_D) \cap L_d(N_D)} = \overline{L_d(H_M)} \cap \overline{L_d(H_N)}$

$\Leftrightarrow \forall d \in B, \overline{L_d(M_D \cap N_D)} = \overline{L_d(M_D)} \cap \overline{L_d(N_D)}$. ♦

Teorema 5: Sejam S_1 e S_2 supervisores admissíveis para G fortemente não-bloqueante e.r.a $B \subseteq D$. Se $\Lambda_D(S_1/G)$ e $\Lambda_D(S_2/G)$ forem fortemente não-conflitantes e.r.a B, $S_1 \wedge S_2$ é um supervisor admissível para G fortemente não-bloqueante e.r.a B.

Prova: Como S_1 e S_2 são admissíveis,

$\forall s \in L(G), \Sigma_u \cap \Gamma(\delta(q_0, s)) \subseteq \Re(S_1(s))$ e $\Sigma_u \cap \Gamma(\delta(q_0, s)) \subseteq \Re(S_2(s))$.

Portanto,

$\forall s \in L(G), \Sigma_u \cap \Gamma(\delta(q_0, s)) \subseteq \Re(S_1(s)) \cap \Re(S_2(s))$

$= \Re(S_1 \wedge S_2(s))$,

isto é, $S_1 \wedge S_2$ é admissível.

Agora, $\forall b \in B$, $\overline{L_b(S_1 \wedge S_2/G)} = \overline{L_b(S_1/G) \cap L_b(S_2/G)}$

$= \overline{L_b(\Lambda_D(S_1/G)) \cap L_b(\Lambda_D(S_2/G))}$

$= \overline{L_b(\Lambda_D(S_1/G) \cap \Lambda_D(S_2/G))}$

$= \overline{L_b(\Lambda_D(S_1/G))} \cap \overline{L_b(\Lambda_D(S_2/G))}$

$= \overline{L_b(S_1/G)} \cap \overline{L_b(S_2/G)}$

$= L(S_1/G) \cap L(S_2/G)$

$= L(S_1 \wedge S_2/G)$. ♦

Proposição 21: Sejam M_D, $N_D \subseteq Pwr(Pwr(\Sigma^*) \times D)$ controláveis e.r.a G. Se M_D e N_D forem fracamente não-conflitantes e.r.a D, então $M_D \cap N_D$ é controlável e.r.a G.

Prova: $\overline{L_D(M_D \cap N_D)}\Sigma_u \cap L(G) \subseteq (\overline{L_D(M_D) \cap L_D(N_D)})\Sigma_u \cap L(G)$

$\subseteq (\overline{L_D(M_D)} \cap \overline{L_D(N_D)})\Sigma_u \cap L(G)$

$= (\overline{L_D(M_D)}\Sigma_u \cap L(G)) \cap (\overline{L_D(N_D)}\Sigma_u \cap L(G))$

$\subseteq \overline{L_D(M_D)} \cap \overline{L_D(N_D)}$

$= \overline{L_D(M_D \cap N_D)}$. ♦

Proposição 22: Sejam M_D, $N_D \subseteq Pwr(Pwr(\Sigma^*) \times D)$ especificações para G. Se $SupC(M_D, G)$ e $SupC(N_D, G)$ forem fracamente não-conflitantes e.r.a D, então $SupC(M_D \cap N_D, G) = SupC(M_D, G) \cap SupC(N_D, G)$.

Prova: De $M_D \cap N_D \subseteq M_D$ tem-se $SupC(M_D \cap N_D, G) \subseteq M_D$ e, portanto,
$SupC(M_D \cap N_D, G) \subseteq SupC(M_D, G)$. Simetricamente, tem-se
$SupC(M_D \cap N_D, G) \subseteq SupC(N_D, G)$.
Daí, $SupC(M_D \cap N_D, G) \subseteq SupC(M_D, G) \cap SupC(N_D, G)$.
Como $SupC(M_D, G)$ e $SupC(N_D, G)$ são controláveis e fracamente não-conflitantes,
$SupC(M_D, G) \cap SupC(N_D, G)$ é controlável. Também está contido em $M_D \cap N_D$.
Por conseguinte, $SupC(M_D, G) \cap SupC(N_D, G) \subseteq SupC(M_D \cap N_D, G)$. ♦

Proposição 23: Sejam M_D, $N_D \subseteq Pwr(Pwr(\Sigma^*) \times D)$ fortemente não-bloqueantes e.r.a $B \subseteq D$. Se M_D e N_D forem fortemente não-conflitantes e.r.a B, então $M_D \cap N_D$ é fortemente não-bloqueante e.r.a B.

Prova: $\forall b \in B$, $\overline{L_b(M_D \cap N_D)} = \overline{L_b(M_D)} \cap \overline{L_b(N_D)}$
$= \overline{L_D(M_D)} \cap \overline{L_D(N_D)}$
$\supseteq \overline{L_D(M_D \cap N_D)}$.
Por outro lado, é claro que $\forall b \in B$, $\overline{L_b(M_D \cap N_D)} \subseteq \overline{L_D(M_D \cap N_D)}$.
Por conseguinte, $\forall b \in B$, $\overline{L_b(M_D \cap N_D)} = \overline{L_D(M_D \cap N_D)}$. ♦

Proposição 24: Sejam M_D, $N_D \subseteq Pwr(Pwr(\Sigma^*) \times D)$. Se $SupSNB(M_D, B)$ e $SupSNB(N_D, B)$ forem fortemente não-conflitantes e.r.a $B \subseteq D$, então $SupSNB(M_D \cap N_D, B) = SupSNB(M_D, B) \cap SupSNB(N_D, B)$.

Prova: De $M_D \cap N_D \subseteq M_D$, tem-se $SupSNB(M_D \cap N_D, B) \subseteq M_D$ e, portanto,
$SupSNB(M_D \cap N_D, B) \subseteq SupSNB(M_D, B)$.
Simetricamente, tem-se $SupSNB(M_D \cap N_D, B) \subseteq SupSNB(N_D, B)$.
Daí, $SupSNB(M_D \cap N_D, B) \subseteq SupSNB(M_D, B) \cap SupSNB(N_D, B)$.
Como $SupSNB(M_D, B)$ e $SupSNB(N_D, B)$ são fortemente não-bloqueantes e.r.a B e
fortemente não-conflitantes e.r.a B, $SupSNB(M_D, B) \cap SupSNB(N_D, B)$ é fortemente não-bloqueante e.r.a B. Também está contido em $M_D \cap N_D$.
Consequentemente, $SupSNB(M_D, B) \cap SupSNB(N_D, B) \subseteq SupSNB(M_D \cap N_D, B)$. ♦

Proposição 25: Sejam M_D, $N_D \subseteq Pwr(Pwr(\Sigma^*) \times D)$ D–fechados e.r.a G. Se M_D e N_D forem fortemente não-conflitantes e.r.a $D \cap C$, então $M_D \cap N_D$ é D–fechado e.r.a G.

Prova: $\forall d \in (D \cap C)$, $L_d(M_D \cap N_D) = L_d(M_D) \cap L_d(N_D)$
$= (\overline{L_d(M_D)} \cap L_d(G)) \cap (\overline{L_d(N_D)} \cap L_d(G))$

$$= \overline{L_d(\mathrm{M}_D)} \cap \overline{L_d(\mathrm{N}_D)} \cap L_d(G)$$
$$= \overline{L_d(\mathrm{M}_D \cap \mathrm{N}_D)} \cap L_d(G). \quad \blacklozenge$$

Proposição 26: Sejam M_D, $\mathrm{N}_D \subseteq Pwr(Pwr(\Sigma^*) \times D)$ especificações para G. Se *SupCSNB*(M_D, G, B) e *SupCSNB*(N_D, G, B) forem fortemente não-conflitantes e.r.a $B \subseteq D$, então *SupCSNB*($\mathrm{M}_D \cap \mathrm{N}_D$, G, B) = *SupCSNB*(M_D, G, B) $\cap$ *SupCSNB*(N_D, G, B).

Prova: De $\mathrm{M}_D \cap \mathrm{N}_D \subseteq \mathrm{M}_D$ tem-se *SupCSNB*($\mathrm{M}_D \cap \mathrm{N}_D$, G, B) $\subseteq \mathrm{M}_D$ e, portanto, *SupCSNB*($\mathrm{M}_D \cap \mathrm{N}_D$, G, B) $\subseteq$ *SupCSNB*(M_D, G, B).
Simetricamente, tem-se *SupCSNB*($\mathrm{M}_D \cap \mathrm{N}_D$, G, B) $\subseteq$ *SupCSNB*(N_D, G, B).
Daí, *SupCSNB*($\mathrm{M}_D \cap \mathrm{N}_D$, G, B) $\subseteq$ *SupCSNB*(M_D, G, B) $\cap$ *SupCSNB*(N_D, G, B).
Como *SupCSNB*(M_D, G, B) e *SupCSNB*(N_D, G, B) são fortemente não-bloqueantes e.r.a B, controláveis e.r.a G e fortemente (e fracamente) não-conflitantes e.r.a B, *SupCSNB*(M_D, G, B) $\cap$ *SupCSNB*(N_D, G, B) é fortemente não-bloqueante e.r.a B e controlável e.r.a G. Também está contido em $\mathrm{M}_D \cap \mathrm{N}_D$.
Assim, *SupCSNB*(M_D, G, B) $\cap$ *SupCSNB*(N_D, G, B) $\subseteq$ *SupCSNB*($\mathrm{M}_D \cap \mathrm{N}_D$, G, B). $\blacklozenge$

Os resultados nessa seção podem ser resumidos da seguinte forma. Dados dois comportamentos coloridos admissíveis e D–fechados M_D e $\mathrm{N}_D \subseteq Pwr(Pwr(\Sigma^*) \times D)$ especificados para um SEDMT G, podem-se calcular supervisores ótimos fortemente não-bloqueantes S_1 e S_2, tais que $\Lambda_D(S_1/G)$ = *SupCSNB*(M_D, G, D) e $\Lambda_D(S_2/G)$ = *SupCSNB*(N_D, G, D). Se *SupCSNB*(M_D, G, D) e *SupCSNB*(N_D, G, D) forem fortemente não-conflitantes, então a conjunção de S_1 e S_2 é minimamente restritiva e fortemente não-bloqueante, isto é, a arquitetura de controle modular restringe a planta ao mesmo comportamento que um supervisor monolítico ótimo.

Exemplo: Para a linha de manufatura do Problema 3, ao invés de se projetar um supervisor monolítico, pode-se sintetizar um supervisor S_1 para $E_{gen,1}$ e S_2 para $E_{gen,2}$. Seguindo-se o procedimento indicado na Seção 4.3.4 para cada especificação, são obtidos os supervisores apresentados na Figura 39 e na Figura 40. Apesar de que individualmente eles sejam fortemente não-bloqueantes em relação à planta, sua ação conjunta não o é. Esse problema pode ser detectado verificando-se que o não-conflito forte entre $\Lambda_D(S_1/G)$ e $\Lambda_D(S_2/G)$, embora verdadeiro para a cor **i**, falha para as cores **a**, **b**, **c** e **e**.

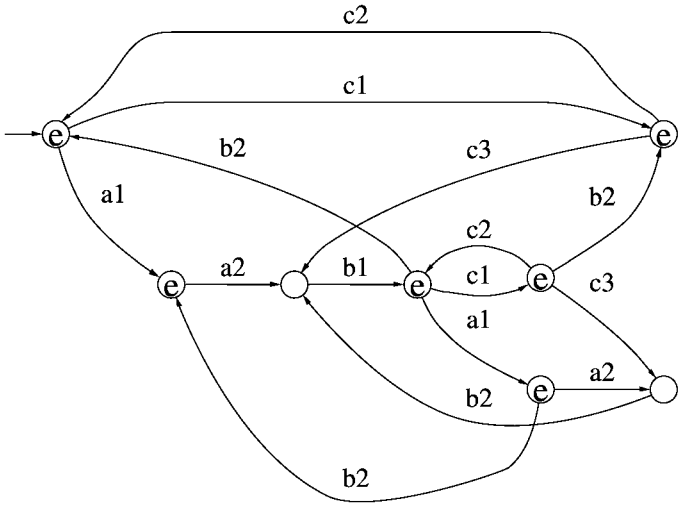


Figura 39: Supervisor pintor S_1 para $E_{gen,1}$

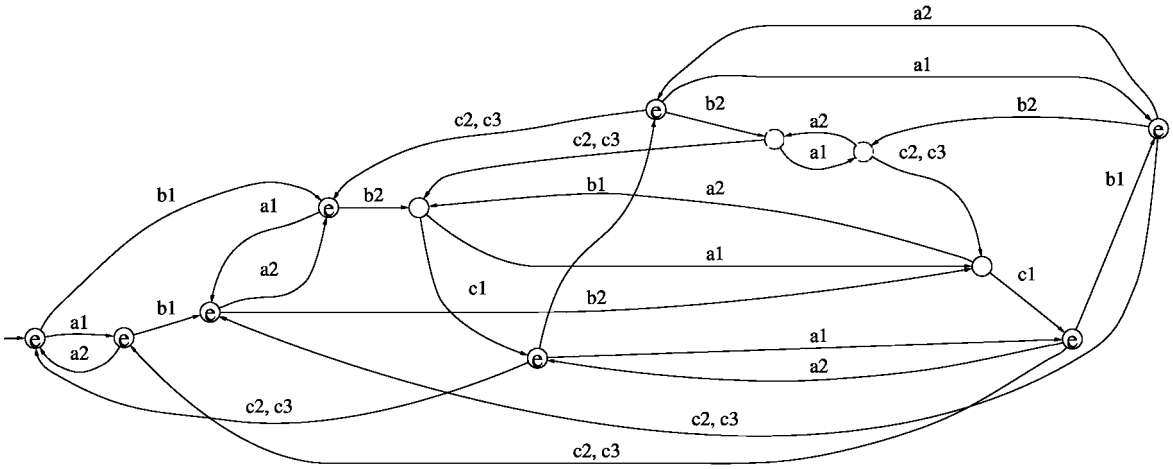


Figura 40: Supervisor pintor S_2 para $E_{gen,2}$

Quando o teste para não-conflito forte falha, a conjunção de supervisores modulares é problemática. O sistema controlado é fortemente bloqueante, pois pode alcançar estados a partir dos quais não há caminhos habilitados por ambos supervisores que levem o sistema a um estado marcado pela cor de cada tarefa relevante. Uma forma conservadora de se resolver tal problema é descartar a solução modular e calcular um supervisor monolítico ótimo fortemente não-bloqueante que atenda a ambas especificações, conforme apresentado na Seção 4.4. Todavia, as boas propriedades da arquitetura modular podem ser preservadas se for introduzido um supervisor adicional, denominado coordenador, com o único objetivo de evitar o conflito forte. Como visto na Seção 2.5.2, a literatura apresenta diversas abordagens alternativas para resolução de conflito (WONG *et al.*, 1995; CHEN *et*

al., 1995; WONG e WONHAM, 1998). Por exemplo, um coordenador para este problema é apresentado na Figura 41. Ele evita, entre outras, a ocorrência da cadeia fortemente bloqueante $a_1a_2b_1b_2a_1a_2$.

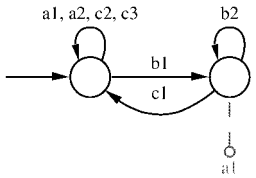


Figura 41: Coordenador para a solução modular da linha de transferência

5.2 Controle Modular Local

A arquitetura modular apresentada na seção anterior não permite tirar proveito do fato de que as especificações em geral não possuem o mesmo alfabeto ou conjunto de cores. Para sistemas a eventos discretos compostos, essa arquitetura implica a síntese de supervisores modulares a partir do modelo global da planta, cujo número de estados cresce exponencialmente com o número de subsistemas. Como sinalizado por QUEIROZ e CURY (2000a), pode ser vantajoso projetar cada supervisor modular sobre uma versão local da planta, que é obtida pela composição dos subsistemas afetados pela respectiva especificação modular. A seguir, define-se a arquitetura para conjunção de supervisores modulares locais para sistemas a eventos discretos multitarefa compostos. Como em QUEIROZ e CURY (2000a), assume-se que a planta seja dada pela composição de múltiplos subsistemas assíncronos (sistema-produto) e que a especificação global seja dada por um conjunto de especificações locais, cada qual sincronizando subconjuntos de subsistemas.

Seja o comportamento em malha aberta de um SEDMT modelado pela composição dos subsistemas do conjunto $\{G_i = (Q_i, \Sigma_i, C_i, \delta_i, \chi_i, q_{0i}),\ i = 1, 2, ..., n\}$, com $\Sigma_i \cap \Sigma_j = \varnothing$, para $i \neq j$. Sejam as especificações expressas como um conjunto de comportamentos admissíveis genéricos $\{M_{gen,j} \subseteq Pwr(Pwr(\Sigma_{gen,j}^*) \times C_{gen,j}),\ j = 1,2,...,m\}$, com $\Sigma_{gen,j} \subseteq \Sigma$. Para $j = 1,...,m$, define-se a planta local $G_{loc,j} = (Q_{loc,j}, \Sigma_{loc,j}, C_{loc,j}, \delta_{loc,j}, \chi_{loc,j}, q_{0loc,j})$, com função de eventos ativos $\Gamma_{loc,i}$, associada ao comportamento admissível $M_{gen,j}$, pela composição (assíncrona) de todos os subsistemas que são afetados por $M_{gen,j}$, isto é, que compartilham eventos com $M_{gen,j}$. Formalmente,

$$G_{loc,j} := \|\{G_i: \Sigma_i \cap \Sigma_{gen,j} \neq \varnothing\}.$$

Para $j = 1,...,m$, define-se também a planta complementar $G_{loc,j}{}^{c}$ respectiva ao comportamento admissível $M_{gen,j}$ por:

$$G_{loc,j}{}^{c} := ||\{G_i: \Sigma_i \cap \Sigma_{gen,j} = \varnothing\}.$$

Como os subsistemas de $\{G_i, i = 1,2,...,n\}$ são assíncronos, sempre é verdade que, para $j = 1,...,m$, $G_{loc,j}$ e $G_{loc,j}{}^{c}$ também são assíncronos. Por conseguinte, para os sistemas aqui abordados, pode-se dizer que a planta complementar não interfere nos problemas locais. Entretanto, é possível que plantas locais distintas sejam síncronas, já que diferentes problemas locais podem afetar o mesmo subsistema. Assim, quando duas plantas locais compartilham um evento, elas compartilham um subsistema contendo esse evento.

A planta global $G = (Q, \Sigma, C, \delta, \chi, q_0)$, com função de eventos ativos Γ, é dada por $G = ||_{j = 1,2,...,m} G_{loc,j}$, ou seja, G é formada pela composição de todos os subsistemas de $\{G_i, i = 1,2,...,n\}$ afetados por alguma especificação. Assim, o alfabeto global é dado por $\Sigma = \cup_{j = 1,...,m} \Sigma_{loc,i}$. Para $j = 1,...,m$, a especificação local $M_{loc,j} \subseteq Pwr(Pwr(\Sigma_{loc,j}{}^{*}) \times D_{loc,j})$ é definida por $M_{loc,j} := M_{gen,j} \,||\, \Lambda_{Cloc,j}(G_{loc,j})$. Define-se $E_{loc,j} := D_{loc,j} - C_{loc,j}$.

