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

CONTRIBUIÇÕES AO PROBLEMA DE CONTROLE SUPERVISÓRIO DE SISTEMAS A EVENTOS DISCRETOS PARAMETRIZÁVEIS E NÃO-REGULARES

CLAUDIO DE OLIVEIRA

FLORIANÓPOLIS
2005

CLAUDIO DE OLIVEIRA

CONTRIBUIÇÕES AO PROBLEMA DE CONTROLE SUPERVISÓRIO DE SISTEMAS A EVENTOS DISCRETOS PARAMETRIZÁVEIS E NÃO-REGULARES

FLORIANÓPOLIS
2005

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

CONTRIBUIÇÕES AO PROBLEMA DE CONTROLE SUPERVISÓRIO DE SISTEMAS A EVENTOS DISCRETOS PARAMETRIZÁVEIS E NÃO-REGULARES

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

CLAUDIO DE OLIVEIRA

Florianópolis, outubro de 2005.

CONTRIBUIÇÕES AO PROBLEMA DE CONTROLE SUPERVISÓRIO DE SISTEMAS A EVENTOS DISCRETOS PARAMETRIZÁVEIS E NÃO-REGULARES

CLAUDIO DE OLIVEIRA

Esta tese foi julgada adequada para a obtenção do título de **Doutor em Engenharia Elétrica** na especialidade **Engenharia Elétrica**, área de concentração **Automação e Sistemas**, e aprovada em sua forma final pelo curso de Pós-Graduação.

Florianópolis, 10 de outubro de 2005.

Dr. José E.R. Cury, orientador

Dr. Celso A.A. Kaestner, coorientador

Dr. Alexandre Trofino
Coordenador do curso de Pós-Graduação em Engenharia Elétrica
da Universidade Federal de Santa Catarina.

Banca Examinadora

Dr. Rafael Santos Mendes

Dr. Fernando Gomide

Dr. Jean-Marie Farines

Dr. Eduardo Camponogara

Dr. Max Hering de Queiroz

DEDICATÓRIA

"De repente a dor
De esperar terminou
E o amor veio em fim
Eu que sempre sonhei
Mas não acreditei
Muito em mim
Vi o tempo passar
O inverno chegar outra vez
Mas dessa vez
Todo o pranto sumiu
Como encanto surgiu
Meu amor ... "

Para Nádia, Claudia, Luisa e Lais. Tudo que eu faço é dedicado a vocês e para expressar o que eu sinto por vocês, faço minhas as palavras do Tim Maia.

"Vocês,
são mais do que sei,
são mais que pensei,
são mais que esperava.

Vocês,
são algo assim,
são tudo pra mim,
são como eu sonhava.

Sou feliz agora."

AGRADECIMENTOS

Ao longo deste trabalho, muitas pessoas contribuíram para minha formação pessoal. Particularmente no departamento de automação de sistemas da UFSC, são exemplos admiráveis de profissionalismo, seriedade e simplicidade, os professores José Eduardo Ribeiro Cury, Edson Roberto de Pieri, Jean-Marie Farines, Guilherme Bittencourt e Joni da Silva Fraga.

Ao professor orientador José Eduardo Ribeiro Cury. Sua enorme paciência e rápida percepção das minhas limitações me permitiu desenvolver um trabalho de meu próprio interesse sem nenhum constrangimento ou pressão de qualquer espécie. Espero, sinceramente, um dia, poder-lhe retribuir.

Ao professor Celso Antônio Alves Kaestner, coorientador, conselheiro, motivador e amigo.

Ao Eduardo Hamerski que um dia, quando eu filosofava sobre o que tinha acontecido, me sugeriu filosofar sobre o que iria acontecer. Sua observação mudou o rumo de minhas pesquisas o que me permitiu resolver muitos problemas.

À Eliana Pantaleão, por suas inúmeras revisões detalhadas dos artigos e desta tese.

À Universidade Católica do Paraná, pela inestimável ajuda financeira.

À Universidade Federal de Santa Catarina pela oportunidade de realizar o curso de doutorado.

Finalmente, aos meus pais, João Maria de Oliveira e Mirian de Oliveira, por me conduzirem, desde os primeiros passos, na direção certa.

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

CONTRIBUIÇÕES AO PROBLEMA DE CONTROLE SUPERVISÓRIO DE SISTEMAS A EVENTOS DISCRETOS PARAMETRIZÁVEIS E NÃO-REGULARES

CLAUDIO DE OLIVEIRA

Orientador: José Eduardo Ribeiro Cury, Dr.

Área de Concentração: Automação e Sistemas

Palavras-Chave: controle supervisório; sistemas a eventos discretos; comportamentos não-regulares; parametrização.

Número de Páginas: 182.

Esta tese apresenta um modelo para sistemas a eventos discretos no qual a planta consiste de um Sistema de Transição de Estados equipado com uma coleção de dados. A coleção de dados introduz variáveis cujos valores são atualizados por operações comandadas pelas transições discretas do Sistema de Transição de Estados. As especificações de comportamentos desejáveis consistem de predicados relacionados às ocorrências dos eventos. O supervisor, baseado na seqüência de eventos ocorrida no passado, controla a evolução do sistema através da habilitação de eventos. A decisão da habilitação de eventos decorre da avaliação de predicados computados a partir da especificação de comportamentos desejáveis. Um método para a síntese de supervisores atendendo uma especificação através de um controle minimamente restritivo é apresentado. As principais características do framework apresentado são a possibilidade da captura de comportamentos não-regulares e a construção de modelos parametrizados os quais conduzem a soluções genéricas para uma dada classe de problemas.

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

CONTRIBUTIONS TO THE SUPERVISORY CONTROL PROBLEM OF PARAMETERIZABLE AND NON-REGULAR DISCRETE EVENT SYSTEMS

CLAUDIO DE OLIVEIRA

Advisor: José Eduardo Ribeiro Cury, Dr.

Área de Concentração: Automação e Sistemas

Keywords: supervisory control; discrete event systems; non-regular behaviors; parametrization.

Number of Pages: 182.

This thesis presents a model for discrete-event systems in which the plant consists of a finite State Transition System, or STS, equipped with a data collection. The data collection introduces variables whose values are updated by operations commanded by the discrete transitions in the STS. The specifications of desirable behaviors consist of predicates related to the occurrence of the events. The supervisor, based on the sequence of events occurred in the past, controls the system evolution through the disabling of events. The decision of disabling events originates from the evaluation of predicates computed from the specification of desirable behaviors. A method for the synthesis of supervisors attaining a specification with minimally restrictive control is presented. The main characteristics of our framework are the possibility of capturing non-regular behaviors and the construction of parameterized models which lead to generic solutions for a given class of problems.

Sumário

Lista de Figuras

Lista de Tabelas

Nomenclatura

Acrônimos

FSMwP	Finite State Machine with Parameters
SED	Sistemas a Eventos Discretos
RW	Ramadge e Wonham
STE	Sistema de Transição de Estados (Determinístico)
STExCD	STE com coleção de dados
STE-PC	STE com Pré-Condições

Diversos

$<$	Menor que ...	
$>$	Maior que ...	
$\geq$	Maior que ou igual a ...	
$\langle \ldots \rangle$	Notação para "tuplas"	
$\Leftrightarrow$	Equivalência lógica	
$\leq$	Menor que ou igual a ...	
$\mathbb{B}$		$\mathbb{B} = \{true, false\}$
$\mathbb{Z}$	Conjunto dos inteiros	$\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$
$\Rightarrow$	Implicação lógica	
$\rightarrow$	Funções em geral (totais ou não)	
$\vee$	Disjunção lógica	
$\wedge$	Conjunção lógica	

Teoria Clássica

G	Gerador	$G = \langle Q, \Sigma, \delta, q_0, Q_m \rangle$	12
Q	Conjunto finito de estados discretos		12
Σ	Conjunto finito de eventos		12
Σ^*	Conjunto de cadeias de símbolos em Σ		12
ε	Cadeia vazia		12
δ	Função parcial de transição de estados	$\delta : Q \times \Sigma \rightarrow Q$	12
$\Sigma(q)$	Eventos possíveis a partir do estado q	$\Sigma(q) = \{\sigma \in \Sigma \mid \delta(q, \sigma)!\}$	12
q_0	Estado inicial	$q_0 \in Q$	12
Q_m	Conjunto de estados marcados	$Q_m \subseteq Q$	12
δ^*	Extensão da função δ para cadeias de eventos	$\delta^* : Q \times \Sigma^* \rightarrow Q$	12
Σ_c	Conjunto de eventos controláveis		12
Σ_u	Conjunto de eventos não controláveis		12
$L(G)$	Linguagem gerada por G	$\{s \in \Sigma^* \mid \delta^*(q_0, s)!\}$	12
$L_m(G)$	Linguagem marcada por G	$\{s \in \Sigma^* \mid \delta^*(q_0, s) \in Q_m\}$	12
$\overline{H}$	Prefixo fechamento de H	$\{s \in \Sigma^* \mid su \in H\}$	12
$Reach^Q(G)$	Subconjunto de estados acessíveis	$\{q \in Q \mid \exists\, s \in \Sigma^*,\ \delta^*(q_0, s) = q\}$	12
$CoReach^Q(G)$	Subconjunto de estados co-acessíveis	$\{q \in Q \mid \exists\, s \in \Sigma^*,\ \delta^*(q, s) \in Q_m\}$	13
Γ_G	Conjunto de entradas de controle válidas	$\{\gamma \in 2^\Sigma \mid \Sigma_u \subseteq \gamma \subseteq \Sigma\}$	16
V	Supervisor	$V : L \rightarrow \Gamma_G$	16
V/G	Sistema supervisionado		16
$\mathcal{C}(E)$	Sub-linguagens $L(G)$-controláveis de E	$\{K \subseteq E \mid K \text{ é } L(G) - \textit{controlável}\}$	21
$sup\mathcal{C}(E)$	Máxima sub-linguagem $L(G)$-controlável de E	$sup\mathcal{C}(E) = \bigcup_{K \in \mathcal{C}(E)} K$	21

Máquinas de Estados Finitos com Parâmetros

$FSMwP$	Máquina de estados finitos com parâmetros	$FSMwP = \langle \Sigma, Q, \delta, P, G, \langle q_0, p_0 \rangle, Q_m \rangle$	26
Σ	Conjunto finito de eventos		26
δ	Função de transição	$\delta : \Sigma \times Q \times G \times P \to Q \times P$	26
Q	Conjunto finito e não vazio de estados discretos		26
r	Uma execução de uma FSMwP	$r = (q_0, p_0) \xrightarrow{l_1} (q_1, p_1) \xrightarrow{l_2} (q_2, p_2) \xrightarrow{l_3} \ldots,$	27
L(FSMwP)	Linguagem gerada por uma FSMwP		27
L_m(FSMwP)	Linguagem marcada por uma FSMwP		28
P	Espaço vetorial		26
Q_b	Conjunto de estados ilegais	$Q_b \subseteq Q$	28
t	Relógio		29
Σ_c	Conjunto de eventos controláveis	$\Sigma_c \subseteq \Sigma$	29
Σ_f	Conjunto de eventos forçáveis	$\Sigma_f \subseteq \Sigma$	29
Γ	Padrões de controle	$\Gamma = \{ g \subseteq \Sigma \mid \Sigma - \Sigma_c \subseteq g \ \wedge \ g \subseteq \Sigma_f \}$	29
C_d	Conjunto de eventos a serem desabilitados no estado q	$C_d : Q \to 2^{\Sigma_c}$	30
C_f	Conjunto de eventos a serem forçados no estado	$C_f : Q \to 2^{\Sigma_f}$	30
p	Vetor de parâmetros	$p \in P$	26
$L(FSMwP, \gamma)$	Comportamento do sistema controlado		30
G	Conjunto de predicados definidos sobre os parâmetros $p \in P$		26
$T(q, Q')$	Conjunto de transições para estados em Q'	$\{ l \in \Sigma \cup G \mid q \xrightarrow{l} q' \in \delta \ \wedge \ q' \in Q' - \{q\} \}$	31
$NB(Q_b)$	Conjunto de estados vizinhos dos estados ilegais	$\{ q \in Q - Q_b \mid \exists\, l, \ l \in T(q, Q_b) \}$	31
g	Guarda	$g \in G$	26
G_t	Conjunto de transições temporais		32
$\tau(g)$	Tempo em que g ocorre		32
p_0	Valor inicial dos parâmetros		26
q_0	Estado inicial do FSMwP	$q_0 \in Q$	26
Q_m	Conjunto de estados marcados ou finais	$Q_m \subseteq Q$	26

Sistemas de Transição de Estados com Pré-Condições

V_Λ/$\mathbf{S}$	STE com Pré-Condições		71
$\mathbb{C}(\mathbf{S})$	Subconjunto de especificações controláveis	$\mathbb{C}(\mathbf{S}) \subseteq \Lambda(\mathbf{S})$	77
$\Lambda^\uparrow$	Menor limite superior de $\mathbb{C}(\Lambda)$	$\Lambda^\uparrow = sup\, \mathbb{C}(\Lambda)$	79
$\Lambda^\downarrow$	Maior limite inferior de $\mathbb{C}(\Lambda)$	$\Lambda^\downarrow = inf\, \mathbb{C}(\Lambda)$	79
$\mathbb{C}(\Lambda)$	Subconjunto das especificações controláveis que implicam Λ	$\mathbb{C}(\Lambda) = \{ \lambda \in \mathbb{C}(\mathbf{S}) \mid \lambda \Rightarrow \Lambda \}$	79
S	Designação genérica de um STE determinístico	$S = \langle Q, \Sigma, \delta \rangle$	42
ξ	Evento de inicialização ou reinicialização de um STE	$\xi \in \Sigma_c$	42
I	Conjunto genérico de instanciações		45
nop	No-operation	$\forall\, i \in I, \ nop(i) = i$	45
$\circ$	Operação de composição de funções	$\forall\, i \in I, \ (f \circ g)(i) = f(g(i))$	45
F	Conjunto finito de funções totais em I		45
F^*	Monóide livre com relação à composição de funções em F		45
M_I	Monóide de operações	$M_I = \langle F_I^*, \circ, nop \rangle$	46
$\mathbf{M}_n$	coleção de dados	$\mathbf{M}_n = \langle M_{I_1}, M_{I_2}, \ldots, M_{I_n} \rangle$	48
$\mathbf{I}_n$	Conjunto de instanciações de $\mathbf{M}_n$	$\mathbf{I}_n = I_1 \times I_2 \times \ldots \times I_n$	48
ι	Instanciação genérica de $\mathbf{M}_n$	$\iota = \langle \iota_1, \ldots, \iota_n \rangle \in \mathbf{I}_n$	48
$\mathbf{F}_n$	Conjunto suporte de uma coleção de dados	$\mathbf{F}_n = F_{I_1} \times F_{I_2} \times \ldots \times F_{I_n}$	48
$\mathbf{F}_n^*$	O conjunto de composições finitas de funções em $\mathbf{F}_n$		48
$\mathbf{S}$	STExCD	$\mathbf{S} = \langle S, \mathbf{M}_n, \Delta, \mathcal{P}_m \rangle$	49
Δ	Função parcial mapeando eventos a operações	$\Delta : Q \times \Sigma \to \mathbf{F}_n$	49
$\mathcal{P}_m$	Predicado marcador	$\mathcal{P}_m : Q \times \mathbf{I}_n \to \mathbb{B}$	49
Δ^*	Extensão da função Δ para cadeias de eventos	$\Delta^* : Q \times \Sigma^* \to \mathbf{F}_n$	49
$\langle q_0, \iota_0 \rangle$	Configuração inicial de $\mathbf{S}$	$\langle q_0, \iota_0 \rangle = \langle \delta(\cdot, \xi), \Delta(\cdot, \xi)(\cdot) \rangle$	49
$\langle q, \iota \rangle$	Configuração genérica de $\mathbf{S}$	$\langle q, \iota \rangle \in Q \times \mathbf{I}_n$	49
$L(\mathbf{S})$	Linguagem gerada pelo STExCD $\mathbf{S}$	$L(\mathbf{S}) = \{ \xi s \mid s \in (\Sigma - \{\xi\})^* \text{ e } \delta^*(q_0, s)! \}$	50
$L_m(\mathbf{S})$	Linguagem marcada pelo STExCD $\mathbf{S}$	$\{ \xi s \in L(\mathbf{S}) \mid \mathcal{P}_m \langle \delta^*(q_0, s), \Delta^*(q_0, s)(\iota_0) \rangle \}$	50
$Reach^{QI}(\xi s)$	Configuração alcançada pelo traço ξs	$Reach^{QI}(\xi s) = \langle \delta^*(q_0, s), \ \Delta^*(q_0, s)(\iota_0) \rangle$	50
$Reach^{QI}(K)$	Conjunto de configurações alcançadas pelos traços em K	$\{ Reach^{QI}(\xi s) \in Q \times \mathbf{I}_n \mid \xi s \in K \}$	50
$Reach^{QI}(L(\mathbf{S}))$	O subconjunto de configurações acessíveis em $\mathbf{S}$		50
$CoReach^{QI}(K)$	Conjunto de configurações co-acessíveis pelos traços em K		50
$\mathcal{P}$	Fórmula lógica	$\mathcal{P} : Q \times \mathbf{I}_n \to \mathbb{B}$	54
$\mathbb{D}_\Lambda$	Conjunto de configurações satisfazendo Λ	$\mathbb{D}_\Lambda \subseteq Q \times \mathbf{I}_n$	54

Λ	Especificação por pré-condições	$\Lambda : \Sigma \to \{\mathcal{P} \mid \mathcal{P} : Q \times \mathbf{I}_n \to \mathbb{B}\}$	54
$\Rightarrow$	Implicação entre especificação por pré-condições		55
$\Leftrightarrow$	Equivalência entre especificação por pré-condições		55
$\gamma_\Lambda\langle q, \iota\rangle$	Controle induzido por Λ	$\{\sigma \in \Sigma(q) \mid \Lambda(\sigma)\langle q, \iota\rangle = true\}$	69
$V_\Lambda(\xi s)$	Supervisor para Λ	$V_\Lambda(\xi s) = \gamma_\Lambda\langle\delta^*(q_0, s), \Delta^*(q_0, s)(\iota_0)\rangle$	70
Inv	Especificação por invariantes	$Inv : Q \times \mathbf{I}_n \to \mathbb{B}$	58
$\mathbb{D}_{Inv}$	Conjunto de configurações satisfazendo o invariante Inv	$\{\langle q, \iota\rangle \in Q \times \mathbf{I}_n \mid Inv(q, \iota)\}$	58
$\Lambda(\mathbb{S})$	Conjunto de todas as especificações por pré-condições		54
$\bot$	Especificação tal que $\mathbb{D}_\bot = \emptyset$		54
$\sqcap$	Operador meet entre especificações por pré-condições	$\sqcap : \Lambda(\mathbb{S}) \times \Lambda(\mathbb{S}) \to \Lambda(\mathbb{S})$	61
$\sqcup$	Operador join entre especificações por pré-condições	$\sqcup : \Lambda(\mathbb{S}) \times \Lambda(\mathbb{S}) \to \Lambda(\mathbb{S})$	61
$\top$	Especificação tal que $\mathbb{D}_\top = Reach^{QI}(L(\mathbb{S}))$		54

Algoritmos e Procedimentos

ω	É um caminho genérico até uma transição σ	$\omega\sigma \in L(\mathbb{S})$	80
$\sqcap_i$	i-ésima aplicação do operador meet		109
$Path(\sigma)$	Conjunto de caminhos até a transição σ	$Path(\sigma) = \{\omega \mid \omega\sigma \in L(\mathbb{S})\}$	80

Capítulo 1

Introdução

"Um sistema é uma combinação de componentes que atuam conjuntamente e realizam um certo objetivo. Um distúrbio é um sinal que tende a afetar adversamente o valor da saída de um sistema. O controle realimentado é uma operação que, na presença de distúrbios, tende a reduzir a diferença entre a saída de um sistema e a entrada de referência (ou um estado desejado, arbitrariamente variado) e que opera nesta diferença. Um sistema de controle realimentado é aquele que tende a manter uma relação prescrita entre a saída e a entrada de referência, comparando-as e utilizando a diferença como um meio de controle [Oga82]."

Embora o (sistema de) controle realimentado seja aplicável a várias áreas do conhecimento, ele é basicamente uma disciplina de engenharia. Como tal, seu progresso está intimamente relacionado com a resolução de problemas práticos que apareceram em algum momento da história humana. Dentre estes momentos, destacam-se (a) a revolução industrial na Europa (século 18), (b) o surgimento das comunicações em massa e as grandes guerras na primeira metade do século 20 e (c) o início da era espacial em 1957. Ao longo de todos estes anos, as metodologias para a análise e o projeto de sistemas de controle evoluíram da tentativa-e-erro a métodos matemáticos bem definidos. As primeiras metodologias matemáticas focaram basicamente a representação de fenômenos físicos através de equações diferenciais. Com o advento dos computadores, estes fenômenos físicos passaram a ser representados por equações dinâmicas em tempo discreto, as quais não alteraram a natureza contínua intrínseca da evolução daqueles fenômenos [Lew92].

Com o avanço da tecnologia, o homem começou a construir sistemas "completamente" artificiais cada vez mais complexos, como, por exemplo, redes de transporte, redes de computadores, redes de comunicação, linhas de montagem, em particular, suas formas "flexíveis" – para citar, é claro,

apenas alguns exemplos. Nestes sistemas, o principal mecanismo dinâmico é a sucessão de tarefas e os principais problemas a serem resolvidos estão relacionados com a sincronização e cooperação de tarefas e a exclusão mútua ou competição pelo uso de algum recurso comum. Para sistemas como estes, a utilização de modelos baseados em equações diferenciais ou por suas análogas no tempo discreto nem sempre são adequados. Esta é certamente a razão pela qual tais sistemas, não obstante, constituírem verdadeiros sistemas dinâmicos, terem sido negligenciados durante algum tempo pelos peritos em controle automático, tendo atraído, entretanto, a atenção de cientistas da computação, especialistas em manufatura, etc., de acordo com o domínio da aplicação de interesse.

O desinteresse por parte dos peritos em controle automático nestes sistemas perdurou até os anos 80. A partir daí, a engenharia de controle passou a utilizar teorias bem definidas pela ciência da computação para modelar e tratar sistemas como os anteriores. Tais sistemas foram referenciados como Sistemas Dinâmicos a Eventos Discretos ou, resumidamente, SED [1]. A palavra "discretos" se refere ao fato de que as dinâmicas são construídas a partir de eventos, ou em outras palavras, dinâmicas dirigidas pela ocorrência de eventos em instantes discretos, não conhecidos à priori. Nestes sistemas, o interesse está no momento e na ordem em que os eventos acontecem.

Em resumo, os SED são sistemas lógicos que representam abstrações dos sistemas físicos e são usados quando a exploração de certas propriedades lógicas (ou qualitativas) e quantitativas é desejável. No contexto das propriedades qualitativas, são tratados problemas tais como o confinamento dos sistemas a um conjunto de estados seguros e a ausência de bloqueio; e, no contexto das propriedades quantitativas, o interesse principal está no desempenho e otimização dos sistemas.

1.1 Do que consiste o estudo e quais são os objetivos do estudo de SED?

Normalmente, o objetivo de qualquer estudo relacionado com SED está na modelagem, análise e/ou controle dos sistemas. Um modelo é uma representação do conhecimento que se tem sobre o sistema e a partir do qual é possível, por exemplo, predizer o comportamento do sistema. Claramente, qualquer estudo cujo objetivo é a análise ou o controle de sistemas, parte de uma representação adequada do sistema. Desta forma, pode-se dizer que a modelagem é a base de qualquer estudo relacionado com SED.

[1] Muito provavelmente, o termo "Sistemas Dinâmicos a Eventos Discretos" foi criado por Yu Chi Ho [HC83].

Como em qualquer área do conhecimento, os modelos são construídos em função do que se deseja realizar e das técnicas a serem usadas nestas realizações. Um modelo pode ser adequado a uma certa aplicação e inadequado a outra. Assim, em termos dos objetivos a atingir, não existe o modelo perfeito, mas apenas o modelo mais adequado. Nos SED, a modelagem não é diferente e, conseqüentemente, muitos modelos foram propostos nos últimos anos. Estes modelos podem, em geral, ser classificados segundo os instrumentos utilizados em suas fundamentações. Dentre os modelos mais utilizados, destacam-se aqueles baseados em linguagens formais, redes de Petri, lógica temporal e máquinas de estados estendidas. Estes e outros modelos interessantes são explorados extensivamente na literatura [VK88, RW89, CR90, SSK96, CQ94].

Como todos os métodos de modelagem naturalmente têm vantagens e desvantagens, a escolha de um ou outro método depende do sistema a ser modelado, dos objetivos da modelagem, da praticidade do método e, principalmente, do conhecimento que se tem sobre o instrumental usado na fundamentação do método. É claro que qualquer estudo relacionado com a modelagem de SED deve intencionar a definição de métodos que possam ser extensivamente usados. Neste sentido, entenda-se a praticidade do método como a facilidade da construção do modelo, a facilidade da aplicação de técnicas (preferencialmente, automáticas) para a análise e o controle do sistema e a facilidade da interpretação dos resultados. Além disso, o instrumental, normalmente matemático, usado na fundamentação do modelo deve ser o mais simples possível, sem o comprometimento da exatidão requerida com o sistema físico.

1.2 Objetivos desta tese

O enfoque deste trabalho está no tratamento das propriedades qualitativas dos SED e, neste contexto, devido à sua abrangência, a abordagem normalmente usada é o controle supervisório para SED, introduzida por Ramadge e Wonham [RW87]. Embora bem consolidada, a abordagem original de Ramadge e Wonham (RW) apresenta alguns problemas principalmente em se tratando de sistemas complexos. Muitos destes problemas já foram abordados na literatura. Outros, porém, ainda permanecem em aberto e não têm sido muito explorados. Em particular, a linguagem formal usada na descrição dos requisitos a serem atendidos por um agente de controle nem sempre possibilita um claro entendimento destes requisitos. Além disso, pouca ênfase foi dada, até o momento, a soluções genéricas para uma classe de problemas com uma mesma estrutura (uma exceção são os trabalhos de Chen e Lin [CL00, CL01b, CL01a]).

Desafio e motivação

As observações anteriores motivaram o desenvolvimento desta tese, a qual basicamente propõe um modelo para os SED adaptável a certas mudanças de configuração dos processos. Com relação à especificação de requisitos, a linguagem usada é simples, clara, flexível e genérica no sentido de permitir o tratamento de especificações parametrizadas. O desafio deste desenvolvimento é atingir estes objetivos sem penalizar demasiadamente a sistematização do projeto dos controladores.

Proposta

Esta tese apresenta um modelo para SED, o qual será denominado *Sistemas de Transição de Estados com Pré-Condições* (STE-PC), baseado na interação entre um sistema de transição de estados e uma coleção de dados. O sistema de transição de estados é usado para capturar a estrutura do sistema bem como servir como âncora para a definição dos comportamentos do sistema enquanto que a coleção de dados é usada para fornecer informações complementares que possam auxiliar a descrição dos comportamentos desejáveis. A linguagem para a descrição destes comportamentos consiste de um conjunto de fórmulas lógicas e simbólicas, também chamadas de pré-condições, restringindo a ocorrência dos eventos. Mais especificamente, a cada transição do sistema de transição de estados pode ser associada uma fórmula lógica cujos termos consistem de variáveis refletindo os estados locais dos subsistemas ou valores dos componentes da coleção de dados.

Em resumo, este trabalho propõe (a) um modelo para SED, (b) uma linguagem para a descrição dos comportamentos desejáveis e (c) algoritmos para a síntese de controladores. Também são requisitos integrantes desta tese, os seguintes:

1. A classe dos sistemas tratados pelos STE-PC deve ser igual ou maior do que a classe dos sistemas tratados pela abordagem RW.

2. O modelo deve permitir que sistemas complexos sejam construídos a partir de subsistemas mais simples (composição de subsistemas).

3. A linguagem para a descrição dos comportamentos desejáveis deve permitir que especificações complexas possam ser construídas a partir de especificações mais simples (composição de especificações).

4. A linguagem para a descrição dos comportamentos desejáveis deve ser intuitiva e flexível.

5. A linguagem para a descrição dos comportamentos desejáveis deve permitir a construção de especificações parametrizadas, isto é, classes de especificações.