Sem perda de generalidade, assume-se que $m = 2$. Sejam $S_{loc,1}: L(G_{loc,1}) \rightarrow Pwr(\Sigma_{loc,1}) \times Pwr(E_{loc,1})$ e $S_{loc,2}: L(G_{loc,2}) \rightarrow Pwr(\Sigma_{loc,2}) \times Pwr(E_{loc,2})$ supervisores pintores para $G_{loc,1}$ e $G_{loc,2}$, respectivamente. Para $i = 1,2$, definem-se as projeções naturais $P_{loc,i}: \Sigma^{*} \rightarrow \Sigma_{loc,i}{}^{*}$. Nota-se que, como $G = G_{loc,1} \,||\, G_{loc,2}$, para qualquer $s \in L(G)$ tem-se $P_{loc,1}(s) \in L(G_{loc,1})$ e $P_{loc,2}(s) \in L(G_{loc,2})$. Assim, para uma cadeia $s \in L(G)$, com $S_{loc,1}(P_{loc,1}(s)) = (\gamma_1, \mu_1)$ e $S_{loc,2}(P_{loc,2}(s)) = (\gamma_2, \mu_2)$, a ação de controle do supervisor representando a conjunção de $S_{loc,1}$ e $S_{loc,2}$, $S_{loc,1} \wedge S_{loc,2}: L(G) \rightarrow Pwr(\Sigma) \times Pwr(E)$, onde $E = E_{loc,1} \cup E_{loc,2}$, é dada por:

$$S_{loc,1} \wedge S_{loc,2}(s) := ((\gamma_1 \cup (\Sigma_{loc,2} - \Sigma_{loc,1})) \cap (\gamma_2 \cup (\Sigma_{loc,1} - \Sigma_{loc,2})),$$
$$(\mu_1 \cup (E_{loc,2} - E_{loc,1})) \cap (\mu_2 \cup (E_{loc,1} - E_{loc,2}))).$$

A conjunção de dois supervisores pintores modulares locais é descrita na Figura 42. A ação de controle dos supervisores locais é estendida para a planta global habilitando-se todos os eventos de Σ que não estão no seu alfabeto local e marcando-se todas as novas cores que não estão no seu conjunto de cores local. Um evento controlável compartilhado é habilitado na planta se e somente se for habilitado pela ação global de ambos supervisores para as cadeias correspondendo às seqüências de eventos ocorridas nas respectivas plantas locais. Em outras palavras, é suficiente que um supervisor desabilite um evento para que este seja desabilitado na planta. Da mesma forma, as novas cores comuns aos conjuntos de cores de ambos supervisores são marcadas no comportamento em malha fechada se e

somente se for marcada por ambos supervisores. Pode-se verificar que o comportamento em malha fechada sob controle modular local é dado por:

$$L(S_{loc,1} \wedge S_{loc,2}/G) = L(S_{loc,1}/G_{loc,1}) \,||\, L(S_{loc,2}/G_{loc,2});$$

$$\forall d \in D,\ L_d(S_{loc,1} \wedge S_{loc,2}/G) = \begin{cases} L_d(S_{loc,1}/G_{loc,1}) \,||\, L_d(S_{loc,2}/G_{loc,2}), \text{ se } d \in D_{loc,1} \cap D_{loc,2} \\ L_d(S_{loc,1}/G_{loc,1}) \,||\, L(S_{loc,2}/G_{loc,2}), \text{ se } d \in D_{loc,1} - D_{loc,2} \\ L(S_{loc,1}/G_{loc,1}) \,||\, L_d(S_{loc,2}/G_{loc,2}), \text{ se } d \in D_{loc,2} - D_{loc,1} \end{cases}.$$

Portanto,

$$S_{loc,1} \wedge S_{loc,2}/G = (S_{loc,1}/G_{loc,1}) \,||\, (S_{loc,2}/G_{loc,2}).$$

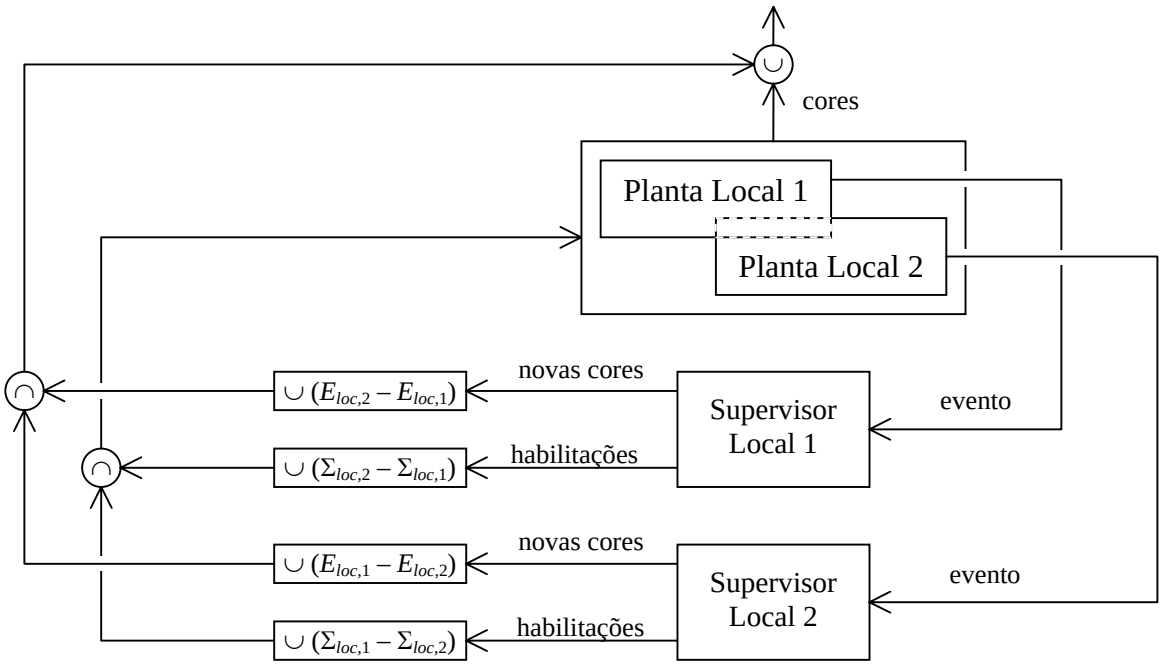


Figura 42: Arquitetura de controle modular local

De acordo com a Seção 4.3.3, se $S_{loc,1}$ e $S_{loc,2}$ forem admissíveis, eles podem ser respectivamente representados pelos geradores com marcação colorida $H_{loc,1}$ e $H_{loc,2}$ tais que $S_{loc,1}/G_{loc,1} = H_{loc,1} \,||\, G_{loc,1}$ e $S_{loc,2}/G_{loc,2} = H_{loc,2} \,||\, G_{loc,2}$. Então, pela definição de composição de supervisores locais, tem-se:

$$\begin{aligned} S_{loc,1} \wedge S_{loc,2}/G &= (S_{loc,1}/G_{loc,1}) \,||\, (S_{loc,2}/G_{loc,2}) \\ &= (H_{loc,1} \,||\, G_{loc,1}) \,||\, (H_{loc,2} \,||\, G_{loc,2}) \\ &= H_{loc,1} \,||\, H_{loc,2} \,||\, G_{loc,1} \,||\, G_{loc,2} \\ &= H_{loc,1} \,||\, H_{loc,2} \,||\, G. \end{aligned}$$

Assim, para que as propriedades do sistema controlado na arquitetura modular local sejam deduzidas das propriedades dos supervisores pintores locais, é necessário estudar as condições sob as quais as propriedades de controlabilidade, *D*–fechamento e não-bloqueio forte são preservadas pela composição síncrona. As condições fundamentais para isso são não-conflito forte e fraco, que são estendidas para comportamentos coloridos com alfabetos locais e conjuntos de cores distintos a seguir.

Sejam os comportamentos coloridos $M_{D1} \subseteq Pwr(Pwr(\Sigma_1^*) \times D_1)$ e $N_{D2} \subseteq Pwr(Pwr(\Sigma_2^*) \times D_2)$. Sejam $B \subseteq D = D_1 \cup D_2$, $B_1 = B \cap D_1$ e $B_2 = B \cap D_2$. Diz-se que M_{D1} e N_{D2} são fracamente não-conflitantes e.r.a *B* sempre que

$$\overline{L_{B1}(M_{D1})} \parallel \overline{L_{B2}(N_{D2})} = \overline{L_B(M_{D1} \parallel N_{D2})}.$$

Em palavras, M_{D1} e N_{D2} são fracamente não-conflitantes e.r.a *B* se toda cadeia de $(\Sigma_1 \cup \Sigma_2)^*$ que for um prefixo de $L_{B1}(M_{D1})$e de $L_{B2}(N_{D2})$ (quando projetada nos respectivos alfabetos) puder ser estendida para uma cadeia que complete uma tarefa de $B_1 \cup B_2$ em $M_{D1} \parallel N_{D2}$, isto é, se a composição síncrona de qualquer prefixo de tarefas "locais" (de B_1 e B_2) puder completar uma tarefa "global" (de B_1 ou de B_2). Além disso, diz-se que M_{D1} e N_{D2} são fortemente não-conflitantes e.r.a *B* quando

$$\forall b \in B, L_b(\overline{M_{D1}} \parallel \overline{N_{D2}}) = L_b(\overline{M_{D1} \parallel N_{D2}}).$$

Equivalentemente, M_{D1} e N_{D2} são fortemente não-conflitantes e.r.a *B* se

$$\begin{aligned} &\forall b \in B_1 \cap B_2, \overline{L_b(M_{D1})} \parallel \overline{L_b(N_{D2})} = \overline{L_b(M_{D1}) \parallel L_b(N_{D2})} \\ \wedge\ &\forall b \in B_1 - B_2, \overline{L_b(M_{D1})} \parallel \overline{L_{D2}(N_{D2})} = \overline{L_b(M_{D1}) \parallel \overline{L_{D2}(N_{D2})}} \\ \wedge\ &\forall b \in B_2 - B_1, \overline{L_{D1}(M_{D1})} \parallel \overline{L_d(N_{D2})} = \overline{\overline{L_{D1}(M_{D1})} \parallel L_d(N_{D2})}\ . \end{aligned}$$

Não-conflito forte de M_{D1} e N_{D2} e.r.a *B* significa que, para qualquer tarefa de $B_1 \cup B_2$, a composição síncrona de cadeias que possam completar uma versão local da tarefa em M_{D1} e N_{D2} pode completar a respectiva tarefa em $M_{D1} \parallel N_{D2}$. Naturalmente, não-conflito forte e.r.a *B* implica não-conflito fraco e.r.a *B*. As seguintes proposições indicam a relação entre não-conflito fraco/forte de comportamento coloridos e não-bloqueio fraco/forte de GMCs com distintos alfabetos e conjuntos de cores.

Proposição 27: Sejam os comportamentos coloridos $M_{D1} \subseteq Pwr(Pwr(\Sigma_1^*) \times D_1)$ e $N_{D2} \subseteq Pwr(Pwr(\Sigma_2^*) \times D_2)$. Sejam $B \subseteq D = D_1 \cup D_2$, $B_1 = B \cap D_1$ e $B_2 = B \cap D_2$. Sejam, respectivamente, H_M e H_N geradores com marcação colorida para M_{D1} e N_{D2} fracamente não-bloqueantes (e.r.a B_1 e B_2). $H_M \parallel H_N$ é fracamente não-bloqueante e.r.a *B* se e somente se M_{D1} e N_{D2} forem fracamente não-conflitantes e.r.a *B*.

Prova: $\overline{L_B(H_M \parallel H_N)} = L(H_M \parallel H_N) \Leftrightarrow \overline{L_B(\Lambda_D(H_M \parallel H_N))} = L(H_M) \parallel L(H_N)$
$\Leftrightarrow \overline{L_B(\Lambda_{D1}(H_M) \parallel \Lambda_{D2}(H_N))} = \overline{L_{B1}(H_M)} \parallel \overline{L_{B2}(H_N)}$
$\Leftrightarrow \overline{L_B(M_{D1} \parallel N_{D2})} = \overline{L_{B1}(M_{D1})} \parallel \overline{L_{B2}(N_{D2})}$. ♦

Proposição 28: Sejam os comportamentos coloridos $M_{D1} \subseteq Pwr(Pwr(\Sigma_1^*) \times D_1)$ e $N_{D2} \subseteq Pwr(Pwr(\Sigma_2^*) \times D_2)$. Seja $B \subseteq D = D_1 \cup D_2$, $B_1 = B \cap D_1$ e $B_2 = B \cap D_2$. Sejam M_{D1} e N_{D2} fortemente não-bloqueantes e.r.a B_1 e B_2, respectivamente. Sejam, respectivamente, H_M e H_N geradores com marcação colorida para M_{D1} e N_{D2} fortemente não-bloqueantes (e.r.a B_1 e B_2) . $H_M \parallel H_N$ é fortemente não-bloqueante e.r.a B se e somente se M_{D1} e N_{D2} forem fortemente não-conflitantes e.r.a B.

Prova:

$\forall b \in B, \overline{L_b(H_M \parallel H_N)} = L(H_M \parallel H_N) \Leftrightarrow \forall b \in B, \overline{L_b(\Lambda_D(H_M \parallel H_N))} = L(H_M) \parallel L(H_N)$
$\Leftrightarrow \forall b \in B_1 \cap B_2, \overline{L_b(\Lambda_{D1}(H_M) \parallel \Lambda_{D2}(H_N))} = \overline{L_b(H_M)} \parallel \overline{L_b(H_N)}$
$\wedge \forall b \in B_1 - B_2, \overline{L_b(\Lambda_{D1}(H_M) \parallel \Lambda_{D2}(H_N))} = \overline{L_b(H_M)} \parallel \overline{L_{D2}(H_N)}$
$\wedge \forall b \in B_2 - B_1, \overline{L_b(\Lambda_{D1}(H_M) \parallel \Lambda_{D2}(H_N))} = \overline{L_{D1}(H_M)} \parallel \overline{L_b(H_N)}$
$\Leftrightarrow \forall b \in B_1 \cap B_2, \overline{L_b(M_{D1} \parallel N_{D2})} = L_b(\overline{M_{D1}}) \parallel L_b(\overline{N_{D2}})$
$\wedge \forall b \in B_1 - B_2, \overline{L_b(M_{D1} \parallel N_{D2})} = L_b(\overline{M_{D1}}) \parallel \overline{L_{D2}(N_{D2})}$
$\wedge \forall b \in B_2 - B_1, \overline{L_b(M_{D1} \parallel N_{D2})} = \overline{L_{D1}(M_{D1})} \parallel L_b(\overline{N_{D2}})$
$\Leftrightarrow \forall b \in B, \overline{L_b(M_{D1} \parallel N_{D2})} = L_b(\overline{M_{D1}} \parallel \overline{N_{D2}})$ ♦

Teorema 6: Para um SEDMT composto definido como acima, sejam $B \subseteq D = D_{loc,1} \cup D_{loc,2}$, $B_{loc,1} = B \cap D_{loc,1}$ e $B_{loc,2} = B \cap D_{loc,2}$. Sejam, respectivamente, os supervisores $S_{loc,1}$ e $S_{loc,2}$ admissíveis para $G_{loc,1}$ e $G_{loc,2}$ fortemente não-bloqueantes (e.r.a $B_{loc,1}$ e $B_{loc,2}$, respectivamente). Se $\Lambda_{D1}(S_{loc,1}/G_{loc,1})$ e $\Lambda_{D2}(S_{loc,2}/G_{loc,2})$ forem fortemente não-conflitantes e.r.a B, $S_{loc,1} \wedge S_{loc,2}$ é um supervisor admissível para G fortemente não-bloqueante e.r.a B.

Prova: Como $S_{loc,1}$ e $S_{loc,2}$ são admissíveis,
$\forall s \in L(G), \Sigma_u \cap \Gamma_{loc,1}(\delta_{loc,1}(q_{0loc,1}, P_{loc,1}(s))) \subseteq \Re(S_{loc,1}(P_{loc,1}(s)))$ e
$\Sigma_u \cap \Gamma_{loc,2}(\delta_{loc,2}(q_{0loc,2}, P_{loc,2}(s))) \subseteq \Re(S_{loc,2}(P_{loc,2}(s)))$.
Da definição de composição síncrona , pode-se verificar que
$\delta(q_0, s) = \delta((q_{0loc,1}, q_{0loc,2}), s) = (\delta_{loc,1}(q_{0loc,1}, P_{loc,1}(s)), \delta_{loc,2}(q_{0loc,2}, P_{loc,2}(s)))$.
Portanto,

$\forall s \in L(G), \Sigma_u \cap \Gamma(\delta(q_0, s)) = \Sigma_u \cap (\Gamma_{loc,1}(\delta_{loc,1}(q_{0loc,1}, P_{loc,1}(s))) \cup (\Sigma_{loc,2} - \Sigma_{loc,1}))$
$\cap (\Gamma_{loc,2}(\delta_{loc,2}(q_{0loc,2}, P_{loc,2}(s))) \cup (\Sigma_{loc,1} - \Sigma_{loc,2}))$
$= (\Sigma_u \cap \Gamma_{loc,1}(\delta_{loc,1}(q_{0loc,1}, P_{loc,1}(s))) \cup \Sigma_u \cap (\Sigma_{loc,2} - \Sigma_{loc,1}))$
$\cap (\Sigma_u \cap \Gamma_{loc,2}(\delta_{loc,2}(q_{0loc,2}, P_{loc,2}(s))) \cup \Sigma_u \cap (\Sigma_{loc,1} - \Sigma_{loc,2}))$
$\subseteq (\Re(S_{loc,1}(P_{loc,1}(s))) \cup \Sigma_u \cap (\Sigma_{loc,2} - \Sigma_{loc,1}))$

$$\begin{aligned}
&\cap (\Re(S_{loc,2}(\mathrm{P}_{loc,2}(s))) \cup \Sigma_u \cap (\Sigma_{loc,1} - \Sigma_{loc,2})) \\
&\subseteq (\Re(S_{loc,1}(\mathrm{P}_{loc,1}(s))) \cup (\Sigma_{loc,2} - \Sigma_{loc,1})) \\
&\quad \cap (\Re(S_{loc,2}(\mathrm{P}_{loc,2}(s))) \cup (\Sigma_{loc,1} - \Sigma_{loc,2})) \\
&= \Re(S_{loc,1} \wedge S_{loc,2}(s)),
\end{aligned}$$

isto é, $S_{loc,1} \wedge S_{loc,2}$ é admissível.

Agora,

$$\begin{aligned}
\forall b \in B,\ \overline{L_b(S_{loc,1} \wedge S_{loc,2}/G)} &= \overline{L_b((S_{loc,1}/G_{loc,1}) \parallel (S_{loc,2}/G_{loc,2}))} \\
&= \overline{L_b(\Lambda_{Dloc,1}(S_{loc,1}/G_{loc,1}) \parallel \Lambda_{Dloc,2}(S_{loc,2}/G_{loc,2}))} \\
&= L_b(\overline{\Lambda_{Dloc,1}(S_{loc,1}/G_{loc,1})} \parallel \overline{\Lambda_{Dloc,2}(S_{loc,2}/G_{loc,2})}) \\
&= L_b(\overline{\Lambda_{Dloc,1}(S_{loc,1}/G_{loc,1})}) \parallel L_b(\overline{\Lambda_{Dloc,2}(S_{loc,2}/G_{loc,2})}) \\
&\qquad \text{se } b \in B_{loc,1} \cap B_{loc,2}, \\
&\quad L_b(\overline{\Lambda_{Dloc,1}(S_{loc,1}/G_{loc,1})}) \parallel L_{Dloc,2}(\overline{\Lambda_{Dloc,2}(S_{loc,2}/G_{loc,2})}) \\
&\qquad \text{se } b \in B_{loc,1} - B_{loc,2}, \\
&\quad L_{Dloc,1}(\overline{\Lambda_{Dloc,1}(S_{loc,1}/G_{loc,1})}) \parallel L_b(\overline{\Lambda_{Dloc,2}(S_{loc,2}/G_{loc,2})}) \\
&\qquad \text{se } b \in B_{loc,2} - B_{loc,1} \\
&= \overline{L_{Dloc,1}(S_{loc,1}/G_{loc,1})} \parallel \overline{L_{Dloc,2}(S_{loc,2}/G_{loc,2})} \\
&= L(S_{loc,1}/G_{loc,1}) \parallel L(S_{loc,2}/G_{loc,2}) \\
&= L(S_{loc,1} \wedge S_{loc,2}/G). \qquad \blacklozenge
\end{aligned}$$

Proposição 29: Sejam os comportamentos coloridos $\mathrm{M} \subseteq Pwr(Pwr(\Sigma_{loc,1}^{\ *}) \times D_{loc,1})$ e $\mathrm{N} \subseteq Pwr(Pwr(\Sigma_{loc,2}^{\ *}) \times D_{loc,2})$ controláveis e.r.a $G_{loc,1}$ e $G_{loc,2}$, respectivamente. Se M e N forem fracamente não-conflitantes e.r.a $D = D_{loc,1} \cup D_{loc,2}$, então M || N é controlável e.r.a G.

Prova:
$$\begin{aligned}
\overline{L_D(\mathrm{M} \parallel \mathrm{N})}\Sigma_u \cap L(G) &\subseteq \overline{L_{Dloc,1}(\mathrm{M}) \parallel L_{Dloc,2}(\mathrm{N})}\Sigma_u \cap L(G) \\
&\subseteq \overline{L_{Dloc,1}(\mathrm{M})} \parallel \overline{L_{Dloc,2}(\mathrm{N})}\Sigma_u \cap L(G_{loc,1}) \| L(G_{loc,2}) \\
&= (\overline{L_{Dloc,1}(\mathrm{M})}\Sigma_u \cap L(G_{loc,1})) \parallel (\overline{L_{Dloc,2}(\mathrm{N})}\Sigma_u \cap L(G_{loc,2})) \\
&\subseteq \overline{L_{Dloc,1}(\mathrm{M})} \parallel \overline{L_{Dloc,2}(\mathrm{N})} \\
&= \overline{L_D(\mathrm{M} \parallel \mathrm{N})}. \qquad \blacklozenge
\end{aligned}$$

Proposição 30: Sejam as especificações $\mathrm{M} \subseteq Pwr(Pwr(\Sigma_{loc,1}^{\ *}) \times D_{loc,1})$ e $\mathrm{N} \subseteq Pwr(Pwr(\Sigma_{loc,2}^{\ *}) \times D_{loc,2})$ para $G_{loc,1}$ e $G_{loc,2}$, respectivamente. Se $SupC(\mathrm{M}, G_{loc,1})$ e $SupC(\mathrm{N}, G_{loc,2})$ forem fracamente não-conflitantes e.r.a $D = D_{loc,1} \cup D_{loc,2}$, então $SupC(\mathrm{M} \parallel \mathrm{N}, G) = SupC(\mathrm{M}, G_{loc,1}) \parallel SupC(\mathrm{N}, G_{loc,2})$.

Prova: De $\mathrm{M} \parallel \mathrm{N} \subseteq \mathrm{M} \parallel G_{loc,2} \subseteq \mathrm{M} \parallel G_{loc,1}{}^{c}$,tem-se $SupC(\mathrm{M} \parallel \mathrm{N}, G) \subseteq \mathrm{M} \parallel G_{loc,1}{}^{c}$ e portanto
$$\begin{aligned}
SupC(\mathrm{M} \parallel \mathrm{N}, G) &\subseteq SupC(\mathrm{M} \parallel G_{loc,1}{}^{c}, G_{loc,1} \parallel G_{loc,1}{}^{c}) \\
&= SupC(\mathrm{M}, G_{loc,1}) \parallel G_{loc,1}{}^{c}.
\end{aligned}$$
Simetricamente, tem-se $SupC(\mathrm{M} \parallel \mathrm{N}, G) \subseteq SupC(\mathrm{N}, G_{loc,2}) \parallel G_{loc,2}{}^{c}$.