6. O modelo do sistema e a linguagem para a descrição dos comportamentos desejáveis devem ser expansíveis, isto é, devem permitir que no futuro uma classe maior de problemas possam ser tratados (tal como sistemas temporizados).

7. Os procedimentos algorítmicos para a síntese de controladores devem tratar especificações parametrizadas. Nestes casos, os resultados produzidos devem ser soluções parametrizadas, isto é, classes de soluções.

8. As soluções produzidas pelos procedimentos algorítmicos para a síntese de controladores devem ser reduzidas.

9. As soluções produzidas pelos procedimentos algorítmicos para a síntese de controladores devem ser ótimas, isto é, minimamente restritivas.

10. Os procedimentos algorítmicos para a síntese de controladores devem ser computáveis e computacionalmente eficientes.

Contribuição

Em comparação com a abordagem RW, a inclusão de uma coleção de dados ao modelo, associada à avaliação on-line de fórmulas, permite uma representação mais compacta dos sistemas, a captura de comportamentos não regulares e a construção e resolução off-line de modelos genéricos para uma classe de problemas com a mesma estrutura. Estes são ganhos quantitativos que já justificariam este trabalho. No entanto, ainda existem ganhos qualitativos que tornam os STE-PC bastante práticos. Além da linguagem usada na construção das especificações ser do domínio de profissionais de várias áreas do conhecimento, as diferentes formas para a composição de diferentes especificações facilitam muito a descrição dos requisitos desejáveis [2].

[2] É claro que estas contribuições qualitativas necessitam uma comprovação prática, a qual está além do escopo desta tese.

Limitações

A classe dos sistemas tratados nesta tese não inclui sistemas temporizados ou híbridos. Na modelagem da coleção de dados, somente funções computáveis [3] podem ser usadas. Os métodos algorítmicos só se aplicam a uma sub-classe de STE-PC, a qual será descrita no Capítulo 4. O Procedimento 5.2.1 para a síntese de controladores livres de deadlocks não elimina livelocks.

1.3 Artigos publicados

Ao longo do desenvolvimento dos STE-PC foram publicados os seguintes artigos:

[OCK03a] Artigo publicado no VI SBAI em 2003.

Resumo: *Este trabalho propõe um modelo para sistemas dirigidos pela ocorrência de eventos discretos (SED) consistindo de um autômato finito determinístico sincronizado com uma coleção de dados. O modelo, denominado sistemas a eventos discretos guardados (SED-G) é realizado através de autômatos guardados, onde a habilitação de cada conjunto de eventos síncronos pode estar condicionada à satisfação de um predicado definido em função do estado discreto do autômato e em função das variáveis definidas pela coleção de dados. O correlacionamento entre estados discretos, eventos e restrições é representado textualmente na forma de regras. Cada regra define as condições que habilitam a ocorrência de um conjunto de eventos síncronos e as condições a serem satisfeitas após a ocorrência daqueles eventos. As principais características dos SED-G são a expressão de comportamentos não regulares e, dependendo das propriedades estruturais dos componentes constituintes do sistema, a construção de modelos genéricos através da parametrização de valorações iniciais, finais, etc.*

[OCK03b] Artigo publicado no VI SBAI em 2003.

Resumo: *Os SED-G são sistemas dirigidos pela ocorrência de eventos discretos cujas restrições de comportamento são expressas em termos de predicados (ou guardas) restringindo a ocorrência de certos eventos. A habilitação dos eventos é condicionada à satisfação dos predicados a eles (possivelmente) associados. Um supervisor garantindo a satisfação dos requisitos dados pode ser construído somente se todas as restrições forem impostas sobre a ocorrência de*

[3] Por computável, entenda-se a existência de uma seqüência finita e contável de passos bem definidos capaz de produzir o resultado correto da função [HMU01].

eventos controláveis. Este artigo utiliza uma técnica denominada antecipação de guardas, a ser aplicada sobre as restrições impostas sobre a ocorrência de eventos não controláveis, para permitir a construção de supervisores nos casos em que uma especificação requer a habilitação condicionada de eventos não controláveis. Esta técnica consiste basicamente do teste antecipado das restrições associadas aos eventos não controláveis para o instante em que o sistema está para habilitar os eventos controláveis que precedem os eventos não controláveis restritos. Como resultado da antecipação de guardas, a satisfação das condições para a habilitação de eventos não controláveis são implicadas pelo teste antecipado realizado durante a habilitação dos eventos controláveis precedentes. Isto garante a existência de supervisores (mais restritivos, é claro) mesmo nos casos onde a especificação original restringe a ocorrência de eventos não controláveis.

[OCK04] Artigo publicado no "11^{th} IFAC Symposium on Information Control Problems in Manufacturing" em 2004.

Abstract: *This work presents a model of discrete event systems (DES) consisted of a synchronized deterministic finite automaton with a data structure. The model, denominated DES with guards is realized through automata with guards, where the habilitation of each group of synchronized events may be conditioned to the satisfaction of a predicate related to the discrete state of the automata and to the defined variables by data structure. The most important characteristics of DES with guards are the expression of non-regular behaviors and, depending on the structural properties of the components of the system, the construction of generic models through the parametrization of initial values, final values, etc.*

[OCK05] Supervisory Control Problem for Parameterized and Non-Regular Discrete Event Systems.

Abstract: *This article presents a model for discrete-event systems in which the plant consists of a finite State Transition System, or STS, equipped with a data collection. The data collection introduces variables whose values are updated by operations commanded by the discrete transitions in the STS. The specifications of desirable behaviors consist of predicates related to the occurrence of the events. The supervisor, based on the sequence of events occurred in the past, controls the system evolution through the disabling of events. The decision of disabling events originates from the evaluation of predicates computed from the specification of desirable behaviors. A method for the synthesis of supervisors attaining a specification with minimally restrictive control is presented. The main characteristics of our framework are the possibility*

of capturing non-regular behaviors and the construction of parameterized models which lead to generic solutions for a given class of problems.

1.4 Organização do documento

A estrutura desta tese consiste dos seguintes capítulos:

- No Capítulo 2 são introduzidos os dois modelos para SED mais relevantes ao modelo a ser apresentado neste documento.
- No Capítulo 3 são introduzidos "teoricamente" os conceitos relacionados com os STE-PC.
- No Capítulo 4 são descritos formalmente os conceitos relacionados com a síntese de controladores. Neste capítulo também é apresentado o algoritmo para a síntese da máxima especificação controlável.
- No Capítulo 5 são comentados os problemas relacionados com a síntese de controladores não bloqueantes.
- No Capítulo 6, os algoritmos para a síntese de controladores são aplicados sobre alguns exemplos.
- Finalmente, no Capítulo 7 são apresentadas as conclusões sobre o trabalho realizado bem como uma série de perspectivas sobre possíveis trabalhos futuros.

Capítulo 2

Modelos para SED

Neste capítulo são introduzidos dois modelos para SED que, no contexto desta tese, são de particular interesse. Devido à sua importância no contexto dos SED, a teoria clássica baseada em autômatos é explorada mais detalhadamente. O modelo FSMwP – *Finite State Machine with Parameters* é introduzido devido à sua similaridade com o modelo a ser apresentado nesta tese.

2.1 Teoria clássica baseada em autômatos

A teoria de controle supervisório para sistemas a eventos discretos, introduzida em [RW87], modela os SED como uma planta interagindo com um supervisor. A planta é capaz de gerar eventos expontaneamente. O supervisor, em função da seqüência de eventos ocorrida no passado, controla a evolução futura do sistema através da inibição da ocorrência de alguns eventos na planta. O comportamento da planta é descrito em termos de duas linguagens consistindo de cadeias de eventos. A linguagem L, de *cadeias parciais*, corresponde às tarefas incompletas do sistema e a linguagem L_m, de *cadeias marcadas*, corresponde às tarefas completas do sistema. Estas linguagens, normalmente infinitas, são geradas por dispositivos finitos denominados *geradores*. Um gerador G é uma estrutura de controle semelhante a um autômato (finito), na qual, em geral, somente um subconjunto próprio da totalidade de eventos pode ocorrer em cada estado [RW87]. As notações $L(G)$ e $L_m(G)$ enfatizam o gerador usado para a geração de L e L_m, respectivamente.

2.1.1 Geradores e a modelagem das plantas

Definição 2.1.1 (Gerador). *Um* gerador G *é a quíntupla* $\langle Q, \Sigma, \delta, q_0, Q_m \rangle$ *consistindo dos seguintes elementos:*

1. Q *é um conjunto finito de* estados *(discretos).*
2. Σ *é um conjunto finito de símbolos denominados* eventos. *A notação* Σ^* *representa o conjunto de todas as cadeias de símbolos em* Σ*, incluindo a cadeia vazia* ε*.*
3. $\delta : Q \times \Sigma \to Q$ *é uma função parcial de transição de estados definida para cada* $q \in Q$ *e um subconjunto de símbolos* $\rho \in \Sigma$*. A notação* $\delta(q, \sigma)!$ *indica que* $\delta(q, \sigma)$ *é definida. Para um estado* $q \in Q$*, o conjunto de eventos fisicamente possíveis de ocorrerem a partir de* q *é* $\Sigma(q) = \{ \sigma \in \Sigma \mid \delta(q, \sigma)! \}$*.*
4. $q_0 \in Q$ *é o* estado inicial *do gerador.*
5. $Q_m \subseteq Q$ *é um conjunto de* estados marcados.

A função δ é estendida para a função parcial $\delta^* : Q \times \Sigma^* \to Q$ pelas regras:

- $\delta^*(q, \varepsilon) = q$, e
- $\delta^*(q, s\sigma) = \delta(\delta^*(q, s), \sigma)$,

contanto que $q' = \delta^*(q, s)!$ e $\delta(q', \sigma)!$.

O conjunto de eventos Σ é particionado em Σ_u e Σ_c. O conjunto Σ_c é o conjunto dos eventos *controláveis* e corresponde ao conjunto de eventos que podem ser inibidos por algum agente externo. O conjunto Σ_u é o conjunto dos eventos *não-controláveis* contendo aqueles eventos que não podem ser inibidos.

O *comportamento fechado* de G é $L(G) = \{ s \in \Sigma^* \mid \delta^*(q_0, s)! \}$ e o *comportamento marcado* de G é $L_m(G) = \{ s \in \Sigma^* \mid \delta^*(q_0, s)! \in Q_m \}$. É importante notar que $\emptyset \subseteq L_m(G) \subseteq L(G)$ e $\varepsilon \in L(G)$. Para $H \subseteq \Sigma^*$, o *prefixo-fechamento* de H é dado por $\overline{H} = \{ s \in \Sigma^* \mid su \in H \text{ para algum } u \in \Sigma \}$. A linguagem H é *prefixo-fechada* se $\overline{H} = H$. A linguagem $L(G)$ é prefixo-fechada.

Um estado $q \in Q$ é *acessível* se existe uma cadeia $s \in \Sigma^*$ com $\delta^*(q_0, s)!$ e $\delta^*(q_0, s) = q$. O subconjunto de todos os estados acessíveis em G é $Reach^Q(G) = \{ q \in Q \mid \exists\, s \in \Sigma^*,\ \delta^*(q_0, s) = q \}$; G é *acessível* se

$Reach^{Q}(G) = Q$. Um estado $q \in Q$ é *co-acessível* se existe $s \in \Sigma^*$ tal que $\delta^*(q,s) \in Q_m$, isto é, q é co-acessível se a partir dele existe um caminho (uma seqüência de eventos possível em G) até um estado marcado. O subconjunto de todos os estados co-acessíveis em G é $CoReach^{Q}(G) = \{ q \in Q \mid \exists\, s \in \Sigma^*, \delta^*(q,s) \in Q_m \}$; G é *co-acessível* se $CoReach^{Q}(G) = Q$. G é *não-bloqueante* se cada estado acessível é co-acessível, ou em outras palavras, G é não-bloqueante se toda cadeia gerada por G é prefixo de uma cadeia marcada também gerada por G, isto é, se $L(G) = \overline{L_m(G)}$. G é trim se todos os seus estados são acessíveis e co-acessíveis.

Graficamente, um gerador é representado como um grafo, cujos vértices correspondem aos estados discretos do sistema. Os eventos do sistema definem (e rotulam) as transições entre os estados. Uma transição definida por um evento controlável é diferenciada de outra, definida por um evento não-controlável, por um pequeno traço cruzando o arco representando aquela transição. Uma transição sem um estado origem define o estado destino como inicial. Os estados marcados são indicados por um círculo duplo ou por um asterisco (*).

Exemplo 2.1.1 (Linha realimentada – LR [Won98]). *Uma linha realimentada é constituída de uma planta com duas máquinas simples (M_1 e M_2), dois armazéns (A_1 e A_2) e um testador (T_1). As máquinas e o testador têm estados* Idle, *indicando ocioso, e* Working, *indicando em operação. O testador pode aceitar um produto ou refutar o produto. No caso da refutação, o produto refutado deve ser reprocessado pela máquina M_2. Os armazéns A_1 e A_2 são definidos como tendo a capacidade de armazenar, respectivamente, até N e M produtos (onde N e M são parâmetros do projeto) e as operações de inserção e de remoção. As mudanças de estado das máquinas são indicadas pelos eventos discretos s_i e f_i para $i = 1,2$. Os eventos s_i sinalizam os inícios das operações das máquinas e os eventos f_i, os finais das operações. O início do teste é indicado pelo evento* t *e o final do teste, pelos eventos* a *(de aceito) e* r *(de refutado).*

Depois que M_1 termina o processamento de uma peça, M_1 deposita a peça temporariamente no armazém A_1. Em seguida, M_2 retira uma peça do armazém A_1, processa a peça e a armazena em A_2. Finalmente, o testador retira uma peça de A_2, testa-a e, caso a peça seja refutada, ela é depositada novamente em A_1. Estas interações entre os diversos componentes da planta são ilustradas na Figura 2.1.

O objetivo do controle da linha realimentada é evitar os underflows *e* overflows *dos armazéns e o* deadlock *da planta.*

Os geradores modelando os subsistemas são ilustrados na Figura 2.2.

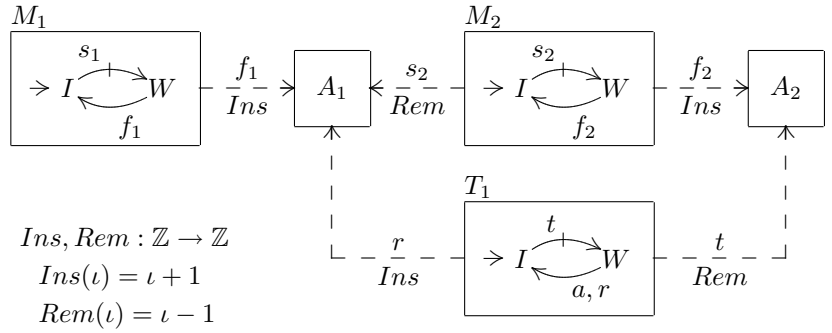

Figura 2.1: Linha realimentada.

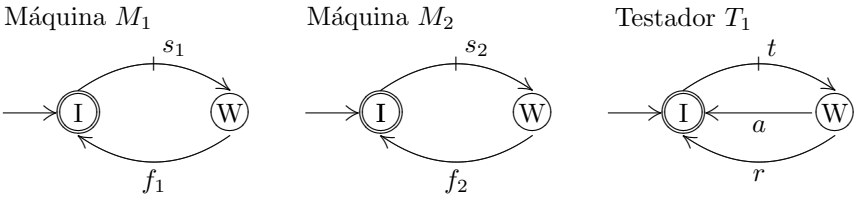

Figura 2.2: Geradores dos subsistemas da linha realimentada.

O modelo da planta é o gerador obtido pela composição sincronizada dos diversos subsistemas. Formalmente, a operação de composição síncrona usada na construção da planta é definida como:

Definição 2.1.2 (Composição síncrona [Hoa85]). *Dadas duas máquinas de estados determinísticas* $M_1 = \langle Q_1, \Sigma_1, \delta_1, q_{1,0}, Q_{1,m}\rangle$ *e* $M_2 = \langle Q_2, \Sigma_2, \delta_2, q_{2,0}, Q_{2,m}\rangle$, *a* composição síncrona *de* M_1 *e* M_2, *denotada* $M_1 \parallel M_2$, *é a máquina* $M_1 \parallel M_2 = \langle Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta, \langle q_{1,0}, q_{2,0}\rangle, Q_{m,1} \times Q_{m,2}\rangle$, *onde, para cada* $q = \langle q_1, q_2\rangle \in Q_1 \times Q_2$ *e* $\sigma \in \Sigma_1 \cup \Sigma_2$*:*

$$\delta(q,\sigma) = \begin{cases} \langle \delta_1(q_1,\sigma), \delta_2(q_2,\sigma)\rangle & \text{se } \delta_1(q_1,\sigma)!,\ \delta_2(q_2,\sigma)! \text{ e } \sigma \in \Sigma_1 \cap \Sigma_2 \\ \langle \delta_1(q_1,\sigma), q_2\rangle & \text{se } \delta_1(q_1,\sigma)! \text{ e } \sigma \in \Sigma_1 - \Sigma_2 \\ \langle q_1, \delta_2(q_2,\sigma)\rangle & \text{se } \delta_2(q_2,\sigma)! \text{ e } \sigma \in \Sigma_2 - \Sigma_1 \end{cases}$$

Assim, os eventos comuns de $M_1 \parallel M_2$ ocorrem sincronamente, enquanto que os demais, assincronamente. Se $\Sigma_1 = \Sigma_2 = \Sigma$, então $L(M_1 \parallel M_2) = L(M_1) \cap L(M_2)$ e $L_m(M_1 \parallel M_2) = L_m(M_1) \cap L_m(M_2)$, uma vez que todos os eventos ocorrem sincronamente. Caso não existam eventos comuns entre os

subsistemas, então todos os eventos ocorrem assincronamente e, neste caso, o produto acima é denominado *produto assíncrono*. O estado inicial da composição $M_1 \,||\, M_2$ é $q_0 = \langle q_{0_1}, q_{0_2} \rangle \in Q_1 \times Q_2$, onde $q_{0_1} \in Q_1$ é o estado inicial de M_1 e $q_{0_2} \in Q_2$ é o estado inicial de M_2. A Figura 2.3 ilustra o gerador modelando a linha realimentada obtido a partir da composição síncrona dos geradores da Figura 2.2. Na figura, os estados são da forma xyz onde x, y e z denotam, respectivamente, os estados locais das máquinas M_1 e M_2 e o estado local do testador T_1.

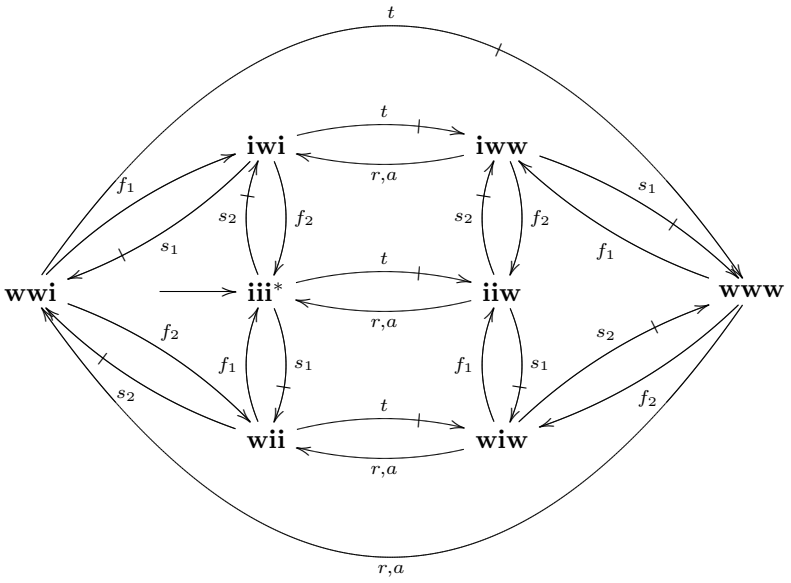

Figura 2.3: Modelo clássico da linha realimentada.

2.1.2 Supervisores e especificações

Um agente externo de controle, baseado na seqüência de eventos que conduziu o sistema a um estado qualquer, pode naquele instante inibir alguns eventos controláveis (se houver algum) e, desta forma, restringir o passo seguinte da evolução do sistema a um subconjunto dos estados possíveis. Formalmente, a seleção dos eventos possíveis no próximo passo da evolução do sistema é expressa em termos de *entradas de controle*.

Definição 2.1.3 (Entradas de controle válidas). *Seja* $G = \langle Q, \Sigma, \delta, q_0, Q_m \rangle$ *um gerador com* $\Sigma = \Sigma_u \cup \Sigma_c$. *O conjunto de todas as* entradas de controle válidas *associado a* G *é o conjunto* $\Gamma_G = \{ \gamma \in 2^{\Sigma} \mid \Sigma_u \subseteq \gamma \subseteq \Sigma \}$.

Em palavras, uma entrada de controle válida é qualquer subconjunto de eventos contendo, no mínimo, todos os eventos não-controláveis ($\Sigma_u \subseteq \gamma$).

O conjunto de entradas de controle válidas é fechado tanto em relação à união de conjuntos quanto em relação à intersecção de conjuntos. Isto é, se γ e γ' estão em Γ_G, então $\gamma \cup \gamma'$ também está em Γ_G (fecho em relação à união de conjuntos [GR87]) e se $\gamma \in \Gamma_G$, $\gamma \subset \gamma' \subset 2^{\Sigma}$, então $\gamma' \in \Gamma_G$ (fecho em relação ao confinamento de conjuntos [TW94]).

Um *supervisor* é um agente externo de controle que associa a cada cadeia de eventos possível em G, uma entrada de controle válida. O sistema resultante da interação entre a planta G e o supervisor V é denotado V/G. O comportamento de V/G é denominado comportamento em *malha fechada* de G. Como o controle exercido por V é apenas restritivo, o comportamento em malha fechada de G está incluído no comportamento em *malha aberta* [1] de G. As definições a seguir formalizam os conceitos relacionados à supervisão dos SED, segundo [RW89].

Definição 2.1.4 (Supervisor). *Um* supervisor V *é um agente externo de controle que realiza o mapa* $V : L \rightarrow \Gamma_G$, *associando a cada cadeia de eventos gerada,* $s \in L$, *uma entrada de controle válida* $\gamma = V(s) \in \Gamma_G$.

O supervisor deve ser projetado para atender os requisitos de uma especificação dada. Tal especificação é dada em termos de uma sub-linguagem da linguagem gerada $L(G)$ ou marcada $L_m(G)$. Considerando-se uma especificação da forma $K_m \subseteq L_m(G)$, representando as tarefas cujas finalizações sob supervisão são permitidas, o problema é encontrar (se possível) um supervisor V, cujo comportamento em malha fechada satisfaça $L_m(V/G) = K_m$. Do mesmo modo, considerando-se uma especificação $K \subseteq L(G)$, representando o comportamento desejado sob supervisão e fisicamente possível em G, o problema é encontrar (se possível) um supervisor V tal que $L(V/G) = K$. Em ambos os casos, a especificação K é denominada *linguagem alvo*. Em geral, a linguagem alvo é obtida a partir da composição síncrona entre os geradores de especificações parciais mais simples. Por exemplo, no caso da linha realimentada, se $M = 1$ e $N = 3$, as especificações evitando os underflows e overflows nos armazéns são geradas pelos geradores da Figura 2.4. O gerador da esquerda está associado ao armazém A_1 que está limitado a armazenar até 3 peças e o gerador da direita está associado ao armazém A_2 que pode conter no máximo 1 peça. Observa-se que no estado u_3, o evento f_1 não está habilitado e no estado v_1 o evento f_2 não está habilitado. A especificação global resultante do produto síncrono entre estes dois geradores é ilustrada na Figura 2.5.

[1] Comportamento possível de G sem a supervisão de V.

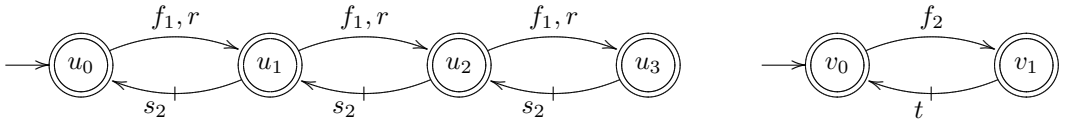

Figura 2.4: Especificações parciais em termos de geradores.

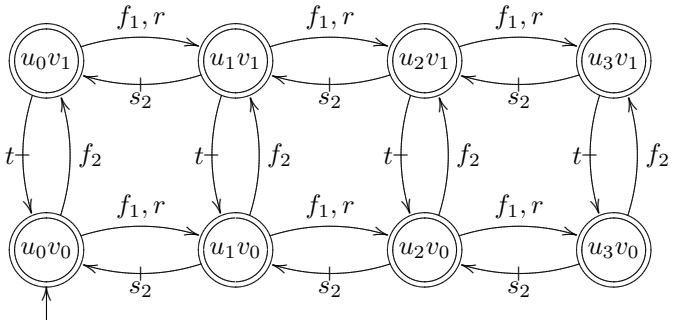

Figura 2.5: Especificação total.

Definição 2.1.5 (Comportamento em malha fechada). *O comportamento em* malha fechada *resultante da interação entre o SED G e o supervisor V (ou, simplesmente, o comportamento do sistema V/G) é a linguagem $L(V/G) \subseteq L(G)$, satisfazendo:*

1. *$\varepsilon \in L(V/G)$, e*
2. *$s\gamma \in L(V/G)$ se, e somente se, $s \in L(V/G)$, $s\gamma \in L(G)$ e $\gamma \in V(s)$.*

O comportamento marcado *de V/G é $L_m(V/G) = L(V/G) \cap L_m(G)$.*

Como comentado anteriormente, o comportamento em malha fechada está contido no comportamento em malha aberta e este último é representado por um gerador. Portanto, é natural que um supervisor também seja representado por um gerador, desde que obedecidos certos requisitos. Os requisitos necessários para que um supervisor V representado como um gerador S atue sobre uma planta, também representada como um gerador G, são:

1. Se $s \in L(V/G)$, então s e $s\sigma$ estão em $L(S)$ somente se $\sigma \in V(s)$.
2. Se $s \in L(V/G)$, $s\sigma \in L(G)$ e $\sigma \in V(s)$, então $s\sigma \in L(S)$.

Em palavras, o primeiro requisito diz que se s é um comportamento em malha fechada, ele é um comportamento de S e os eventos σ, habilitados após a ocorrência de s pelo supervisor S, são eventos

possíveis na planta G após a ocorrência de s. O segundo requisito diz que se s é um comportamento em malha fechada, o comportamento $s\sigma$ na planta G só é possível se σ é habilitado pelo supervisor S após a ocorrência de s.

Enquanto o gerador G tem a função de capturar as seqüências de eventos que definem tarefas parciais e completas, o gerador S tem a função de associar a cada seqüência de eventos possível na planta sob supervisão, uma entrada de controle válida. Portanto, o conceito de tarefa não se aplica ao gerador S e, tão pouco, o conceito de marcação de estados. Apesar disto, convenientemente, considera-se que todos os estados do gerador S são marcados, isto é, $S = \langle X, \Sigma, \xi, x_0, X\rangle$. A evolução de S é sincronizada com a evolução de G, isto é, se na planta G ocorre uma transição σ para um estado $q \in Q$, simultaneamente, no supervisor S ocorre uma transição σ para um estado $x \in X$. Imediatamente após a ocorrência da transição para x, S inibe em G os eventos $\sigma' \notin \Sigma(x)$, enquanto S permanecer no estado x [2].

A evolução sincronizada entre S e G define o comportamento em malha fechada do sistema. Na prática, o comportamento em malha fechada é dado pelo gerador obtido da composição síncrona $S \parallel G$, significando que em malha fechada são permitidas somente aquelas transições fisicamente possíveis em G e habilitadas por S.

2.1.3 Controlabilidade e existência de supervisores

Pelo que foi exposto até aqui, intuitivamente se percebe que uma condição necessária para a existência de um supervisor é que qualquer prefixo da linguagem alvo K, seguido de qualquer evento não-controlável, deve produzir um prefixo de K. Este requisito qualifica K como controlável em relação à G. Esta noção de controlabilidade é formalmente introduzida a seguir.

Definição 2.1.6 (Controlabilidade). *Seja $G = \langle Q, \Sigma, \delta, q_0, Q_m\rangle$ um gerador e $K \subseteq L(G)$ uma especificação do comportamento desejável sob supervisão e fisicamente possível em G. A linguagem K é* controlável *em relação à $L(G)$ ou, K é $L(G)$-*controlável *se, e somente se,*

$$\forall\, s \in \overline{K},\ \forall\, \sigma \in \Sigma_u,\ s\sigma \in L(G) \Rightarrow s\sigma \in \overline{K}$$

ou, mais sucintamente, K é $L(G)$-controlável se, e somente se, $\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}$.

[2] Obrigatoriamente, $\sigma' \in \Sigma(q)$.

É importante notar que a noção de controlabilidade está intimamente relacionada ao mapa (realizado pelo supervisor) de entradas de controle válidas às cadeias geradas pela planta. Se K é não-controlável, é porque alguma cadeia em K tenta inibir na planta um evento não-controlável. A especificação da Figura 2.5 não é controlável uma vez que nos estados u_3v_0 e u_3v_1 os eventos f_1 e r são desabilitados e nos estados u_iv_1, para $0 \leq i \leq 3$, o evento f_2 é desabilitado.

A controlabilidade é necessária, mas não suficiente para garantir a existência de supervisores. Particularmente no caso de uma especificação controlável, dada em termos de uma linguagem marcada, a existência de supervisores só pode ser garantida se a especificação for fechada em relação à linguagem marcada L_m. Esta noção de fecho é sucintamente introduzida a seguir.

Seja $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ um conjunto de (sub)-tarefas completas e T uma tarefa completa cuja realização consiste da realização de cada $t \in \mathcal{T}$. Para um supervisor realizar uma especificação controlável $K_m \subseteq L_m(G)$, a qual permite a realização de T, a especificação deve permitir também, a realização de cada (sub)-tarefa $t \in \mathcal{T}$. Diz-se que K é *fechada em relação à linguagem* $L_m(G)$ quando para cada tarefa completa T, permitida por K_m, K_m permite também, a realização de todas as (sub)-tarefas necessárias a realização de T.

Definição 2.1.7 (L_m–fechamento). *Seja $G = \langle Q, \Sigma, \delta, q_0, Q_m\rangle$ um gerador e $K_m \subseteq L_m(G)$ uma linguagem marcada. K_m é* fechada em relação à linguagem $L_m(G)$ *ou, K_m é $L_m(G)$–*fechada*, se, e somente se, $K_m = \overline{K}_m \cap L_m(G)$* [3].

Finalmente, as condições necessárias e suficientes para a existência de supervisores são estabelecidas no teorema a seguir, baseado em [Won98, KG95].

Teorema 2.1.1 (Existência de supervisores para linguagens marcadas). *Seja $G = \langle Q, \Sigma, \delta, q_0, Q_m\rangle$ um gerador e $K_m \subseteq L_m(G)$ uma linguagem marcada não-vazia. Então, existe um supervisor V não-bloqueante para G, satisfazendo $K_m = L_m(V/G)$, se, e somente se,*

1. *K_m é $L(G)$–controlável, e*
2. *K_m é $L_m(G)$–fechada.*

Para especificações formuladas em termos da linguagem gerada, as condições necessárias e suficientes para a existência de supervisores são estabelecidas em [Won98, KG95] como um caso particular

[3] *Normalmente, a definição de $L_m(G)$–fechamento é tratada como um caso particular do conceito de $L(G)$-fechamento, o qual relaciona duas linguagens quaisquer definidas sob o mesmo alfabeto.*

do Teorema 2.1.1, onde todos os estados da planta são considerados marcados. Estas condições são reproduzidas aqui, pelo corolário a seguir.

Corolário 2.1.1 (Existência de supervisores para linguagens geradas). *Seja* $G = \langle Q, \Sigma, \delta, q_0, Q_m \rangle$ *um gerador e* $K \subseteq L(G)$ *uma linguagem não-vazia. Então, existe um supervisor* V *não-bloqueante para* G*, satisfazendo* $K = L(V/G)$*, se, e somente se,* K *é prefixo-fechada e controlável em relação à* $L(G)$*.*

2.1.4 Máxima linguagem controlável

Como visto anteriormente, um dos requisitos necessários à existência de supervisores é a controlabilidade da especificação em relação à $L(G)$. Evidentemente, nem todas as especificações são controláveis. Entretanto, lembrando-se que as especificações são dadas em termos de linguagens geradas ou marcadas e que qualquer linguagem não vazia contém linguagens mais simples, uma especificação controlável aproximada pode ser obtida. Esta seção, baseada em [Won98], demonstra como pode ser obtida a *máxima linguagem controlável* correspondendo à maior linguagem $L(G)$-controlável contida na especificação. Para tanto, seja $E \subseteq \Sigma^*$ uma linguagem especificando o comportamento desejável da planta G. O conjunto de sub-linguagens $L(G)$-controláveis de E é:

$$\mathcal{C}(E) = \{ K \subseteq E \mid K \text{ é controlável em relação à } G \}.$$

Em [Won98] é demonstrado que para quaisquer K_1 e K_2 em $\mathcal{C}(E)$, $K_1 \cup K_2$ também está em $\mathcal{C}(E)$, isto é, $\mathcal{C}(E)$ é não-vazio e fechado em relação à união. Em particular, [Won98] demonstra que existe um único elemento supremo, $sup\mathcal{C}(E) \in \mathcal{C}(E)$, igual à união de todas as sub-linguagens controláveis de $\mathcal{C}(E)$, isto é, $sup\mathcal{C}(E) = \bigcup_{K \in \mathcal{C}(E)} K$. Também, se E é $L_m(G)$-fechada, então cada $K \in \mathcal{C}(E)$ é $L_m(G)$-fechada e, conseqüentemente, $sup\mathcal{C}(E)$ é $L_m(G)$-fechada. Da mesma forma, se E é prefixo-fechada, isto é, $E = \overline{E}$, então cada $K \in \mathcal{C}(E)$ é prefixo-fechada, bem como $sup\mathcal{C}(E)$.

Com base nestas propriedades de $sup\mathcal{C}(E)$, em relação às propriedades de E, [Won98] conclui que:

- Para uma especificação prefixo-fechada e não vazia E, existe um supervisor V tal que $L(V/G) = sup\mathcal{C}(E)$.
- Para uma especificação $L_m(G)$-fechada e não vazia E, existe um supervisor V tal que $L_m(V/G) = sup\mathcal{C}(E)$.

Portanto, para uma especificação E, controlável [4] ou não, um supervisor V para G pode ser obtido, tal que o comportamento em malha fechada de G é $sup\mathcal{C}(E)$. Neste caso, $sup\mathcal{C}(E)$ é a máxima linguagem controlável contida em E e V é o *supervisor menos restritivo* que pode ser obtido para a especificação E. A computação de $sup\mathcal{C}(E)$ pode ser realizada pelo Algoritmo 2.1.1.

Algoritmo 2.1.1 Síntese da máxima linguagem controlável.

Dada uma planta $G = \langle Q, \Sigma, \delta, q_0, Q_m \rangle$ *com* $\Sigma = \Sigma_u \cup \Sigma_c$ *e uma linguagem regular* $E \subseteq L(G)$, *o algoritmo retorna a máxima linguagem controlável em* C. S *é usado como um gerador para* E.

/* $L(G') = L_m(G') = L(G)$ */
$G' \leftarrow G$ com todos os estados marcados.
/* *Tanto para* E *quanto para* G' *o alfabeto é* Σ. *Portanto,* $L_m(C_0) = L_m(G') \cap L_m(S) = L(G) \cap L_m(S) = L_m(S) = E$. */
$C_0 \leftarrow S \parallel G'$
$i \leftarrow -1$
Repita
  $i \leftarrow i + 1$
  Identificar os maus estados de C_i.
  /* *Os estados de* C_i *são da forma* (x, y), *onde* x *denota um estado de* S *e* y, *um estado de* G'. *Um mau estado de* C_i *é um estado* (x, y) *cujas transições não contém as transições não-controláveis de* y. */
  Obter C'_i eliminando os maus estados de C_i, bem como, suas transições.
  $C_{i+1} \leftarrow trim(C'_i)$
Até que $C_{i+1} = C'_i$
$C \leftarrow C'_i$
/* $sup\mathcal{C}(E) = L_m(C)$ */

A aplicação do Algoritmo 2.1.1 à linha realimentada e à especificação da Figura 2.5, produz o supervisor ilustrado na Figura 2.6. Na figura, os estados do supervisor são denotados $xiyjz$ com x, y e z representando, respectivamente, os estados locais de M_1, M_2 e T_1, i representando o estado do armazém A_1 e j, o estado do armazém A_2. É importante observar que, sob supervisão, o estado bloqueante $i3i1i$ não é acessível.

2.1.5 Resumo e observações

A abordagem RW utiliza linguagens (na prática, geradores) para modelar tanto a planta quanto o supervisor. A linguagem utilizada para a especificação dos requisitos desejáveis também é dada

[4] Evidentemente, se E é controlável, então $sup\mathcal{C}(E) = E$.

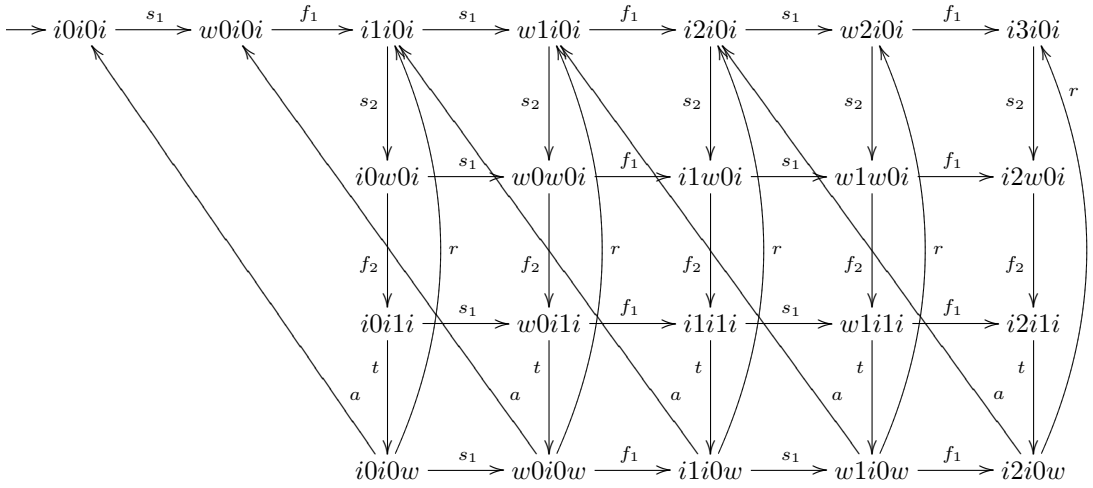

Figura 2.6: Supervisor para a linha realimentada.

em termos de geradores. Os geradores são autômatos com funções parciais de transição de estados. Portanto, todas as linguagens anteriores são regulares [HMU01].

Como pode ser percebido no exemplo descrito anteriormente, no modelo da planta não existem informações explícitas sobre as peças correntemente armazenadas nos armazéns. Tais informações são mantidas implicitamente pelos estados do supervisor. A conseqüência imediata disto é que uma alteração na capacidade dos armazéns implica o reprojeto completo do supervisor. A outra implicação direta da falta de informações explícitas sobre as peças nos armazéns é que dependendo da capacidade dos armazéns, o número de estados do supervisor pode crescer demasiadamente.

Apesar da generalidade da abordagem RW, sua aplicação é dificultada em função da dificuldade de se expressar os requisitos desejáveis do sistema. A especificação dos comportamentos, em geral, não é uma tarefa simples quando diferentes requisitos são incorporados em uma única especificação. O controle modular de SED, introduzido por [WR88] (primeiro trabalho relativo ao controle supervisório modular) objetiva simplificar estas especificações. Basicamente, a idéia por trás do controle modular é a conjunção de supervisores, cada qual atendendo uma especificação local, de modo que o supervisor global atenda simultaneamente todos os requisitos do conjunto de especificações locais [dQC00, CL91, TMHL97, CLL00, FK00, Li97].

Outro problema que dificulta a aplicação da abordagem RW em casos práticos é que um sistema complexo é construído como um produto de componentes mais simples e, conseqüentemente, o esforço computacional para a computação automática de controladores cresce ex-

ponencialmente com o número destes componentes [5]. Com um dos objetivos sendo o de diminuir este esforço computacional, técnicas para a descentralização do controle foram propostas [LW88, RW92, YL00b, YL02, JCK01, YL00a, BL00, JK99, KS97, OvS01, CL99].

Um outro enfoque que tenta reduzir a complexidade dos sistemas quando estes possuem espaço de estados e funções de transições estruturadas são os SED vetoriais [LW89, LW93, LW94, LW95, Li97, Hua91]. Nestes sistemas as estruturas são exploradas para obtenção de modelos compactos e computacionalmente eficientes. Os SED vetoriais definem uma sub-classe de SED com uma estrutura especial de estados. Nos SED vetoriais, cada estado é um vetor de inteiros ou naturais. As especificações são dadas em termos de desigualdades lineares ou predicados lineares. A ferramenta matemática empregada é a álgebra linear elementar e, sob certas condições, para diminuir a complexidade computacional dos métodos de busca, a programação linear inteira é usada. Entretanto, uma deficiência inerente aos SED vetoriais é que a propriedade de não bloqueio de um SED não pode ser facilmente verificada. Nos SED vetoriais, a listagem explícita do espaço de estados é evitada. Isto dá aos SED vetoriais a vantagem do desempenho, com relação à abordagem RW. Porém, ao mesmo tempo, torna a síntese de controladores não bloqueantes muito mais difícil devido à impossibilidade da checagem da alcançabilidade e co-alcançabilidade de cada estado.

Embora tanto a abordagem RW quanto os SED vetoriais sejam usados com sucesso na resolução de muitos problemas de controle, ambos têm estruturas homogêneas. Em outras palavras, o espaço de estados dos SED é modelado como um conjunto de nós abstratos e dos SED vetoriais, como um conjunto de inteiros não negativos. Na prática, os sistemas físicos podem consistir de sub-plantas com propriedades estruturais diferentes.

2.2 Máquina de estados finitos com parâmetros

O modelo das *Máquinas de Estados Finitos com Parâmetros* (FSMwP – Finite State Machines with Parameters), introduzidas em [CL00], será descrito nesta seção – não por se tratar de um modelo clássico para SED, mas sim, por ser, possivelmente, o modelo existente mais próximo daquele que será apresentado nesta tese.

Uma FSMwP consiste de uma máquina de estados finitos cujas transições são da forma $g \wedge \sigma \,/\, p := f(p)$, onde $\sigma \in \Sigma$ é o *evento* rotulando a transição, $p \in P$ é um vetor de *parâmetros*, $g \in G$ é um

[5] Para uma revisão sobre a complexidade da abordagem RW, ver [GW00].

predicado, o qual será denominado *guarda*, definido sobre os parâmetros p e $f : P \to P$ é uma *função de atualização dos parâmetros*. A típica transição ilustrada na parte superior da Figura 2.7 é interpretada como: se no estado q a guarda g é verdade e o evento σ ocorre, então o próximo estado é q' e os parâmetros são atualizados para $f(p)$.

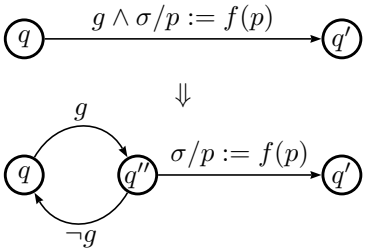

Figura 2.7: Uma transição FSMwP.

Na transição $g \wedge \sigma / p := f(p)$ todos os elementos são opcionais e diferentes combinações são permitidas. Se g é ausente, então a transição acontece quando o evento σ ocorre. Esta transição é chamada *transição de evento*. Se σ é ausente, então a transição acontece quando g se torna *true*. Esta transição é chamada *transição dinâmica*. Se $p := f(p)$ é ausente, então nenhuma atualização de parâmetros é realizada durante a transição. Em qualquer transição, no mínimo, ou σ ou g deve estar presente.

As transições contendo tanto o evento quanto a guarda podem ser decompostas em transições de evento e dinâmicas como mostra a Figura 2.7. Após estas decomposições, qualquer transição é denotada $q \xrightarrow{l} q'$, onde l pode ser, conforme o tipo da transição, um evento ou uma guarda.

Formalmente, uma FSMwP é definida como:

Definição 2.2.1 (FSMwP). *Uma FSMwP, é a 7-upla FSMwP* $= \langle \Sigma, Q, \delta, P, G, \langle q_0, p_0 \rangle, Q_m \rangle$, *na qual:*

- Σ *é um conjunto finito de* eventos.
- Q *é um conjunto finito e não vazio de* estados *discretos.*
- P *é um espaço vetorial. Um elemento* $p \in P$ *é um* vetor de parâmetros.
- G *é um conjunto de predicados definidos sobre os parâmetros* $p \in P$. *Um predicado* $g \in G$ *é denominado* guarda.

- p_0 *é o* valor inicial dos parâmetros *no* estado inicial q_0.
- $Q_m \subseteq Q$ *é um conjunto de estados* marcados *ou* finais.
- $\delta : \Sigma \times Q \times G \times P \to Q \times P$ *é a* função de transição. *Cada transição em* δ *é definida como ilustra a Figura 2.7.*

2.2.1 Modelo das plantas

Os sistemas consistindo de duas ou mais FSMwP podem ser modelados por uma única FSMwP resultante da composição paralela definida a seguir:

Definição 2.2.2 (Composição paralela de FSMwP). *Dados* $FSMwP_1 = \langle \Sigma_1, Q_1, \delta_1, P_1, G_1, \langle q_{0,1}, p_{0,1} \rangle, Q_{m,1} \rangle$ *e* $FSMwP_2 = \langle \Sigma_2, Q_2, \delta_2, P_2, G_2, \langle q_{0,2}, p_{0,2} \rangle, Q_{m,2} \rangle$, *a composição paralela de* $FSMwP_1$ *com* $FSMwP_2$ *é:*

$$FSMwP = FSMwP_1 || FSMwP_2 = \langle \Sigma_1 \cup \Sigma_2, Q_1 \times Q_2, \delta_1 \times \delta_2, P_1 \cup P_2, G_1 \cup G_2, (q_{0,1}q_{0,2}, p_{0,1}p_{0,2}), Q_{m,1} \times Q_{m,2} \rangle,$$

onde a função de transição $\delta = \delta_1 \times \delta_2$ *é definida como mostram as Figuras 2.8 e 2.9. Na figuras,* l_i *podem ser eventos ou guardas. A Figura 2.8 ilustra o caso em que* $l_1 \neq l_2$ *e a Figura 2.9 ilustra o caso em que* $l_1 = l_2$.

Para evitar ambigüidades na atualização dos parâmetros, nas composições paralelas é suposto que qualquer parâmetro em p é atualizado por, no máximo, uma FSMwP.

Para descrever o comportamento de uma FSMwP, uma *execução* da FSMwP é definida como a seqüência:

$$r = (q_0, p_0) \xrightarrow{l_1} (q_1, p_1) \xrightarrow{l_2} (q_2, p_2) \xrightarrow{l_3} (q_3, p_3) \xrightarrow{l_4} \ldots, \tag{2.1}$$

onde l_i é a i-ésima transição e (q_i, p_i) é o estado e os valores dos parâmetros depois da i-ésima transição. O *traço* s da execução r da Equação 2.1 é a seqüência de eventos e/ou guardas $s = l_1 l_2 l_3 l_4 \cdots$. A

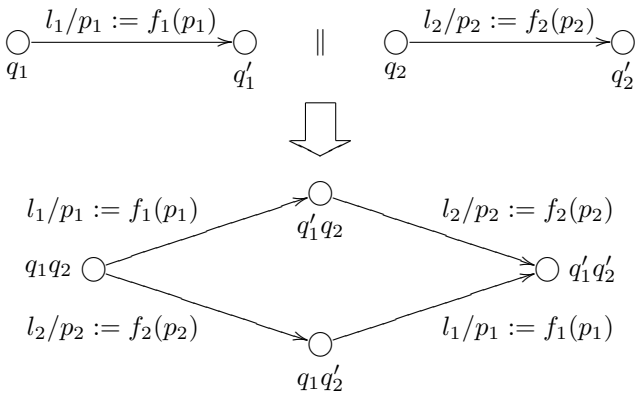

Figura 2.8: Função de transição δ para o caso $l_1 \neq l_2$.

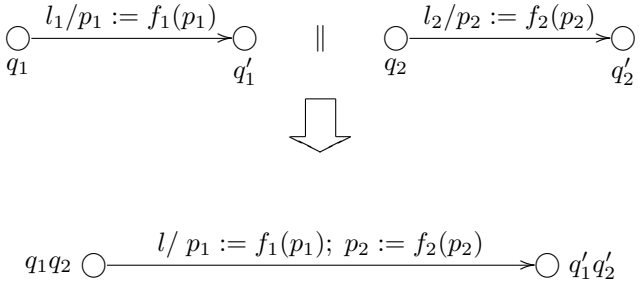

Figura 2.9: Função de transição δ para o caso $l_1 = l_2 = l$.

linguagem gerada por uma FSMwP é:

$$L(\mathsf{FSMwP}) = \{\, s \mid s \text{ é um traço da FSMwP} \,\}$$

e a *linguagem marcada* por uma FSMwP é:

$$L_m(\mathsf{FSMwP}) = \{\, s \in L(\mathsf{FSMwP}) \mid s \text{ termina em um estado marcado } q \in Q_m \,\}.$$

A composição paralela definida anteriormente requer a decomposição das transições. Isto implica que o número de estados da FSMwP composta pode crescer demasiadamente. Em [CL00], uma outra forma de composição paralela, sem a decomposição das transições, é apresentada. Entretanto, aquela forma não é geral e não permite o caso ilustrado na Figura 2.9.

2.2.2 Especificações para FSMwP

Em [CL01b] são descritos dois procedimentos para a síntese de controladores para sistemas modelados por FSMwP. Em ambos os casos, as especificações são do tipo estados ilegais (proibidos). Mais precisamente, dado um conjunto de estados ilegais $Q_b \subseteq Q$, todos os requisitos desejáveis do sistema supervisionado são descritos por transições indesejáveis para os estados em Q_b. Por exemplo, a Figura 2.10 ilustra a especificação "no estado q, $p \geq c$ é ilegal."

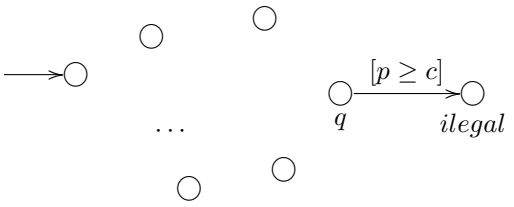

Figura 2.10: Especificação de estado ilegal $p \geq c$.

Na linha realimentada, a especificação evitando o overflow do armazém A_1 pode ser definida pela transição dinâmica $n > N$ partindo dos estados tal que M_1 e/ou T_1 estejam em idle para algum estado ilegal q_1. A especificação evitando o underflow em A_1, pode ser definida pela transição dinâmica $n < 0$ partindo dos estados tal que M_2 esteja em working para algum estado ilegal q_2. A especificação evitando o overflow de A_2, pode ser definida pela transição dinâmica $m > M$ partindo dos estados tal que M_2 esteja em idle para algum estado ilegal q_3. Finalmente, a especificação evitando o underflow em A_2, pode ser definida pela transição dinâmica $m < 0$ partindo dos estados tal que T_1 esteja em working para algum estado ilegal q_4.

2.2.3 Controladores para FSMwP

Dos dois procedimentos para a síntese de controladores descritos em [CL01b], apenas o procedimento off-line será explorado nesta seção. O procedimento on-line descrito em [CL01b] está fora do contexto deste trabalho, o qual trata da síntese de supervisores off-line.

O procedimento para a síntese off-line de controladores pode ser aplicado aos sistemas pertencentes a classe das FSMwP satisfazendo:

1. O sistema é determinístico.

2. O único parâmetro que não é atualizado pela própria FSMwP é o relógio global t.

3. As guardas são combinações de desigualdades da forma p ($>$, $\geq$, $<$, $\leq$) c, onde p é um parâmetro e c é uma constante.

O objetivo da supervisão é impedir que a FSMwP modelando a planta visite qualquer estado ilegal em Q_b. Assim, o comportamento legal do sistema é descrito pela linguagem:

$$E = \{\, s \in L(FSMwP) \mid s \text{ não visita estados ilegais em } Q_b \,\}.$$

Para atingir seu objetivo o controlador utiliza dois mecanismos de controle: a inibição de eventos *controláveis* em $\Sigma_c \subseteq \Sigma$ e a preempção de eventos *forçáveis* em $\Sigma_f \subseteq \Sigma$. A seleção dos eventos a serem habilitados ou forçados em cada passo da evolução do sistema é expressa em termos de *padrões de controle* definidos como:

$$\Gamma = \{\, g \subseteq \Sigma \mid \Sigma - \Sigma_c \subseteq g \ \wedge\ g \subseteq \Sigma_f \,\}.$$

O controlador é definido para ser o mapa $\gamma : L(FSMwP) \to \Gamma$ ou, em termos de realimentações de estados:

- $C_d : Q \to 2^{\Sigma_c}$ é o conjunto de eventos a serem desabilitados em q e
- $C_f : Q \to 2^{\Sigma_f}$ é o conjunto de eventos a serem forçados em q.

Então, se um traço s termina no estado q, tem-se:

$$\gamma(s) = \begin{cases} C_f(q) & \ldots \text{ se } C_f(q) \neq \emptyset \\ \Sigma - C_d(q) & \ldots \text{ em caso contrário.} \end{cases}$$

O comportamento do sistema controlado, denotado $L(FSMwP, \gamma)$, é definido indutivamente como:

1. $\varepsilon \in L(FSMwP, \gamma)$, onde ε denota o traço vazio, e
2. para todo traço $s \in L(FSMwP, \gamma)$ e todo evento $\sigma \in \Sigma$,

$$s\sigma \in L(FSMwP, \gamma) \quad \Leftrightarrow \quad s\sigma \in L(FSMwP) \ \wedge\ \sigma \in \gamma(s).$$

Se um controlador pode ou não ser construído para atender os requisitos descritos por uma linguagem K de modo que $L(FSMwP, \gamma) = K$ depende da controlabilidade de K. Esta noção de controlabilidade é definida como:

Definição 2.2.3 (Controlabilidade). *Uma linguagem $K \subseteq L(FSMwP, \gamma)$ é* controlável *com relação à $L(FSMwP, \gamma)$ e Γ se*

$$\forall\, s \in \overline{K}, \quad \exists\, g \in \Gamma, \qquad \Sigma_{L(FSMwP)}(s) - \Sigma_K(s) = \Sigma_{L(FSMwP)}(s) - g,$$

onde, $\Sigma_{L(FSMwP)}(s) = \{\, \sigma \in \Sigma \mid s\sigma \in L(FSMwP) \,\}$ e similarmente para $\Sigma_K(s)$.

Se uma linguagem legal E não é controlável, a máxima sub-linguagem controlável de E existe, pois Γ é fechado em relação à união. Com este resultado, o controlador menos restritivo pode ser encontrado. O algoritmo off-line para sintetizar tal controlador é descrito informalmente em seguida.

2.2.4 Síntese de controladores

A síntese do controlador seguro menos restritivo com relação a uma linguagem legal E, não controlável, inicia-se pela identificação dos estados ilegais e a análise dos seus estados vizinhos, isto é, os estados conectados a algum estado ilegal por meio de transições de evento ou dinâmicas.

- Se todas as transições partindo de um estado vizinho podem ser evitadas ou por inibição ou por preempção, então aquele estado vizinho é legal.
- Se as transições não puderem ser evitadas, então o estado vizinho é marcado ilegal.
- Se as transições puderem ser evitadas por algumas condições mas não por outras, então o estado deve ser particionado em sub-estados legais e ilegais.

Para uma descrição mais detalhada do procedimento para a síntese de controladores, faz-se necessário duas definições. São elas, o conjunto das transições do estado q para estados pertencentes a um conjunto de estados Q' é:

$$T(q, Q') = \{\, l \in \Sigma \cup G \mid q \xrightarrow{l} q' \in \delta \ \wedge\ q' \in Q' - \{q\} \,\}$$

e o conjunto de estados vizinhos dos estados ilegais é:

$$NB(Q_b) = \{\, q \in Q - Q_b \mid \exists\, l,\ \ l \in T(q, Q_b) \,\}.$$

Com estas definições, os três casos descritos anteriormente podem ser discutidos.