Daí, $SupC(\mathrm{M} \parallel \mathrm{N}, G) \subseteq SupC(\mathrm{M}, G_{loc,1}) \parallel G_{loc,1}{}^{c} \cap SupC(\mathrm{N}, G_{loc,2}) \parallel G_{loc,2}{}^{c}$
$= SupC(\mathrm{M}, G_{loc,1}) \parallel SupC(\mathrm{N}, G_{loc,2})$.
Como $SupC(\mathrm{M}, G_{loc,1})$ e $SupC(\mathrm{N}, G_{loc,2})$ são controláveis e fracamente não-conflitantes e.r.a D, $SupC(\mathrm{M}, G_{loc,1}) \parallel SupC(\mathrm{N}, G_{loc,2})$ é controlável e.r.a G. Também está contido em $\mathrm{M} \parallel \mathrm{N}$. Portanto, $SupC(\mathrm{M}, G_{loc,1}) \parallel SupC(\mathrm{N}, G_{loc,2}) \subseteq SupC(\mathrm{M} \parallel \mathrm{N}, G)$. ♦

Proposição 31: Sejam $\mathrm{M}_{D1} \subseteq Pwr(Pwr(\Sigma_1^*) \times D_1)$ e $\mathrm{N}_{D2} \subseteq Pwr(Pwr(\Sigma_2^*) \times D_2)$ comportamentos coloridos. Sejam $B \subseteq D = D_1 \cup D_2$, $B_1 = B \cap D_1$ e $B_2 = B \cap D_2$. Sejam M_{D1} e N_{D2} fortemente não-bloqueantes e.r.a B_1 e B_2, respectivamente. Se M_{D1} e N_{D2} forem fortemente não-conflitantes e.r.a B, então M_{D1} e N_{D2} são fracamente não-conflitantes e.r.a D.

Prova: $\overline{L_{D1}(\mathrm{M}_{D1})} \parallel \overline{L_{D2}(\mathrm{N}_{D2})} = \overline{L_b(\mathrm{M}_{D1})} \parallel \overline{L_b(\mathrm{N}_{D2})}$,$\forall b \in B_1 \cap B_2$,
$\overline{L_b(\mathrm{M}_{D1})} \parallel \overline{L_{D2}(\mathrm{N}_{D2})}$, $\forall b \in B_1 - B_2$,
$\overline{L_{D1}(\mathrm{M}_{D1})} \parallel \overline{L_b(\mathrm{N}_{D2})}$, $\forall b \in B_2 - B_1$
$= L_b(\overline{\mathrm{M}_{D1}} \parallel \overline{\mathrm{N}_{D2}})$, $\forall b \in B$
$= \overline{L_b(\mathrm{M}_{D1} \parallel \mathrm{N}_{D2})}$, $\forall b \in B$
$\subseteq \overline{L_D(\mathrm{M}_{D1} \parallel \mathrm{N}_{D2})}$.
Por outro lado, é claro que $\overline{L_D(\mathrm{M}_{D1} \parallel \mathrm{N}_{D2})} \subseteq \overline{L_{D1}(\mathrm{M}_{D1})} \parallel \overline{L_{D2}(\mathrm{N}_{D2})}$.
Portanto, $\overline{L_D(\mathrm{M}_{D1} \parallel \mathrm{N}_{D2})} = \overline{L_{D1}(\mathrm{M}_{D1})} \parallel \overline{L_{D2}(\mathrm{N}_{D2})}$. ♦

Proposição 32: Sejam $\mathrm{M}_{D1} \subseteq Pwr(Pwr(\Sigma_1^*) \times D_1)$ e $\mathrm{N}_{D2} \subseteq Pwr(Pwr(\Sigma_2^*) \times D_2)$ comportamentos coloridos. Sejam $B \subseteq D = D_1 \cup D_2$, $B_1 = B \cap D_1$ e $B_2 = B \cap D_2$. Sejam M_{D1} e N_{D2} fortemente não-bloqueantes e.r.a B_1 e B_2, respectivamente. Se M_{D1} e N_{D2} forem fortemente não-conflitantes e.r.a B, então $\mathrm{M}_{D1} \parallel \mathrm{N}_{D2}$ é fortemente não-bloqueante e.r.a B.

Prova: $\forall b \in B$, $\overline{L_b(\mathrm{M}_{D1} \parallel \mathrm{N}_{D2})} = L_b(\overline{\mathrm{M}_{D1}} \parallel \overline{\mathrm{N}_{D2}})$
$= \overline{L_b(\mathrm{M}_{D1})} \parallel \overline{L_b(\mathrm{N}_{D2})}$ se $b \in B_1 \cap B_2$,
$\overline{L_b(\mathrm{M}_{D1})} \parallel \overline{L_{D2}(\mathrm{N}_{D2})}$ se $b \in B_1 - B_2$,
$\overline{L_{D1}(\mathrm{M}_{D1})} \parallel \overline{L_b(\mathrm{N}_{D2})}$ se $b \in B_2 - B_1$
$= \overline{L_{D1}(\mathrm{M}_{D1})} \parallel \overline{L_{D2}(\mathrm{N}_{D2})}$
$= \overline{L_D(\mathrm{M}_{D1} \parallel \mathrm{N}_{D2})}$. ♦

Proposição 33: Sejam $\mathrm{M}_{D1} \subseteq Pwr(Pwr(\Sigma_1^*) \times D_1)$ e $\mathrm{N}_{D2} \subseteq Pwr(Pwr(\Sigma_2^*) \times D_2)$. Sejam $B \subseteq D = D_1 \cup D_2$, $B_1 = B \cap D_1$ e $B_2 = B \cap D_2$. Se $SupSNB(\mathrm{M}_{D1}, B_1)$ e $SupSNB(\mathrm{N}_{D2}, B_2)$ forem fortemente não-conflitantes e.r.a B_1 e B_2, respectivamente, então $SupSNB(\mathrm{M}_{D1} \parallel \mathrm{N}_{D2}, B) = SupSNB(\mathrm{M}_{D1}, B_1) \parallel SupSNB(\mathrm{N}_{D2}, B_2)$.

Prova: Definem-se os comportamentos coloridos $\mathrm{M}_{\Sigma 1} := \{(\Sigma_1^*, d), d \in D_1\}$ e $\mathrm{N}_{\Sigma 2} := \{(\Sigma_2^*, d), d \in D_2\}$. De $\mathrm{M}_{D1} \parallel \mathrm{N}_{D2} \subseteq \mathrm{M}_{D1} \parallel \mathrm{N}_{\Sigma 2}$ tem-se

$SupSNB(M_{D1} \parallel N_{D2}, B) \subseteq M_{D1} \parallel N_{\Sigma 2}$ e, portanto,

$SupSNB(M_{D1} \parallel N_{D2}, B) \subseteq SupSNB(M_{D1} \parallel N_{\Sigma 2}, B)$

$= SupSNB(M_{D1}, B_1) \parallel N_{\Sigma 2}$.

Simetricamente, tem-se $SupSNB(M_{D1} \parallel N_{D2}, B) \subseteq SupSNB(M_{D2}, B_2) \parallel M_{\Sigma 1}$.

Daí, $SupSNB(M_{D1} \parallel N_{D2}, B) \subseteq SupSNB(M_{D1}, B_1) \parallel N_{\Sigma 2} \cap SupSNB(M_{D2}, B_2) \parallel M_{\Sigma 1}$

$= SupSNB(M_{D1}, B_1) \parallel SupSNB(M_{D2}, B_2)$.

Como $SupSNB(M_{D1}, B_1)$ e $SupSNB(M_{D2}, B_2)$ são fortemente não-bloqueantes e.r.a B_1 e B_2, respectivamente, e fortemente não-conflitantes e.r.a B, $SupSNB(M_{D1}, B_1) \parallel SupSNB(M_{D2}, B_2)$ é fortemente não-bloqueante e.r.a B. Também contido em $M_{D1} \parallel N_{D2}$. Por conseguinte,

$SupSNB(M_{D1}, B_1) \parallel SupSNB(M_{D2}, B_2) \subseteq SupSNB(M_{D1} \parallel N_{D2}, B)$. ♦

Proposição 34: Sejam as especificações $M \subseteq Pwr(Pwr(\Sigma_{loc,1}^*) \times D_{loc,1})$ e $N \subseteq Pwr(Pwr(\Sigma_{loc,2}^*) \times D_{loc,2})$ para $G_{loc,1}$ e $G_{loc,2}$, respectivamente. Seja M $D_{loc,1}$–fechada e.r.a $G_{loc,1}$ e seja M $D_{loc,2}$–fechada e.r.a $G_{loc,2}$. Se M e N forem fortemente não-conflitantes e.r.a $D \cap C$ então M || N é D–fechada e.r.a G.

Prova:

$$
\begin{aligned}
\forall d \in (D \cap C),\ L_d(M \parallel N) = {} & L_d(M) \parallel L_d(N) \text{ se } d \in D_{loc,1} \cap D_{loc,2},\\
& L_d(M) \parallel \overline{L_{Dloc,2}(N)} \text{ se } d \in D_{loc,1} - D_{loc,2},\\
& \overline{L_{Dloc,1}(M)} \parallel L_d(N) \text{ se } d \in D_{loc,2} - D_{loc,1}\\
= {} & (\overline{L_d(M)} \cap L_d(G_{loc,1})) \parallel (\overline{L_d(N)} \cap L_d(G_{loc,2}))\\
& \qquad \text{se } d \in D_{loc,1} \cap D_{loc,2},\\
& (\overline{L_d(M)} \cap L_d(G_{loc,1})) \parallel \overline{L_{Dloc,2}(N)} \text{ se } d \in D_{loc,1} - D_{loc,2},\\
& \overline{L_{Dloc,1}(M)} \parallel (\overline{L_d(N)} \cap L_d(G_{loc,2})) \text{ se } d \in D_{loc,2} - D_{loc,1}\\
= {} & (\overline{L_d(M)} \cap L_d(G_{loc,1})) \parallel (\overline{L_d(N)} \cap L_d(G_{loc,2}))\\
& \qquad \text{se } d \in D_{loc,1} \cap D_{loc,2},\\
& (\overline{L_d(M)} \cap L_d(G_{loc,1})) \parallel (\overline{L_{Dloc,2}(N)} \cap \overline{L_{Dloc,2}(G_{loc,2})})\\
& \qquad \text{se } d \in D_{loc,1} - D_{loc,2},\\
& (\overline{L_{Dloc,1}(M)} \cap \overline{L_{Dloc,1}(G_{loc,1})}) \parallel (\overline{L_d(N)} \cap L_d(G_{loc,2}))\\
& \qquad \text{se } d \in D_{loc,2} - D_{loc,1}\\
= {} & (\overline{L_d(M)} \parallel \overline{L_d(N)}) \cap (L_d(G_{loc,1}) \parallel L_d(G_{loc,2}))\\
& \qquad \text{se } d \in D_{loc,1} \cap D_{loc,2},\\
& (\overline{L_d(M)} \parallel \overline{L_{Dloc,2}(N)}) \cap (L_d(G_{loc,1}) \parallel \overline{L_{Dloc,2}(G_{loc,2})})\\
& \qquad \text{se } d \in D_{loc,1} - D_{loc,2},\\
& (\overline{L_{Dloc,1}(M)} \parallel \overline{L_d(N)}) \cap (\overline{L_{Dloc,1}(G_{loc,1})} \parallel L_d(G_{loc,2}))\\
& \qquad \text{se } d \in D_{loc,2} - D_{loc,1}
\end{aligned}
$$

$$= L_d(\overline{\mathrm{M}} \parallel \overline{\mathrm{N}}) \cap L_d(G_{loc,1} \parallel G_{loc,2})$$
$$= \overline{L_d(\mathrm{M} \parallel \mathrm{N})} \cap L_d(G). \qquad \blacklozenge$$

Proposição 35: Sejam as especificações $\mathrm{M} \subseteq Pwr(Pwr(\Sigma_{loc,1}^*) \times D_{loc,1})$ e $\mathrm{N} \subseteq Pwr(Pwr(\Sigma_{loc,2}^*) \times D_{loc,2})$ para $G_{loc,1}$ e $G_{loc,2}$, respectivamente. Sejam $B \subseteq D = D_{loc,1} \cup D_{loc,2}$, $B_1 = B \cap D_{loc,1}$ e $B_2 = B \cap D_{loc,2}$. Se *SupCSNB*(M, $G_{loc,1}$, B_1) e *SupCSNB*(N, $G_{loc,2}$, B_2) forem fortemente não-conflitantes e.r.a B, então *SupCSNB*(M || N, G, B) = *SupCSNB*(M, $G_{loc,1}$, B_1) || *SupCSNB*(N, $G_{loc,2}$, B_2).

Prova: De $\mathrm{M} \parallel \mathrm{N} \subseteq \mathrm{M} \parallel G_{loc,2} \subseteq \mathrm{M} \parallel G_{loc,1}^c$, tem-se *SupCSNB*(M || N, G, B) $\subseteq$ M || $G_{loc,1}^c$ e, portanto, *SupCSNB*(M || N, G, B) $\subseteq$ *SupCSNB*(M || N, G, B_1)

$$\subseteq SupCSNB(\mathrm{M} \parallel G_{loc,1}^c, G_{loc,1} \parallel G_{loc,1}^c, B_1)$$
$$\subseteq SupCSNB(\mathrm{M}, G_{loc,1}, B_1) \parallel G_{loc,1}^c.$$

Simetricamente, tem-se *SupCSNB*(M || N, G, B) $\subseteq$ *SupCSNB*(N, $G_{loc,2}$, B_2) || $G_{loc,2}^c$.
Daí, *SupCSNB*(M || N, G, B) $\subseteq$ *SupCSNB*(M, $G_{loc,1}$, B_1) || $G_{loc,1}^c$

$$\cap\ SupCSNB(\mathrm{N}, G_{loc,2}, B_2) \parallel G_{loc,2}^c$$
$$= SupCSNB(\mathrm{M}, G_{loc,1}, B_1) \parallel SupCSNB(\mathrm{N}, G_{loc,2}, B_2).$$

Como *SupCSNB*(M, $G_{loc,1}$, B_1) e *SupCSNB*(N, $G_{loc,2}$, B_2) são fortemente não-bloqueante e.r.a B_1 e B_2, respectivamente, e fortemente não-conflitantes e.r.a B, eles são fracamente não-conflitantes e.r.a D. Como também são controláveis e.r.a $G_{loc,1}$ e $G_{loc,2}$, respectivamente, *SupCSNB*(M, $G_{loc,1}$, B_1) || *SupCSNB*(N, $G_{loc,2}$, B_2) é controlável e.r.a G e fortemente não-bloqueante e.r.a B. Também está contido em M || N. Portanto, *SupCSNB*(M, $G_{loc,1}$, B_1) || *SupCSNB*(N, $G_{loc,2}$, B_2) $\subseteq$ *SupCSNB*(M || N, G, B).
Assim, *SupCSNB*(M, $G_{loc,1}$, B_1) || *SupCSNB*(N, $G_{loc,2}$, B_2) = *SupCSNB*(M || N, G, B). $\blacklozenge$

Os resultados acima podem ser generalizados para múltiplas especificações usando-se as seguintes extensões das definições de não-conflito forte e fraco. Diz-se que um conjunto de comportamento coloridos $\{\mathrm{M}_i \subseteq Pwr(Pwr(\Sigma_i^*) \times D_i),\ i = 1,...,m\}$ é fracamente não-conflitante e.r.a B sempre que

$$\|_{i=1,...,m} \overline{L_{B \cap Di}(\mathrm{M}_i)} = \overline{L_B(\|_{i=1,...,m} \mathrm{M}_i)}.$$

O conjunto é fortemente não-conflitante e.r.a B quando

$$\forall b \in B,\ L_b(\|_{i=1,...,m} \overline{\mathrm{M}_i}) = L_b(\overline{\|_{i=1,...,m} \mathrm{M}_i}).$$

Segundo os resultados apresentados nesta seção, é possível usar uma abordagem local para a síntese de supervisores modulares desde que a condição de não-conflito seja satisfeita. Propõe-se então uma metodologia de síntese que é resumida da seguinte forma.

Seja um conjunto de GMCs assíncronos representando o comportamento em malha aberta de um SEDMT. Seja a especificação dada por um conjunto de m comportamentos admissíveis genéricos, cada qual sincronizando alguns subsistemas da planta. Para $i = 1,...,m$, obtém-se a planta local $G_{loc,i}$ pela composição de todos os subsistemas que compartilham eventos com o respectivo comportamento admissível genérico. O conjunto de especificações locais $\{M_{loc,j} \subseteq Pwr(Pwr(\Sigma_{loc,j}^{*}) \times D_{loc,j}),\ i = 1,...,m\}$ é calculado pela composição de cada comportamento admissível genérico com sua planta local correspondente. Assumindo-se que as especificações locais sejam D–fechadas por construção, pode-se computar supervisores ótimos fortemente não-bloqueantes $S_{loc,i}$, tais que $\Lambda_D(S_{loc,i}/G) = SupCSNB(M_{loc,i}, G_{loc,i}, D_{loc,i})$. Se o conjunto $\{SupCSNB(M_{loc,i}, G_{loc,i}, D_{loc,i}),\ i = 1,...,m\}$ for fortemente não-conflitante e.r.a D, então a conjunção de $S_{loc,i}$ para $i = 1,...,m$ é minimamente restritiva e fortemente não-bloqueante e.r.a D, isto é, a arquitetura de controle modular local restringe a planta ao mesmo comportamento que um supervisor monolítico ótimo. Caso o conjunto $\{SupCSNB(M_{loc,i}, G_{loc,i}, D_{loc,i}),\ i = 1,...,m\}$ seja fortemente conflitante, deve-se resolver o conflito calculando, por exemplo, um coordenador.

Problema 5: O Sistema Flexível de Manufatura (SFM) apresentado na Figura 43 transforma blocos brutos e tarugos brutos em dois tipos de produtos: um bloco com um pino cônico no topo (Produto A) e um bloco com um pino cilíndrico pintado (Produto B). O SFM é composto de oito equipamentos: três esteiras C_1, C_2 e C_3, uma Fresa, um Torno, um Robô, uma Máquina de Pintura (MP) e uma Máquina de Montagem (MM). Os equipamentos são conectados através de depósitos unitários B_i, $i = 1,...,8$.

As setas na Figura 43 indicam o fluxo de peças inacabadas pelo SFM. Blocos brutos entram na esteira C1 (evento 11) e alcançam B_1 (evento 12). Tarugos brutos entram na esteira C2 (evento 21) e chegam em B_2 (evento 22). O Robô pega um bloco bruto de B_1 (31) e o coloca em B_3 (32) ou move um tarugo bruto de B_2 (33) para B_4 (34). A Fresa começa a processar um bloco de B_3 pelo evento 41 e retorna uma peça com forma geométrica e um buraco no topo pelo evento 42. O Torno pode fazer dois tipos de pinos com os tarugos de B_4: um pino cônico (eventos 51 e 52) ou pino cilíndrico (eventos 53 e 54). Na seqüência, o Robô move um bloco acabado de B_3 para B_5 (eventos 35 e 36), move um pino cônico de B_4 para B_6 (eventos 37 e 38) ou move um pino cilíndrico de B_4 para B_7 (eventos 39 e 30). A esteira C3 transporta o pino de B_7 para B_8 (eventos 71 e 72), onde o pino é pintado (eventos 81 e 82), e o retorna para B_7 (eventos 73 e 74). Finalmente, a Máquina de Montagem pega um bloco de B_5 (evento 61) e põe sobre ele um pino cônico de B_6 (evento 63), gerando um Produto A (evento 64), ou a MM insere um pino cilíndrico

pintado de B_7 no topo do bloco, retornando um Produto B (evento 66). Deve-se sintetizar uma lógica de controle que dê o maior grau de liberdade ao SFM evitando, porém, que ocorram *overflow* ou *underflow* de peças nos depósitos e assegurando que sempre seja possível manufaturar produtos dos tipos A e B. Para garantir a eficiência da produção, deseja-se também que a lógica de controle nunca impeça o SFM de eventualmente operar a Fresa e o Torno simultaneamente.

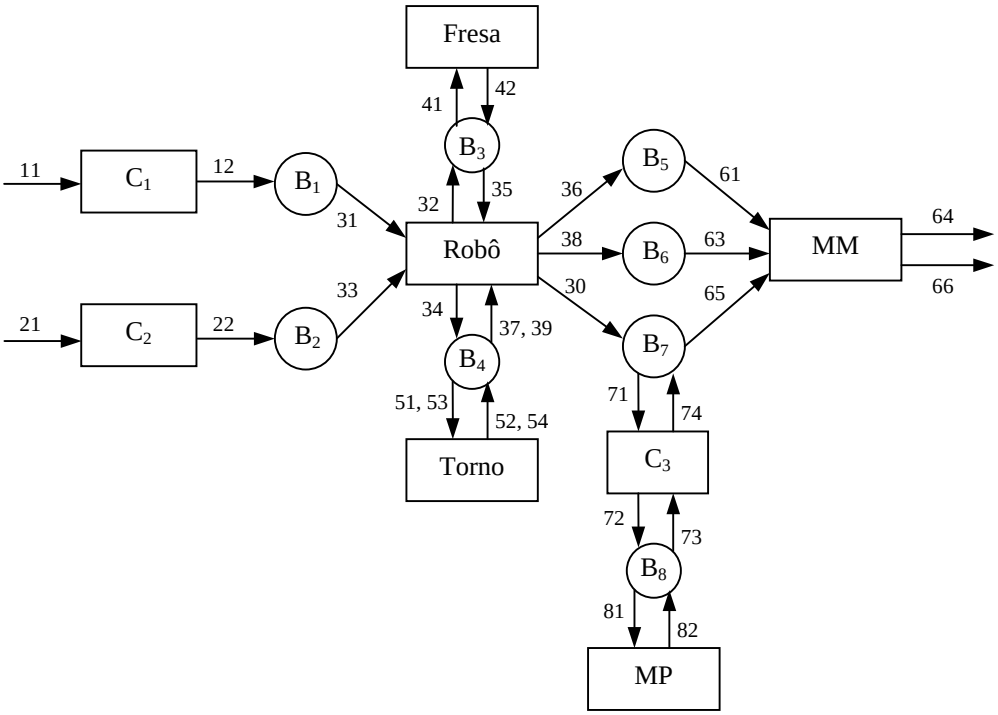


Figura 43: Sistema Flexível de Manufatura (SFM)

O comportamento em malha aberta do SFM é modelado pelo conjunto de oito GMCs assíncronos mostrados na Figura 44, onde os eventos representados por números ímpares são controláveis. A manufatura de um Produto A e de um Produto B é indicada respectivamente pelas tarefas **a** e **b** no modelo para a MM. A tarefa **o** representa a operação simultânea do Torno e da Fresa. As especificações genéricas de segurança $E_{gen,i}$, $i = 1,...,8$ para evitar *overflow* e *underflow* nos depósitos B_i, $i = 1,...,8$, respectivamente, são expressas pelas linguagens prefixo-fechadas geradas pelos autômatos na Figura 45. Essas especificações definem uma coordenação desejável para subconjuntos de eventos do alfabeto global da planta composta. Para assegurar que o SFM sempre seja capaz de gerar produtos dos tipos **a** e **b** e que sempre permita a operação paralela do Torno e da Fresa, o sistema controlado deve ser fortemente não-bloqueante em relação ao conjunto de cores $C = \{\mathbf{a}, \mathbf{b}, \mathbf{o}\}$.