1. O estado vizinho é legal. Um estado vizinho $q \in NB(Q_b)$ é legal se ou existe uma transição de evento forçável $\sigma \in \Sigma_f$ que conduz o sistema a um estado seguro ou todas as transições conduzindo o sistema a estados ilegais são transições de evento controláveis. Formalmente, $q \in NB(Q_b)$ é legal se:

$$(\exists\, l' \in T(q, Q - Q_b))\ (l' \in \Sigma_f) \quad \vee \quad (\forall\, l \in T(q, Q_b))\ (l \in \Sigma_c).$$

2. O estado vizinho é ilegal. Um estado vizinho $q \in NB(Q_b)$ é ilegal se não existe nenhuma transição de evento forçável $\sigma \in \Sigma_f$ que conduz o sistema a um estado seguro e existem transições de evento não controláveis conduzindo o sistema a estados ilegais. Formalmente, $q \in NB(Q_b)$ é ilegal se:

$$(\forall\, l' \in T(q, Q - Q_b))\ (l' \notin \Sigma_f) \quad \wedge \quad (\exists\, l \in T(q, Q_b))\ (l \in \Sigma - \Sigma_c).$$

3. O estado vizinho pode ser particionado. Se nem o caso 1 e nem do caso 2 são verificados, então:

$$(\forall\, l' \in T(q, Q - Q_b))\ (l' \notin \Sigma_f) \quad \wedge \\ (\forall\, l'' \in T(q, Q_b))\ (l'' \notin \Sigma - \Sigma_c) \quad \wedge \quad (\exists\, l \in T(q, Q_b))\ (l \in G).$$

   Em palavras, não existem transições de evento forçáveis a partir de q a estados legais e nem transições de evento não controláveis de q para estados ilegais, mas existem transições dinâmicas de q para estados ilegais.

Neste último caso, a forma como o particionamento do estado vizinho é realizado depende da transição dinâmica ser gerada pelo ambiente ou pela própria FSMwP. Ambos os casos são analisados em seguida.

- Transições temporais.

Se l é disparado pelo ambiente, então l deve ser uma transição temporal, isto é, uma transição disparada pelo relógio global t. Seja G_t o conjunto de todas as transições temporais. Como l é disparada pelo relógio global, ela ocorrerá se não for possível preemptá-la por outras transições. Como não existem transições de evento forçáveis disponíveis, apenas uma outra transição temporal pode preemptar l. Dependendo da existência ou não desta outra transição temporal preemptando l, pode-se determinar se q é legal. Para este fim, para a transição temporal com guarda $g \in G_t$, denota-se o tempo em que g ocorre por $\tau(g)$. $\tau(g)$ pode ser facilmente calculado a partir de g. Por exemplo, $\tau([t \geq c]) = c - t$, $\tau(g_1 \vee g_2) = min\{\tau(g_1), \tau(g_2)\}$.

A condição para que todas as transições temporais de q para estados ilegais possam ser preemptadas por outras transições temporais de q para estados seguros, é dada por:

$$\tau\left(\bigvee_{g \in T(q,Q-Q_b) \cap G_t} g\right) \leq \tau\left(\bigvee_{g' \in T(q,Q_b) \cap G_t} g'\right).$$

Se esta condição não é satisfeita, então q é ilegal.

- Transições internas ao FSMwP.

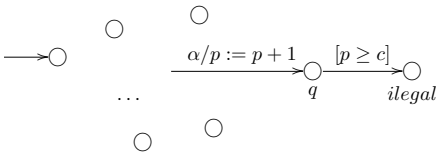

Figura 2.11: Transições dinâmicas internas.

Considere-se a situação ilustrada na Figura 2.11, onde α é um evento atualizando p como $p \leftarrow p + 1$ e $l = [p \geq c]$. Se depois de atualizar p, l se torna verdade, então α não pode ocorrer. Isto é, se $p + 1 \geq c$, então α deve ser desabilitado, se possível. Dependendo de α ser ou não um self-loop, o estado q pode ser particionado como:

  - Se α não é um self-loop, então q deve ser particionado no sub-estado ilegal q' e no sub-estado legal q como mostra a Figura 2.12. Na próxima iteração, a transição de evento restrita $\alpha \wedge [p + 1 \geq c]$ e $\alpha \wedge [p + 1 < c]$ será decomposta em transições de evento e transições dinâmicas. O estado q_a pode ser particionado no passo seguinte, indiferentemente.
  - Se α é um self-loop e não é controlável, então α pode ocorrer inúmeras vezes até disparar l. Portanto, o estado q deve ser considerado ilegal. Por outro lado, se α é controlável, então

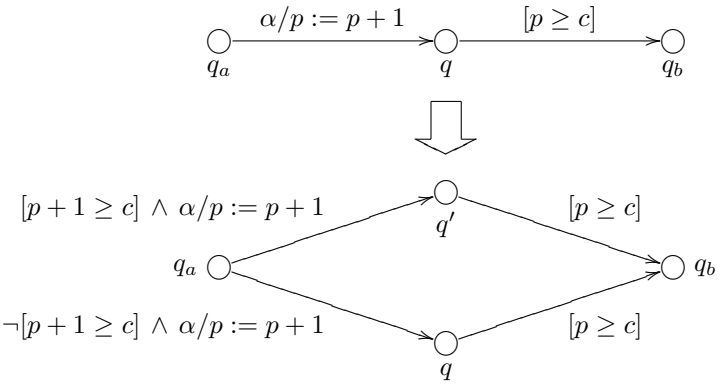

Figura 2.12: Particionamento do estado q (α não é um self-loop).

q precisa ser particionado nos sub-estados q e q' como ilustra a Figura 2.13, onde α deve ser desabilitado em q'.

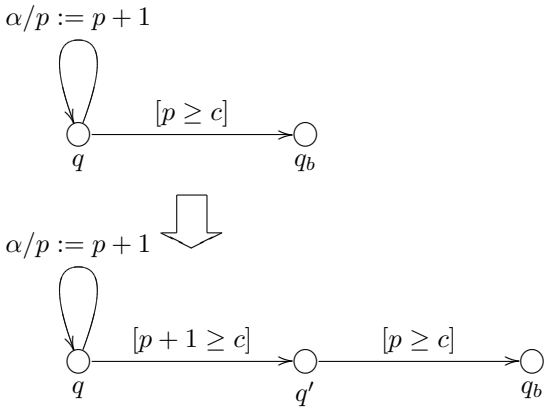

Figura 2.13: Particionamento do estado q (α é um self-loop).

A partir destas discussões, Chen e Lin obtiveram o procedimento 2.2.1 para a síntese off-line de controladores. No algoritmo, $[\alpha \rightarrow g]$ indica que a ocorrência de $\alpha \in \Sigma^*$ pode tornar verdadeira a guarda g [6].

Exemplo 2.2.1. *A Figura 2.14 ilustra uma máquina simples acoplada a um armazém com a capacidade de armazenar até N-peças. O evento s é controlável e o evento f é não controlável. Quando*

[6] Uma descrição mais detalhada deste algoritmo pode ser encontrada em [CL01b].

f *acontece, uma peça é armazenada no armazém. A especificação evitando o* overflow *no armazém consiste da transição dinâmica* $n > N$ *do estado* I *para o estado ilegal* q_b.

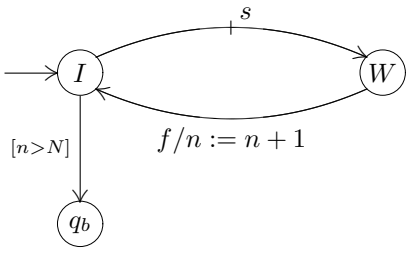

Figura 2.14: Exemplo de uma FSMwP.

Na primeira iteração do algoritmo, o estado I *é particionado em* I *e* I' *como mostra a Figura 2.15. Na figura, aparecem duas transições que podem ser decompostas em transições dinâmicas e de evento. O sistema resultante desta decomposição é mostrado na Figura 2.16*

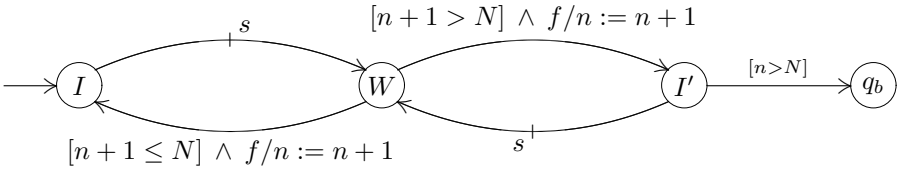

Figura 2.15: FSMwP *resultante do particionamento do estado* I *da Figura 2.14.*

Na Figura 2.16, os estados ilegais são q_2, I' *e* q_b. *Como existe uma transição dinâmica do estado* W *para o estado* q_2, W *deve ser particionado. O sistema resultante deste particionamento é ilustrado na Figura 2.17. A Figura 2.18 mostra o mesmo sistema após a decomposição das transições e a Figura 2.19 mostra o sistema simplificado.*

Finalmente, o último estado a ser particionado é o estado q_1 *da Figura 2.19. O sistema resultante e final é apresentado simplificadamente na Figura 2.20. Este sistema é para ser interpretado da seguinte forma: no estado* I, *o supervisor avalia as guardas* $n + 1 \leq N$ *e* $n + 1 > N$; *se a primeira é verdade, o sistema transita para o estado* I_1; *se a segunda é verdade, o sistema transita para o estado* I_2; *no estado* I_1 *a ação de controle é fixa e consiste da habilitação de* s; *no estado* I_2 *a ação de controle também é fixa e consiste da desabilitação de* s.

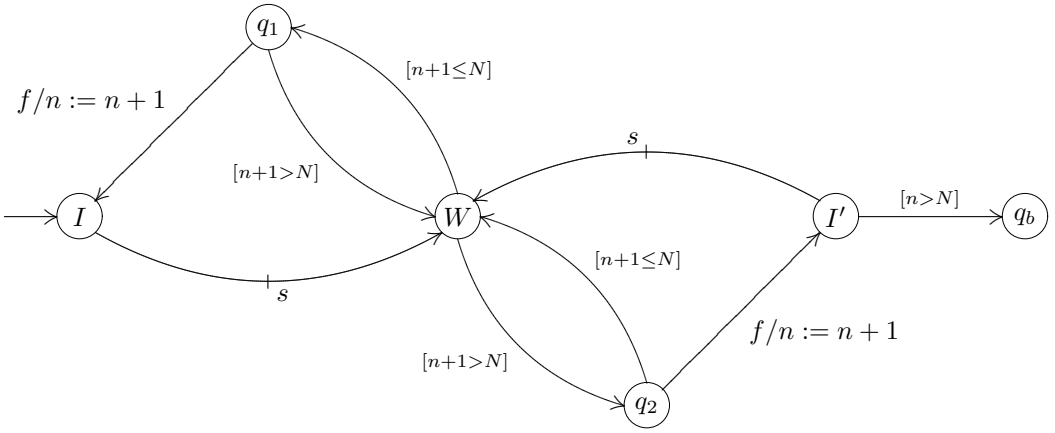

Figura 2.16: Decomposição das transições com guardas da Figura 2.15.

2.2.5 Observações

Como se pôde observar no Exemplo 2.2.1, mesmo para exemplos simples, a síntese de supervisores para FSMwP requer a utilização de ferramentas automatizadas. Tais ferramentas devem manipular, dependendo do sistema e da especificação, uma grande quantidade de transições de estado. O supervisor resultante deve avaliar expressões relacionais para determinar o estado do sistema e, conseqüentemente, fixar a ação de controle a ser exercida naquele momento.

Como ficará claro no decorrer dos próximos capítulos, o algoritmo de síntese proposto nesta tese é mais simples, pois não requer o particionamento de estados e nem a decomposição de transições. Entretanto, desconsiderando-se as características das FSMwP não presentes no modelo sendo proposto, o resultado final do algoritmo de síntese a ser apresentado no Capítulo 4 e do Algoritmo 2.2.1 são equivalentes. A principal diferença entre os dois supervisores está no objetivo da avaliação das guardas – no supervisor para FSMwP a avaliação das guardas determina o estado do sistema e no modelo a ser apresentado, a avaliação das guardas (pré-condições) induz a ação de controle.

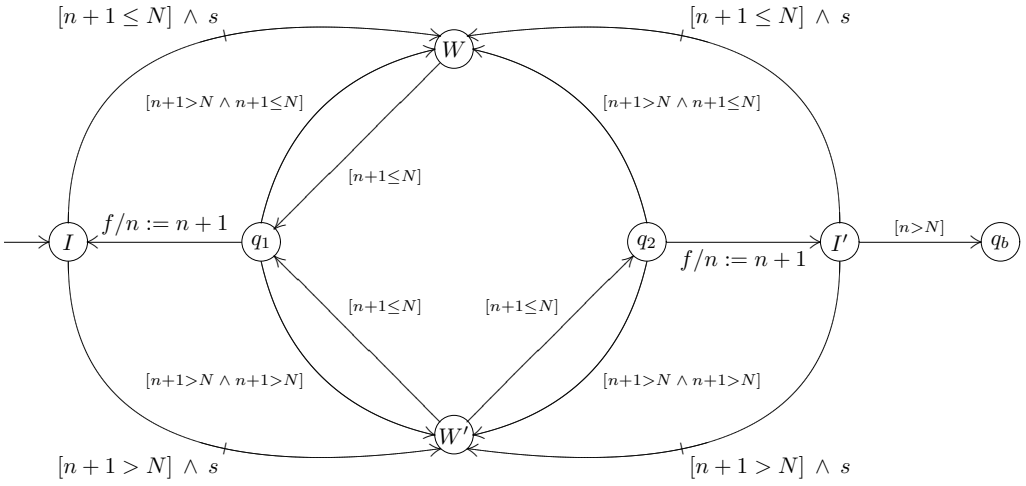

Figura 2.17: FSMwP resultante do particionamento do estado W da Figura 2.15.

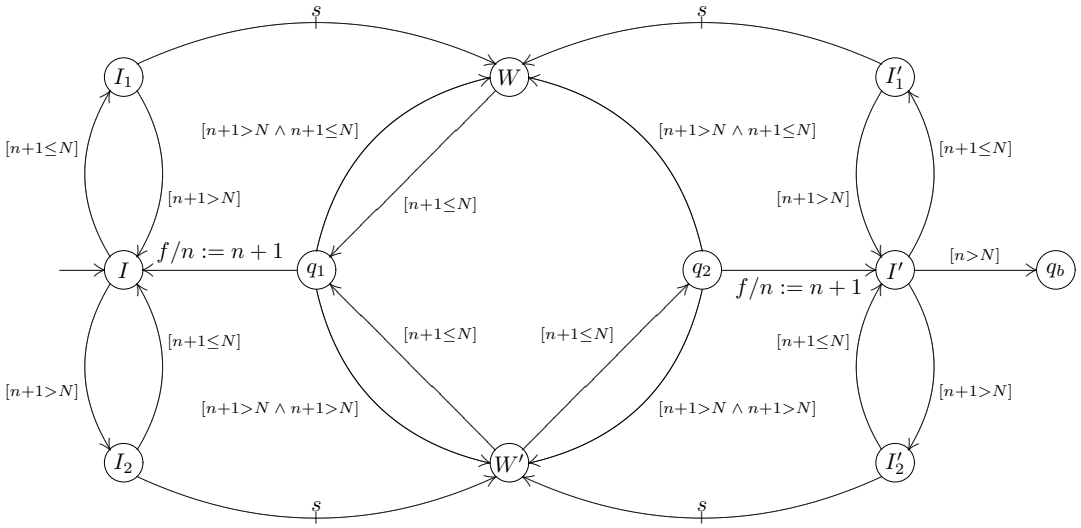

Figura 2.18: Decomposição das transições com guardas da Figura 2.16.

Algoritmo 2.2.1 Síntese de controladores seguros para FSMwP.

Dado o FSMwP = $\langle \Sigma,\ Q,\ \delta,\ P,\ G,\ (q_0, p_0),\ Q_m \rangle$, modelando o sistema, e um conjunto de estados ilegais $Q_b \subseteq Q$, este algoritmo retorna os mapas $C_d : Q \to 2^{\Sigma_c}$ e $C_f : Q \to 2^{\Sigma_f}$ de ações de controle associadas a cada estado $q \in Q$.

/* ***Inicialização*** */
$BS \leftarrow Q_b$ /* *Conjunto de estados ilegais* */
$NBS \leftarrow BS$ /* *Novo conjunto de estados ilegais* */
$PS \leftarrow Q - Q_b$ /* *Conjunto de estados pendentes* */
$NPS \leftarrow \emptyset$ /* *Novo conjunto de estados pendentes* */
$(\forall\ q\ \in\ Q)\,(C_d(q) = \emptyset$ e $C_f(q) = \emptyset)$ /* *Ações de controle* */

/* ***Decomposições de transições*** */
Enquanto $(\exists\ q \xrightarrow{g \wedge \sigma} q')\,(q,\ q'\ \in\ PS)$ **Faça**
$PS \leftarrow (PS - \{q\}) \cup \{q_1,\ q_2\}$
$\delta \leftarrow (\delta - \{q \xrightarrow{g \wedge \sigma} q'\}) \cup \{q_1 \xrightarrow{g} q_2,\ q_2 \xrightarrow{\neg g} q_1,\ q_2 \xrightarrow{\sigma} q'\}$
Para cada transição da forma $q \xrightarrow{l} q''\ \in\ \delta$ **Faça**
$\delta \leftarrow (\delta - \{q \xrightarrow{l} q''\}) \cup \{q_1 \xrightarrow{l} q'',\ q_2 \xrightarrow{l} q''\}$
Fim Para
Para cada transição da forma $q'' \xrightarrow{l} q \in \delta$ **Faça**
$\delta \leftarrow (\delta - \{q'' \xrightarrow{l} q\}) \cup \{q'' \xrightarrow{l} q_1\}$
Fim Para
Fim Enquanto

Para cada estado pendente $q \in PS$ **Faça**
Se $(\forall\ l\ \in\ T(q, BS))\,(l\ \in\ \Sigma_c)$ **então**
$NPS \leftarrow NPS \cup \{q\}$ e $C_d(q) \leftarrow \Sigma_c \cap T(q, BS)$
Senão
Se $(\exists\ l'\ \in\ T(q, BS))\,(l' \in \Sigma_f)$ **então**
$NPS \leftarrow NPS \cup \{q\}$ e $C_f(q) \leftarrow \Sigma_f \cap T(q, PS)$
Senão
Se $(\forall\ l'\ \in\ T(q, BS))\,(l'\ \notin\ \Sigma_f)\ \wedge\ (\exists\ l\ \in\ T(q, BS))\,(l\ \in\ \Sigma - \Sigma_c)$ **então**
$NBS \leftarrow NBS \cup \{q\}$
Senão
Se $(\tau(\bigvee_{g\ \in\ T(q,PS)\ \cap\ G_t}\ g)\ >\ \tau(\bigvee_{g'\ \in\ T(q,BS)\ \cap\ G_t}\ g'))$ **então**
$NBS \leftarrow NBS \cup \{q\}$
Senão
Se $(\exists\ \sigma \in \Sigma - \Sigma_c)\,(\exists\ g\ \in\ T(q, BS)\ \cap\ G)\,(q \xrightarrow{\sigma} q\ \in\ \delta\ \wedge\ [\sigma^* \longrightarrow g])$ **então**
$NBS \leftarrow NBS \cup \{q\}$
Senão
Se $(\exists\ q_a \xrightarrow{\sigma/p:=f(p)} q\ \in\ \delta)\,(\exists\ q \xrightarrow{\sigma} q_b\ \in\ \delta)\,(q_a\ \in\ PS\ \wedge\ q_b\ \in\ BS\ \wedge\ [\sigma \longrightarrow g])$ **então**
$NPS \leftarrow NPS \cup \{q\}$ e $NBS \leftarrow NBS \cup \{q'\}$
$\delta \leftarrow (\delta - \{q_a \xrightarrow{\sigma/p:=f(p)} q,\ q \xrightarrow{g} q_b\}) \cup \{q_a \xrightarrow{g(p \to f(p)) \wedge \sigma/p:=f(p)} q',\ q_a \xrightarrow{\neg g(p \to f(p)) \wedge \sigma/p:=f(p)} q,\ q' \xrightarrow{g} q_b\}$
Para cada transição $q \xrightarrow{l} q''\ \in\ \delta$ **Faça**
$\delta \leftarrow \delta \cup \{q' \xrightarrow{l} q''\}$
Fim Para
Para cada transição $q'' \xrightarrow{l} q\ \in\ \delta$ **Faça**
$\delta \leftarrow (\delta\ -\ \{q'' \xrightarrow{l} q\}) \cup \{q'' \xrightarrow{g(p \to f(p)) \wedge l} q',\ q'' \xrightarrow{\neg g(p \to f(p)) \wedge l} q\}$
Fim Para
Senão
Se $(\exists\ \sigma\ \in\ \Sigma_c)\,(\exists\ g\ \in\ T(q, BS)\ \cap\ G)\,(q \xrightarrow{\sigma/p:=f(p)} q\ \in\ \delta\ \wedge\ [\sigma^* \longrightarrow g])$ **então**
$NPS \leftarrow NPS \cup \{q,\ q'\}$
$(\delta - \{q \xrightarrow{g} q_b\}) \cup \{q \xrightarrow{\neg g(p \to f(p))} q',\ q' \xrightarrow{g} q_b\}$
$C_d(q') \leftarrow C_d(q) \cup \{\sigma\}$ e $C_f(q') \leftarrow C_f(q)$
Para cada $q \xrightarrow{l} q''\ \in\ \delta$ **Faça**
$\delta \leftarrow \delta \cup \{q' \xrightarrow{l} q''\}$
Fim Para
Fim Se
Fim Se
Fim Se
Fim Se
Fim Se
Fim Se
Fim Se
Fim Se
Fim Para
Se $BS \neq NBS$ **então**
$BS \leftarrow NBS$
$PS \leftarrow NPS$
$NPS \leftarrow \emptyset$
Volta ao passo das decomposições de transições
Fim Se

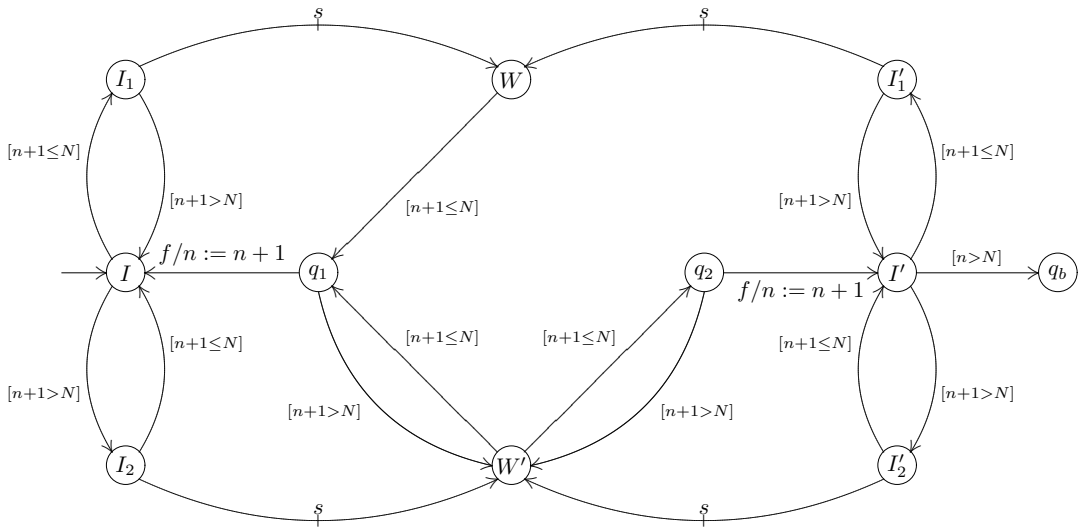

Figura 2.19: FSMwP resultante do particionamento do estado W da Figura 2.18.

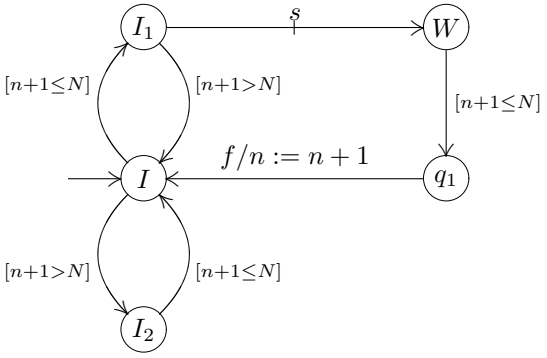

Figura 2.20: Supervisor do FSMwP da Figura 2.14.

Capítulo 3

STE com Coleção de Dados

Similarmente à teoria clássica [RW87, RW89], os sistemas de transição de estados com pré-condições podem ser vistos como uma planta interagindo com um supervisor. A planta gera eventos expontaneamente. O supervisor, em função da seqüência de eventos ocorrida no passado, controla a evolução do sistema através da inibição da ocorrência de alguns eventos na planta. Entretanto, diferentemente da teoria clássica, a decisão da inibição ou não dos eventos resulta da avaliação de certas fórmulas lógicas denominadas *pré-condições*. Os termos destas pré-condições consistem de parâmetros e variáveis representando informações sincronizadas com a ocorrência de eventos na planta.

Neste capítulo são descritos o modelo da planta, o qual provê construtores tais como variáveis e procedimentos, e a linguagem usada para a especificação dos requisitos desejáveis.

3.1 Modelo da planta

Para a modelagem da planta será utilizado um sistema de transição de estados determinístico equipado com uma coleção de dados. Tal composição será denominada Sistema de Transição de Estados com Coleção de Dados (STExCD). O objetivo da introdução da coleção de dados ao modelo é prover o sistema com informações que possam ser utilizadas na especificação de comportamentos desejáveis e, ao mesmo tempo, evitar o problema de explosão de estados discretos dos STE normalmente decorrente das especificações baseadas em linguagens formais. Por exemplo, ao invés de se representar discretamente cada estado de um armazém, uma variável n e as operações necessárias para sua manutenção podem ser introduzidas na coleção de dados. Associando-se convenientemente aos eventos, operações, tais como *insert* e *remove*, o valor de n, refletindo o número de itens cor-

rentemente armazenados no armazém, pode ser usado por algum agente de controle externo para restringir a ocorrência de certos eventos na planta de modo a evitar situações indesejáveis de overflow e underflow. O STE, a coleção de dados e os STExCD são apresentados em seguida.

3.1.1 Sistemas de transição de estados

O *Sistema de Transição de Estados determinístico* (STE) considerado é a tripla $S = \langle Q, \Sigma, \delta \rangle$, tal que:

1. Q é um conjunto finito e não vazio de *estados* discretos.
2. Σ é um conjunto finito e não vazio de símbolos denominados *eventos*. ξ é um evento distinto de Σ denominado *evento de inicialização* do STE. O conjunto de eventos Σ é particionado em eventos *controláveis*, Σ_c, e *não controláveis*, Σ_u. O evento ξ é controlável.
3. $\delta : Q \times \Sigma \to Q$ é uma *função parcial de transições de estado* definida para cada $q \in Q$ e um subconjunto de eventos $\rho \subseteq \Sigma$. É suposto que existe um estado especial $q_0 \in Q$, denominado *estado inicial* de S, tal que $q_0 = \delta(q, \xi)$ para cada $q \in Q$. Como antes, $\delta(q, \sigma)!$ denota que a transição é definida.

Um evento sinaliza mudanças significativas no estado da planta. Um evento é controlável quando ele sinaliza alterações nos estados da planta que podem ser inibidas por um agente de controle externo. O evento ξ foi convenientemente incluído ao modelo para garantir a convergência do algoritmo de síntese (a ser detalhado no Capítulo 4) e para definir valores iniciais do estado do STE e da coleção de dados; o evento ξ não está necessariamente relacionado com alguma característica física da planta.

Os *grafos de transições* da Figura 3.1 ilustram uma máquina simples com estados Idle e Working e eventos ξ, start e finish. Como nos geradores da teoria clássica, um arco cortado por um pequeno traço indica uma transição controlável. O grafo da esquerda representa explicitamente todas as transições associadas a ξ e o da direita representa simplificadamente as transições associadas a ξ como uma seta sem uma origem. Em qualquer representação, o estado inicial é Idle, os eventos controláveis são ξ e start e o único evento não controlável é finish.