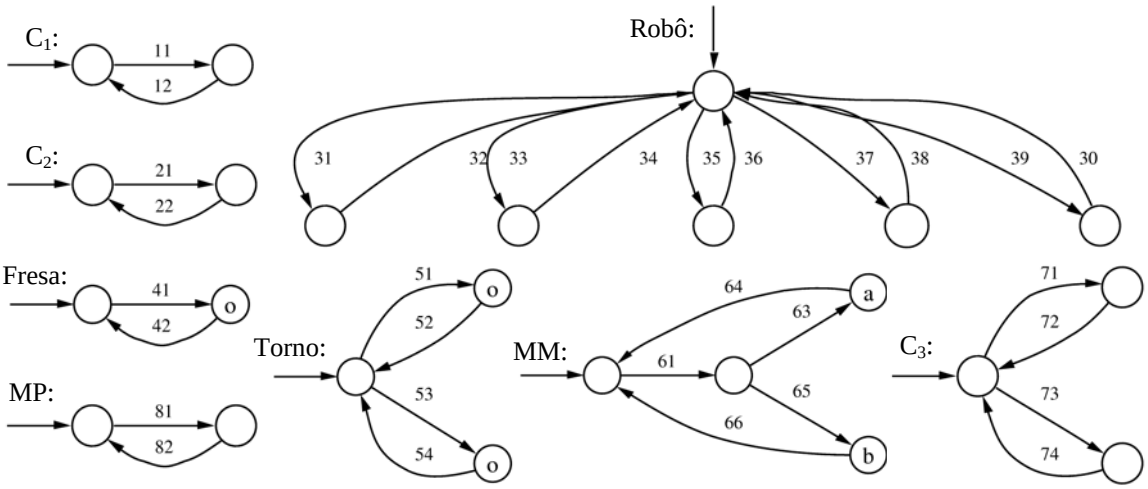


Figura 44: Planta composta para o SFM

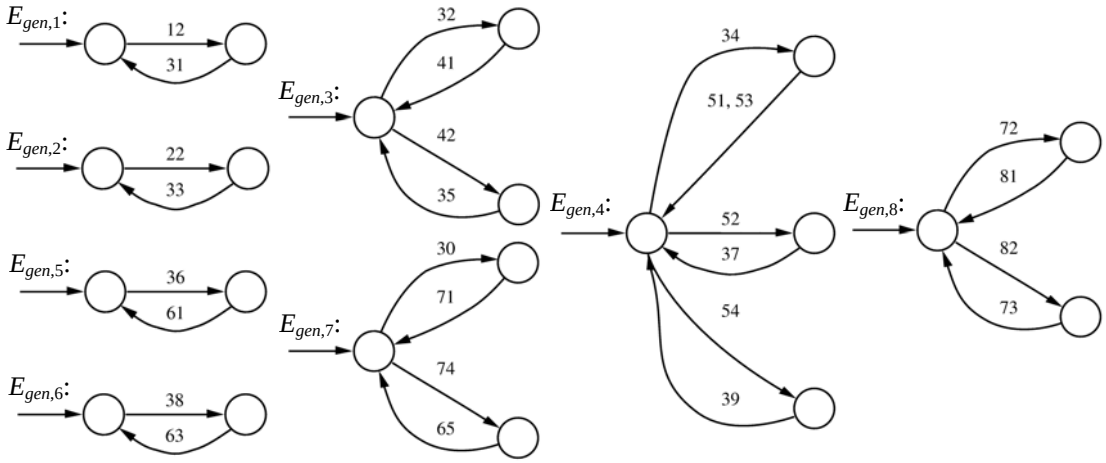


Figura 45: Especificações genéricas $E_{gen,i}$ respectivas aos depósitos B_i, $i = 1,...,8$

Para cada especificação genérica $E_{gen,i}$, $i = 1,...,8$, sua planta local $G_{loc,i}$ é obtida pela composição de todos os subsistemas afetados, isto é, subsistemas compartilhando eventos com a respectiva especificação. Os subsistemas afetados por cada especificação são indicados na Tabela 6. Como apenas os modelos para a Máquina de Montagem, o Torno e a Fresa possuem cores, as plantas locais $G_{loc,5}$, $G_{loc,6}$ e $G_{loc,7}$ possuem o mesmo conjunto de cores $D_{loc,5} = D_{loc,6} = D_{loc,7} = \{\mathbf{a}, \mathbf{b}\}$, enquanto $D_{loc,3} = D_{loc,4} = \{\mathbf{o}\}$ e as outras plantas locais são incolores. Para $i = 3,...,7$, no intuito de se representar as especificações $E_{gen,i}$ como subcomportamentos admissíveis $M_{loc,i}$ das respectivas plantas locais, calcula-se $M_{loc,i} = \{(E_{gen,i} \,||L_d(G_{loc,i}), d), \forall d \in D_{loc,i}\}$. Conforme discussão anterior (páginas 65 e 79), para $i = 1,2$ e 8, representa-se a especificação local $M_{loc,i}$ pelo comportamento colorido $M_{loc,i} = \{(E_{gen,i} \,||L(G_{loc,i}), \mathbf{v})\}$, onde a cor vácuo **v** é apenas um artifício para representar

uma especificação prefixo-fechada como um comportamento colorido. O número de estados dos geradores com marcação colorida $K_{loc,i}$ para as especificações locais $M_{loc,i}$, $i = 1,...,8$, são mostrados na Tabela 6.

Tabela 6: Número de estados dos geradores envolvidos na síntese de supervisores modulares locais reduzidos

i	$E_{gen,i}$	**Subsistemas afetados**	$G_{loc,i}$	$K_{loc,i}$	$H_{loc,i}$	$R_{loc,i}$
1	2	C_1, Robô	12	24	18	2
2	2	C_2, Robô	12	24	18	2
3	3	Fresa, Robô	12	26	18	3
4	4	Torno, Robô	18	34	21	4
5	2	MM, Robô	24	48	44	2
6	2	MM, Robô	24	48	44	2
7	3	C_3, MM, Robô	72	192	128	4
8	3	C_3, MP	6	11	6	3

Para as especificações locais $M_{loc,i}$, $i = 1,2$ e 8, calculam-se supervisores modulares locais (incolores) $H_{loc,i}$ tais que

$$\begin{aligned}\Lambda_{\{\mathbf{v}\}}(H_{loc,i}) &= SupCSNB(M_{loc,i}, G_{loc,i}, \{\mathbf{v}\}) \\ &= SupC(M_{loc,i}, G_{loc,i}) \\ &= \{(SupC(E_{gen,i} \,||L(G_{loc,i}), G_{loc,i}), \{\mathbf{v}\})\}.\end{aligned}$$

Para $i = 3$ e 4, obtêm-se supervisores incolores não-bloqueantes $H_{loc,i}$ tais que $\Lambda_{Dloc,i}(H_{loc,i}) = SupCSNB(M_{loc,i}, G_{loc,i}, \{\mathbf{o}\})$ e, para $i = 5$, 6 e 7, computam-se supervisores incolores $H_{loc,i}$ fortemente não-bloqueantes e.r.a $\{\mathbf{a}, \mathbf{b}\}$ tais que $\Lambda_{Dloc,i}(H_{loc,i}) = SupCSNB(M_{loc,i}, G_{loc,i}, \{\mathbf{a}, \mathbf{b}\})$. Usando o algoritmo de SU e WONHAM (2004), é possível reduzir o tamanho dos supervisores modulares locais para o máximo de 4 estados. A Tabela 6 mostra o número de estados dos supervisores antes e após a redução. O conjunto de supervisores modulares locais reduzidos $R_{loc,i}$, $i = 1,...,8$, é ilustrado na Figura 46, onde as setas tracejadas indicam os eventos a desabilitar nos estados de onde se originam.

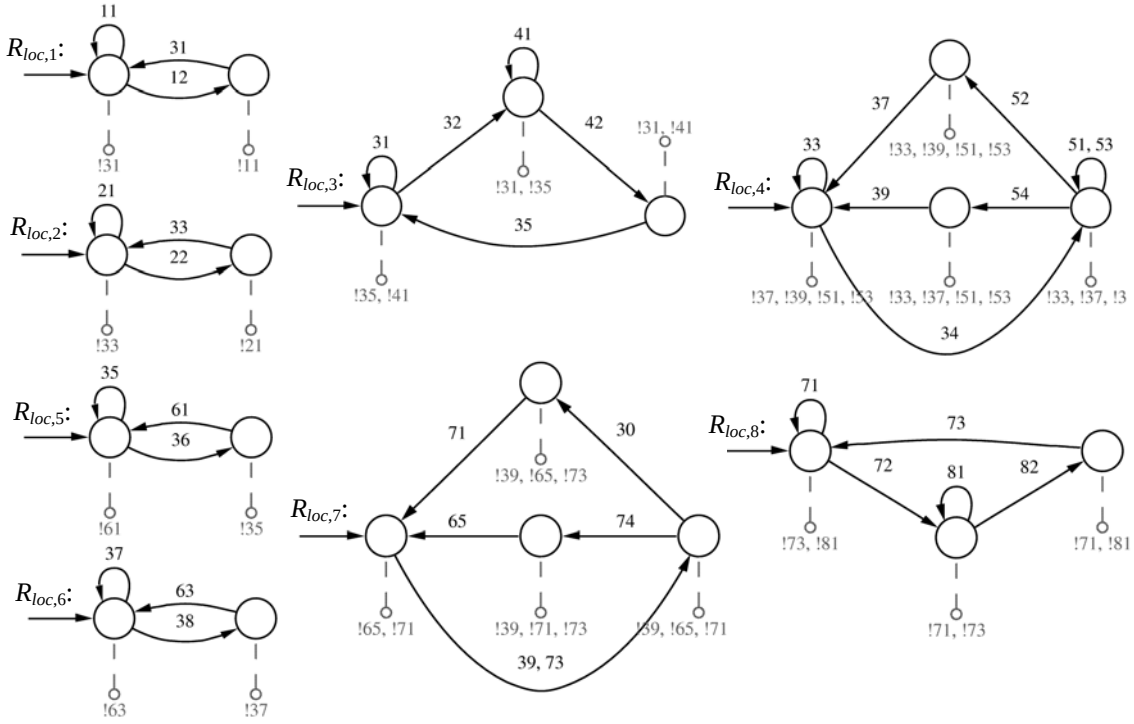


Figura 46: Supervisores locais reduzidos $R_{loc,i}$ para as especificações $E_{gen,i}$, i = 1,...,8

Para se verificar a vivacidade das tarefas **a**, **b** e **o** após a ação conjunta dos oito supervisores modulares locais, é preciso testar se eles são fortemente não-conflitantes e.r.a {**a**, **b**, **o**}. Para isso, calcula-se o GMC $H = ||_{i = 1,...,8} H_{loc,i}$ (com 70272 estados e 310452 transições). Ao verificar que *H* não é fracamente aparado e.r.a {**a**, **b**, **o**}, conclui-se que a supervisão modular é fracamente (e também fortemente) conflitante e.r.a {**a**, **b**, **o**}. Os cálculos foram feitos com a ajuda do CTCT (WONHAM, 2003), num *PC Pentium II* com 128MB de RAM. O tempo máximo de computação, referente ao teste de não-conflito forte, foi menor do que sete segundos.

Prossegue-se com a síntese calculando um coordenador que resolva o conflito global, de acordo com o esquema indicado na Seção 2.5.2. Para isso, toma-se como nova planta o GMC *H*, representando a conjunção conflitante dos supervisores modulares locais sobre a planta original. O gerador *HW* para o máximo comportamento contido em *H* que seja controlável e.r.a *H* e fracamente não-bloqueante e.r.a {**a**, **b**, **o**} tem 63936 estados e 281244 transições. Esse gerador é não-bloqueante para as tarefas **o** e **a**, mas pode bloquear a manufatura de produtos do tipo B. Tomando *HW* como um supervisor para *H*, tem-se um coordenador fracamente não-bloqueante para o SFM. Alternativamente, pode-se computar um gerador *HS* para o maior comportamento contido em *H* que seja controlável e.r.a *H* e fortemente não-bloqueante e.r.a {**a**, **b**, **o**}, o qual possui 45504 estados e 200124

transições. O gerador *HS* pode ser considerado um coordenador fortemente não-bloqueante para *H*.

Para evitar a complexidade da redução dos coordenadores *HW* e *HS*, procura-se resolver o conflito numa seção mais localizada do problema. Raciocinando sobre o problema, observa-se que o conflito que bloqueia a geração de Produto B está relacionada com o fluxo de pinos. Com isso, calcula-se o gerador *H2*, com 1912 estados e 6958 transições, como a composição de todos os supervisores relacionados com o fluxo de pinos: $H2 = H_{loc,4} \parallel H_{loc,6} \parallel H_{loc,7} \parallel H_{loc,8}$. A partir de *H2* calcula-se um coordenador *H2W* fracamente não-bloqueante e.r.a {**a**, **b**, **o**}, com 1720 estados, e um coordenador *H2S*, com 1256 estados, que é fortemente não-bloqueante e.r.a {**a**, **b**, **o**}. Operando um algoritmo para redução de supervisores em Matlab por 42 minutos, obtém-se de *H2W* o simples coordenador fracamente não-bloqueante *CW* da Figura 47. O mesmo algoritmo leva 18 minutos para gerar a partir de *H2S* o coordenador fortemente não-bloqueante *CS* com apenas 2 estados, mostrado na Figura 48. Pode-se verificar que *CW* || *H* é equivalente a *HW* e *CS* || *H* é equivalente a *HS*. Portanto, os coordenadores reduzidos *CW* e *CS* resolvem o conflito global (fraco e forte, respectivamente) de forma minimamente restritiva.

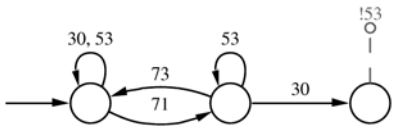


Figura 47: Coordenador fracamente não-bloqueante *CW* para o SFM

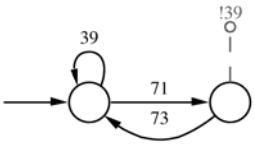


Figura 48: Coordenador fortemente não-bloqueante *CS* para o SFM

5.3 Conclusão do capítulo

Este capítulo estende os resultados de controle supervisório multitarefa para uma arquitetura de controle modular de forma a proporcionar uma metodologia eficiente para a solução de problemas envolvendo SEDMTs compostos de múltiplos subsistemas e múltiplas especificações. Os resultados apresentados na primeira seção permitem dividir o problema de controle global em problemas menores, cada qual tratando uma especificação

separadamente. O não-bloqueio forte e a otimidade da composição das soluções modulares estão condicionados à propriedade de não-conflito forte.

A abordagem proposta na seção seguinte permite a síntese de supervisores fortemente não-bloqueantes sobre uma versão local da planta. Essa característica restringe a complexidade de cada problema local à escala da respectiva planta local, que pode ser muito menor do que a planta global em problemas reais. Para sistemas de grande porte, a maior complexidade recai sobre a verificação de não-conflito e a eventual síntese de coordenador. Esta complexidade, no entanto, pode ser menor do que a síntese de um supervisor monolítico global.

Finalmente, a solução de um problema com múltiplos objetivos de controle para um sistema flexível de manufatura, evidencia a conveniência da metodologia proposta. Esse exemplo destaca também a importância da verificação e resolução de conflitos para a viabilidade da abordagem modular. No caso do SFM, o uso de marcações coloridas contribuiu para o entendimento do problema de conflito, que pôde ser resolvido sobre uma parte localizada do sistema. Porém, em geral, o conflito é um problema global. Assim, para poder tratar com sistemas de grande porte, é fundamental que se desenvolvam métodos eficientes para identificar e resolver tal problema. Uma alternativa nesse sentido é o uso de interfaces que evitam a ocorrência de conflitos, como proposto por LEDUC (2002). Outra alternativa interessante é explorar estruturas como *Binary Decision Diagrams* (BRYANT, 1986) e *Integer Decision Diagrams* (ZHANG e WONHAM, 2001) para manipular eficientemente geradores com grande número de estados.

A resolução do sistema flexível de manufatura mostra ainda que os algoritmos para redução de supervisores, apesar de sua grande complexidade, podem ser explorados na abordagem modular local para obter soluções com poucos estados. A extensão desses algoritmos para supervisores pintores com múltiplas marcações é um tema interessante para pesquisas futuras.

6. Conclusão Final e Perspectivas

A soma dos resultados teóricos e práticos apresentados na presente Tese de Doutorado constitui uma metodologia viável para síntese de supervisores ótimos para problemas de controle envolvendo múltiplas tarefas e especificações sobre SEDs compostos. O uso de geradores com marcação colorida permite distinguir as classes de tarefas completas no processo de síntese de supervisor, de forma que é possível obter por um algoritmo iterativo a solução minimamente restritiva que evita transgredir as especificações e garante a vivacidade de cada objetivo de controle. Já a abordagem de controle modular local permite explorar a arquitetura modular das especificações e da planta em sistemas compostos para evitar explosão exponencial do número de estados do modelo global na síntese e implementação de supervisores modulares reduzidos. A solução teórica e a implementação prática do sistema de controle para a célula de manufatura servem de referência para futuras aplicações da metodologia.

Assim, os resultados deste trabalho podem ser resumidos na forma de três principais contribuições à Teoria de Controle Supervisório: a consolidação da metodologia de controle modular local proposta no trabalho de mestrado (QUEIROZ, 2000), o desenvolvimento de uma nova abordagem para controle multitarefa de SEDs e a composição da abordagem modular local com o novo modelo multitarefa. A seguir, faz-se uma discussão sobre as principais vantagens dos resultados, bem como sobre suas limitações, que abrem espaço para novas pesquisas na área de controle supervisório de sistemas a eventos discretos.

6.1 Controle modular local

O uso da arquitetura modular é uma forma eficiente de se representar com clareza sistemas complexos em problemas em que sistemas a eventos discretos com muitos estados possam ser representados pela composição de múltiplos módulos com poucos estados. Na direção inversa, o número de estados do modelo monolítico, que representa a interação de todos os módulos, pode crescer exponencialmente com o número de módulos. A abordagem modular local é aplicada a problemas em que tanto o modelo da planta quanto o modelo das especificações são representados de forma modular. Essa abordagem permite a obtenção de um supervisor ótimo para cada especificação fazendo apenas a composição

dos subsistemas diretamente afetados por ela. Evita-se assim a composição das especificações e o uso do modelo monolítico da planta na síntese de supervisor para especificações locais, isto é, que afetem apenas um subconjunto dos módulos da planta. Portanto, a abordagem modular local evita a composição de módulos o que reduz a complexidade computacional da síntese dos supervisores modulares bem como o tamanho dos supervisores resultantes.

A combinação da abordagem modular local com algoritmos de redução de supervisores é uma técnica que promove maior clareza às soluções de controle. Naturalmente, um supervisor com poucos estados resolvendo um problema particular pode ser bem mais compreensível ao projetista do que um supervisor imenso para todas as especificações de controle. No entanto, a complexidade dos algoritmos de redução de supervisores limita o tamanho dos supervisores sobre os quais podem ser aplicados. O programa CTCT em sua versão mais recente (WONHAM, 2003), por exemplo, normalmente não permite reduzir supervisores com mais de 1000 estados. O fato de o tamanho dos supervisores modulares locais ser proporcional ao tamanho das plantas locais, que não dependem do modelo global, pode viabilizar a redução de supervisores em problemas de grande porte.

Cada supervisor modular local, em sua ação de controle isolada, desabilita eventos controláveis dos subsistemas de sua planta local para evitar transgredir a respectiva especificação e ocorrência de bloqueio local, de forma minimamente restritiva. Quando múltiplos supervisores agem em conjunto, evita-se transgredir todas as especificações pela desabilitação de eventos controláveis, porém perde-se a garantia de não-bloqueio visto que a ação conjunta pode gerar conflito. Portanto, o bloqueio é em geral um problema global, ou seja, deve-se considerar a interação de todos os supervisores e todos os subsistemas afetados para verificar a existência de bloqueio global (conflito). É por essa razão que a complexidade do teste de modularidade local, como condição necessária e suficiente para não-conflito, cresce exponencialmente com o número de especificações e subsistemas. Pela mesma razão, a resolução de conflitos é também um problema de complexidade global.