Normalmente, as plantas consistem da interação sincronizada entre subsistemas mais simples. É conveniente e geralmente necessário que o modelo das plantas seja construído a partir dos modelos mais simples dos seus subsistemas. A composição síncrona entre STE definida a seguir permite a automação de tais construções.

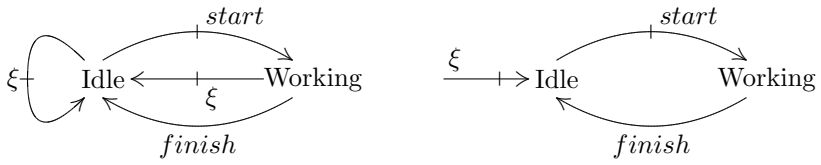

Figura 3.1: Grafo de transições de um STE.

Definição 3.1.1 (Composição síncrona de STE). *Dados dois STE* $S_1 = \langle Q_1, \Sigma_1, \delta_1 \rangle$ *e* $S_2 = \langle Q_2, \Sigma_2, \delta_2 \rangle$, a composição síncrona *de* S_1 *e* S_2, *denotada* $S_1 \parallel S_2$, *é o STE* $S = S_1 \parallel S_2 = \langle Q_1 \times Q_1, \Sigma_1 \cup \Sigma_2, \delta \rangle$, *onde, para cada* $q = \langle q_1, q_2 \rangle \in Q_1 \times Q_2$ *e* $\sigma \in \Sigma_1 \cup \Sigma_2$, $\delta(q, \sigma)$ *é igual à:*

- $\langle \delta_1(q_1, \sigma), \delta_2(q_2, \sigma) \rangle$, *se* $\delta_1(q_1, \sigma)!$, $\delta_2(q_2, \sigma)!$ *e* $\sigma \in \Sigma_1 \cap \Sigma_2$
- $\langle \delta_1(q_1, \sigma), q_2 \rangle$, *se* $\delta_1(q_1, \sigma)!$ *e* $\sigma \in \Sigma_1 - \Sigma_2$
- $\langle q_1, \delta_2(q_2, \sigma) \rangle$, *se* $\delta_2(q_2, \sigma)!$ *e* $\sigma \in \Sigma_2 - \Sigma_1$

O estado inicial de $S = S_1 \parallel S_2$ é $q_0 = \langle \delta_1(q_1, \xi), \delta_2(q_2, \xi) \rangle$ para todo $\langle q_1, q_2 \rangle \in Q_1 \times Q_2$.

A função δ é estendida para a função parcial $\delta^* : Q \times \Sigma^* \to Q$ pelas regras já definidas na Página 10 e re-apresentadas a seguir:

- $\delta^*(q, \varepsilon) = q$, e
- $\delta^*(q, s\sigma) = \delta(\delta^*(q, s), \sigma)$,

contanto que $q' = \delta^*(q, s)!$ e $\delta(q', \sigma)!$.

Exemplo 3.1.1 (Linha de Produção Simples – LPS). *A Figura 3.2 ilustra uma linha de produção simples consistindo de duas máquinas (*M_1 *e* M_2*). A máquina* M_1 *possui o armazém interno A e pode operar em dois modos distintos. No modo 1, as peças são pré-processadas e temporariamente armazenadas em A. Quando o evento* m_2 *acontece, a máquina* M_1 *muda para o modo 2 onde as peças pré-processadas no modo 1 são removidas de A e processadas no modo 2. Ao final do processamento no modo 2 (evento* f_2*), as peças são armazenadas no armazém externo B. No estado* i_2, M_1 *pode retornar ao modo de operação 1 através do evento* m_1*. Os eventos controláveis de* M_1 *são* s_1, s_2 *e* m_1.

A máquina M_2 *é uma máquina simples com dois estados (ver Figura 3.1) e opera assincronamente com* M_1*. A função de* M_2 *é realizar o acabamento ou pós-processamento das peças. A máquina*

M_2 pode pós-processar uma peça de cada vez. As peças pós-processadas por M_2 são removidas do armazém B pelo evento s_3 e, posteriormente, descartadas do sistema pelo evento f_3. O processamento completo de uma peça consiste do pré-processamento no modo 1, o processamento no modo 2, ambas realizadas pela máquina M_1, e o pós-processamento realizado pela máquina M_2.

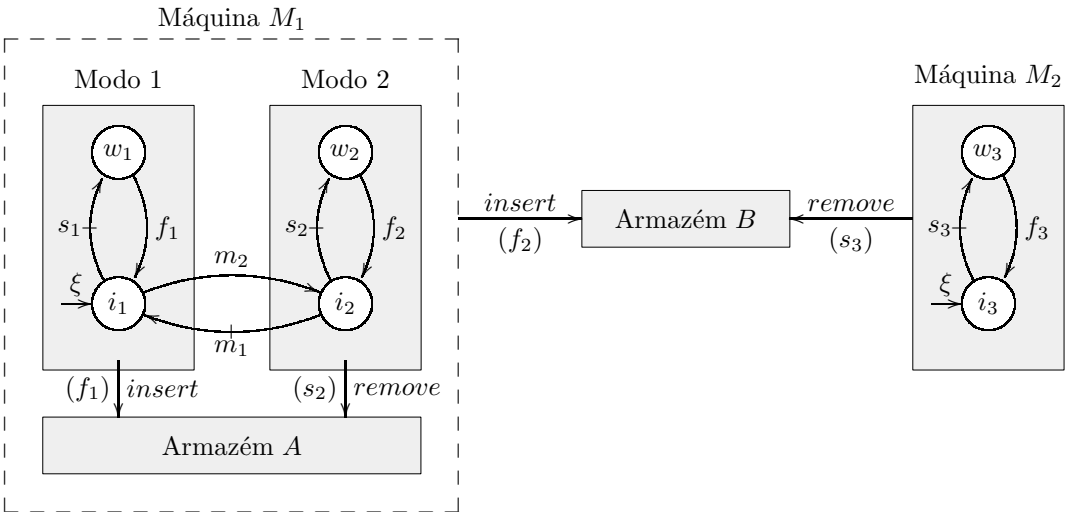

Figura 3.2: Planta de uma linha de produção simples consistindo das máquinas M_1 e M_2 e dois armazéns.

Compondo-se os STE para as máquinas M_1 e M_2, obtém-se o STE da Figura 3.3. Os estados do STE resultante da composição $M_1 \| M_2$ são denotados por xy, onde x é o estado local de M_1 e y é o estado local de M_2.

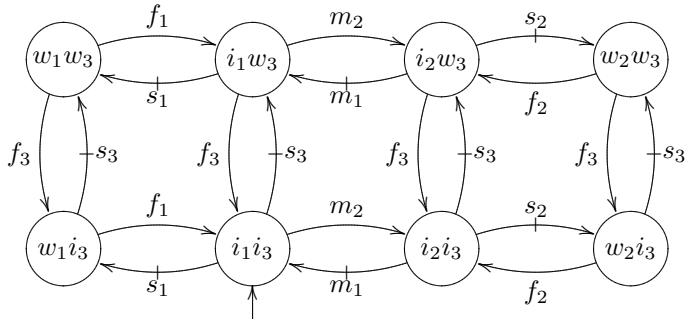

Figura 3.3: STE resultante da composição síncrona entre as máquinas $M_1 \| M_2$ da Figura 3.2.

3.1.2 Coleção de Dados

Como comentado anteriormente, o objetivo da incorporação de uma coleção de dados ao modelo é prover o sistema com informações que possam ser utilizadas como suporte formal para a descrição das especificações de comportamento desejável. Na modelagem que segue, um componente é introduzido formalmente como um conjunto de operações definidas sobre um conjunto, possivelmente infinito, de *instanciações* I. Tais operações devem possuir as propriedades descritas em seguida.

Seja F um conjunto finito de funções totais em I, computáveis [HMU01], contendo, no mínimo, a função $nop : I \to I$ definida para todo $i \in I$ como $nop(i) = i$. A composição da função f com a função g, ambas em F, denotada $g \circ f$, é uma função total em I. A composição de funções é associativa e não comutativa, isto é, para todo f, g, $h \in F$, $(f \circ g) \circ h = f \circ (g \circ h)$ e, de um modo geral, $g \circ h \neq h \circ g$. O elemento neutro da composição de funções em F é a função nop. Assim, para todo $f \in F$, $f \circ nop = nop \circ f = f$.

Seja F^* o conjunto de todas as composições finitas de funções em F. Então, a tripla $\langle F^*, \circ, nop \rangle$ é um *monóide livre* [MB99] com relação à composição de funções em F. O conjunto finito de funções F é denominado conjunto *gerador* do monóide livre.

Um *monóide de operações* para um conjunto de instanciações I é o monóide livre $M_I = \langle F_I^*, \circ, nop \rangle$, no qual:

1. F_I é um conjunto finito e não vazio de *operações*, ou funções totais, definidas no conjunto de instanciações I,
2. $\circ$ é a operação de *composição* de funções, e
3. $nop \in F_I$ é a função *identidade* (polimórfica), correspondente à operação no-operation, definida em cada conjunto de instanciações I como $\forall\, i \in I,\, nop(i) = i$.

Nos monóides de operações, funções simbólicas tais como $g(x) = 4 \times x^2 + x$ e $f(x) = 3 \times x + 2$ são usadas para relacionar as pré e pós-condições das operações. A composição (simbólica) $(g \circ f)(x) = 4 \times (3 \times x + 2)^2 + (3 \times x + 2)$ é obtida pela simples substituição de todas as ocorrências de x em $g(x)$ por $f(x)$.

Exemplo 3.1.2 (Armazém como um monóide de operações (1)). *Um armazém pode ser representado pelo monóide $A_{\mathbb{Z}} = \langle F_{\mathbb{Z}}^*, \circ, nop \rangle$, onde $F_{\mathbb{Z}}$ é o conjunto das seguintes operações:*

1. *$insert : \mathbb{Z} \to \mathbb{Z}$, definida para todo $n \in \mathbb{Z}$ como $insert(n) = n + 1$,*

2. *remove* : $\mathbb{Z} \to \mathbb{Z}$, *definida para todo* $n \in \mathbb{Z}$ *como* $remove(n) = n - 1$,

3. *nop* : $\mathbb{Z} \to \mathbb{Z}$, *definida para todo* $n \in \mathbb{Z}$ *como* $nop(n) = n$ *e*

4. *init* : $\mathbb{Z} \to \mathbb{Z}$, *definida para todo* $n \in \mathbb{Z}$ *como* $init(n) = 0$.

Seja qual for a capacidade de armazenamento do armazém, o conjunto de instanciações $\mathbb{Z}$ *inclui tanto as condições normais de operação do armazém quanto as possíveis condições anormais de* underflow *e* overflow.

A representação anterior é bastante genérica e não enfatiza qualquer modo de operação. Condições anormais de operações, tais como underflow e overflow, requerem uma interpretação dos resultados das operações. Se o resultado de uma inclusão for $N + 1$, onde N representa a capacidade máxima de armazenamento do armazém, ocorreu o overflow. Se o resultado de uma remoção for -1, ocorreu o underflow. Uma representação alternativa para um armazém, a qual considera explicitamente as condições de underflow e overflow, é descrita no exemplo a seguir.

Exemplo 3.1.3 (Armazém como um monóide de operações (2)). *Uma outra representação para o armazém, pode ser definida sobre o conjunto de instanciações definido como* $I = \{underflow, 0, 1, ..., N, overflow, undefined\}$. *Neste caso, as operações de inserção e remoção podem ser definidas como:*

1. *insert* : $I \to I$ *definida para todo* $n \in I$ *como:*

$$insert(n) = \begin{cases} n+1 & se\ n \geq 0 \ \wedge \ n < N \\ overflow & se\ n = N \\ undefined & em\ caso\ contrário. \end{cases}$$

2. *remove* : $I \to I$ *definida para todo* $n \in I$ *como:*

$$remove(n) = \begin{cases} n-1 & se\ n > 0 \ \wedge \ n \leq N \\ underflow & se\ n = 0 \\ undefined & em\ caso\ contrário. \end{cases}$$

Nesta segunda representação, as operações $\{+, -\}$ e as relações $\{>, \geq, <, \leq\}$ devem ser precisamente definidas sobre todas as combinações de operandos em I. Por exemplo, quais devem ser os resultados de $overflow - 1$ e $overflow > 0$? Isto é necessário para que a operação de composição de funções possa ser definida. Na primeira representação (Exemplo 3.1.2) o conjunto de instanciações é o conjunto dos inteiros $\mathbb{Z}$, sobre o qual, as operações e relações anteriores são definidas da forma usual. Neste sentido, a primeira representação é mais simples e mais geral, isto é, ela também pode (provavelmente) ser usada quando o enfoque não está sobre o overflow e/ou underflow.

Por outro lado, na representação 1, composições tais como $(insert \circ remove)(n)$ podem implicar operações anormais (intrínsecas à composição de funções) as quais não são detectadas. Por exemplo, $(insert \circ remove)(0)$ causa temporariamente um underflow, embora o resultado final 0 indique uma condição normal de operação. Na representação 2, tais operações anormais podem ser detectadas, desde que as operações e relações comentadas anteriormente sejam convenientemente definidas. Entretanto, a não detecção de condições anormais de operação, intrínsecas à composição de funções, não implica que tais condições indesejáveis não possam ser evitadas por alguma lógica de controle e também não implica que esta lógica de controle seja mais complexa. Deste modo, a representação 2 não tem qualquer vantagem significativa em relação à representação 1. No projeto de um monóide de operações, a simplicidade e a generalidade devem ser os principais quesitos a serem considerados. Portanto, no caso do armazém, a representação 1 geralmente é mais adequada.

Definição 3.1.2 (Coleção de Dados). *Uma* Coleção de Dados *é uma coleção ordenada e finita de monóides de operações da forma* $\mathbf{M}_n = \langle M_{I_1}, M_{I_2}, \ldots, M_{I_n} \rangle$, *onde* $n \geq 0$ [1].

Dada uma coleção de dados $\mathbf{M}_n = \langle M_{I_1}, M_{I_2}, \ldots, M_{I_n} \rangle$, o conjunto $\mathbf{I}_n = I_1 \times I_2 \times \ldots \times I_n$ é chamado *conjunto de instanciações de* $\mathbf{M}_n$ e cada n-upla $\iota = \langle \iota_1, \ldots, \iota_n \rangle \in \mathbf{I}_n$ é chamada *instanciação de* $\mathbf{M}_n$. O *conjunto suporte* de $\mathbf{M}_n$ é $\mathbf{F}_n = F_{I_1} \times F_{I_2} \times \ldots \times F_{I_n}$. Para $f = \langle f_1, \ldots, f_n \rangle$, $g = \langle g_1, \ldots, g_n \rangle \in \mathbf{F}_n$, a composição de f e g é definida como $g \circ f = \langle g_1, \ldots, g_n \rangle \circ \langle f_1, \ldots, f_n \rangle = \langle g_1 \circ f_1, \ldots, g_n \circ f_n \rangle$. O conjunto de todas as composições finitas de n-upla de funções em $\mathbf{F}_n$ é denotado por $\mathbf{F}_n^*$. A aplicação de $f = \langle f_1, \ldots, f_n \rangle \in \mathbf{F}_n^*$ a uma instanciação $\iota = \langle \iota_1, \ldots, \iota_n \rangle \in \mathbf{I}_n$ é $f(\iota) = \langle f_1(\iota_1), \ldots, f_n(\iota_n) \rangle$.

[1] *Uma coleção de dados pode ser vazia (*$\mathbf{M}_0 = \langle\ \rangle$*).*

Exemplo 3.1.4 (Coleção de Dados). *A coleção de dados representando dois armazéns A e B é o par* $\mathbf{M}_2 = \langle A_{\mathbb{Z}}, B_{\mathbb{Z}} \rangle$ *tal que* $A_{\mathbb{Z}}$ *e* $B_{\mathbb{Z}}$ *são dois monóides de operações iguais e definidos no Exemplo 3.1.2.*

O conjunto de instanciações de $\mathbf{M}_2$ *é* $\mathbb{Z} \times \mathbb{Z}$. *O par* $\langle a, b \rangle$, *com* a *denotando o número de itens armazenados em A e* b, *o número de itens armazenados em B, é uma instanciação de* $\mathbf{M}_2$.

O conjunto suporte de $\mathbf{M}_2$ *é o conjunto de pares de operações* $\mathbf{F}_2 = F_{\mathbb{Z}} \times F_{\mathbb{Z}}$. *O conjunto* $\mathbf{F}_2$ *inclui quaisquer pares de funções* $\langle f_a, f_b \rangle$, *interpretados como "*f_a *é uma operação em A e* f_b *é uma operação em B".*

A composição de $f = \langle insert, remove \rangle$ *com* $g = \langle insert, nop \rangle$ *é* $g \circ f = \langle insert, nop \rangle \circ \langle insert, remove \rangle = \langle insert \circ insert, nop \circ remove \rangle \in \mathbf{F}_2^*$.

3.1.3 STE com coleção de dados

No modelo sendo proposto, a planta é modelada por um STE equipado com uma coleção de dados. Tal composição é definida formalmente a seguir.

Definição 3.1.3 (STE com coleção de dados). *Um STE com coleção de dados (STExCD) é a quádrupla* $\mathbb{S} = \langle S, \mathbf{M}_n, \Delta, \mathcal{P}_m \rangle$ *consistindo dos seguintes elementos:*

1. $S = \langle Q, \Sigma, \delta \rangle$ *é um STE, geralmente resultante do produto síncrono entre subsistemas mais simples.*
2. $\mathbf{M}_n = \langle M_{I_1}, M_{I_2}, \ldots, M_{I_n} \rangle$ *é uma* coleção de dados.
3. $\Delta : Q \times \Sigma \to \mathbf{F}_n$ *é uma função parcial mapeando cada par* $\langle q, \sigma \rangle \in Q \times \Sigma$, *tal que* $\delta(q, \sigma)!$, *a uma n-upla de* operações *do conjunto suporte de* $\mathbf{M}_n$. *Em particular,* $\Delta(q, \xi)$ *é definida para cada estado* $q \in Q$ *como* $\Delta(q, \xi) = \langle init_1, init_2, \ldots, init_n \rangle$, *onde cada* $init_i : I_i \to I_i$ *é a operação de inicialização do monóide* M_{I_i}.
4. $\mathcal{P}_m : Q \times \mathbf{I}_n \to \mathbb{B}$, *onde* $\mathbb{B} = \{true, false\}$, *é um* predicado marcador *representado como uma fórmula lógica definida sobre todas as instanciações do STExCD.*

Tal como a função δ, Δ é estendida para a função $\Delta^* : Q \times \Sigma^* \to \mathbf{F}_n^*$ pelas regras:

- $\Delta^*(q, \varepsilon) = \langle nop_1, \ldots, nop_n \rangle$, e

- $\Delta^*(q, s\sigma) = \Delta(\delta^*(q, s), \sigma) \circ \Delta^*(q, s)$,

contanto que $\delta^*(q, s)!$ e $\delta(\delta^*(q, s), \sigma)!$.

Uma *configuração* em $\mathbb{S}$ é o par $\langle q, \iota \rangle \in Q \times \mathbf{I}_n$ composto por um estado discreto de S e uma instanciação de $\mathbf{M}_n$. A *configuração inicial* de $\mathbb{S}$ é a configuração $\langle q_0, \iota_0 \rangle$, tal que para cada estado q e instanciação $\iota = \langle \iota_1, \iota_2, \ldots, \iota_n \rangle$, $q_0 = \delta(q, \xi)$ é o estado inicial de S e $\iota_0 = \Delta(q, \xi)(\iota) = \langle init_1(\iota_1), init_2(\iota_2), \ldots, init_n(\iota_n) \rangle$ é a instanciação inicial de $\mathbf{M}_n$. A configuração $\langle q, \iota \rangle$ é *marcada* (ou *final*) se $\mathcal{P}_m\langle q, \iota \rangle = true$. Uma configuração marcada representa uma tarefa completa.

Uma *execução* de um STExCD é a seqüência finita de configurações

$$\xrightarrow{\xi} \langle q_0, \iota_0 \rangle \xrightarrow{\sigma_1} \langle q_1, \iota_1 \rangle \xrightarrow{\sigma_2} \langle q_2, \iota_2 \rangle \ldots \xrightarrow{\sigma_n} \langle q_n, \iota_n \rangle,$$

na qual, $\sigma_i \neq \xi$ é a i-ésima transição e $\langle q_i, \iota_i \rangle$ é a configuração depois da i-ésima transição após ξ. A seqüência $s = \xi\sigma_1\sigma_2\sigma_3 \ldots \sigma_n$ de eventos em uma execução é o *traço* daquela execução. Para um STExCD (determinístico), uma execução é unicamente determinada pelo seu traço. A *linguagem gerada* por um STExCD $\mathbb{S}$ é o subconjunto de traços $L(\mathbb{S}) = \{ \xi s \mid s \in (\Sigma - \{\xi\})^* \text{ e } \delta^*(q_0, s)! \}$ e a *linguagem marcada* é o subconjunto de traços $L_m(\mathbb{S}) = \{ \xi s \in L(\mathbb{S}) \mid \langle \delta^*(q_0, s), \Delta^*(q_0, s)(\iota_0) \rangle \text{ é uma configuração marcada } \}$.

A configuração alcançada por um traço $\xi s \in L(\mathbb{S})$ é $Reach^{QI}(\xi s) = \langle \delta^*(q_0, s), \Delta^*(q_0, s)(\iota_0) \rangle$. Estendendo-se a notação anterior para um subconjunto de traços $K \subseteq L(\mathbb{S})$, tem-se $Reach^{QI}(K) = \{ Reach^{QI}(\xi s) \in Q \times \mathbf{I}_n \mid \xi s \in K \}$. O subconjunto de *configurações acessíveis* em $\mathbb{S}$ é $Reach^{QI}(L(\mathbb{S}))$. O subconjunto de configurações *co-acessíveis* em $\mathbb{S}$ é $CoReach^{QI}(L(\mathbb{S})) = \{ \langle q, \iota \rangle \in Q \times \mathbf{I}_n \mid (\exists\, s \in (\Sigma - \{\xi\})^*)$ $\langle \delta^*(q, s), \Delta^*(q, s)(\iota) \rangle$ é uma configuração marcada $\}$. $\mathbb{S}$ é *não-bloqueante* se cada configuração acessível é co-acessível, isto é, $\mathbb{S}$ é não-bloqueante se $Reach^{QI}(L(\mathbb{S})) = Reach^{QI}(\overline{L_m(\mathbb{S})})$.

Exemplo 3.1.5 (Sistema de transição de estados – LPS). *A linha de produção simples, descrita no Exemplo 3.1.1, pode ser modelada pelo STE da Figura 3.3, o qual modela o comportamento assíncrono das máquinas M_1 e M_2, e pela coleção de dados $\mathbf{M}_2 = \langle A_{\mathbb{Z}}, B_{\mathbb{Z}} \rangle$ do Exemplo 3.1.4. Como ilustra a Figura 3.2, a função Δ sincronizando eventos da planta com operações em $\mathbf{M}_2$ deve ser:*

$$\Delta(w_1 -, f_1) = \langle insert, nop \rangle, \qquad \Delta(i_2 -, s_2) = \langle remove, nop \rangle,$$
$$\Delta(w_2 -, f_2) = \langle nop, insert \rangle, \qquad \Delta(- i_2, s_3) = \langle nop, remove \rangle,$$

onde o traço −, dependendo de sua posição, indica qualquer estado local da máquina M_1 ou da máquina M_2.

Exemplo 3.1.6 (Sistema de transição de estados – LR). *A Figura 3.4 ilustra o STExCD da linha realimentada introduzida no Exemplo 2.1.1 (página 9). Na figura, os estados da planta são denotados por xyz, onde x e y representam, respectivamente, os estados locais das máquinas M_1 e M_2, e z, o estado local do testador T_1. A coleção de dados é novamente um par de monóides do tipo $\langle L_{\mathbb{Z}}, R_{\mathbb{Z}} \rangle$, onde $L_{\mathbb{Z}}$ e $R_{\mathbb{Z}}$ representam, respectivamente, os armazéns A_1 e A_2. Os estados iniciais das máquinas e do testador são Idle e as instanciações iniciais dos armazéns são 0. O predicado marcador é false para todas as configurações, isto é, nenhuma tarefa deve ser realizada. Uma interpretação para este predicado marcador é a produção contínua de peças; esta interpretação difere da interpretação de marcação da teoria clássica. No Capítulo 5, retornar-se-á a este assunto.*

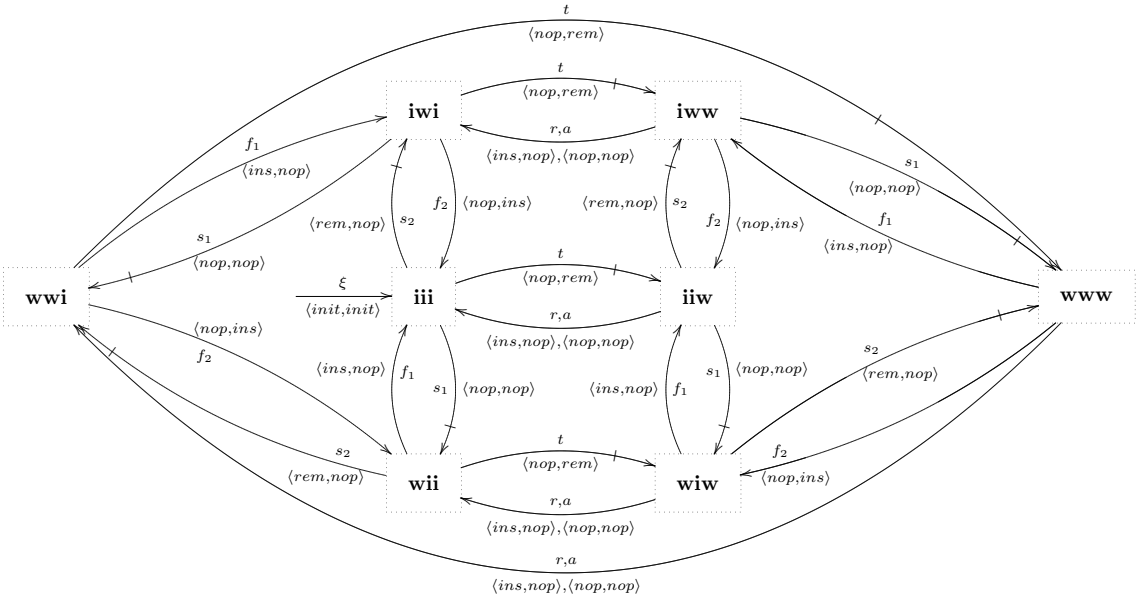

Figura 3.4: STExCD para a linha realimentada.

3.1.4 Composição síncrona entre STExCD

Tão importante quanto a composição síncrona entre STE, já discutida na Seção 3.1.1, é a composição síncrona entre STExCD.

Sejam $\mathbf{M}^1_n = \langle M^1_1, M^1_2, \ldots, M^1_n \rangle = \left\langle M^1_i \right\rangle_{i=1,\ldots,n}$ e $\mathbf{M}^2_m = \langle M^2_1, M^2_2, \ldots, M^2_m \rangle = \left\langle M^2_j \right\rangle_{j=1,\ldots,m}$ duas coleções ordenadas e não vazias de dados. A união de $\mathbf{M}^1_n$ e $\mathbf{M}^2_m$, denotada $\mathbf{M}_K = \mathbf{M}^1_n \uplus \mathbf{M}^2_m = \langle M_k \rangle_{k=1,\ldots,K}$, produz uma nova coleção $\mathbf{M}_K$, onde K é igual ao número de componentes distintos de $\mathbf{M}^1_n$ e $\mathbf{M}^2_m$, contendo cada componente de uma ou outra coleção. Se existem componentes duplicados entre as duas coleções $\mathbf{M}^1_n$ e $\mathbf{M}^2_m$, apenas uma das duplicatas aparece na coleção resultante [2].

A definição de $\uplus$ é feita com o auxílio de dois mapas de índices. Os mapas de índices $p_1 : \{1,\ldots,n\} \to \{1,\ldots,K\}$ e $p_2 : \{1,\ldots,m\} \to \{1,\ldots,K\}$, são definidos de tal forma que $p_1(i) = p_2(j)$ se, e somente se, $M^1_i \doteq M^2_j$, onde $\doteq$ é verdade se M^1_i e M^2_j são dois monóides iguais representando o mesmo componente. Assim, a união de $\mathbf{M}^1_n$ com $\mathbf{M}^2_m$ forma $\mathbf{M}_K = \mathbf{M}^1_n \uplus \mathbf{M}^2_m = \langle M_k \rangle_{k=1,\ldots,K}$, tal que, para cada $k = 1, \ldots, K$:

- $M_k \doteq M^1_i$, se existe i tal que $k = p_1(i)$ e não existe j tal que $k = p_2(j)$.
- $M_k \doteq M^2_j$, se existe j tal que $k = p_2(j)$ e não existe i tal que $k = p_1(i)$.
- $M_k \doteq M^1_i \doteq M^2_j$, se existe i tal que $k = p_1(i)$ e existe j tal que $k = p_2(j)$.

Trivialmente, para coleções vazias, tem-se $\mathbf{M}_K = \mathbf{M}_n \uplus \langle\,\rangle = \mathbf{M}_n$.

O operador $\uplus$ representa uma operação sobre listas que depende única e exclusivamente dos mapas de índices p_1 e p_2. Assim, fixados p_1 e p_2, o operador $\uplus$ pode ser usado sobre outras listas, tais como $\mathbf{F}_K = \mathbf{F}^1_n \uplus \mathbf{F}^2_m$ e $\mathbf{I}_K = \mathbf{I}^1_n \uplus \mathbf{I}^2_m$. A Figura 3.5 ilustra a união de duas listas quaisquer.

Dois STExCD, $\mathbb{S}^1 = \langle S^1, \mathbf{M}^1_n, \Delta^1, \mathcal{P}^1_m \rangle$ e $\mathbb{S}^2 = \langle S^2, \mathbf{M}^2_m, \Delta^2, \mathcal{P}^2_m \rangle$, são ditos *consistentes* se para todo evento $\sigma \in \Sigma_1 \cap \Sigma_2$, sempre que $p_1(i) = p_2(j)$, a operação em M^1_i associada ao evento σ é igual à operação em M^2_j associada ao mesmo evento σ. Assim, para dois STExCD consistentes, nenhuma transição síncrona entre os dois subsistemas tenta realizar diferentes operações em um mesmo componente.

Definição 3.1.4 (Composição síncrona entre STExCD). *Sejam* $\mathbb{S}^1 = \langle S^1, \mathbf{M}^1_n, \Delta^1, \mathcal{P}^1_m \rangle$ *e* $\mathbb{S}^2 = \langle S^2, \mathbf{M}^2_m, \Delta^2, \mathcal{P}^2_m \rangle$ *dois STExCD consistentes. A* composição síncrona *de* $\mathbb{S}^1$ *e* $\mathbb{S}^2$ *é o STExCD* $\mathbb{S} = \mathbb{S}^1 \parallel \mathbb{S}^2 = \langle S^1 \parallel S^2, \mathbf{M}^1_n \uplus \mathbf{M}^2_m, \Delta^1 \uplus \Delta^2, \mathcal{P}^1_m \wedge \mathcal{P}^2_m \rangle$.

[2] Aqui, fala-se componentes e não monóides porque em uma ou mais coleções podem existir monóides iguais, porém, cada qual representando (semanticamente) um componente diferente. Em casos como estes, monóides iguais não são considerados duplicatas.

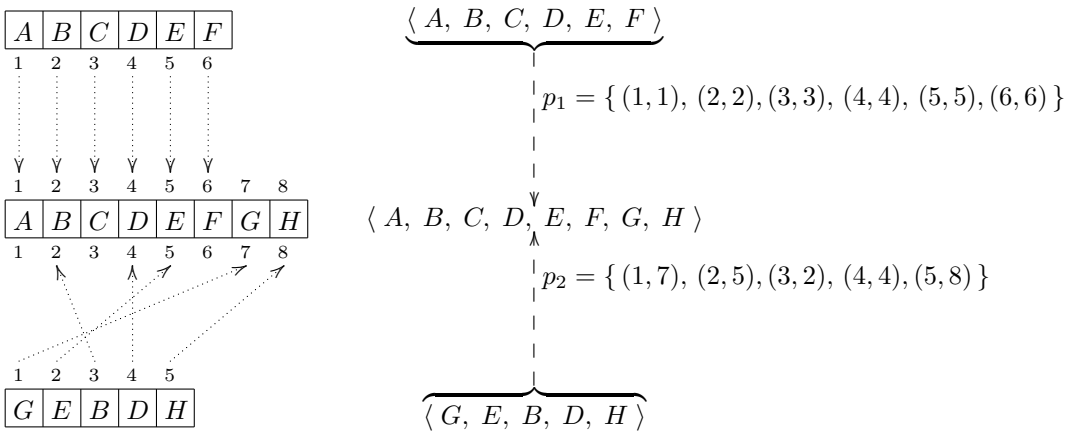

Figura 3.5: União de listas ($\langle A,B,C,D,E,F,G,H\rangle = \langle A,B,C,D,E,F\rangle \uplus \langle G,E,B,D,H\rangle$).

Tanto as definições de p_1 e p_2 quanto a composição síncrona de STExCD podem ser automatizadas. Sem perda de generalidade, deste ponto em diante, serão considerados apenas sistemas compostos.

3.2 Especificações

Nesta seção são descritos dois tipos de especificações, ambos definidos a partir de fórmulas lógicas. O primeiro tipo, chamado *especificação por pré-condições*, determina as condições para a habilitação dos eventos. O segundo tipo, chamado *especificação por invariante*, consiste de uma fórmula lógica que deve ser satisfeita por todas as configurações que sobrevivem à supervisão. Ambos os tipos de especificação são do tipo configurações proibidas e não estão relacionadas com a descrição de tarefas a serem realizadas – estas devem ser descritas pelo predicado marcador.

3.2.1 Especificação por pré-condições

Definição 3.2.1 (Especificação por pré-condições). *Uma* especificação por pré-condições *para um STExCD* $\mathbb{S}$ *é o mapa*

$$\Lambda : \Sigma \to \{\mathcal{P} \mid \mathcal{P} : Q \times \mathbf{I}_n \to \mathbb{B}\},$$

no qual $\mathcal{P} : Q \times \mathbf{I}_n \to \mathbb{B}$ *é uma fórmula lógica (simbólica), definida sobre as configurações de* $\mathbb{S}$.

Cada fórmula $\mathcal{P}$ de uma especificação por pré-condições é chamada *pré-condição*. As pré-condições podem conter variáveis livres, as quais são denominadas *parâmetros*. As pré-condições são avaliadas somente durante o processo de supervisão.

O conjunto de configurações satisfazendo uma especificação Λ, denotado $\mathbb{D}_\Lambda$, é aquele que satisfaz:

1. a configuração inicial $\langle q_0, \iota_0 \rangle$ está em $\mathbb{D}_\Lambda$,
2. para todo traço $\xi\omega \in L(\mathbb{S})$ e $\omega \in \Sigma^*$, a configuração $\langle q, \iota \rangle = \langle \delta^*(q_0, \omega), \Delta^*(q_0, \omega)(\iota_0) \rangle$ está em $\mathbb{D}_\Lambda$ se:
   (a) para $\omega = \phi\sigma$, $\phi \in \Sigma^*$ e $\sigma \in \Sigma$, a configuração $\langle \delta^*(q_0, \phi), \Delta^*(q_0, \phi)(\iota_0) \rangle$ está em $\mathbb{D}_\Lambda$, e
   (b) $\Lambda(\sigma)\langle \delta^*(q_0, \phi), \Delta^*(q_0, \phi)(\iota_0) \rangle = \mathit{true}$.

Em palavras, para $\mathbb{D}_\Lambda \neq \emptyset$, a Condição 1 diz que a configuração inicial deve satisfazer Λ. A segunda condição diz que a configuração $\langle q, \iota \rangle$, alcançada por um traço $\xi\omega$, satisfaz Λ se a configuração $\langle q', \iota' \rangle = \langle \delta^*(q_0, \phi), \Delta^*(q_0, \phi)(\iota_0) \rangle$, anterior à configuração $\langle q, \iota \rangle$, satisfaz Λ e a restrição sobre o evento σ, conduzindo o sistema da configuração $\langle q', \iota' \rangle$ à configuração $\langle q, \iota \rangle$, é verificada.

O conjunto de todas as especificações por pré-condições para $\mathbb{S}$ é denotado $\Lambda(\mathbb{S})$. $\bot \in \Lambda(\mathbb{S})$ é a especificação tal que $\mathbb{D}_\bot = \emptyset$ (por definição) e $\top \in \Lambda(\mathbb{S})$ é a especificação tal que $\mathbb{D}_\top = \mathit{Reach}^{QI}(L(\mathbb{S}))$. Em termos práticos, $\bot$ é a especificação que associa a pré-condição false a todas as transições ξ de $\mathbb{S}$ e $\top$ é a especificação que associa a pré-condição true a todas as transições de $\mathbb{S}$. As especificações $\Lambda \in \Lambda(\mathbb{S})$ diferentes de $\bot$ são denominadas *próprias*.

Para duas especificações por pré-condições Λ_1 e Λ_2, quando $\mathbb{D}_{\Lambda_1} \subseteq \mathbb{D}_{\Lambda_2}$, diz-se que a especificação Λ_1 implica Λ_2, escrito como $\Lambda_1 \Rightarrow \Lambda_2$. Além disso, se $\mathbb{D}_{\Lambda_1} = \mathbb{D}_{\Lambda_2}$, isto é, se $\Lambda_1 \Rightarrow \Lambda_2$ e $\Lambda_2 \Rightarrow \Lambda_1$, diz-se que Λ_1 e Λ_2 são equivalentes, ou $\Lambda_1 \Leftrightarrow \Lambda_2$. Claramente, $\Leftrightarrow$ é uma relação de equivalência sobre o conjunto de especificações. No que segue, uma especificação não será distinguida de sua classe de equivalência.

Exemplo 3.2.1 (Especificação por pré-condições – LPS). *Voltando-se à linha de produção simples e, supondo-se que A tem (teoricamente) uma capacidade ilimitada para o armazenamento de peças, que B tem uma capacidade limitada para armazenar até N peças e que os requisitos de controle sejam:*

E1. evitar o underflow *em A,*

E2. *evitar o* ***underflow*** *e* ***overflow*** *em B e*

E3. *minimizar as mudanças de modo de operação em M_1, isto é, permitir que M_1 retorne ao modo 1 somente depois que todas as peças no armazém A foram processadas,*

como ficaria a especificação por pré-condições?

Denotando-se por a e b as variáveis introduzidas pela coleção de dados para refletir, respectivamente, o número de peças armazenados em A e B, os requisitos anteriores podem ser descritos pelas seguintes pré-condições:

$$\begin{cases} \Lambda_1(s_2)\langle q, \langle a, b\rangle\rangle = (a > 0) & \textit{evita } \textbf{\textit{underflow}} \textit{ em } A \textit{ e } \ldots \\ \Lambda_1(m_1)\langle q, \langle a, b\rangle\rangle = (a = 0) & \ldots \textit{ minimiza mudanças de modo em } M_1 \\ \Lambda_2(f_2)\langle q, \langle a, b\rangle\rangle = (b < N) & \textit{evita } \textbf{\textit{overflow}} \textit{ e } \ldots \\ \Lambda_2(s_3)\langle q, \langle a, b\rangle\rangle = (b > 0) & \ldots \textbf{\textit{underflow}} \textit{ em } B. \end{cases} \tag{3.1}$$

É importante enfatizar que na teoria clássica [RW87], a especificação baseada em linguagens evitando o ***underflow*** *em A e minimizando as mudanças do modo de operação em M_1 seria parecida com o autômato infinito da Figura 3.6. O autômato é infinito pois representa um comportamento não regular da forma $(f_1^m\ m_2\ s_2^m\ m_1)^*$ [HMU01]. Como na teoria clássica a síntese de supervisores é baseada na composição síncrona entre os geradores dos comportamentos desejáveis com o gerador do comportamento livre da planta, especificações não regulares como a anterior não podem ser manipuladas.*

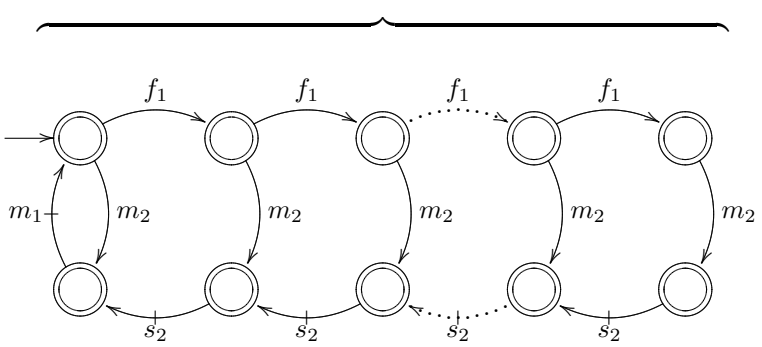

Figura 3.6: Especificação baseada em linguagens restringindo o comportamento de M_1.

Uma das vantagens das especificações por pré-condições sobre as especificações através de linguagens formais é a expressão de comportamentos não regulares. Na forma de uma especificação por pré-condições, o comportamento não regular descrito na Figura 3.6 é especificado através das restrições impostas sobre s_2 e m_1, isto é, $\Lambda_1(s_2)\langle q, \langle a, b\rangle\rangle = (a > 0)$ e $\Lambda_1(m_1)\langle q, \langle a, b\rangle\rangle = (a = 0)$. Estas restrições, utilizam a variável a como um contador; durante a ocorrência dos eventos f_1 o contador é incrementado e durante a ocorrência dos eventos s_2 o contador é decrementado. A restrição sobre s_2 previne o underflow em A e a restrição sobre m_1 é satisfeita somente quando o número de eventos f_1 ocorridos for igual ao número de eventos s_2 ocorridos.

Outra vantagem das especificações por pré-condições com relação às especificações através de linguagens formais, diz respeito à parametrização das especificações. Na teoria clássica os supervisores são implementados como tabelas estáticas. Estas tabelas definem os eventos a serem habilitados em cada estado discreto do sistema supervisionado. Por serem estáticas, qualquer alteração no espaço de estados do sistema supervisionado requer o reprojeto e a reconstrução de todo o supervisor. Por exemplo, em termos de linguagens formais, a especificação evitando o underflow e overflow no armazém B, teria a forma do gerador ilustrado na Figura 3.7. A exata definição da linguagem gerada por este gerador requer a fixação do valor de N. Se N é igual a 1, a linguagem gerada é:

$$f_2(s_3f_2)^*s_3 + f_2(s_3f_2)^* + \varepsilon$$

e se N é igual a 2:

$$f_2(s_3f_2)^*f_2(s_3(s_3f_2)^*f_2)^*s_3(s_3f_2)^*s_3 + f_2(s_3f_2)^*f_2(s_3(s_3f_2)^*f_2)^*s_3(s_3f_2)^* + f_2(s_3f_2)^*f_2(s_3(s_3f_2)^*f_2)^* + f_2(s_3f_2)^*s_3 + f_2(s_3f_2)^* + \varepsilon.$$

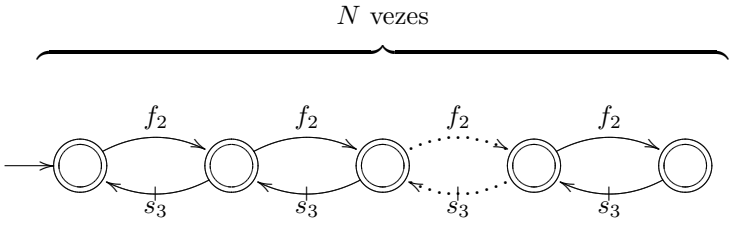

Figura 3.7: Especificação baseada em linguagens evitando overflow e underflow em A_2.

No caso das especificações por pré-condições, o valor de N não precisa ser fixado na especificação. Assim, o par de pré-condições:

$$\Lambda_2(f_2)\langle q, \langle a, b\rangle\rangle = (b < N) \quad e \quad \Lambda_2(s_3)\langle q, \langle a, b\rangle\rangle = (b > 0) \tag{3.2}$$

evitando, respectivamente, o overflow *e o* underflow *em B, vale para quaisquer valores razoáveis de N. Por valores razoáveis de N, entenda-se* $N > 0$.

Exemplo 3.2.2 (Especificação por pré-condições – LR). *A especificação por pré-condições evitando os* overflows *e* underflows *nos armazéns* A_1 *e* A_2 *da linha realimentada (Exemplo 3.1.6) é definida como:*

> *Para todos os estados* $q \in Q$ *e instanciações* $\langle a_1, a_2\rangle \in \mathbb{Z} \times \mathbb{Z}$, *tal que* a_1 *e* a_2 *denotam, respectivamente, o número de itens correntemente armazenados em* A_1 *e* A_2,
>
> $$\begin{cases} \Lambda(f_1)\langle q, \langle a_1, a_2\rangle\rangle = a_1 < N & \text{evita } \textit{overflow} \text{ em } A_1, \\ \Lambda(s_2)\langle q, \langle a_1, a_2\rangle\rangle = a_1 > 0 & \text{evita } \textit{underflow} \text{ em } A_1, \\ \Lambda(f_2)\langle q, \langle a_1, a_2\rangle\rangle = a_2 < M & \text{evita } \textit{overflow} \text{ em } A_2, \\ \Lambda(t)\langle q, \langle a_1, a_2\rangle\rangle = a_2 > 0 & \text{evita } \textit{underflow} \text{ em } A_2 \text{ e} \\ \Lambda(r)\langle q, \langle a_1, a_2\rangle\rangle = a_1 < N & \text{evita } \textit{overflow} \text{ em } A_1. \end{cases}$$

*Para os demais eventos as pré-condições são verdadeiras (*true*).*

3.2.2 Especificação por invariante

Seja $Q \times \mathbf{I}_n$ o conjunto de configurações de um STExCD $\mathbb{S}$. Um *invariante* para $\mathbb{S}$ é a fórmula lógica $Inv : Q \times \mathbf{I}_n \to \mathbb{B}$. Uma especificação por invariante descreve propriedades comuns de um subconjunto de configurações. As configurações satisfazendo um invariante Inv são $\mathbb{D}_{Inv} = \{\langle q, \iota\rangle \in Q \times \mathbf{I}_n \mid Inv(q, \iota) = true\}$.

Exemplo 3.2.3 (Especificação por invariante – LR). *A especificação por invariante evitando os* ***overflows*** *e* ***underflows*** *nos armazéns A_1 e A_2 da linha realimentada (Exemplo 3.1.6) é a fórmula:*

> *Para todos os estados $q \in Q$ e instanciações $\langle a_1, a_2 \rangle \in \mathbb{Z} \times \mathbb{Z}$, tal que a_1 e a_2 denotam, respectivamente, o número de itens correntemente armazenados em A_1 e A_2,*
>
> $$a_1 \geq 0 \ \wedge \ a_1 \leq N \ \wedge \ a_2 \geq 0 \ \wedge \ a_2 \leq M.$$

3.2.3 Equivalência entre especificações

Cada especificação por invariante pode ser transformada em uma especificação por pré-condições equivalente. A construção de uma especificação por pré-condições equivalente a uma especificação por invariante se inicia pela observação, na Figura 3.8, de que para cada configuração $\langle q, \iota \rangle$ e cada transição $\delta(q_i, \sigma_i)$, tal que $q = \delta(q_i, \sigma_i)$ e $\iota = \Delta(q_i, \sigma_i)(\iota_i)$, existe uma pré-condição da forma $\Lambda(\sigma_i)\langle q_i, \iota_i \rangle$ implicando o invariante $Inv(q, \iota)$, isto é,

$$\Lambda(\sigma_i)\langle q_i, \iota_i \rangle = Inv\langle \delta(q_i, \sigma_i), \Delta(q_i, \sigma_i)(\iota_i) \rangle \Rightarrow Inv\langle q, \iota \rangle. \tag{3.3}$$

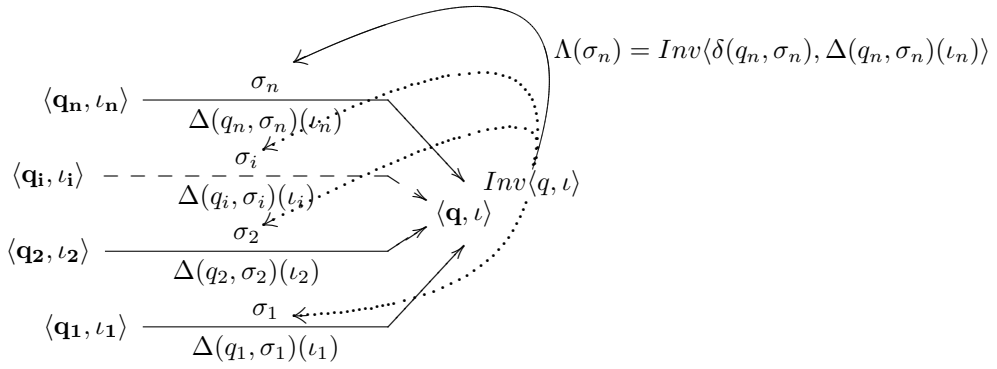

Figura 3.8: Transformação de um invariante em um conjunto de pré-condições.

A implicação anterior decorre do fato de que nem todas as instanciações ι, associadas ao estado q, são obtidas a partir da instanciação ι_i, associada ao estado q_i. A igualdade relacionando o conjunto

de todas as possíveis instanciações ι, no estado q, com as instanciações anteriores à ocorrência dos n eventos σ_i, conduzindo o sistema ao estado q é:

$$\iota = \Delta(q_1, \sigma_1)(\iota_1) \vee \Delta(q_2, \sigma_2)(\iota_2) \vee \ldots \vee \Delta(q_n, \sigma_n)(\iota_n).$$

Assim, o invariante $Inv\langle q, \iota\rangle$ pode ser reescrito como:

$$Inv\langle q, \iota\rangle = Inv\langle \delta(q_1, \sigma_1), \Delta(q_1, \sigma_1)(\iota_1)\rangle \vee Inv\langle \delta(q_2, \sigma_2), \Delta(q_2, \sigma_2)(\iota_2)\rangle \vee \ldots \vee Inv\langle \delta(q_n, \sigma_n), \Delta(q_n, \sigma_n)(\iota_n)\rangle$$

ou, de acordo com a Equação 3.3,

$$Inv\langle q, \iota\rangle = \Lambda(\sigma_1)\langle q_1, \iota_1\rangle \vee \Lambda(\sigma_2)\langle q_2, \iota_2\rangle \vee \ldots \vee \Lambda(\sigma_n)\langle q_n, \iota_n\rangle. \tag{3.4}$$

A igualdade anterior diz que se existem eventos habilitados pelas pré-condições à direita da igualdade 3.4, então existem configurações associadas ao estado q que não invalidam o invariante. Por outro lado, se existem configurações associadas ao estado q que não invalidam o invariante, então existem pré-condições à direita do invariante habilitando pelo menos um dos eventos σ_i ($i = 1, 2, \ldots, n$).

Estendendo-se a Igualdade 3.4 a todos os estados do sistema, uma especificação por pré-condições equivalente à especificação por invariantes é obtida.

Exemplo 3.2.4 (Equivalência entre especificações – LR). *A especificação por invariante evitando os underflows e overflows na linha realimentada (Exemplo 3.2.3) consiste do invariante* $a_1 \geq 0 \wedge a_1 \leq N \wedge a_2 \geq 0 \wedge a_2 \leq M$.

Analisando-se as configurações cujo estado discreto é iii *(ver Figura 3.4), o invariante anterior deve ser transformado em pré-condições associadas aos eventos* ξ, f_1, f_2, r *e* a, *todos rotulando transições para o estado* iii. *Os resultados desta análise estão apresentados na Tabela 3.1. Na tabela, as duas primeiras colunas identificam a transição para o estado* iii. *A coluna* $\Delta(q', \sigma)$ *denota as operações sobre os armazéns associadas à transição correspondente. Finalmente, a última coluna representa, sem nenhuma simplificação, as pré-condições restringindo as transições correspondentes.*

Procedendo-se de forma semelhante com a análise dos demais estados da planta, a especificação por pré-condições equivalente ao invariante anterior é obtida. Aparentemente, esta especificação por

q'	σ	$\Delta(q', \sigma)$	$\Lambda(\sigma)\langle q', \langle a_1, a_2\rangle\rangle$
$\cdot$	ξ	$\langle nop, nop\rangle$	$a_1 \geq 0 \wedge a_1 \leq N \wedge a_2 \geq 0 \wedge a_2 \leq M$
wii	f_1	$\langle ins, nop\rangle$	$a_1 + 1 \geq 0 \wedge \mathbf{a_1 + 1} \leq \mathbf{N} \wedge a_2 \geq 0 \wedge a_2 \leq M$
iwi	f_2	$\langle nop, ins\rangle$	$a_1 \geq 0 \wedge a_1 \leq N \wedge a_2 + 1 \geq 0 \wedge \mathbf{a_2 + 1} \leq \mathbf{M}$
iiw	r	$\langle ins, nop\rangle$	$a_1 + 1 \geq 0 \wedge \mathbf{a_1 + 1} \leq \mathbf{N} \wedge a_2 \geq 0 \wedge a_2 \leq M$
iiw	a	$\langle nop, nop\rangle$	$a_1 \geq 0 \wedge a_1 \leq N \wedge a_2 \geq 0 \wedge a_2 \leq M$

Tabela 3.1: Especificação por pré-condições equivalente à especificação por invariante $a_1 \geq 0 \wedge a_1 \leq N \wedge a_2 \geq 0 \wedge a_2 \leq M$.

pré-condições é mais complexa do que aquela do Exemplo 3.2.2. No entanto, ambas as especificações por pré-condições são equivalentes como se pode perceber após as simplificações das pré-condições obtidas a partir do invariante. Considerando-se o contexto da especificação completa, tais simplificações reduzem as pré-condições $a_1 \geq 0 \wedge a_1 \leq N \wedge a_2 \geq 0 \wedge a_2 \leq M$, *associadas aos eventos* ξ *e* a, *a* true *e as pré-condições associadas aos demais eventos, àquelas indicadas em negrito na Tabela 3.1.*

Na especificação para a linha de produção simples (Exemplo 3.2.1), os requisitos foram descritos por duas especificações locais. A especificação Λ_1 descreve os requisitos E_1 e E_3 e a especificação Λ_2 descreve o requisito E_2. A seção a seguir discute como uma especificação global, mais complexa, pode ser obtida a partir de um conjunto de especificações locais mais simples.

3.2.4 Composição de especificações

Independentemente da linguagem usada na construção das especificações, o projeto de uma especificação incorporando diferentes requisitos geralmente não é uma tarefa simples. Portanto, a composição de especificações mais simples, cada qual descrevendo requisitos locais, é desejável. Devido à natureza lógica da linguagem usada pelas especificações por pré-condições, dois tipos de composição de especificações serão definidas: a *composição conjuntiva* e a *composição disjuntiva*.