Por conseguinte, o teste de modularidade e a resolução de conflitos são os principais desafios para a aplicação da abordagem modular local na resolução de problemas de grande porte. Essa limitação pode motivar novas pesquisas na busca de metodologias para verificação e tratamento eficientes de conflito. Uma alternativa interessante é explorar o uso de estruturas eficientes, como *Binary Decision Diagrams* (BRYANT, 1986), para manipulação de autômatos com muitos estados. Outra alternativa que pode ser investigada é a combinação sucessiva de sincronização e encapsulamento por

projeção, como sugerido por WONHAM (2003, p. 128), para a resolução iterativa do conflito.

Outra limitação importante da abordagem modular local é que a planta deve ser representada na forma de um conjunto de subsistemas completamente assíncronos, denominado sistema-produto. A sincronização entre os módulos é feita exclusivamente pela ação dos supervisores. Isso garante que cada problema local possa ser resolvido isoladamente sobre o conjunto dos subsistemas diretamente afetados. Essa estrutura é encontrada em muitos problemas reais, porém é fato que a abordagem se aplica diretamente apenas a uma classe particular de sistemas compostos. Em geral, fazendo-se a composição dos subsistemas síncronos pode-se representar qualquer sistema composto na forma de sistema-produto. Contudo, a síntese modular local teria menor complexidade se essa composição pudesse ser evitada. Algumas pesquisas recentes (BRANDIN *et al.*, 2000; ÅKESSON *et al.*, 2002) exploram o fato de que a controlabilidade de linguagens se preserva pela sincronização de geradores com a planta para evitar a composição de subsistemas na síntese de supervisores locais em sistemas compostos. Entretanto, esses trabalhos não consideram a questão do bloqueio. Parece natural que a abordagem modular local possa ser estendida para sistemas compostos em geral se houver uma forma de verificar e resolver o bloqueio global (conflito). Essa questão merece ser melhor investigada. No intuito de aumentar o domínio de uso da metodologia, pode-se investigar também a extensão da abordagem modular local para SEDs temporizados (BRANDIN e WONHAM, 1993).

O uso bem sucedido do controle supervisório modular local para a solução do problema da célula de manufatura com mesa giratória é um resultado fundamental para consolidação dessa abordagem. Esse exemplo real, apesar de simples, serve de modelo e motivação para novas aplicações da Teoria de Controle Supervisório. As dificuldades práticas encontradas no processo de implementação física do sistema de controle correspondente à solução teórica obtida motivaram a proposta de uma nova metodologia para implementação estruturada de controle supervisório. A estrutura proposta tem a vantagem de preservar as características modulares dos supervisores reduzidos e da planta, e de abstrair os detalhes operacionais do modelo da planta usado na síntese de supervisores. Para sua consolidação efetiva, é importante que esse novo modelo genérico de implementação seja aplicado à resolução de diversos problemas reais, de preferência em diferentes ambientes de programação do sistema de controle. Nessa linha, outras perspectivas para pesquisas futuras são o desenvolvimento de programa para geração

automática do código de controle e a extensão da arquitetura para implementação distribuída do sistema de controle.

Além da clareza dos supervisores, a abordagem modular local proporciona flexibilidade e segurança ao sistema de controle. Se houver mudança em alguma especificação (ou subsistema) particular, pode-se atualizar o sistema de controle alterando apenas o(s) supervisor(es) correspondente. Nesse caso, deve-se ter o cuidado de refazer o teste de modularidade e eventualmente alterar o esquema para resolução de conflitos. No caso de falha de um supervisor, as outras especificações são preservadas pela ação dos demais supervisores. Pode-se inclusive projetar um sistema de controle que possua supervisores redundantes para as mesmas especificações.

6.2 Controle multitarefa

Nesta Tese, os geradores com marcação colorida foram introduzidos como um modelo que permite identificar de forma conveniente os diversos objetivos de controle em problemas envolvendo múltiplas classes de tarefas. A definição de comportamento colorido como forma de representar o conjunto de linguagens marcadas por um GMC mostrou-se também bastante adequada para o desenvolvimento formal da teoria que fundamenta a abordagem de controle multitarefa.

A composição síncrona de GMCs foi concebida de forma a preservar as informações das tarefas completas nos subsistemas, o que promove melhor entendimento dos estados dos modelos compostos. Uma vez que a composição de GMCs sincroniza as cores compartilhadas, é possível fazer a definição de tarefas globais nos próprios estados dos subsistemas. Esse aspecto é bastante conveniente para a modelagem de sistemas compostos, visto que permite que a estrutura modular da planta seja preservada para a síntese de supervisores.

Com a distinção das classes de tarefas completas, foi possível refinar o conceito de não-bloqueio para os casos em que todos os objetivos de controle são sempre alcançáveis (forte) e para os casos em que sempre existe ao menos uma tarefa alcançável (fraco). Em analogia à abordagem clássica de controle supervisório com marcação única, o modelo da planta pode ser representado por um GMC, as restrições de controle (especificações de segurança) podem ser representadas na forma de um comportamento colorido admissível e a vivacidade do sistema pode ser especificada pela imposição de não-bloqueio forte a um conjunto de tarefas relevantes. Claramente, tal especificação pode ser mais restritiva do que não-bloqueio em relação a uma única marcação representando qualquer tarefa

completa e menos restritiva do que não-bloqueio em relação a uma marcação para a finalização simultânea de todas as tarefas. Portanto, a solução de controle multitarefa é potencialmente mais refinada do que o controle supervisório clássico.

Esse fato não implica que o uso de múltiplas marcações leva necessariamente a um sistema de controle mais restritivo. Há casos em que a vivacidade de múltiplas tarefas pode ser garantida pelo não-bloqueio de uma única tarefa e, assim, pode-se obter equivalente solução ótima usando uma modelagem com marcação única. Entretanto, a identificação de tal tarefa antes de se conhecer a solução de controle pode ser muito difícil.

Como apresentado na Seção 4.5, a reversibilidade do sistema controlado garante a vivacidade das tarefas alcançáveis. Quando imposta ao problema, a reversibilidade pode ser tratada como uma tarefa única que substitui a especificação de não-bloqueio forte em relação a um certo conjunto de cores, no qual ela se inclui. Essa propriedade foi explorada na Seção 4.5 para simplificar o cálculo de supervisor. Observa-se, no entanto, que a imposição de reversibilidade pode ser demasiadamente restritiva em alguns casos, como no Problema 4. Como conseqüência, essa abordagem particular não pode ser usada como metodologia geral para problemas multitarefa.

Uma metodologia formal para solução geral do problema de controle supervisório multitarefa foi desenvolvida na Seção 4.3. A definição do conceito de supervisor pintor permite fazer uma distinção clara entre o tratamento de tarefas definidas pela planta ou introduzidas pelo supervisor. Como o supervisor pintor não pode agir sobre a sinalização de ocorrência de tarefas da planta, a condição de *D*–fechamento para existência de supervisor deve ser verificada na especificação para as respectivas cores. Essa condição não é necessária para as cores definidas pelo supervisor pintor. Além do *D*–fechamento e da controlabilidade, o não-bloqueio forte do comportamento colorido admissível é condição fundamental para existência de supervisor pintor fortemente não-bloqueante. Essa propriedade só tem sentido prático em problemas envolvendo múltiplas tarefas.

O algoritmo para síntese de supervisor pintor fortemente não-bloqueante ótimo para um dado comportamento colorido admissível foi formalmente apresentado na Seção 4.3.4. Esse algoritmo foi aplicado com sucesso na resolução de três problemas acadêmicos. Para viabilizar sua aplicação a problemas reais, é imprescindível que o algoritmo seja implementado junto com ferramentas para manipulação de GMCs em ambiente computacional. Um estudo de complexidade computacional deve acompanhar esse trabalho. Uma ferramenta muito útil para controle multitarefa que pode ser desenvolvida no âmbito de uma pesquisa futura é a redução de supervisores pintores.

6.3 Controle modular local multitarefa

Finalmente, a abordagem modular foi combinada com a abordagem multitarefa no Capítulo 5. O conceito de não-conflito forte surgiu naturalmente como condição para preservação do não-bloqueio forte pela composição de supervisores pintores. A abordagem modular local foi usada para restringir a síntese modular de supervisores pintores locais aos subsistemas da planta diretamente afetados pela respectiva especificação. A metodologia proposta foi elucidada pela síntese de uma solução modular de controle compreensível para um sistema flexível de manufatura. Esse exemplo mostrou também que as marcações coloridas facilitam o entendimento do bloqueio global e, portanto, podem auxiliar no processo de resolução de conflito. A extensão do modelo multitarefa para uma arquitetura hierárquica (ZHONG e WONHAM, 1990; CUNHA e CURY; 2002; TORRICO e CURY; 2002) em conjunção com a modular é um tema interessante para próximas pesquisas. Nessa arquitetura, as cores do nível operacional poderiam, por exemplo, definir o alfabeto do nível gerencial.

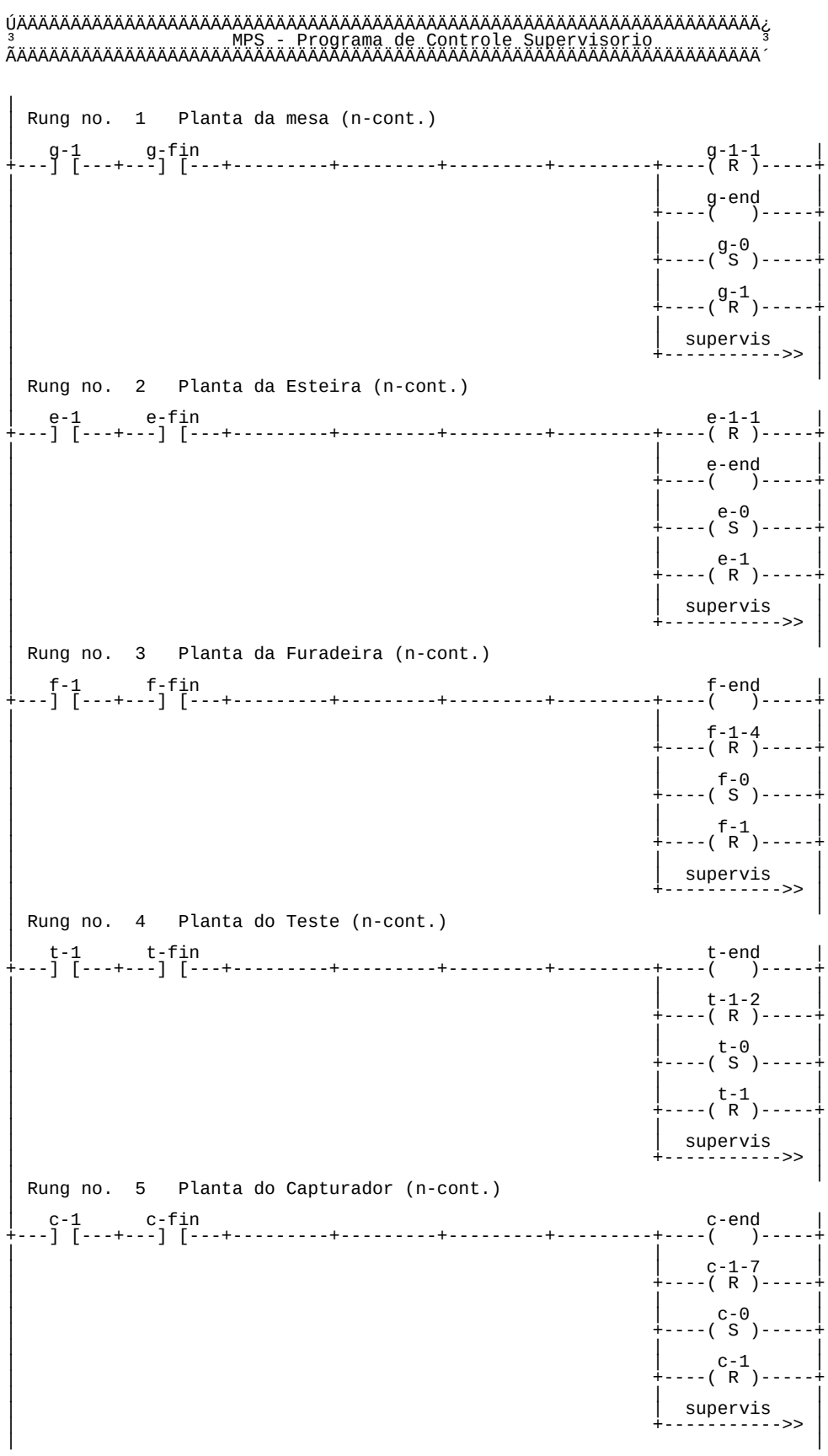
Conclui-se que, com a introdução de uma nova abordagem para controle modular multitarefa de sistemas a eventos discretos compostos, a presente Tese de Doutorado executou algumas transições na direção de levar a teoria de controle supervisório a um estado em que possa ser eficientemente aplicada aos problemas complexos do mundo real. Muitas outras transições ainda devem ocorrer para que esta seja considerada uma tarefa completa.

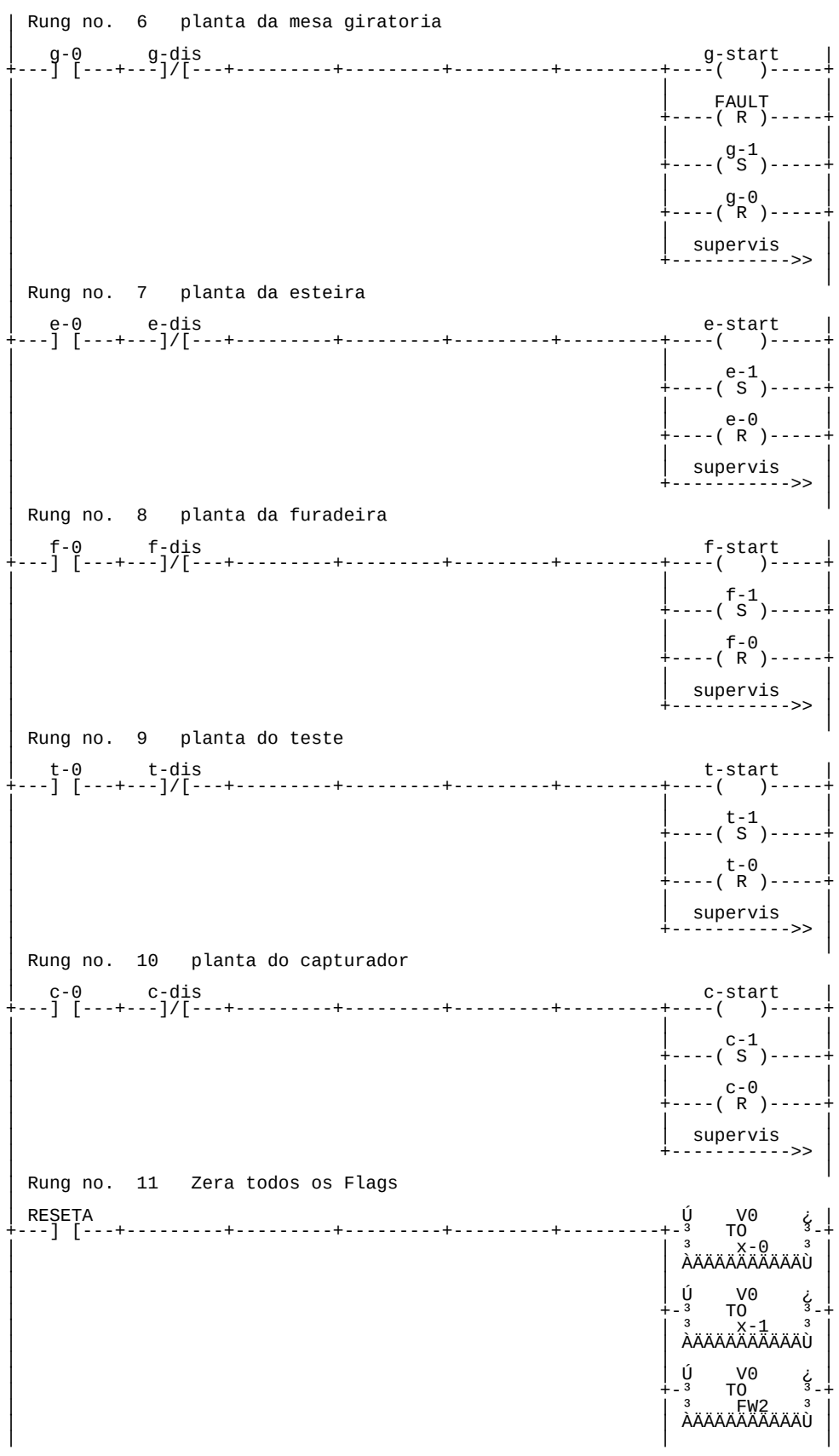
Anexo 1. Programa de Controle da Célula de Manufatura

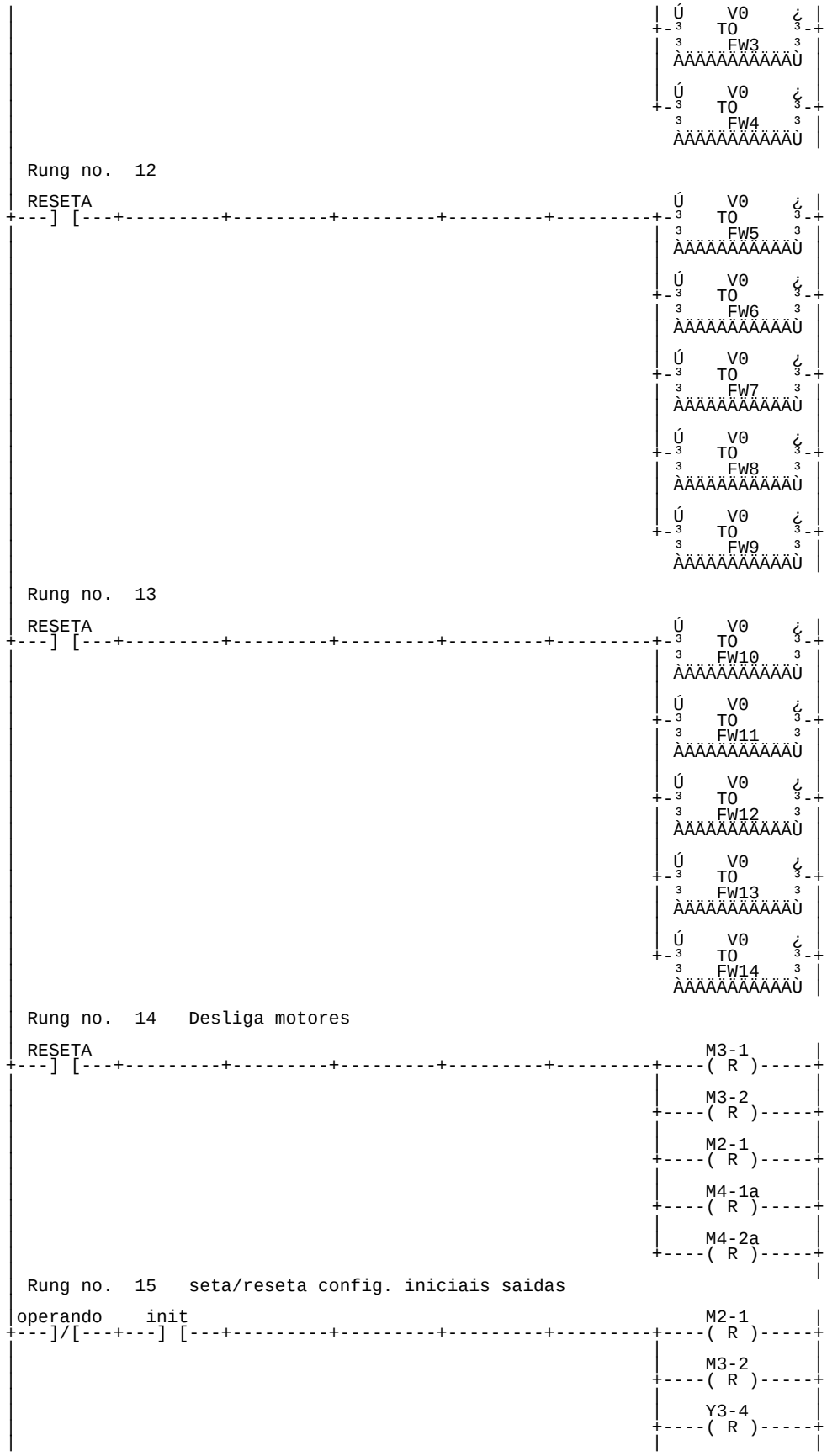
```
ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿
³                     FESTO  Software  Tool   FST                    ³
Ã====================================================================´
³                     MPS - Programa de Controle Supervisorio        ³
Ã====================================================================´
³                     Autor: Max Hering de Queiroz                   ³
ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ´


  Operadores da lista de alocacao
  Absoluto   Simbolico        Comentario

  O0.0       Y3-1             levanta furadeira
  O0.1       Y3-2             abaixa furadeira
  O0.2       Y3-3             avanca cilindro de teste
  O0.3       M3-1             liga furadeira
  O0.4       Y3-4             avanca o cilindro fixador
  O0.5       M3-2             liga o motor da mesa
  O0.6       M2-1             liga esteira
  O10.0      Y4-1             manipulador para baixo
  O10.1      Y4-2             manipulador para cima
  O10.2      Y4-3             vacuo on
  O10.3      Y4-4             vacuo off
  O10.4      M4-1a            manipulador esquerda/direita ligado
  O10.5      M4-1b            manipulador esquerda(0)/direita(1)
  O10.6      M4-2a            manipulador frente/tras ligado
  O10.7      M4-2b            manipulador frente(0)/tras(1)
  I0.0       B3-1             furadeira levantada
  I0.2       B3-3             cilindro de teste levantado
  I0.3       B3-4             cilindro de teste abaixado
  I0.4       B3-5             cilindro fixador recuado
  I0.5       B3-6             cilindro fixador avancado
  I0.6       B3-7             mesa giratoria em posicao
  I0.7       B3-8             Peca na posicao P1
  I1.0       init             botao de inicializacao
  I1.1       e-peca           botao indicando peca na esteira
  I1.2       RESETA           botao RESET
  I10.1      B4-8             longitude dos magazines
  I10.2      B4-3             latitude mesa/magazineOK
  I10.3      B4-7             longitude da mesa
  I10.4      B4-4             latitude magazineNOK
  I10.5      S4-1             vacuo prendendo peca
  I10.6      B4-2             manipulador abaixado
  I11.6      B4-1             manipulador levantado
  F0.1       e-0              esteira parada
  F0.2       g-0              mesa parada
  F0.3       f-0              furadeira parada
  F0.4       t-0              teste parado
  F0.5       c-0              capturador parado
  F1.1       e-1              esteira operando
  F1.2       g-1              mesa girando 90 graus
  F1.3       f-1              furadeira operando
  F1.4       t-1              testando
  F1.5       c-1              capturando peca da mesa
  F2.1       e-dis            desabilita a esteira
  F2.2       g-dis            desabilita o giro
  F2.3       f-dis            desabilita a furadeira
  F2.4       t-dis            desabilita o teste
  F2.5       c-dis            desabilita a captura
  F2.6       f-dis1           supervisor 1 desabilita furadeira
  F2.7       f-dis2           supervisor 2 desabilita furadeira
  F2.8       t-dis2           supervisor 2 desabilita teste
  F2.9       t-dis3           supervisor 3 desabilita teste
  F2.10      g-dis1           supervisor 1 desabilita giro da mesa
  F2.11      g-dis2           supervisor 2 desabilita giro da mesa
  F2.12      g-dis3           supervisor 3 desabilita giro da mesa
  F2.13      g-dis4           supervisor 4 desabilita giro da mesa
  F3.1       e-start          sinaliza ao sup. o inicio da esteira
  F3.2       g-start          sinaliza ao sup. o inicio do giro
  F3.3       f-start          sinaliza ao sup. o inicio da furadei
  F3.4       t-start          sinaliza ao sup. o inicio do teste
  F3.5       c-start          sinaliza ao sup. o inicio da captura
  F4.1       e-end            sinaliza ao sup. o final da esteira
  F4.2       g-end            sinaliza ao sup. o final do giro
  F4.3       f-end            sinaliza ao sup. o final da furadeir
```