Definição 3.2.2 (Composições de especificações). *Dadas duas especificações,* Λ_1 *e* Λ_2 *em* $\Lambda(\mathbb{S})$, *as operações* meet ($\sqcap$) *e* join ($\sqcup$) *entre especificações são definidas como:*

$$\forall\, \sigma \in \Sigma, \quad \begin{cases} (\Lambda_1 \sqcap \Lambda_2)(\sigma) = \Lambda_1(\sigma) \wedge \Lambda_2(\sigma) \\ (\Lambda_1 \sqcup \Lambda_2)(\sigma) = \Lambda_1(\sigma) \vee \Lambda_2(\sigma). \end{cases}$$

A composição conjuntiva *entre as especificações* Λ_1 *e* Λ_2 *é a especificação* $(\Lambda_1 \sqcap \Lambda_2)$ *e a* composição disjuntiva *entre* Λ_1 *e* Λ_2 *é a especificação* $(\Lambda_1 \sqcup \Lambda_2)$. *Em termos de conjuntos de configurações,*

$$\mathbb{D}_{\Lambda_1 \sqcap \Lambda_2} = \mathbb{D}_{\Lambda_1} \cap \mathbb{D}_{\Lambda_2} \quad e \quad \mathbb{D}_{\Lambda_1 \sqcup \Lambda_2} = \mathbb{D}_{\Lambda_1} \cup \mathbb{D}_{\Lambda_2}.$$

Exemplo 3.2.5 (Composição de especificações – LPS). *A especificação por pré-condições* Λ, *descrevendo todos os requisitos da linha de produção simples (Exemplo 3.2.1) pode ser obtida como* $\Lambda = \Lambda_1 \sqcap \Lambda_2$, *onde:*

1. Λ_1 *é a especificação descrevendo os requisitos* E_1 *e* E_3, *isto é,*

$$\Lambda_1(s_2)\langle q, \langle a, b\rangle\rangle = (a > 0) \quad e \quad \Lambda_1(m_1)\langle q, \langle a, b\rangle\rangle = (a = 0).$$

2. Λ_2 *é a especificação descrevendo o requisito* E_2, *isto é,*

$$\Lambda_2(f_2)\langle q, \langle a, b\rangle\rangle = (b < N) \quad e \quad \Lambda_2(s_3)\langle q, \langle a, b\rangle\rangle = (b > 0).$$

Portanto, $\Lambda = \Lambda_1 \sqcap \Lambda_2$ *consiste das seguintes pré-condições:*

$$\forall\, q \in Q, \quad \forall\, a, b \in \mathbb{Z} \quad \begin{cases} \Lambda(s_2)\langle q, \langle a, b\rangle\rangle = (a > 0) \\ \Lambda(m_1)\langle q, \langle a, b\rangle\rangle = (a = 0) \\ \Lambda(f_2)\langle q, \langle a, b\rangle\rangle = (b < N) \\ \Lambda(s_3)\langle q, \langle a, b\rangle\rangle = (b > 0). \end{cases}$$

Exemplo 3.2.6 (Composição de especificações – LR). *A especificação por pré-condições* Λ *evitando os* overflows *e* underflows *na linha realimentada (Exemplo 3.2.2) pode ser interpretada como* $\Lambda = \Lambda_1 \sqcap \Lambda_2 \sqcap \Lambda_3 \sqcap \Lambda_4 \sqcap \Lambda_5$, *onde:*

1. Λ_1 *e* Λ_2 *são as especificações evitando o* overflow *em* A_1, *isto é,* $\Lambda_1(f_1)\langle q, \langle a_1, a_2\rangle\rangle = a_1 < N$ *e* $\Lambda_2(r)\langle q, \langle a_1, a_2\rangle\rangle = a_1 < N$.

2. Λ_3 *é a especificação evitando o* underflow *em* A_1, *isto é,* $\Lambda_3(s_2)\langle q, \langle a_1, a_2\rangle\rangle = a_1 > 0$.

3. Λ_4 *é a especificação evitando o* **overflow** *em* A_2*, isto é,* $\Lambda_4(f_2)\langle q, \langle a_1, a_2\rangle\rangle = a_2 < M$.

4. Λ_5 *é a especificação evitando o* **underflow** *em* A_2*, isto é,* $\Lambda_5(t)\langle q, \langle a_1, a_2\rangle\rangle = a_2 > 0$.

Para todos os pares de especificações $\Lambda_1, \Lambda_2 \in \Lambda(\mathbb{S})$ e eventos $\sigma \in \Sigma$ é fácil verificar as seguintes propriedades (para reticulados [MB99]) das composições de especificações:

1. Idempotência: $\begin{cases}(\Lambda_1 \sqcap \Lambda_1)(\sigma) = \Lambda_1(\sigma)\\ (\Lambda_1 \sqcup \Lambda_1)(\sigma) = \Lambda_1(\sigma).\end{cases}$

2. Comutatividade: $\begin{cases}(\Lambda_1 \sqcap \Lambda_2)(\sigma) = (\Lambda_2 \sqcap \Lambda_1)(\sigma)\\ (\Lambda_1 \sqcup \Lambda_2)(\sigma) = (\Lambda_2 \sqcup \Lambda_1)(\sigma).\end{cases}$

3. Associatividade: $\begin{cases}(\Lambda_1 \sqcap (\Lambda_2 \sqcap \Lambda_3))(\sigma) = ((\Lambda_1 \sqcap \Lambda_2) \sqcap \Lambda_3)(\sigma)\\ (\Lambda_1 \sqcup (\Lambda_2 \sqcup \Lambda_3))(\sigma) = ((\Lambda_1 \sqcup \Lambda_2) \sqcup \Lambda_3)(\sigma).\end{cases}$

4. Absorção: $\begin{cases}(\Lambda_1 \sqcap (\Lambda_1 \sqcup \Lambda_2))(\sigma) = \Lambda_1(\sigma)\\ (\Lambda_1 \sqcup (\Lambda_1 \sqcap \Lambda_2))(\sigma) = \Lambda_1(\sigma).\end{cases}$

As operações meet e join sobre $\Lambda(\mathbb{S})$ definem a ordem parcial $\Rightarrow$ sobre $\Lambda(\mathbb{S})$ pela regra $\Lambda_1 \Rightarrow \Lambda_2$ se, e somente se, $(\Lambda_1 \sqcup \Lambda_2) \Rightarrow \Lambda_2$, ou, equivalentemente, $(\Lambda_1 \sqcap \Lambda_2) \Rightarrow \Lambda_1$ (consistência). Então, $\Lambda(\mathbb{S})$ parcialmente ordenado pela implicação $\Rightarrow$ e sob as operações meet e join forma um reticulado completo [MB99].

Para encerrar a discussão sobre as especificações por pré-condições é importante comentar que tais especificações consistem da associação de restrições aos eventos. Tais restrições são expressas por pré-condições definidas sobre as variáveis introduzidas pela coleção de dados e/ou sobre a variável de estado discreto. Portanto, como as restrições dependem dos estados discretos do sistema, é possível a associação de restrições diferentes a diferentes transições rotuladas pelo mesmo evento.

3.3 Equivalência com a teoria clássica

Nos SED, em geral, as especificações podem ser classificadas em (a) especificações de segurança (ou configurações proibidas) e (b) especificações seqüenciais (ou seqüências proibidas). As linguagens alvo da teoria clássica podem descrever requisitos pertencentes a qualquer uma destas classes. As especificações por pré-condições têm (no mínimo) o mesmo poder de expressividade. Para comprovar esta afirmação, os parágrafos a seguir mostram como um problema da teoria clássica pode ser transformado em um STExCD e em uma especificação por pré-condições equivalentes. Tais transformações consistem da (1) obtenção do STE a partir do gerador modelando a planta na teoria clássica, (2) da transformação da linguagem alvo em uma coleção de dados, (3) da construção do STExCD, e (4) da associação de pré-condições aos eventos.

1. **Obtenção de um STE a partir de um gerador**. Seja $G = \langle Q, \Sigma, \delta, q_0, Q_m \rangle$ o gerador usado na teoria clássica para modelar uma planta. O gerador G pode ser transformado no STE $S' = \langle Q', \Sigma', \delta' \rangle$, de acordo com as seguintes regras:

   (a) $Q' = Q$

   (b) $\Sigma' = \Sigma \cup \{ \xi \}$

   (c) $\delta' = \delta \cup \{ \langle q, \xi, q_0 \rangle \mid q \in Q' \}$.

2. **Transformação das linguagens alvo em uma coleção de dados**. Geralmente, as especificações da teoria clássica são obtidas a partir de um conjunto de especificações locais mais simples. Seja $\{ K_1, K_2, \ldots, K_n \}$ este conjunto de especificações locais. Então, a linguagem alvo correspondente à especificação K é gerada por $K = L(H) = L(H_1 \,\|\, H_2 \,\|\, \ldots \,\|\, H_n)$, onde H e H_i, $1 \leq i \leq n$, são geradores tais que $K = L(H)$ e $K_i = L(H_i)$. Para a transformação, é requerido que o conjunto de eventos de H_i seja um subconjunto dos eventos de H; entretanto, não é requerido que os conjuntos sejam iguais. A coleção de dados construída a partir de H consiste da n-tupla $\langle M_{Q_1}, M_{Q_2}, \ldots, M_{Q_n} \rangle$, onde cada M_{Q_i} é o monóide construído da maneira descrita a seguir.

   Para $K_i = L(H_i)$, tal que $H_i' = \langle Q_i, \Sigma_i, \delta_i, q_{i,0}, Q_i \rangle$ é o gerador de $\overline{K}_i$, defina-se o monóide $M_{Q_i} = \langle F_I^*, \circ, nop \rangle$ como:

   (a) $I = Q_i \cup \{ q_\perp \}$, isto é, o conjunto de instanciações I consiste dos estados discretos de H_i' mais a instanciação especial $q_\perp$.

(b) $F_I = \{\ nop\ \} \cup \{\ f_\sigma^{(i)} : Q_i \to Q_i \mid \sigma \in \Sigma'\ \}$. A operação de inicialização $f_\xi^{(i)}$ é definida para cada instanciação $\iota \in Q_i$ como $f_\xi^{(i)}(\iota) = q_{i,0}$. Todas as demais operações $f_\sigma^{(i)}$, tal que $\sigma \in \Sigma_i - \{\xi\}$, são definidas como:

$$\forall\, \iota \in Q_i,\ f_\sigma^{(i)}(\iota) = \begin{cases} \iota_1 & \text{se } \delta_i(\iota, \sigma) = \iota_1 \\ q_\perp & \text{em caso contrário.} \end{cases}$$

3. **Construção do STExCD**. O STExCD equivalente a um problema da teoria clássica é $\mathbb{S} = \langle S', \mathbf{M}_n, \Delta, \mathcal{P}_m \rangle$, onde:

   (a) $S' = \langle Q', \Sigma', \delta' \rangle$ é o STE obtido de acordo com o Item 1.

   (b) $\mathbf{M}_n = \langle M_{Q_1}, M_{Q_2}, \ldots, M_{Q_n} \rangle$ é a coleção de dados obtida de acordo com o Item 2.

   (c) Para todo $q \in Q'$ e $\sigma \in \Sigma'$, se $\delta'(q, \sigma)!$ então $\Delta(q, \sigma) = \langle\ f_\sigma^{(1)}, f_\sigma^{(2)}, \ldots, f_\sigma^{(n)}\ \rangle$.

   (d) $\mathcal{P}_m$ é o predicado $\mathcal{P}_m\langle q, \iota \rangle$, onde q é um estado marcado na planta G e ι é um estado marcado na composição $H_1 \| H_2 \| \ldots \| H_n$.

4. **Especificação por pré-condições**. Finalmente, a especificação Λ é definida para todo evento $\sigma \in \Sigma'$ como:

$$\Lambda(\sigma)\langle q, \iota \rangle = \begin{cases} \mathit{false} & \text{se } \Delta(q, \sigma)(\iota) = q_\perp \\ \mathit{true} & \text{em caso contrário.} \end{cases}$$

Exemplo 3.3.1 (Equivalência com a teoria clássica – LR). *O modelo clássico da linha realimentada consiste do gerador $G = \langle Q, \Sigma, \delta, q_0, Q_m \rangle$ ilustrado na Figura 3.9.*

1. Obtenção do STE. *O STE $S' = \langle Q', \Sigma', \delta' \rangle$ equivalente à planta G anterior é facilmente obtido como:*

   (a) $Q' = Q = \{\mathit{iii}, \mathit{iiw}, \mathit{iwi}, \mathit{iww}, \mathit{wii}, \mathit{wiw}, \mathit{wwi}, \mathit{www}\}$.

   (b) $\Sigma' = \Sigma \cup \{\xi\} = \{s_1, f_1, s_2, f_2, t, r, a, \xi\}$.

   (c) $\delta' = \delta \cup \{\langle q, \xi, q_0 \rangle \mid q \in Q'\}$.

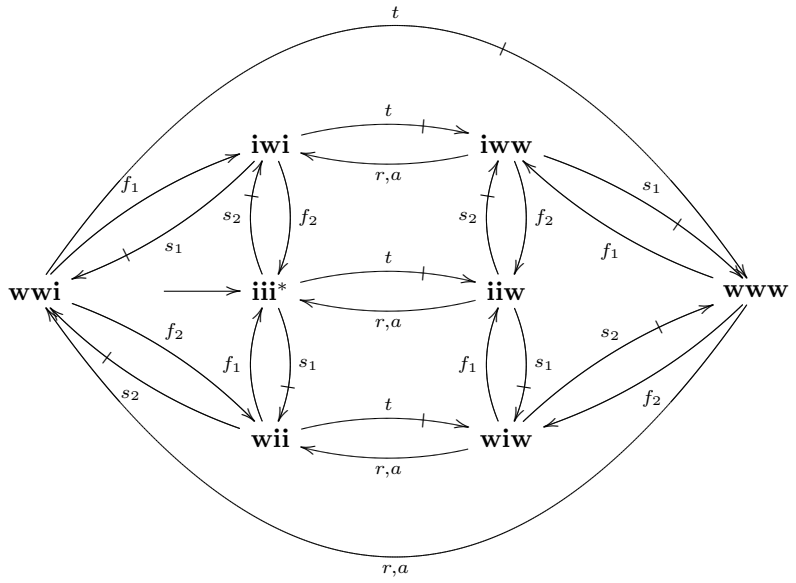

Figura 3.9: Modelo clássico da linha realimentada.

2. Construção da coleção de dados. *Considerando-se que a capacidade dos armazéns A_1 e A_2 são, respectivamente, 3 e 1* [3]*, as especificações locais $K_1 = L(H_1)$ e $K_2 = L(H_2)$ evitando os* underflows *e* overflows *nos armazéns, são representadas pelos geradores H_1 e H_2 da Figura 3.10 (ver [Won98]).*

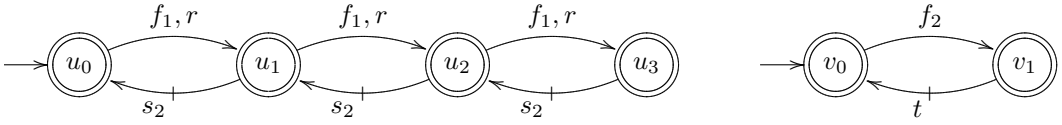

Figura 3.10: Geradores das especificações locais K_1 e K_2 (baseadas em linguagens).

Os monóides M_{Q_1} e M_{Q_2} equivalentes aos geradores H_1 e H_2, são ilustrados graficamente na Figura 3.11. É importante notar que a figura ilustra dois autômatos completos. Os conjuntos de instanciações são $I_1 = Q_1 \cup \{q_\perp\} = \{u_0, u_1, u_2, u_3, q_\perp\}$ e $I_2 = Q_2 \cup \{q_\perp\} = \{v_0, v_1, q_\perp\}$ e as operações em I_1 e I_2 são:

[3] *Aqui, faz-se necessário fixar as capacidades dos armazéns pois a teoria clássica requer a definição exata das linguagens geradas.*

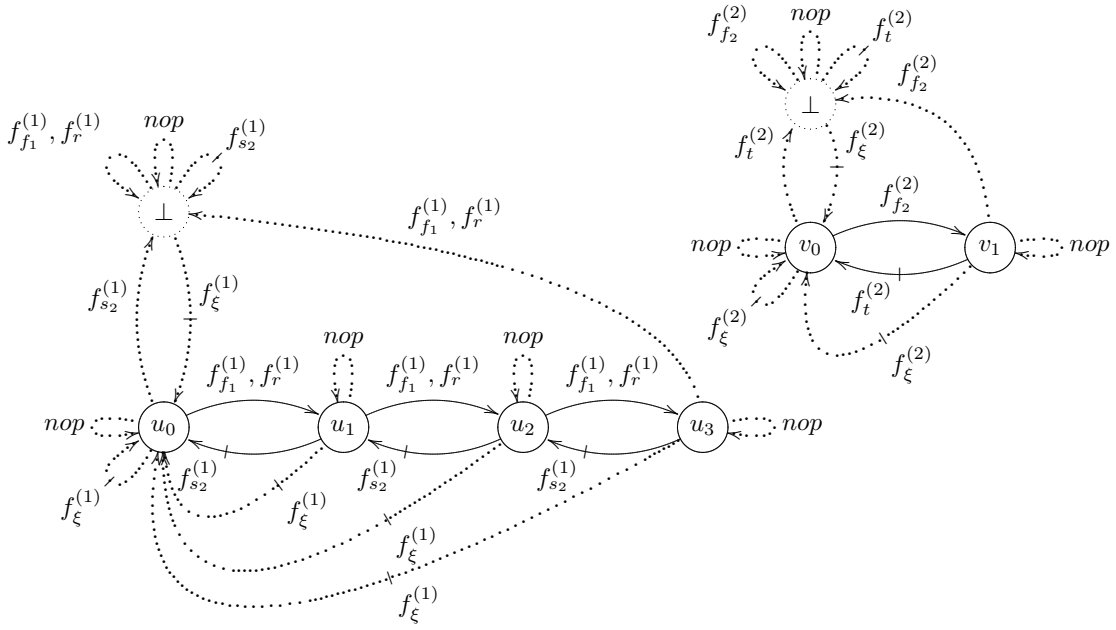

Figura 3.11: Monóides equivalentes às linguagens alvo da Figura 3.10.

- *em* I_1: $\begin{cases} f_{f_1}^{(1)} = f_r^{(1)} = \{\langle u_0, u_1\rangle, \langle u_1, u_2\rangle, \langle u_2, u_3\rangle, \langle u_3, q_\perp\rangle\} \\ f_{s_2}^{(1)} = \{\langle u_3, u_2\rangle, \langle u_2, u_1\rangle, \langle u_1, u_0\rangle, \langle u_0, q_\perp\rangle\} \\ f_\xi^{(1)} = \{\langle u_0, u_0\rangle, \langle u_1, u_0\rangle, \langle u_2, u_0\rangle, \langle u_3, u_0\rangle, \langle q_\perp, u_0\rangle\} \\ nop = \{\langle u_0, u_0\rangle, \langle u_1, u_1\rangle, \langle u_2, u_2\rangle, \langle u_3, u_3\rangle, \langle q_\perp, q_\perp\rangle\} \end{cases}$

- *em* I_2: $\begin{cases} f_{f_2}^{(2)} = \{\langle v_0, v_1\rangle, \langle v_1, q_\perp\rangle\} \\ f_t^{(2)} = \{\langle v_1, v_0\rangle, \langle v_0, q_\perp\rangle\} \\ f_\xi^{(2)} = \{\langle v_0, v_0\rangle, \langle v_1, v_0\rangle, \langle q_\perp, v_0\rangle\} \\ nop = \{\langle v_0, v_0\rangle, \langle v_1, v_1\rangle, \langle q_\perp, q_\perp\rangle\}. \end{cases}$

3. Construção do STExCD. *O STE S' obtido anteriormente equipado com a coleção de dados $M_2 = \langle M_{Q_1}, M_{Q_2} \rangle$, produz o STExCD $\mathbb{S} = \langle S', \mathbf{M}_2, \Delta, \mathcal{P}_m \rangle$, onde Δ é o mapa definido para todo*

estado $q \in Q'$ *e instanciação* $\langle \iota_1, \iota_2 \rangle \in I_1 \times I_2$, *como:*

$$\Delta(q,\xi) = \langle\ f_{\xi}^{(1)},\ f_{\xi}^{(2)}\ \rangle, \qquad \Delta(q,s_1) = \langle\ nop,\ nop\ \rangle, \qquad \Delta(q,f_1) = \langle\ nop,\ f_{f_1}^{(2)}\ \rangle,$$
$$\Delta(q,s_2) = \langle\ f_{s_2}^{(1)},\ nop\ \rangle, \qquad \Delta(q,f_2) = \langle\ nop,\ f_{f_2}^{(2)}\ \rangle, \qquad \Delta(q,t) = \langle\ nop,\ f_{t}^{(2)}\ \rangle,$$
$$\Delta(q,r) = \langle\ f_{r}^{(1)},\ nop\ \rangle, \qquad \Delta(q,a) = \langle\ nop,\ nop\ \rangle.$$

4. Especificação por pré-condições. *A especificação por pré-condições equivalente às especificações locais* K_1 *e* K_2 *é o conjunto de pré-condições impedindo que alguma instanciação contendo* $q_\perp$ *seja alcançada, isto é, para toda instanciação* $\langle \iota_1, \iota_2 \rangle \in I_1 \times I_2$,

$$\Lambda(f_1)\langle \iota_1, \iota_2 \rangle = (\iota_1 \neq u_3), \qquad \Lambda(r)\langle \iota_1, \iota_2 \rangle = (\iota_1 \neq u_3), \qquad \Lambda(s_2)\langle \iota_1, \iota_2 \rangle = (\iota_1 \neq u_0),$$
$$\Lambda(t)\langle \iota_1, \iota_2 \rangle = (\iota_2 \neq v_0), \qquad \Lambda(f_2)\langle \iota_1, \iota_2 \rangle = (\iota_2 \neq v_1).$$

3.4 Observações

Os STExCD possuem uma estrutura de mais alto nível do que as linguagens regulares, provendo o sistema com variáveis, parâmetros, procedimentos e a habilitação condicionada logicamente dos eventos. Tais construtores permitem grande flexibilidade na modelagem de uma extensa classe de SED e seus respectivos comportamentos desejáveis. Em contra partida, a sistematização da construção do modelo da planta e da especificação correspondente foi penalizada. Esta penalização permite diferentes formas de modelar e especificar um mesmo problema.

Capítulo 4

Supervisores e controlabilidade

Como visto na Seção 3.2.3 e sem perda de generalidade, no que segue, apenas especificações por pré-condições são consideradas. Além disso, se nada for dito em contrário, as especificações são próprias, isto é, diferentes de $\bot$. Deste modo, as ações do agente de controle consistem da habilitação/inibição de eventos controláveis da planta de acordo com a satisfação das respectivas pré-condições.

Dada uma especificação por pré-condições $\Lambda \in \Lambda(\mathbb{S})$, a ação de um agente de controle externo, doravante denominado supervisor, é habilitar a partir de cada configuração alcançada um subconjunto dos eventos possíveis de modo que as configurações acessíveis sob supervisão satisfaçam Λ. O subconjunto dos eventos fisicamente possíveis a partir de um estado q é denotado $\Sigma(q)$. O subconjunto dos eventos a serem habilitados a partir de cada configuração é determinado pelo controle induzido por Λ, o qual é definido a seguir.

Definição 4.0.1 (Controle induzido por Λ). *Seja Λ uma especificação por pré-condições para um STExCD $\mathbb{S}$. O* controle induzido por Λ *é o mapa $\gamma_\Lambda : Q \times \mathbf{I}_n \to 2^\Sigma$ definido para toda configuração $\langle q, \iota \rangle \in Reach^{QI}(L(\mathbb{S}))$ como:*

$$\gamma_\Lambda \langle q, \iota \rangle = \{ \sigma \in \Sigma(q) \mid \Lambda(\sigma)\langle q, \iota \rangle = true \}.$$

Em palavras, o controle induzido por Λ associa a cada configuração $\langle q, \iota \rangle$ da planta um subconjunto dos eventos fisicamente possíveis a partir do estado q. Os eventos em $\gamma_\Lambda \langle q, \iota \rangle$ são aqueles cujas pré-condições, avaliadas na configuração $\langle q, \iota \rangle$, resultam verdadeiras.

4.1 Supervisores

O controle induzido por Λ permite uma única estratégia de controle: a habilitação de um subconjunto dos eventos possíveis. Em geral, o subconjunto dos eventos a serem habilitados em cada configuração é determinado pelo supervisor durante o processo de supervisão através das avaliações das pré-condições. A definição a seguir introduz formalmente o supervisor.

Definição 4.1.1 (Supervisor). *Seja $\mathbb{S}$ um STExCD e Λ uma especificação por pré-condições para $\mathbb{S}$. Um* supervisor *para $\mathbb{S}$ e Λ é o mapa $V_\Lambda : L(\mathbb{S}) \to 2^\Sigma$ definido para cada traço $\xi s \in L(\mathbb{S})$ como:*

$$V_\Lambda(\xi s) = \gamma_\Lambda\langle\delta^*(q_0, s), \Delta^*(q_0, s)(\iota_0)\rangle.$$

Informalmente, $V_\Lambda(\xi s)$ define o conjunto de eventos a serem habilitados após o traço ξs. Considerando-se que um evento não controlável, na prática, não pode ser impedido, um supervisor só pode ser construído se todos os eventos a serem desabilitados forem controláveis. Isto leva à seguinte definição de supervisores próprios:

Definição 4.1.2 (Supervisor próprio). *Um supervisor $V_\Lambda : L(\mathbb{S}) \to 2^\Sigma$ é denominado* próprio *se cada subconjunto de eventos habilitados por V_Λ inclui todos os eventos não controláveis fisicamente possíveis, ou seja, se:*

$$\forall\, \xi s \in L(\mathbb{S}), \quad \forall\, \sigma \in \Sigma(\delta^*(q_0, s)), \quad \sigma \in \Sigma_u \Rightarrow \sigma \in V_\Lambda(\xi s).$$

A interação entre uma planta $\mathbb{S}$ e um supervisor próprio V_Λ pode ser informalmente descrita como:

"Cada seqüência de eventos gerada pela planta é memorizada pelo supervisor na forma de uma configuração. No momento em que o supervisor observa (na planta) a ocorrência de um evento α, ele atualiza a configuração corrente $\langle q, \iota\rangle$ de acordo com a função de transição $\delta(q, \alpha)$ e a operação $\Delta(q, \alpha)(\iota)$. Em seguida, a nova configuração $\langle q',\ \iota'\rangle = \langle\delta(q, \alpha),\ \Delta(q, \alpha)(\iota)\rangle$ é usada na avaliação das pré-condições associadas aos eventos $\sigma \in \Sigma(q')$. Se a avaliação da pré-condição $\Lambda(\sigma)\langle q', \iota'\rangle$, onde $\sigma \in \Sigma(q')$, resultar verdade, o evento σ é habilitado. Caso contrário, o evento σ é desabilitado. O tempo decorrido desde a ocorrência (na planta) do evento α até as habilitações dos eventos σ é considerado desprezível."

Formalmente, a interação descrita é modelada pelos sistemas de transição de estados com pré-condições definidos a seguir.

Definição 4.1.3 (STE com pré-condições). *Um* STE com Pré-Condições *(STE-PC) é o sistema* $V_\Lambda/\mathbb{S}$*, consistindo de um STExCD* $\mathbb{S}$ *e um supervisor* V_Λ*, o qual foi construído para satisfazer os requisitos de uma especificação por pré-condições própria* Λ *para* $\mathbb{S}$*.*

Em termos de linguagem gerada, o *comportamento em malha fechada* de $V_\Lambda/\mathbb{S}$ é a linguagem $L(V_\Lambda/\mathbb{S}) \subseteq L(\mathbb{S})$ descrita como:

1. $\xi \in L(V_\Lambda/\mathbb{S})$.
2. Se $\xi s \in L(V_\Lambda/\mathbb{S})$, $\sigma \in V_\Lambda(\xi s)$ e $\xi s\sigma \in L(\mathbb{S})$, então $\xi s\sigma \in L(V_\Lambda/\mathbb{S})$.
3. Nenhum outro traço pertence a $L(V_\Lambda/\mathbb{S})$.

Estendendo-se a linguagem $L(V_\Lambda/\mathbb{S})$ com a cadeia vazia ε, obtém-se $L^\varepsilon(V_\Lambda/\mathbb{S})$. Claramente $\{\varepsilon\} \subseteq L^\varepsilon(V_\Lambda/\mathbb{S}) \subseteq L(\mathbb{S})$ é não vazia e prefixo fechada.