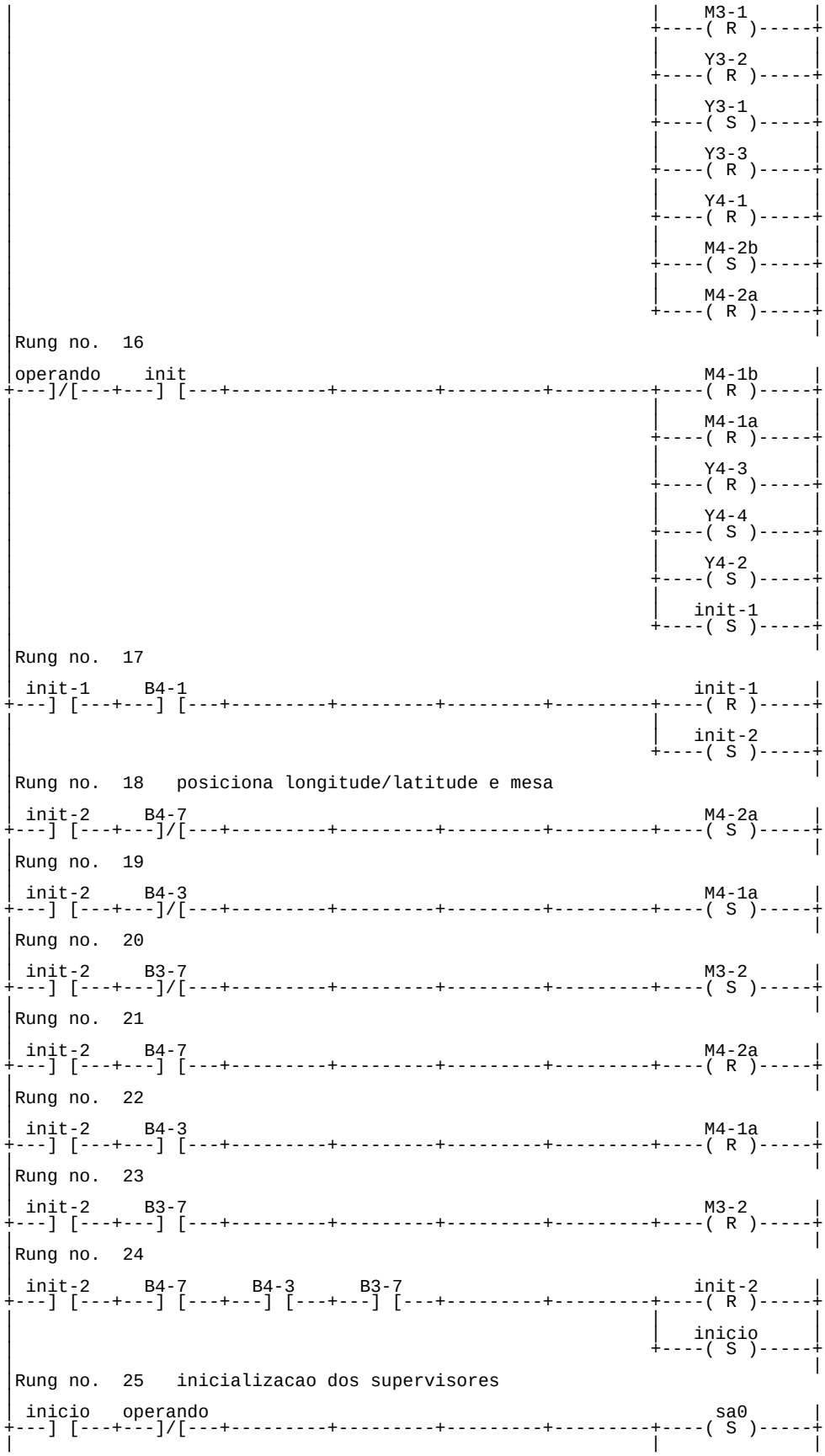
```
  F4.4       t-end            sinaliza ao sup. o final do teste
  F4.5       c-end            sinaliza ao sup. o final da captura
  F5.0       sa0              supervisor a - estado 0
  F5.1       sa1              supervisor a - estado 1
  F6.0       sb10             supervisor b1 - estado 0
  F6.1       sb11             supervisor b1 - estado 1
  F6.2       sb20             supervisor b2 - estado 0
  F6.3       sb21             supervisor b2 - estado 1
  F6.4       sb30             supervisor b3 - estado 0
  F6.5       sb31             supervisor b3 - estado 1
  F6.6       sb40             supervisor b4 - estado 0
  F6.7       sb41             supervisor b4 - estado 1
  F7.0       sc10             supervisor c1 - estado 0
  F7.1       sc11             supervisor c1 - estado 1
  F7.2       sc12             supervisor c1 - estado 2
  F7.3       sc13             supervisor c1 - estado 3
  F7.4       sc20             supervisor c2 - estado 0
  F7.5       sc21             supervisor c2 - estado 1
  F7.6       sc22             supervisor c2 - estado 2
  F7.7       sc23             supervisor c2 - estado 3
  F7.8       sc30             supervisor c3 - estado 0
  F7.9       sc31             supervisor c3 - estado 1
  F7.10      sc32             supervisor c3 - estado 2
  F7.11      sc33             supervisor c3 - estado 3
  F8.1       g-1-1            1o. passo da mesa
  F8.2       e-1-1            1o. passo da esteira
  F9.1       f-1-1            1o. passo da furadeira
  F9.2       f-1-2            2o. passo da furadeira
  F9.3       f-1-3            3o. passo da furadeira
  F9.4       f-1-4            4o. passo da furadeira
  F10.1      t-1-1            1o. passo do teste
  F10.2      t-1-2            2o. passo do teste
  F11.0      c-1-1            1o. passo da captura
  F11.1      c-1-2            2o. passo da captura
  F11.2      c-1-3            3o. passo da captura
  F11.3      c-1-4            4o. passo da captura
  F11.4      c-1-5            5o. passo da captura
  F11.5      c-1-6            6o. passo da captura
  F11.6      c-1-7            7o. passo da captura
  F12.1      g-fin            fim do giro da mesa
  F12.2      e-fin            fim de operacao da esteira
  F12.3      f-fin            fim de operacao da furadeira
  F12.4      t-fin            fim do teste
  F12.5      c-fin            fim da captura de uma peca na mesa
  F13.0      init-1           1o. passo da inicializacao
  F13.1      init-2           2o. passo da inicializacao
  F14.0      fault-P3         peca falhada em P3
  F14.1      operando         programa em andamento
  F14.2      inicio           inicializacao do sistema supervisor
  F14.4      FAULT            peca falhada em P4
  F14.5      ERRO             sinal de erro no programa
  FW0        x-0              estados iniciais das plantas
  FW1        x-1              estados 1 das plantas
  FW2                         desabilitacoes
  FW3                         sinalizacoes de inicio de operacao
  FW4                         sinalizacoes de final de operacao
  FW5                         estados do supervisor a
  FW6                         estados dos supervisores bi
  FW7                         estados dos supervisores ci
  FW8                         passos de operacao da mesa/esteira
  FW9                         passos de operacao da furadeira
  FW10                        passos de operacao do teste
  FW11                        passos de operacao da captura
  FW12                        sinalizacoes de final sequencia op.
  FW13                        passos da inicializacao
  FW14                        extras
  T1         t-delay          tempo de teste
  T2         g-delay          tempo minimo de giro
  T3         f-delay          tempo de furacao
  T4         e-delay          tempo minimo da esteira
  T5         c-delay          tempo para prender a peca

ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿
³                    MPS - Programa de Controle Supervisorio          ³
ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ´


                 SISTEMA-PRODUTO

Rung No.     1   Planta da mesa (n-cont.)
Rung No.     2   Planta da Esteira (n-cont.)
Rung No.     3   Planta da Furadeira (n-cont.)
Rung No.     4   Planta do Teste (n-cont.)
Rung No.     5   Planta do Capturador (n-cont.)
Rung No.     6   planta da mesa giratoria
Rung No.     7   planta da esteira
Rung No.     8   planta da furadeira
Rung No.     9   planta do teste
Rung No.    10   planta do capturador
```

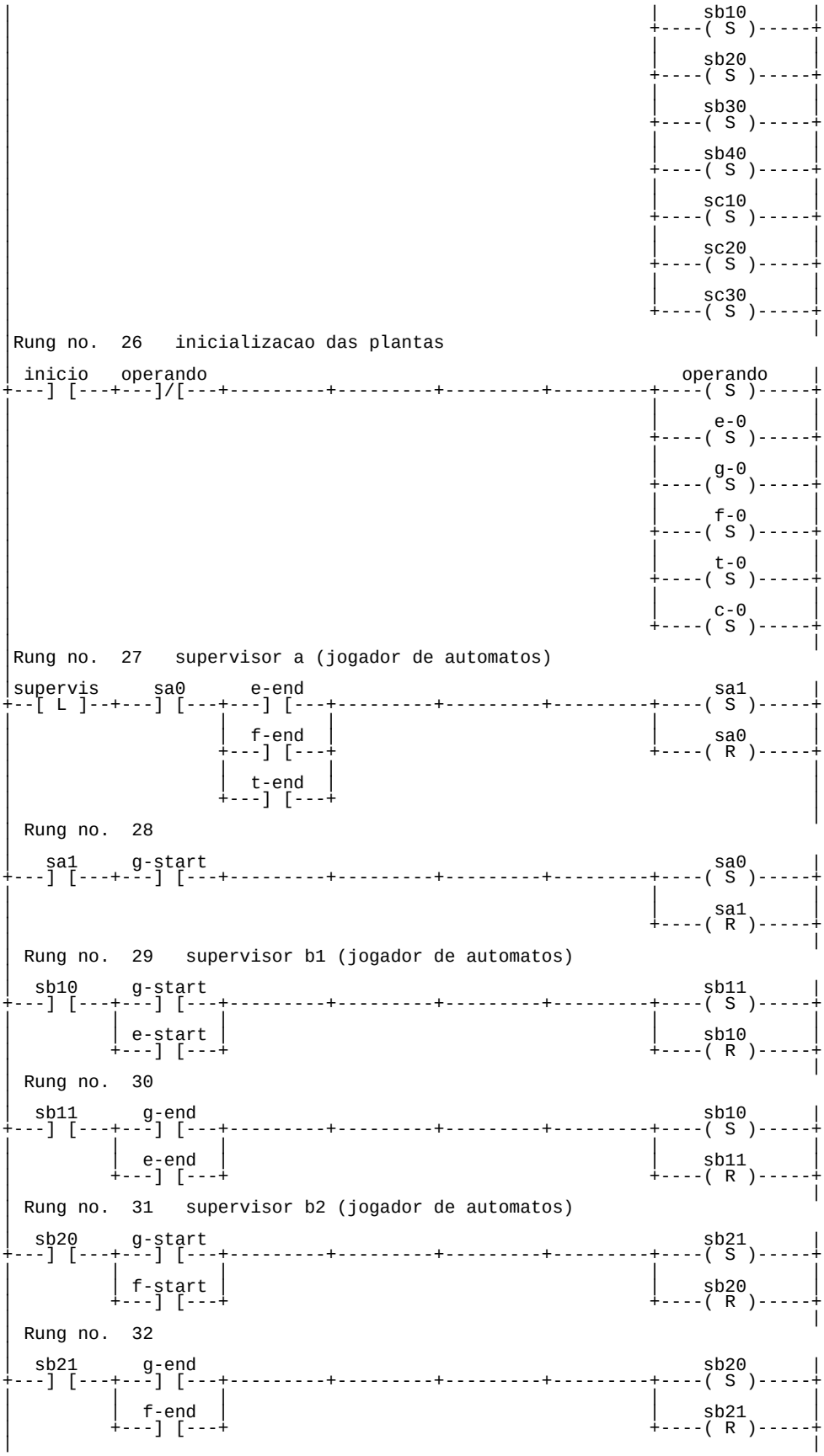
```
                    INICIALIZACAO

Rung No.     11     Zera todos os Flags
Rung No.     12
Rung No.     13
Rung No.     14     Desliga motores
Rung No.     15     seta/reseta config. iniciais saidas
Rung No.     16
Rung No.     17
Rung No.     18     posiciona longitude/latitude e mesa
Rung No.     19
Rung No.     20
Rung No.     21
Rung No.     22
Rung No.     23
Rung No.     24
Rung No.     25     inicializacao dos supervisores
Rung No.     26     inicializacao das plantas

                    SUPERVISORES MODULARES

Rung No.     27     supervisor a (jogador de automatos)
Rung No.     28
Rung No.     29     supervisor b1 (jogador de automatos)
Rung No.     30
Rung No.     31     supervisor b2 (jogador de automatos)
Rung No.     32
Rung No.     33     supervisor b3 (jogador de automatos)
Rung No.     34
Rung No.     35     supervisor b4 (jogador de automatos)
Rung No.     36
Rung No.     37     supervisor c1 (jogador de automato)
Rung No.     38
Rung No.     39
Rung No.     40
Rung No.     41
Rung No.     42     supervisor c2 (jogador de automato)
Rung No.     43
Rung No.     44
Rung No.     45
Rung No.     46
Rung No.     47     supervisor c3 (jogador de automato)
Rung No.     48
Rung No.     49
Rung No.     50
Rung No.     51
Rung No.     52     desabilitacao da mesa
Rung No.     53
Rung No.     54
Rung No.     55     desabilitacao da esteira
Rung No.     56     desabilitacao da furadeira
Rung No.     57     desabilitacao do teste
Rung No.     58     desabilitacao do capturador
Rung No.     59     atual. valor peca em P4eP3 ao girar

                    SEQUENCIAS OPERACIONAIS

Rung No.     60     sequencia operacional da mesa
Rung No.     61
Rung No.     62     sequencia operacional da esteira
Rung No.     63
Rung No.     64     sequencia operacional da furadeira
Rung No.     65
Rung No.     66
Rung No.     67
Rung No.     68
Rung No.     69     sequencia operacional do teste
Rung No.     70
Rung No.     71
Rung No.     72
Rung No.     73     sequencia operacional do capturador
Rung No.     74
Rung No.     75
Rung No.     76
Rung No.     77
Rung No.     78
Rung No.     79
Rung No.     80
Rung No.     81
Rung No.     82
Rung No.     83
Rung No.     84
Rung No.     85
Rung No.     86
Rung No.     87
```

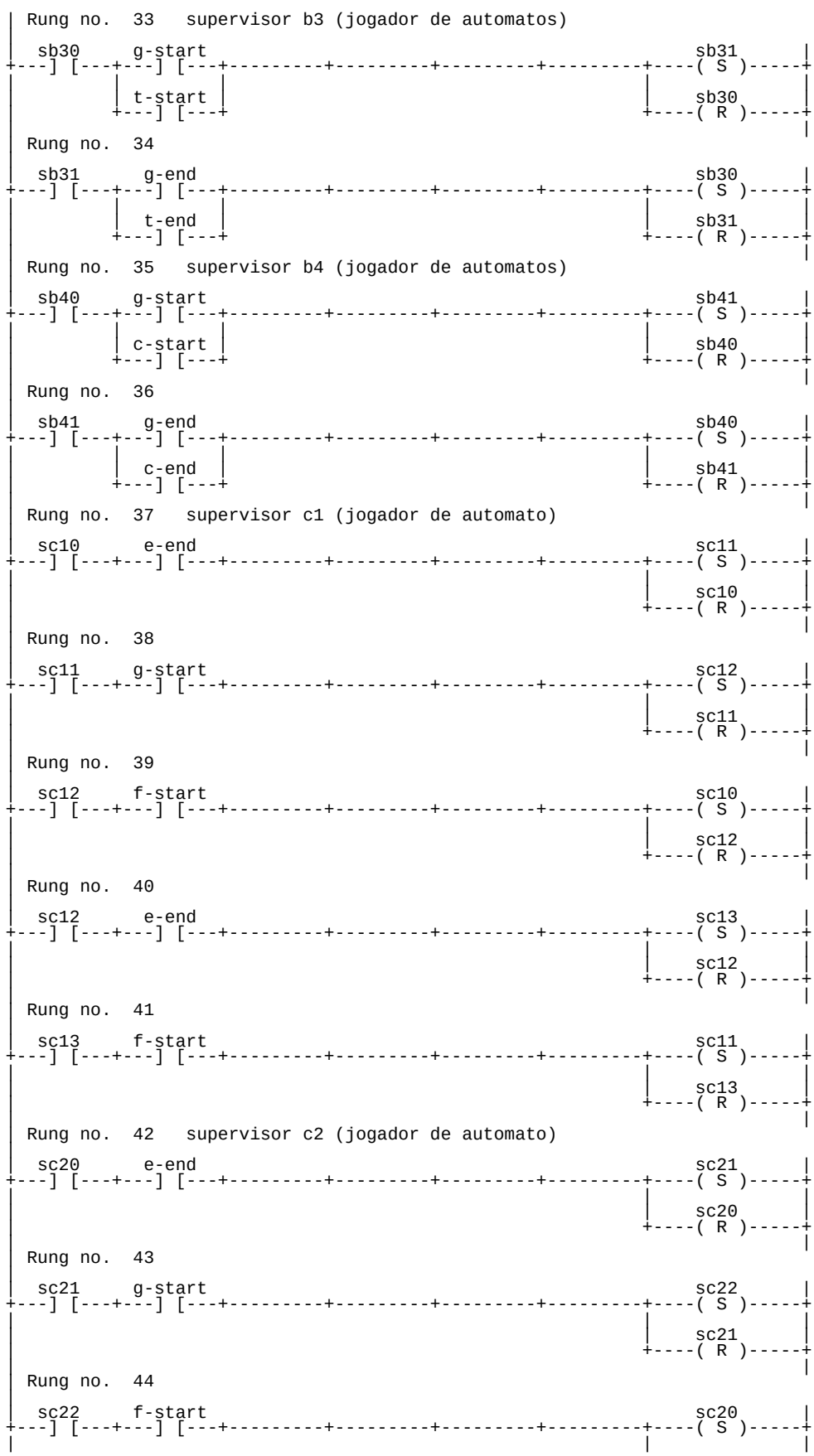

MPS - Programa de Controle Supervisorio
Rung no. 1 Planta da mesa (n-cont.)
g-1
g-fin
g-1-1
R
g-end
g-0
S
g-1
R
supervis
Rung no. 2 Planta da Esteira (n-cont.)
e-1
e-fin
e-1-1
R
e-end
e-0
S
e-1
R
supervis
Rung no. 3 Planta da Furadeira (n-cont.)
f-1
f-fin
f-end
f-1-4
R
f-0
S
f-1
R
supervis
Rung no. 4 Planta do Teste (n-cont.)
t-1
t-fin
t-end
t-1-2
R
t-0
S
t-1
R
supervis
Rung no. 5 Planta do Capturador (n-cont.)
c-1
c-fin
c-end
c-1-7
R
c-0
S
c-1
R
supervis

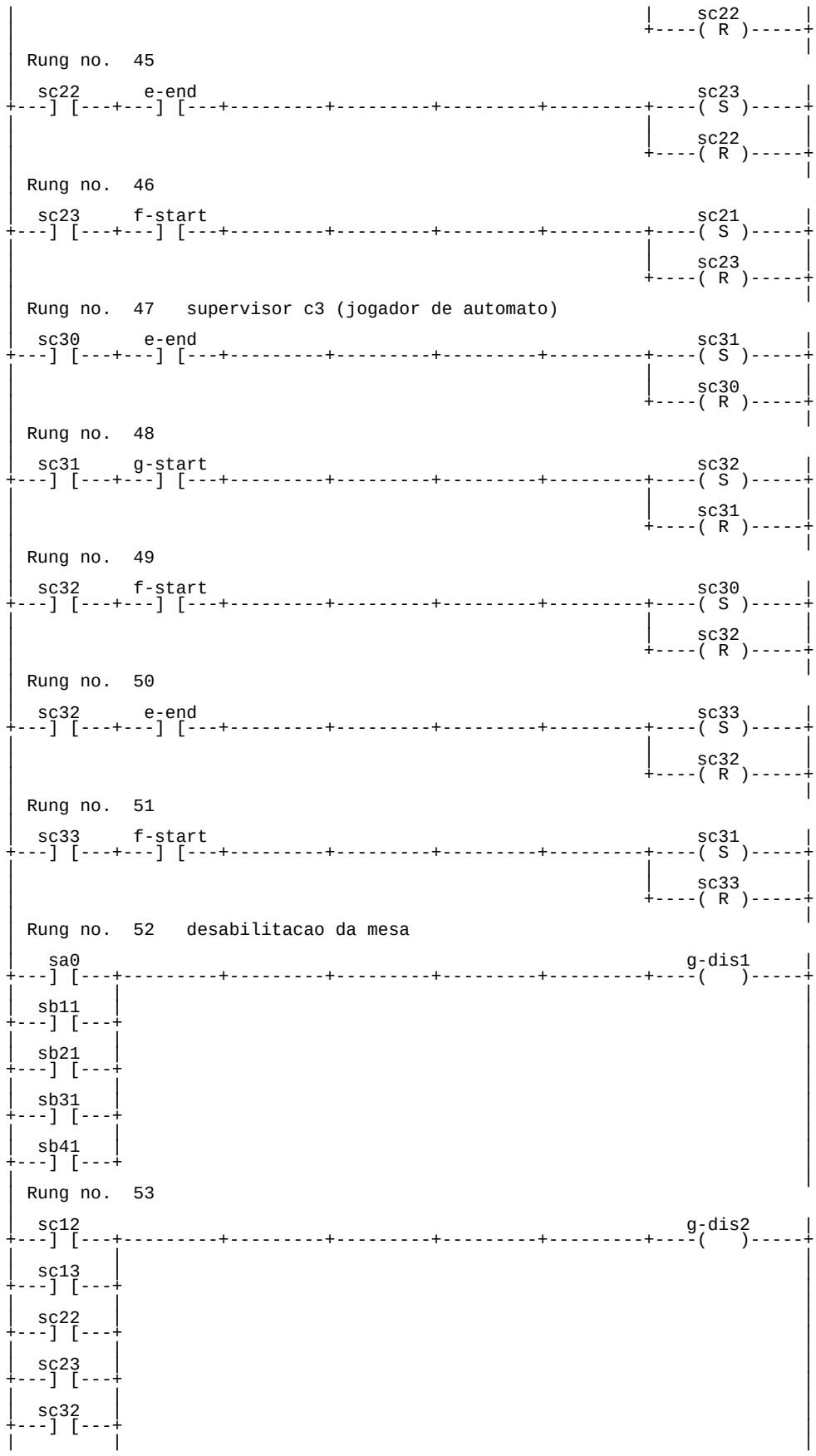

Rung no. 6 planta da mesa giratoria
g-0
g-dis
g-start
FAULT
g-1
g-0
supervis
Rung no. 7 planta da esteira
e-0
e-dis
e-start
e-1
e-0
supervis
Rung no. 8 planta da furadeira
f-0
f-dis
f-start
f-1
f-0
supervis
Rung no. 9 planta do teste
t-0
t-dis
t-start
t-1
t-0
supervis
Rung no. 10 planta do capturador
c-0
c-dis
c-start
c-1
c-0
supervis
Rung no. 11 Zera todos os Flags
RESETA
V0
TO
x-0
V0
TO
x-1
V0
TO
FW2

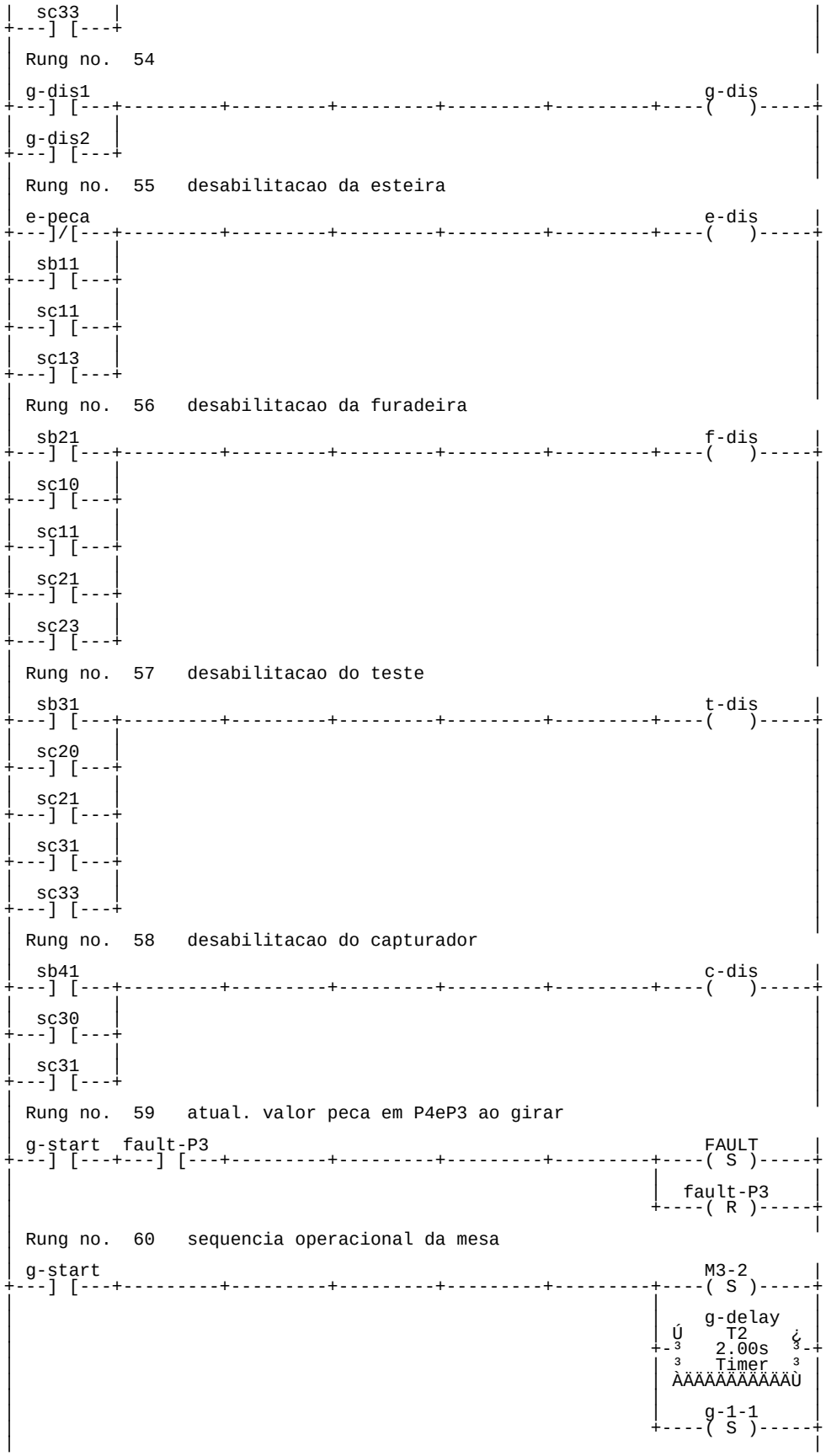

V0
TO
FW3
V0
TO
FW4
Rung no. 12
RESETA
V0
TO
FW5
V0
TO
FW6
V0
TO
FW7
V0
TO
FW8
V0
TO
FW9
Rung no. 13
RESETA
V0
TO
FW10
V0
TO
FW11
V0
TO
FW12
V0
TO
FW13
V0
TO
FW14
Rung no. 14 Desliga motores
RESETA
M3-1
(R)
M3-2
(R)
M2-1
(R)
M4-1a
(R)
M4-2a
(R)
Rung no. 15 seta/reseta config. iniciais saidas
operando
init
M2-1
(R)
M3-2
(R)
Y3-4
(R)

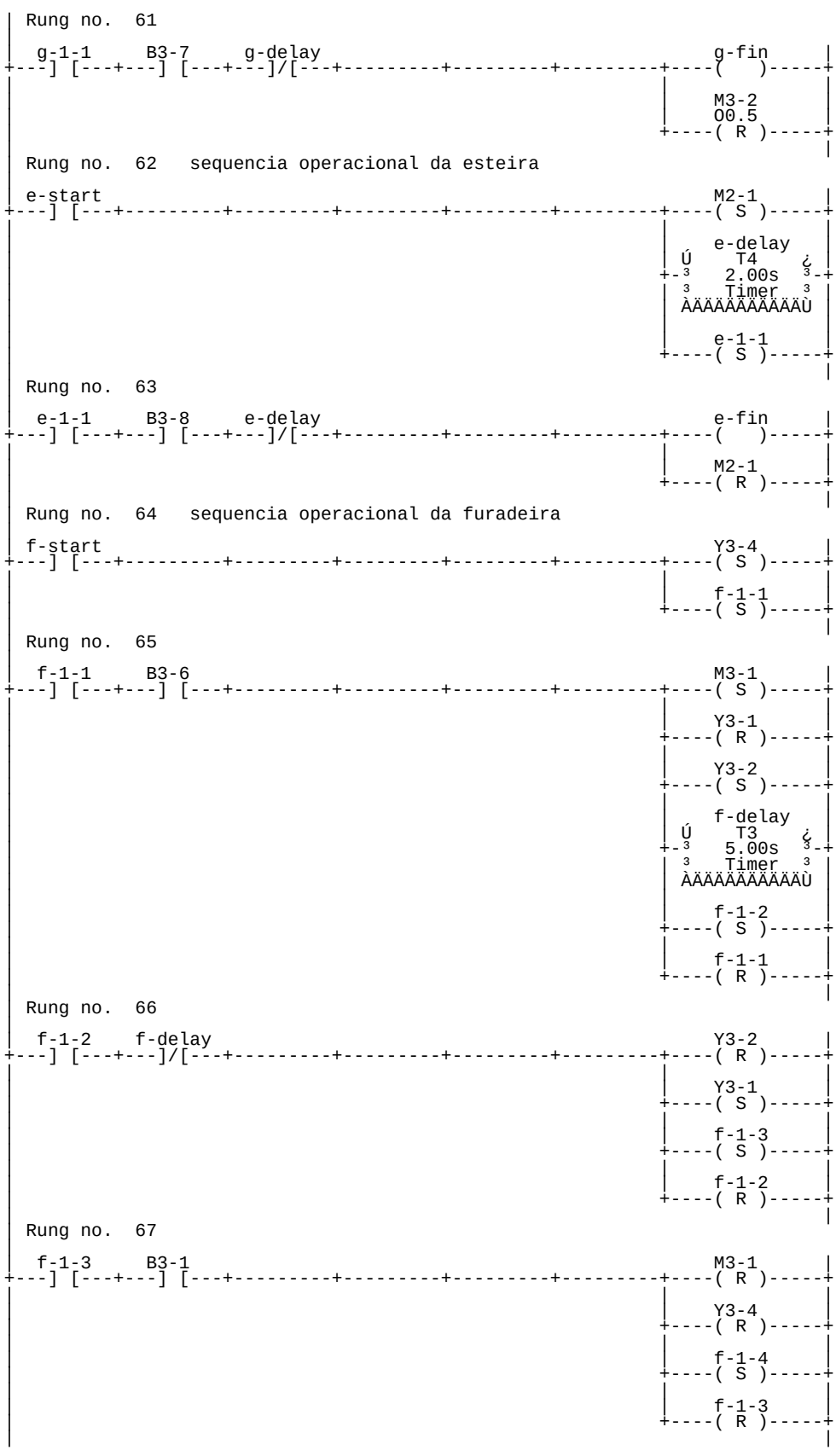

M3-1
(R)
Y3-2
(R)
Y3-1
(S)
Y3-3
(R)
Y4-1
(R)
M4-2b
(S)
M4-2a
(R)
Rung no. 16
operando init
M4-1b
(R)
M4-1a
(R)
Y4-3
(R)
Y4-4
(S)
Y4-2
(S)
init-1
(S)
Rung no. 17
init-1 B4-1
init-1
(R)
init-2
(S)
Rung no. 18 posiciona longitude/latitude e mesa
init-2 B4-7
M4-2a
(S)
Rung no. 19
init-2 B4-3
M4-1a
(S)
Rung no. 20
init-2 B3-7
M3-2
(S)
Rung no. 21
init-2 B4-7
M4-2a
(R)
Rung no. 22
init-2 B4-3
M4-1a
(R)
Rung no. 23
init-2 B3-7
M3-2
(R)
Rung no. 24
init-2 B4-7 B4-3 B3-7
init-2
(R)
inicio
(S)
Rung no. 25 inicializacao dos supervisores
inicio operando
sa0
(S)

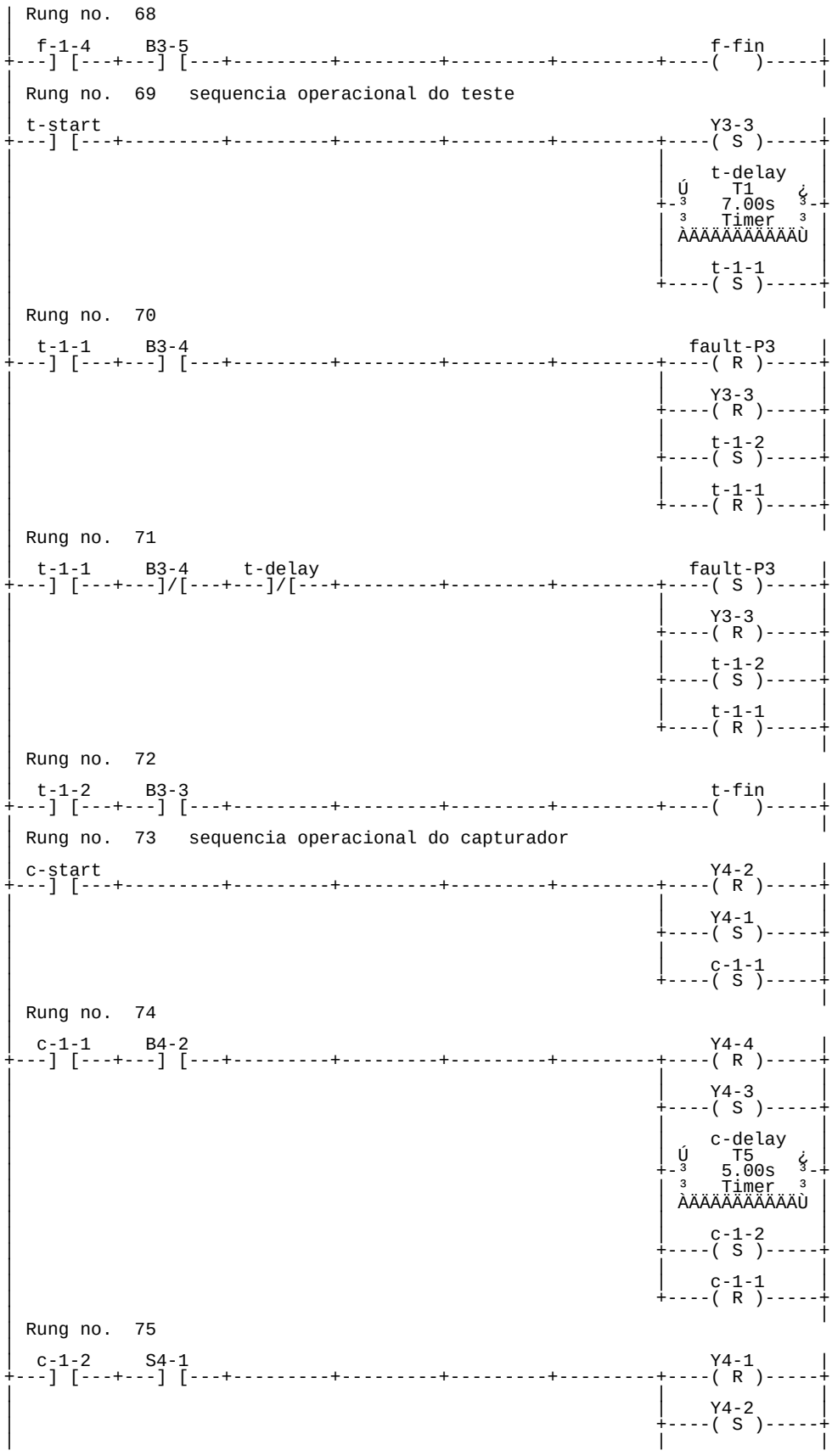

sb10
sb20
sb30
sb40
sc10
sc20
sc30
Rung no. 26 inicializacao das plantas
inicio operando
operando
e-0
g-0
f-0
t-0
c-0
Rung no. 27 supervisor a (jogador de automatos)
supervis
sa0
e-end
f-end
t-end
sa1
sa0
Rung no. 28
sa1
g-start
sa0
sa1
Rung no. 29 supervisor b1 (jogador de automatos)
sb10
g-start
sb11
e-start
sb10
Rung no. 30
sb11
g-end
sb10
e-end
sb11
Rung no. 31 supervisor b2 (jogador de automatos)
sb20
g-start
sb21
f-start
sb20
Rung no. 32
sb21
g-end
sb20
f-end
sb21

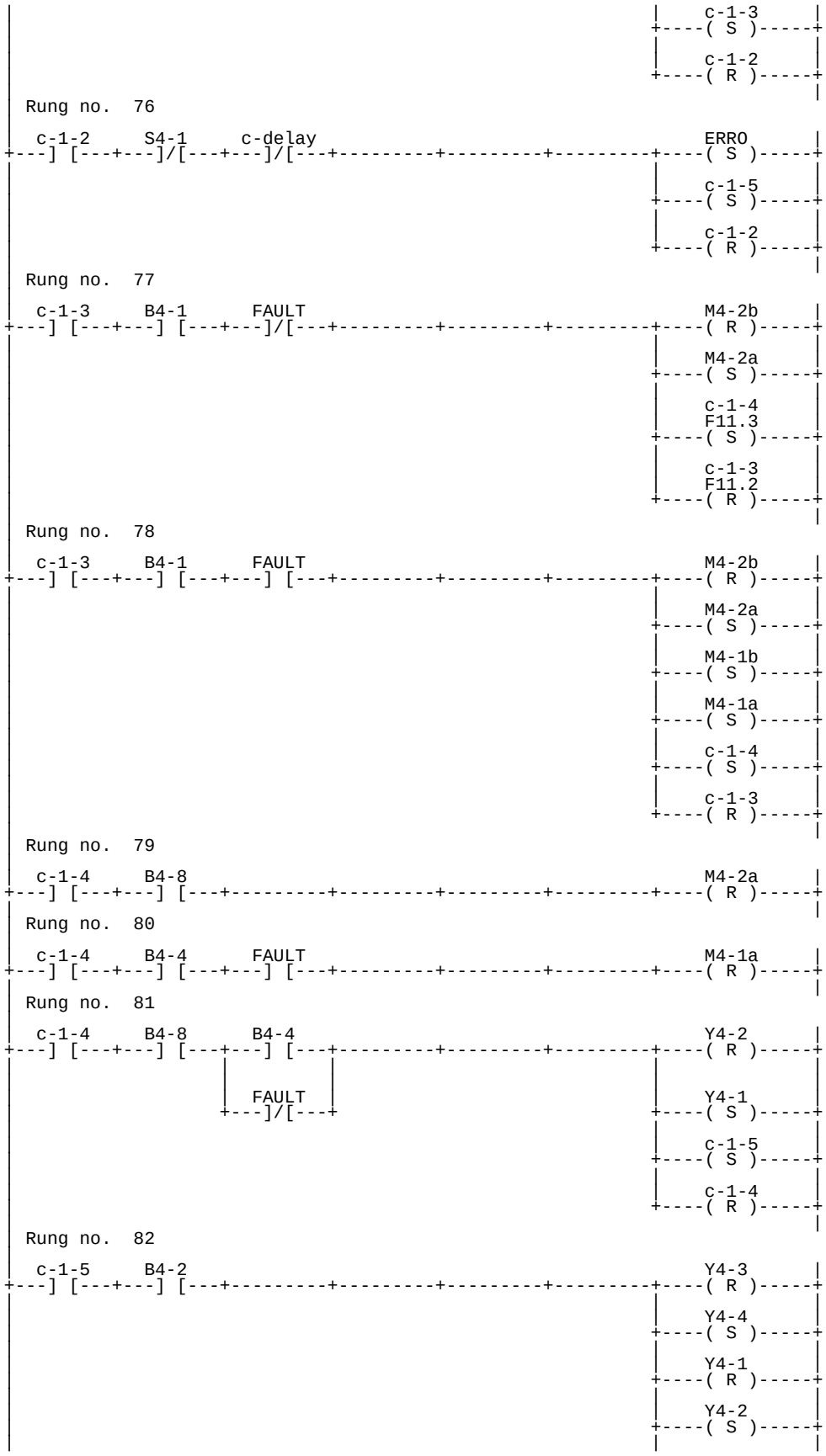

Rung no. 33 supervisor b3 (jogador de automatos)
sb30
g-start
t-start
sb31
S
sb30
R
Rung no. 34
sb31
g-end
t-end
sb30
S
sb31
R
Rung no. 35 supervisor b4 (jogador de automatos)
sb40
g-start
c-start
sb41
S
sb40
R
Rung no. 36
sb41
g-end
c-end
sb40
S
sb41
R
Rung no. 37 supervisor c1 (jogador de automato)
sc10
e-end
sc11
S
sc10
R
Rung no. 38
sc11
g-start
sc12
S
sc11
R
Rung no. 39
sc12
f-start
sc10
S
sc12
R
Rung no. 40
sc12
e-end
sc13
S
sc12
R
Rung no. 41
sc13
f-start
sc11
S
sc13
R
Rung no. 42 supervisor c2 (jogador de automato)
sc20
e-end
sc21
S
sc20
R
Rung no. 43
sc21
g-start
sc22
S
sc21
R
Rung no. 44
sc22
f-start
sc20
S

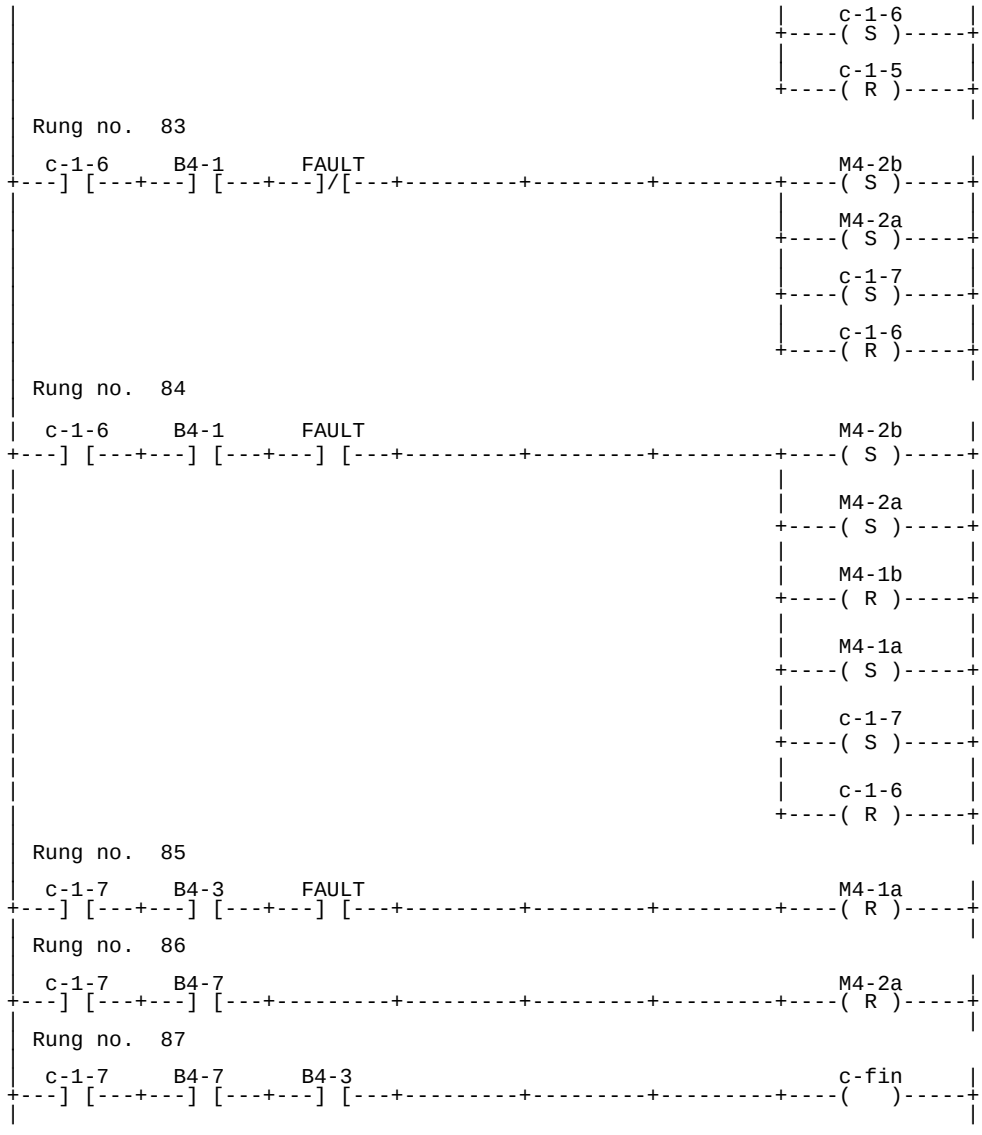

sc22
(R)
Rung no. 45
sc22
e-end
sc23
(S)
sc22
(R)
Rung no. 46
sc23
f-start
sc21
(S)
sc23
(R)
Rung no. 47 supervisor c3 (jogador de automato)
sc30
e-end
sc31
(S)
sc30
(R)
Rung no. 48
sc31
g-start
sc32
(S)
sc31
(R)
Rung no. 49
sc32
f-start
sc30
(S)
sc32
(R)
Rung no. 50
sc32
e-end
sc33
(S)
sc32
(R)
Rung no. 51
sc33
f-start
sc31
(S)
sc33
(R)
Rung no. 52 desabilitacao da mesa
sa0
g-dis1
sb11
sb21
sb31
sb41
Rung no. 53
sc12
g-dis2
sc13
sc22
sc23
sc32


sc33
Rung no. 54
g-dis1
g-dis
g-dis2
Rung no. 55 desabilitacao da esteira
e-peca
e-dis
sb11
sc11
sc13
Rung no. 56 desabilitacao da furadeira
sb21
f-dis
sc10
sc11
sc21
sc23
Rung no. 57 desabilitacao do teste
sb31
t-dis
sc20
sc21
sc31
sc33
Rung no. 58 desabilitacao do capturador
sb41
c-dis
sc30
sc31
Rung no. 59 atual. valor peca em P4eP3 ao girar
g-start
fault-P3
FAULT
S
fault-P3
R
Rung no. 60 sequencia operacional da mesa
g-start
M3-2
S
g-delay
T2
2.00s
Timer
g-1-1
S


Rung no. 61
g-1-1
B3-7
g-delay
g-fin
M3-2
00.5
Rung no. 62 sequencia operacional da esteira
e-start
M2-1
e-delay
T4
2.00s
Timer
e-1-1
Rung no. 63
e-1-1
B3-8
e-delay
e-fin
M2-1
Rung no. 64 sequencia operacional da furadeira
f-start
Y3-4
f-1-1
Rung no. 65
f-1-1
B3-6
M3-1
Y3-1
Y3-2
f-delay
T3
5.00s
Timer
f-1-2
f-1-1
Rung no. 66
f-1-2
f-delay
Y3-2
Y3-1
f-1-3
f-1-2
Rung no. 67
f-1-3
B3-1
M3-1
Y3-4
f-1-4
f-1-3


Rung no. 68
f-1-4
B3-5
f-fin
Rung no. 69 sequencia operacional do teste
t-start
Y3-3
t-delay
T1
7.00s
Timer
t-1-1
Rung no. 70
t-1-1
B3-4
fault-P3
Y3-3
t-1-2
t-1-1
Rung no. 71
t-1-1
B3-4
t-delay
fault-P3
Y3-3
t-1-2
t-1-1
Rung no. 72
t-1-2
B3-3
t-fin
Rung no. 73 sequencia operacional do capturador
c-start
Y4-2
Y4-1
c-1-1
Rung no. 74
c-1-1
B4-2
Y4-4
Y4-3
c-delay
T5
5.00s
Timer
c-1-2
c-1-1
Rung no. 75
c-1-2
S4-1
Y4-1
Y4-2


c-1-3
(S)
c-1-2
(R)
Rung no. 76
c-1-2
S4-1
c-delay
ERRO
(S)
c-1-5
(S)
c-1-2
(R)
Rung no. 77
c-1-3
B4-1
FAULT
M4-2b
(R)
M4-2a
(S)
c-1-4
F11.3
(S)
c-1-3
F11.2
(R)
Rung no. 78
c-1-3
B4-1
FAULT
M4-2b
(R)
M4-2a
(S)
M4-1b
(S)
M4-1a
(S)
c-1-4
(S)
c-1-3
(R)
Rung no. 79
c-1-4
B4-8
M4-2a
(R)
Rung no. 80
c-1-4
B4-4
FAULT
M4-1a
(R)
Rung no. 81
c-1-4
B4-8
B4-4
Y4-2
(R)
FAULT
Y4-1
(S)
c-1-5
(S)
c-1-4
(R)
Rung no. 82
c-1-5
B4-2
Y4-3
(R)
Y4-4
(S)
Y4-1
(R)
Y4-2
(S)



End of the ladder diagram

Error(s): 0

program length = 5233 Bytes

Referências Bibliográficas

ÅKESSON, K.; FLORDAL, H.; FABIAN, M. Exploiting modularity for synthesis and verification of supervisors. *In: Proc. of the 2002 IFAC 15th Triennial World Congress*, Barcelona, Spain, 2002.

BALEMI, S. *Control of Discrete Event Systems: Theory and Application.* Thesis (Doctor of Technical Sciences) - Swiss Federal Institute of Technology. Zurich, 1992.

BALEMI, S.; HOFFMANN, G. J.; GYUGYI, P.; WONG-TOI, H.; FRANKLIN, G. F. Supervisory control of a rapid thermal multiprocessor. *IEEE Trans. on Automatic Control*, v. 38, n. 7, pp. 1040-1059, 1993.

BRANDIN, B. A. The real-time supervisory control of an experimental manufacturing cell. *IEEE Trans. Robotics and Autom.,* v. 12, n. 1, pp. 1-14, 1996.

BRANDIN, B.; MALIK, R.; DIETRICH, P. Incremental system verification and synthesis of minimally restrictive behaviors. *In: Proc. of the 2000 American Control Conference*, Chicago, USA, pp. 4056–4061, 2000.

BRANDIN, B. A.; WONHAM, W. M. Modular Supervisory Control of Timed Discrete-Event Systems. *In: Proceedings of the 32nd IEEE Conference On Decision and Control*, p. 2230-2235, Dec. 1993.

BRYANT, R. E. Graph Based Algorithms for Boolean Function Manipulation. *IEEE Transaction on Computers,* v. 35, n. 8, pp. 677-691, 1986.

CASSANDRAS, C. G.; LAFORTUNE, S. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, USA, 1999.

CASTELLS, M. *A sociedade em rede* (*A era da informação: economia, sociedade e cultura, v. 1*), Ed. Paz e Terra, São Paulo, 1999.

CHARBONNIER, F.; ALLA, H.; DAVID, R. The supervised control of DEDSs. *IEEE Trans. on Control System Technology*, v. 7, n. 2, pp. 175-187, 1999.

CHEN, Y.-L.; LAFORTUNE, S.; LIN, F. Modular supervisory control with priorities for discrete event systems. *In: Proceedings of the 34th IEEE Conference On Decision and Control*, pp. 409-415, December 1995.

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

CUNHA, A. E. C. DA; CURY, J. E. R. Hierarchically consistent controlled discrete event systems. *In: Proceedings of the IFAC*, Barcelona, Spain, July 2002.

EYZELL, J. M.; CURY, J. E. R. Symmetry in the Supervisory Control Problem. *In: Proceedings of the Workshop on Discrete Event Systems*, Cagliari, Italy, August 1998a.

————— Exploiting Symmetry in the Synthesis of Supervisors for Discrete Event Systems. *In: Proceedings of the American Control Conference*, Philadelphia, USA, June 1998b.

FABIAN, M.; HELLGREN, A. PLC-based implementation of supervisory control for discrete event systems. *In: Proc. of the 37th Conf. on Decision and Control.* Tampa, USA, 1998.

FABIAN, M.; KUMAR, R. Mutually Nonblocking Supervisory Control of Discrete Event Systems, *In: Proc. of the 36th IEEE Conference on Decision e Control*, CDC'97, San Diego, CA, USA, December 1997.

FESTO. *Programmable Logic Controllers Types FPC100 Series.* Germany, 1999.

GOHARI, P. *Fair Supervisory Control of Discrete Event Systems*, Ph.D. Thesis, Dept. of Electrical & Computer Engineering, University of Toronto, September 2002.

HOPCROFT, J. E.; ULLMAN, J. D. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, USA, 1979.

ISO/IEC. *International Standard IEC 61131-3, Programmable Controllers – Part 3: Programming Languages*, ed. 2.0, 2003.

JENSEN, K. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use. - Volume 1: Basic Concepts.* Monographs in theoretical computer science, Springer-Verlag, Berlin, 1992.

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

KROGH, B. H.; HOLLOWAY, L. E. Synthesis of Feedback Control Logic for Discrete Manufacturing Systems. *Automatica*, v. 27, n. 4, pp. 641-651, 1991.

KUMAR, R.; GARG, V. K. *Modeling and Control of Logical Discrete Event Systems*. Kluwer Academic Publishers, USA, 1995.

LAW, A. M.; KELTON, W. D. *Simulation Modeling and Analysis*. 3rd ed., McGraw Hill, 2000.

LEDUC, R. J. *PLC Implementation of a DES Supervisor for a Manufacturing Testbed: An Implementation Perspective*. M.A.Sc. Thesis, Dept. of Electrical & Computer Engineering, University of Toronto, Canada, 1996.

LEDUC, R. J. *Hierarchical Interface-based Supervisory Control*, Ph.D. Thesis, Dept. of Electrical & Computer Engineering, Univ. of Toronto, April 2002.

LIN, F.; WONHAM, W. M. Decentralized control and coordination of discrete-event systems with partial observation. *IEEE Transactions on Automatic Control*, v. 35, n. 12, pp. 1330-1337, 1990.

MA, C.; WONHAM, W. M. Control of State Tree Structures. *In: Proc. 11th Mediterranean Conference on Control and Automation*, Rhodes (Greece), paper T4-005 (6pp.), June 2003.

MANNA, Z.; PNUELI, A. *The temporal logic of reactive and concurrent systems.* Springer-Verlag, 1992.

MINHAS, R. S. *Complexity Reduction in Discrete Event Systems*, Ph.D. Thesis, Dept. of Electrical & Computer Engineering, Univ. of Toronto, September 2002.

MOORE, E. F. *Sequential Machines Selected Papers*. Addison-Wesley, Reading, USA, 1964.

MURATA, T. Petri Nets: Properties, Analysis and Applications. *In: Proceedings of the IEEE*, v. 77, n. 4, pp. 541-580, 1989.

OZVEREN, C. M.; WILLSKY, A. S. Output stabilizability of discrete-event dynamic systems. *IEEE Trans. on Automatic Control*, v. 36, n. 8, pp. 925-935, 1991.

PLAGEMANN, B. *Ladder Diagram for Festo Controllers.* (Manual) 2nd ed., Festo KG, Esslingen, Germany, 1991.

PUTERMAN, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

QUEIROZ, M. H. DE. *Controle supervisório modular de sistemas de grande porte*. Dissertação de Mestrado, Universidade Federal de Santa Catarina, Departamento de Engenharia Elétrica, Florianópolis, março de 2000.

QUEIROZ, M. H. DE; CURY, J. E. R. Modular Control of Composed Systems. *In: Proceedings of the American Control Conference*. Chicago, June 2000a.

————— Modular Supervisory Control of Large Scale Discrete-Event Systems. *In: Discrete Event Systems: Analysis and Control.* Kluwer Academic Publishers, pp. 103-110, 2000b. (*Proc. of WODES 2000*, Ghent, Belgium)

————— Controle Modular de Sistemas de Manufatura Discretos. *In: Anais do Congresso Brasileiro de Automática*. Florianópolis, 2000c.

————— Controle Supervisório Modular de Sistemas de Manufatura. *Revista Controle & Automação,* SBA, 2002a.

—————— Synthesis and implementation of local modular supervisory control for a manufacturing cell. *In: Proc. of WODES 2002*, 2002b.

—————— Modular multitasking supervisory control of discrete-event systems. *In: Proc. of the 43rd IEEE Conference on Decision and Control*, 2004.

QUEIROZ, M. H. DE; CURY, J. E. R.; WONHAM, W. M. Multi-tasking supervisory control of discrete-event systems. *In: Proc. of WODES 2004*, 2004.

QUEIROZ, M. H. DE; SANTOS, E. A. P.; CURY, J. E. R. Síntese Modular do Controle Supervisório em Diagrama Escada para uma Célula de Manufatura. *In: Anais do V Simpósio Brasileiro de Automação Inteligente*, Canela, novembro de 2001.

RAMADGE, P. J. Some tractable supervisory control problems for discrete-event systems modeled by Büchi automata. *IEEE Transactions on Automatic Control*, v. 34, n. 1, pp. 10-19, 1989.

RAMADGE, P. J.; WONHAM, W. M. Supervisory control of a class of discrete event process. *SIAM Journal of Control and Optimization*, v. 25, n. 1, pp. 206-230, 1987.

—————— The control of discrete event systems. *Proceedings IEEE, Special Issue on Discrete Event Dynamic Systems*, v. 77, n. 1, pp. 81-98, Jan. 1989.

RAMIREZ, A.; ZHU, S. C.; BENHABIB, B. Moore Automata for Flexible Routing and Flow Control in Manufacturing Workcells, *In: Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA'99)*, Monterey, CA, pp. 119-124, , November 1999.

RUDIE, K.; WONHAM, W. M. Think globally, act locally: decentralized supervisory control. *IEEE Transactions on Automatic Control*, v. 37, n. 11, pp. 1692-1708, 1992.

SIMON, H. A. The architecture of complexity. *Proc. Amer. Phil. Soc.*, v. 106, pp. 467-482, December 1967. Reprinted in: Herbert A. Simin, *The Sciences of the Artificial* (Chapt. 7, pp. 193-229), Second ed., The MIT Press, Cambridge MA, 1981.

SU, R.; WONHAM, W. M. Supervisor Reduction For Discrete Event Systems. *In: Proceedings of 2001 Conference on Information Sciences and Systems*, The Johns Hopkins University, pp. 786-791, March 2001.

—————— Supervisor Reduction for Discrete-Event Systems. *Discrete Event Dynamic Systems*, v. 14, n. 1, pp. 31-53, January 2004.

TARSKI, A. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*. 5, 1955.

THISTLE, J. G. R.; MALHAME, P. Multiple Marked Languages and Noninterference in Modular Supervisory Control. *In: Proc. of the 35th Annual Allerton Conference*. pp. 523-532, 1997.

THISTLE, J. G. R.; MALHAME, P.; HOANG, H.-H.; LAFORTUNE, S. *Supervisory control of distributed systems part I: modeling, specification, and synthesis.* Technical Report EPM/RT-97/08, Ecole Polytechnique de Montreal, 1997.

TORRICO, C. R. C.; CURY, J. E. R. Hierarchical supervisory control of discrete event systems based on state aggregation. *In: Proceedings of the IFAC*, Barcelona, Spain, July 2002.

VAZ, A. F.; WONHAM, W. M. On supervisor reduction in discrete-event systems. *International Journal of Control*, v. 44, n. 2, pp. 475-491, 1986.

WILLNER, Y.; HEYMANN, M. Supervisory control of concurrent discrete-event systems. *International Journal on Control*, v. 54, n.5, pp. 1143-1169, 1991.

WONG, K. C.; THISTLE, J. G.; HOANG, H.-H.; MALHAMÉ, R. P. Conflict resolution in modular control with application to feature interaction. *In: Proceedings of the 34th IEEE Conference on Decision and Control*, pp. 416-421, Dec. 1995.

WONG, K. C.; WONHAM, W. M. Hierarchical control of discrete-event systems. *Discrete Event Dynamic Systems*, v. 6, n. 3, pp. 241-273, July 1996.

———————— Modular Control and Coordination of Discrete-Event Systems. *Discrete Event Dynamic Systems*, v. 8, n. 3, pp. 247-297, Oct. 1998.

WONHAM, W. M. *Notes on Control of Discrete-Event Systems*. Dept. of Electrical & Computer Engineering, University of Toronto, 2003. http://www.control.utoronto.ca/DES

WONHAM, W. M.; RAMADGE, P. J. On the supremal controllable sublanguage of a given language. *SIAM Journal of Control and Optimization*, v. 25, n. 3, pp. 637-659, 1987.

———————— Modular supervisory control of discrete event systems. *Mathematics of control of discrete event systems*, v. 1, n. 1, pp. 13-30, 1988.

ZAD, S. H.; KWONG, R. H.; WONHAM, W. M. Fault diagnosis in discrete-event systems: framework and model reduction. *In: Proc. 1998 IEEE Conference on Decision and Control (CDC'98)*, pp. 3769-3774, December 1998.

ZHANG, Z. H.; WONHAM, W. M.. STCT: An Efficient Algorithm for Supervisory Control Design, *In: Proc. Symposium on Supervisory Control of Discrete Event Systems (SCODES2001)*, Paris, 12 pp., 2001.

ZHONG, H.; WONHAM, W. M. On the Consistency of Hierarchical Supervision in Discrete-Event Systems. *IEEE Transactions on Automatic Control*, v. 35, n. 10, pp. 1125-1134, 1990.

ZHOU, M. C.; VENKATESH, K. *Modeling, Simulation and Control of Manufacturing Systems: A Petri Net Approach.* World Scientific, Singapore, 1999.