Proposição 4.1.1 (Configurações acessíveis sob supervisão). *O conjunto de configurações acessíveis sob supervisão é* $Reach^{QI}(V_\Lambda/\mathbb{S}) = \mathbb{D}_\Lambda$*.*

Demonstração. *O que deve ser demonstrado é que* $Reach^{QI}(V_\Lambda/\mathbb{S}) \subseteq \mathbb{D}_\Lambda$ *e* $\mathbb{D}_\Lambda \subseteq Reach^{QI}(V_\Lambda/\mathbb{S})$*.*

- $Reach^{QI}(L(V_\Lambda/\mathbb{S})) \subseteq \mathbb{D}_\Lambda$

  De acordo com a definição de $\mathbb{D}_\Lambda$ *a configuração inicial* $\langle q_0, \iota_0\rangle$ *satisfaz* Λ*, isto é,* $\langle q_0, \iota_0\rangle \in \mathbb{D}_\Lambda$*.*

  Seja $\xi s \in L(\mathbb{S})$ *um traço satisfazendo a proposição. Para todo evento* $\sigma \in \Sigma$*, se* $\xi s\sigma \in L(V_\Lambda/\mathbb{S})$ *então, de acordo com a definição do comportamento em malha fechada,* $\sigma \in V_\Lambda(\xi s)$*. Mas,* $\sigma \in V_\Lambda(\xi s)$ *se* $\Lambda(\sigma)\langle\delta^*(q_0, s), \Delta^*(q_0, s)(\iota_0)\rangle = true$*. Portanto, de acordo com a definição de* $\mathbb{D}_\Lambda$*,* $\langle\delta^*(q_0, s\sigma), \Delta^*(q_0, s\sigma)(\iota_0)\rangle \in \mathbb{D}_\Lambda$*.*

- $\mathbb{D}_\Lambda \subseteq Reach^{QI}(L(V_\Lambda/\mathbb{S}))$

  Por contradição, seja $\langle q, \iota\rangle$ *uma configuração pertencente a* $\mathbb{D}_\Lambda$*, mas não pertencente a* $Reach^{QI}(L(V_\Lambda/\mathbb{S}))$*.*

  Pela definição de $\mathbb{D}_\Lambda$*, tem-se* $\langle q_0, \iota_0\rangle \in \mathbb{D}_\Lambda$*. Pela definição do comportamento em malha fechada, tem-se* $\xi \in L(V_\Lambda/\mathbb{S})$ *e* $\langle q_0, \iota_0\rangle \in Reach^{QI}(L(V_\Lambda/\mathbb{S}))$*. Portanto,* $\langle q, \iota\rangle$ *não pode ser igual à configuração inicial* $\langle q_0, \iota_0\rangle$*.*

Para $\langle q, \iota \rangle = \langle \delta^*(q_0, s\sigma),\ \Delta^*(q_0, s\sigma)(\iota_0) \rangle$, *onde* $s \in \Sigma^*$ *e* $\sigma \in \Sigma(\delta^*(q_0, s))$, *se* $\langle q, \iota \rangle \in Reach^{QI}(L(\mathbb{S}))$, *então* $\xi s\sigma \in L(\mathbb{S})$[1]. *Pela definição de* $\mathbb{D}_\Lambda$, *se* $\langle q, \iota \rangle \in \mathbb{D}_\Lambda$, *então* $\langle \delta^*(q_0, s),\ \Delta^*(q_0, s)(\iota_0) \rangle \in \mathbb{D}_\Lambda$ *e* $\Lambda(\sigma)\langle \delta^*(q_0, s),\ \Delta^*(q_0, s)(\iota_0) \rangle = true$[2]. *Pela definição de comportamento em malha fechada, se* $\langle q, \iota \rangle \notin Reach^{QI}(L(V_\Lambda/\mathbb{S}))$, *uma das condições a seguir deve ser verdadeira:*

1. $\xi s\sigma \notin L(\mathbb{S})$ *(contradição* [1]*).*
2. $\sigma \notin V_\Lambda(\xi s)$, *isto é,* $\Lambda(\sigma)\langle \delta^*(q_0, s),\ \Delta^*(q_0, s)(\iota_0) \rangle = false$ *(contradição* [2]*).*
3. $\xi s \notin L(V_\Lambda/\mathbb{S})$, *isto é, ou* $\xi s\sigma \notin L(\mathbb{S})$ *(contradição* [1]*), ou* $\sigma \notin V_\Lambda(\xi s)$ *(contradição* [2]*).*

Portanto, não existem configurações pertencentes a $\mathbb{D}_\Lambda$ *que não pertençam simultaneamente a* $Reach^{QI}(L(V_\Lambda/\mathbb{S}))$ *e vice-versa.*

■

4.2 Controlabilidade

Um dos principais objetivos da síntese de supervisores é a caracterização das especificações para as quais um supervisor pode ser construído. Como em outros modelos para SED, a característica que determina se um supervisor pode ou não ser construído está relacionado com a noção de controlabilidade das especificações.

Definição 4.2.1 (Especificações controláveis). *Uma especificação* $\Lambda \neq \bot$ *é* controlável *se para todo traço* $\xi s \in L(\mathbb{S})$, *tal que* $\langle q, \iota \rangle = \langle \delta^*(q_0, s), \Delta^*(q_0, s)(\iota_0) \rangle \in \mathbb{D}_\Lambda$, *tem-se:*

$$\Sigma(q) - \gamma_\Lambda\langle q, \iota \rangle \subseteq \Sigma_c.$$

Como um caso particular, $\bot$ *é considerada controlável.*

A subtração do conjunto $\gamma_\Lambda\langle q, \iota \rangle$ (de eventos habilitados) do conjunto de eventos (fisicamente possíveis) $\Sigma(q)$ resulta o conjunto de eventos que devem ser desabilitados pelo supervisor. Para uma especificação controlável, a partir de cada configuração, todos os eventos a serem desabilitados devem ser controláveis. A condição de controlabilidade é satisfeita, por exemplo, pela especificação Λ_1 evitando o underflow no armazém A e minimizando as mudanças de modo na máquina M_1 da linha de produção simples (Exemplo 3.2.1), mas não é satisfeita pela especificação Λ_2, a qual tenta, em

algum momento, restringir a ocorrência do evento não controlável f_2. Portanto, Λ_1 é controlável e Λ_2 é não controlável.

A proposição a seguir estabelece as condições para a existência de supervisores.

Proposição 4.2.1 (Existência de supervisores). *Seja Λ uma especificação por pré-condições e $\mathbb{D}_\Lambda$ um conjunto não vazio de configurações satisfazendo Λ. O supervisor $V_\Lambda : L(\mathbb{S}) \to 2^\Sigma$ é próprio se, e somente se, Λ for controlável. Neste caso, $Reach^{QI}(L(V_\Lambda/\mathbb{S})) = \mathbb{D}_\Lambda$.*

Demonstração. • Se existe um supervisor V_Λ então Λ é controlável.

Por contradição, seja Λ uma especificação não controlável e V_Λ um supervisor para Λ, construído da seguinte forma:

$$\forall \xi s \in L(\mathbb{S}),$$
$$V_\Lambda(\xi s) = \gamma_\Lambda\langle\delta^*(q_0, s), \Delta^*(q_0, s)(\iota_0)\rangle =$$
$$\{\, \sigma \in \Sigma \mid \sigma \in \Sigma_u(\delta^*(q_0, s)) \Rightarrow \Lambda(\sigma)\langle\delta^*(q_0, s),\ \Delta^*(q_0, s)(\iota_0)\rangle = \mathit{true}\,\}.$$

Se Λ é não controlável, então existe $\xi s \in L(\mathbb{S})$ tal que

$$\Sigma(\delta^*(q_0, s)) - \gamma_\Lambda\langle\delta^*(q_0, s), \Delta^*(q_0, s)(\iota_0)\rangle \not\subseteq \Sigma_c,$$

isto é, existe pelo menos um evento não controlável $\sigma \in \Sigma(\delta^*(q_0, s))$ tal que $\sigma \notin \gamma_\Lambda\langle\delta^*(q_0, s), \Delta^*(q_0, s)(\iota_0)\rangle$.

Se V_Λ é um supervisor (próprio) para Λ, então, para o traço ξs anterior, tem-se:

$$V_\Lambda(\xi s) = \{\, \sigma \in \Sigma \mid \sigma \in \Sigma_u(\delta^*(q_0, s)) \Rightarrow \Lambda(\sigma)\langle\delta^*(q_0, s), \Delta^*(q_0, s)(\iota_0)\rangle = \mathit{true}\,\}.$$

Como $\sigma \in \Sigma(\delta^*(q_0, s))$ e $\sigma \in \Sigma_u$, a parte da esquerda da implicação anterior é verdadeira. Como a parte da direita é falsa, o evento σ não pode pertencer a $V_\Lambda(\xi s)$. Porém, pela definição de supervisor próprio, todos os eventos não controláveis fisicamente possíveis após o traço ξs devem pertencer a $V_\Lambda(\xi s)$. Logo, V_Λ não é um supervisor para Λ.

- Se Λ é controlável então existe um supervisor V_Λ.

Seja Λ uma especificação controlável, isto é,

$$\forall\, \xi s \in L(\mathbb{S}), \quad \Sigma(\delta^*(q_0, s)) - \gamma_\Lambda\langle\delta^*(q_0, s), \Delta^*(q_0, s)(\iota_0)\rangle \subseteq \Sigma_c$$

ou ainda,

$$\forall\, \xi s \in L(\mathbb{S}), \quad \forall\, \sigma \in \Sigma, \quad \sigma \in \Sigma(\delta^*(q_0, s)) \wedge \sigma \notin V_\Lambda(\xi s) \Rightarrow \sigma \in \Sigma_c.$$

Desenvolvendo-se $\sigma \notin V_\Lambda(\xi s)$, tem-se:

$$\begin{aligned} &\forall\, \xi s \in L(\mathbb{S}), \quad \forall\, \sigma \in \Sigma, \\ &\qquad \sigma \in \Sigma(\delta^*(q_0, s)) \wedge (\Lambda(\sigma)\langle\delta^*(q_0, s), \Delta^*(q_0, s)(\iota_0)\rangle = \mathit{false}) \Rightarrow \sigma \in \Sigma_c \end{aligned}$$

$$\begin{aligned} \therefore \quad &\forall\, \xi s \in L(\mathbb{S}), \quad \forall\, \sigma \in \Sigma, \\ &\qquad \sigma \in \Sigma(\delta^*(q_0, s)) \wedge \underbrace{\sigma \notin \Sigma_c}_{\sigma \in \Sigma_u} \Rightarrow (\Lambda(\sigma)\langle\delta^*(q_0, s), \Delta^*(q_0, s)(\iota_0)\rangle = \mathit{true}), \end{aligned}$$

ou seja,

$$\forall\, \xi s \in L(\mathbb{S}), \quad \forall\, \sigma \in \Sigma, \quad \sigma \in \Sigma_u(\delta^*(q_0, s)) \Rightarrow (\Lambda(\sigma)\langle\delta^*(q_0, s), \Delta^*(q_0, s)(\iota_0)\rangle = \mathit{true}).$$

Esta última implicação é exatamente a definição de supervisor próprio. Portanto, para uma especificação Λ controlável, existe um supervisor V_Λ.

■

Exemplo 4.2.1 (Supervisor – LR). *Nas Tabelas 4.1 e 4.2 são ilustradas, respectivamente, uma especificação não controlável (Λ) e uma especificação controlável (Λ^{ctl}) para a linha realimentada. Na Tabela 4.1, observam-se associações de pré-condições aos eventos não controláveis. Na Tabela 4.2, as pré-condições associadas aos eventos não controláveis, fisicamente possíveis a partir de um estado*

qualquer, são todas verdades.

	s_1	f_1	s_2	f_2	t	r	a
iii	*true*	–	$a_1 > 0$	–	$a_2 > 0$	–	–
wii	–	$a_1 < N$	$a_1 > 0$	–	$a_2 > 0$	–	–
iwi	*true*	–	–	$a_2 < M$	$a_2 > 0$	–	–
wwi	–	$a_1 < N$	–	$a_2 < M$	$a_2 > 0$	–	–
iiw	*true*	–	$a_1 > 0$	–	–	$a_1 < N$	*true*
wiw	–	$a_1 < N$	$a_1 > 0$	–	–	$a_1 < N$	*true*
iww	*true*	–	–	$a_2 < M$	–	$a_1 < N$	*true*
www	–	$a_1 < N$	–	$a_2 < M$	–	$a_1 < N$	*true*

Tabela 4.1: Especificação Λ (não controlável) para a linha realimentada.

	s_1	f_1	s_2	f_2	t	r	a
iii	$a_1 < N$	–	$a_1 > 0 \wedge a_2 < M$	–	$a_1 < N \wedge a_2 > 0$	–	–
wii	–	*true*	$a_1 > 0 \wedge a_2 < M$	–	$a_1 + 1 < N \wedge a_2 > 0$	–	–
iwi	$a_1 < N$	–	–	*true*	$a_1 < N \wedge a_2 > 0$	–	–
wwi	–	*true*	–	*true*	$a_1 + 1 < N \wedge a_2 > 0$	–	–
iiw	$a_1 + 1 < N$	–	$a_1 > 0 \wedge a_2 < M$	–	–	*true*	*true*
wiw	–	*true*	$a_1 > 0 \wedge a_2 < M$	–	–	*true*	*true*
iww	$a_1 + 1 < N$	–	–	*true*	–	*true*	*true*
www	–	*true*	–	*true*	–	*true*	*true*

Tabela 4.2: Especificação Λ^{ctl} (controlável) para a linha realimentada.

Com relação à especificação Λ^{ctl}, deve-se observar o seguinte. De acordo com a Tabela 4.2, na configuração tal que $q = iii$, $a_1 = N - 1$ e $a_2 = M$, os eventos s_1 e t são habilitados. Supondo-se que s_1 acontece, o estado do sistema muda para wii, a partir do qual é habilitado o evento f_1 e desabilitados os eventos s_2 e t. Portanto, apenas f_1 pode acontecer. Quando f_1 acontece, o valor de a_1 muda para $a_1 = N$ e o sistema volta para o estado iii. A nova configuração é tal que $q = iii$, $a_1 = N$ e $a_2 = M$.

A partir da configuração $\langle iii, N, M\rangle$, o sistema não pode mais evoluir, pois as pré-condições associadas aos eventos s_1, s_2 e t, são todas falsas. Se o predicado marcador for $q = iii \wedge a_1 = 0 \wedge a_2 = 0$, pode-se construir um supervisor $V_{\Lambda^{ctl}}$ para atender os requisitos de Λ^{ctl} de acordo com a Tabela 4.2, porém ele é bloqueante e não garante a realização das tarefas descritas pelo predicado marcador.

No exemplo anterior, a especificação Λ^{ctl} foi obtida intuitivamente a partir da especificação Λ. Ambas as especificações evitam os overflows e underflows nos armazéns. A especificação Λ^{ctl} é mais restritiva do que Λ, ou seja, $\mathbb{D}_{\Lambda^{ctl}} \subset \mathbb{D}_{\Lambda}$ ($\Lambda^{ctl} \Rightarrow \Lambda$). Por exemplo, as configurações tais que $q = wii$, $a_1 = N$ e $a_2 \geq 0 \wedge a_2 \leq M$, satisfazem Λ mas não satisfazem Λ^{ctl}. Outra observação importante sobre transformações intuitivas de especificações é que as especificações controláveis obtidas podem não ser minimamente restritivas. Na seção a seguir serão exploradas transformações automáticas de especificações não controláveis de modo que a especificação controlável obtida seja minimamente restritiva. As especificações controláveis minimamente restritivas obtidas destas transformações automáticas podem ser bloqueantes. A eliminação de bloqueios será discutida mais adiante, no Capítulo 5.

4.3 Síntese de supervisores

Um supervisor só pode garantir os requisitos impostos por uma especificação controlável. Entretanto, de uma maneira geral, uma especificação projetada para atender determinados requisitos é não controlável. Nestes casos, freqüentemente, uma especificação não controlável pode ser transformada em uma outra especificação mais restritiva, porém controlável. O problema da síntese de supervisores para STE-PC pode ser enunciado como:

Problema 4.3.1. *Dada uma especificação não controlável Λ, como obter uma outra especificação Λ^{ctl}, tal que:*

1. *Λ^{ctl} seja controlável,*
2. *$\Lambda^{ctl} \Rightarrow \Lambda$, e*
3. *para quaisquer outras especificações controláveis Λ' tal que $\Lambda' \Rightarrow \Lambda$, $\Lambda' \Rightarrow \Lambda^{ctl}$, isto é, Λ^{ctl} deve ser a especificação menos restritiva.*

A idéia fundamental a ser usada na computação de Λ^{ctl} consiste da antecipação dos testes das pré-condições associadas aos eventos não controláveis para os instantes em que os respectivos eventos controláveis precedentes estiverem para serem habilitados. Tais antecipações implicam um controle indireto dos eventos não controláveis. Os conceitos e definições relacionadas com estas antecipações para uma classe de STExCD são apresentados nesta seção. A classe de sistemas considerada inclui os STExCD apresentando as seguintes propriedades:

1. Todas as operações definidas sobre os componentes da coleção de dados são computáveis.

2. Não existem laços fechados de eventos não controláveis.

4.3.1 Existência da máxima especificação controlável

O conjunto de todas as especificações controláveis com relação a um STExCD $\mathbb{S}$, o qual será denotado por $\mathbb{C}(\mathbb{S}) = \{ \Lambda \in \Lambda(\mathbb{S}) \mid \Lambda \text{ é controlável} \}$, apresenta algumas propriedades importantes. Em particular, $\mathbb{C}(\mathbb{S})$ é fechado tanto em relação à composição disjuntiva quanto à composição conjuntiva de especificações controláveis.

Teorema 4.3.1 (Fecho disjuntivo de $\mathbb{C}(\mathbb{S})$). *$\mathbb{C}(\mathbb{S})$ é fechado com relação à composição disjuntiva entre especificações controláveis.*

Demonstração. *Para todo traço $\xi s \in L(\mathbb{S})$, tal que $\langle q, \iota \rangle = \langle \delta^*(q_0, s), \Delta^*(q_0, s)(\iota_0) \rangle$ e toda especificação própria $\Lambda \in \mathbb{C}(\mathbb{S})$, tem-se $\Sigma(q) - \gamma_\Lambda\langle q, \iota \rangle \subseteq \Sigma_c$. Claramente,*

$$\bigcap_{\Lambda \in \mathbb{C}(\mathbb{S})} (\Sigma(q) - \gamma_\Lambda\langle q, \iota \rangle) \subseteq \Sigma_c,$$

ou seja,

$$\left(\Sigma(q) - \bigcup_{\Lambda \in \mathbb{C}(\mathbb{S})} \gamma_\Lambda\langle q, \iota \rangle \subseteq \Sigma_c \right) = \left(\Sigma(q) - \gamma_{\left(\bigsqcup_{\Lambda \in \mathbb{C}(\mathbb{S})} \Lambda \right)}\langle q, \iota \rangle \subseteq \Sigma_c \right).$$

Logo, $\bigsqcup_{\Lambda \in \mathbb{C}(\mathbb{S})} \Lambda$ é controlável e, portanto, $\bigsqcup_{\Lambda \in \mathbb{C}(\mathbb{S})} \Lambda \in \mathbb{C}(\mathbb{S})$.

■

Teorema 4.3.2 (Fecho conjuntivo de $\mathbb{C}(\mathbb{S})$). *$\mathbb{C}(\mathbb{S})$ é fechado com relação à composição conjuntiva entre especificações controláveis.*

Demonstração. *Para todo traço $\xi s \in L(\mathbb{S})$, tal que $\langle q, \iota \rangle = \langle \delta^*(q_0, s), \Delta^*(q_0, s)(\iota_0) \rangle$ e toda especificação própria $\Lambda \in \mathbb{C}(\mathbb{S})$, tem-se $\Sigma(q) - \gamma_\Lambda\langle q, \iota \rangle \subseteq \Sigma_c$. Claramente,*

$$\bigcup_{\Lambda \in \mathbb{C}(\mathbb{S})} (\Sigma(q) - \gamma_\Lambda\langle q, \iota \rangle) \subseteq \Sigma_c,$$

ou seja,

$$\left(\Sigma(q) - \bigcap_{\Lambda \in \mathbb{C}(\mathbb{S})} \gamma_\Lambda \langle q, \iota \rangle \subseteq \Sigma_c\right) = \left(\Sigma(q) - \gamma_{\left(\prod\limits_{\Lambda \in \mathbb{C}(\mathbb{S})} \Lambda\right)} \langle q, \iota \rangle \subseteq \Sigma_c\right).$$

Logo, $\bigsqcap_{\Lambda \in \mathbb{C}(\mathbb{S})} \Lambda$ *é controlável e, portanto,* $\bigsqcap_{\Lambda \in \mathbb{C}(\mathbb{S})} \Lambda \in \mathbb{C}(\mathbb{S})$.

■

Resumidamente, para quaisquer duas (ou mais) especificações Λ_1, $\Lambda_2 \in \mathbb{C}(\mathbb{S})$, tem-se $\Lambda_1 \sqcup \Lambda_2 \in \mathbb{C}(\mathbb{S})$ e $\Lambda_1 \sqcap \Lambda_2 \in \mathbb{C}(\mathbb{S})$. Além disso, para Λ_1, $\Lambda_2 \in \mathbb{C}(\mathbb{S})$ e $\Lambda \in \Lambda(\mathbb{S})$, se $\Lambda_1 \Rightarrow \Lambda$ e $\Lambda_2 \Rightarrow \Lambda$, então, de acordo com a definição dos operadores join e meet, pode-se demonstrar que $\Lambda_1 \sqcup \Lambda_2 \Rightarrow \Lambda$ e $\Lambda_1 \sqcap \Lambda_2 \Rightarrow \Lambda$.

A noção de join (e, por dualidade, meet) pode ser estendida para subconjuntos de especificações controláveis, possivelmente infinitos, conforme as seguintes definições.

Definição 4.3.1 (Especificação controlável máxima e mínima). *Seja* $\Lambda \in \Lambda(\mathbb{S})$ *uma especificação por pré-condições e* $\mathbb{C}(\Lambda) = \{\, \lambda \in \mathbb{C}(\mathbb{S}) \mid \lambda \Rightarrow \Lambda \,\}$ *o subconjunto das especificações controláveis que implicam* Λ.

O menor limite superior *de* $\mathbb{C}(\Lambda)$ *é a especificação* $\Lambda^{\uparrow} = sup\, \mathbb{C}(\Lambda)$ *satisfazendo:*

1. *Para todas as especificações controláveis* $\lambda \in \mathbb{C}(\Lambda)$, *tem-se* $\lambda \Rightarrow \Lambda^{\uparrow}$.
2. *Para qualquer especificação* $\lambda' \in \Lambda(\mathbb{S})$, *se é verdade que* $\lambda \Rightarrow \lambda'$ *para todas as especificações* $\lambda \in \mathbb{C}(\Lambda)$ *então, também é verdade que* $\Lambda^{\uparrow} \Rightarrow \lambda'$.

O maior limite inferior *de* $\mathbb{C}(\Lambda)$ *é a especificação* $\Lambda^{\downarrow} = inf\, \mathbb{C}(\Lambda)$ *satisfazendo:*

1. *Para todas as especificações controláveis* $\lambda \in \mathbb{C}(\Lambda)$, *tem-se* $\Lambda^{\downarrow} \Rightarrow \lambda$.
2. *Para qualquer especificação* $\lambda' \in \Lambda(\mathbb{S})$, *se é verdade que* $\lambda' \Rightarrow \lambda$ *para todas as especificações* $\lambda \in \mathbb{C}(\Lambda)$ *então, também é verdade que* $\lambda' \Rightarrow \Lambda^{\downarrow}$.

Proposição 4.3.3 (Existência da especificação máxima e mínima). *Dada uma especificação* $\Lambda \in \Lambda(\mathbb{S})$, *cada subconjunto de especificações controláveis* $\mathbb{C}(\Lambda) \subseteq \mathbb{C}(\mathbb{S})$ *inclui um elemento máximo, dado por:*

$$\Lambda^{\uparrow} = sup\, \mathbb{C}(\Lambda) = \bigsqcup_{\lambda \in \mathbb{C}(\Lambda)} \lambda$$

e um elemento mínimo, dado por:

$$\Lambda^{\downarrow} = inf\, \mathbb{C}(\Lambda) = \bigsqcap_{\lambda \in \mathbb{C}(\Lambda)} \lambda.$$

Demonstração. *Decorre diretamente de* $\mathbb{C}(\Lambda) \subseteq \mathbb{C}(\mathbb{S})$ *e dos fechos de* $\mathbb{C}(\mathbb{S})$ *em relação à composição disjuntiva (Teorema 4.3.1) e conjuntiva (Teorema 4.3.2).*

■

Portanto, o Problema 4.3.1 sempre tem uma e apenas uma solução. O algoritmo para a computação desta solução é apresentado em seguida.

4.3.2 Algoritmo para computação de $sup\, \mathbb{C}(\Lambda)$

Em geral, um evento σ pode rotular diferentes transições de estado. Por exemplo, no STE da Figura 3.3, o evento f_3 rotula quatro transições diferentes. Como os STE considerados são determinísticos, as transições podem ser totalmente identificadas pelos pares (q_σ, σ), onde q_σ é um estado discreto do STE e σ é um evento, desde que $\delta(q_\sigma, \sigma)!$. No que segue, o foco das discussões em geral estará na transição e não no evento. Entretanto, por questões de clareza, a notação de pares anterior só será usada para identificar transições específicas. Quando for necessário se falar sobre uma transição causada pela ocorrência de um evento σ, sem a necessidade de se enfatizar sua origem, simplesmente será dito 'transição σ'. Portanto, 'evento σ' e 'transição σ' são coisas distintas. Além disso, dependendo da controlabilidade de σ, será dito 'transição controlável σ', se $\sigma \in \Sigma_c$, ou 'transição não controlável σ', se $\sigma \in \Sigma_u$.

Um traço ω é um *caminho* até a transição σ se $\omega\sigma \in L(\mathbb{S})$. O conjunto de caminhos até a transição σ é $Path(\sigma) = \{\omega \mid \omega\sigma \in L(\mathbb{S})\}$. A transição não controlável σ é *indiretamente controlável com relação ao caminho* $\omega \in Path(\sigma)$ se existe uma transição controlável α, tal que, $\omega = \phi_1\alpha\phi_2$, com $\phi_2 \in \Sigma_u^*$. O *controle indireto da transição* σ *com relação ao caminho* $\omega \in Path(\sigma)$ associa a pré-condição $\Lambda(\alpha)\langle q', \iota'\rangle$ à transição α de modo que $\Lambda(\alpha)\langle q', \iota'\rangle$ implique logicamente $\Lambda(\sigma)\langle q, \iota\rangle$.

A implicação lógica $\Lambda(\alpha)\langle q', \iota'\rangle \Rightarrow \Lambda(\sigma)\langle q, \iota\rangle$ pode ser facilmente obtida, fazendo-se:

$$\Lambda(\alpha)\langle q',\ \iota'\rangle = \Lambda(\sigma)\langle \delta^*(q', \alpha\phi_2),\ \Delta^*(q', \alpha\phi_2)(\iota')\rangle,$$

isto é, a restrição em α é igual a restrição em σ, com q substituído por $\delta^*(q', \alpha\phi_2)$ e ι substituído por $\Delta^*(q', \alpha\phi_2)(\iota')$, onde $\alpha\phi_2$ é a subcadeia de eventos desde o estado q' até o estado q. Enfatiza-se que $\Lambda(\sigma)\langle\delta^*(q', \alpha\phi_2), \Delta^*(q', \alpha\phi_2)(\iota')\rangle$ é uma fórmula definida sobre a configuração $\langle q', \iota'\rangle$, a qual implica logicamente $\Lambda(\sigma)\langle q, \iota\rangle$ uma vez que $q = \delta^*(q', \alpha\phi_2)$ e $\iota = \Delta^*(q', \alpha\phi_2)(\iota')$.

A transição σ é *indiretamente controlável* se ela é indiretamente controlável com relação a cada um dos seus caminhos. Como todos os caminhos iniciam com as transições controláveis ξ, todos as transições são (teoricamente) diretamente ou indiretamente controláveis.

De uma maneira geral, o conjunto $Path(\sigma)$ é infinito. Entretanto, para a classe de STE-PC, tal que não existem laços fechados de eventos não controláveis, o controle indireto de uma transição não controlável σ pode ser facilmente computado a partir de sucessivas antecipações de pré-condições, cada qual de uma transição não controlável qualquer (iniciando pela transição σ) para suas transições predecessoras imediatas. Por exemplo, considerando-se sempre apenas uma única transição predecessora, a Figura 4.1 ilustra a antecipação da pré-condição associada à transição σ_3 para a transição controlável α. Esta antecipação é realizada em partes: primeiramente, a pré-condição associada à transição σ_3 é antecipada para a transição σ_2, a qual é uma transição predecessora da transição σ_3. A pré-condição assim obtida é

$$\Lambda(\sigma_2)\langle q_3, \iota_3\rangle = \Lambda(\sigma_3)\langle\delta(q_3, \sigma_2), \Delta(q_3, \sigma_2)(\iota_3)\rangle,$$

a qual é uma fórmula definida em função de $\langle q_3, \iota_3\rangle$ satisfazendo $\Lambda(\sigma_2)\langle q_3, \iota_3\rangle \Rightarrow \Lambda(\sigma_3)\langle q_4, \iota_4\rangle$. É importante observar que a configuração $\langle q_4, \iota_4\rangle$ é igual à configuração $\langle\delta(q_3, \sigma_2), \Delta(q_3, \sigma_2)(\iota_3)\rangle$.

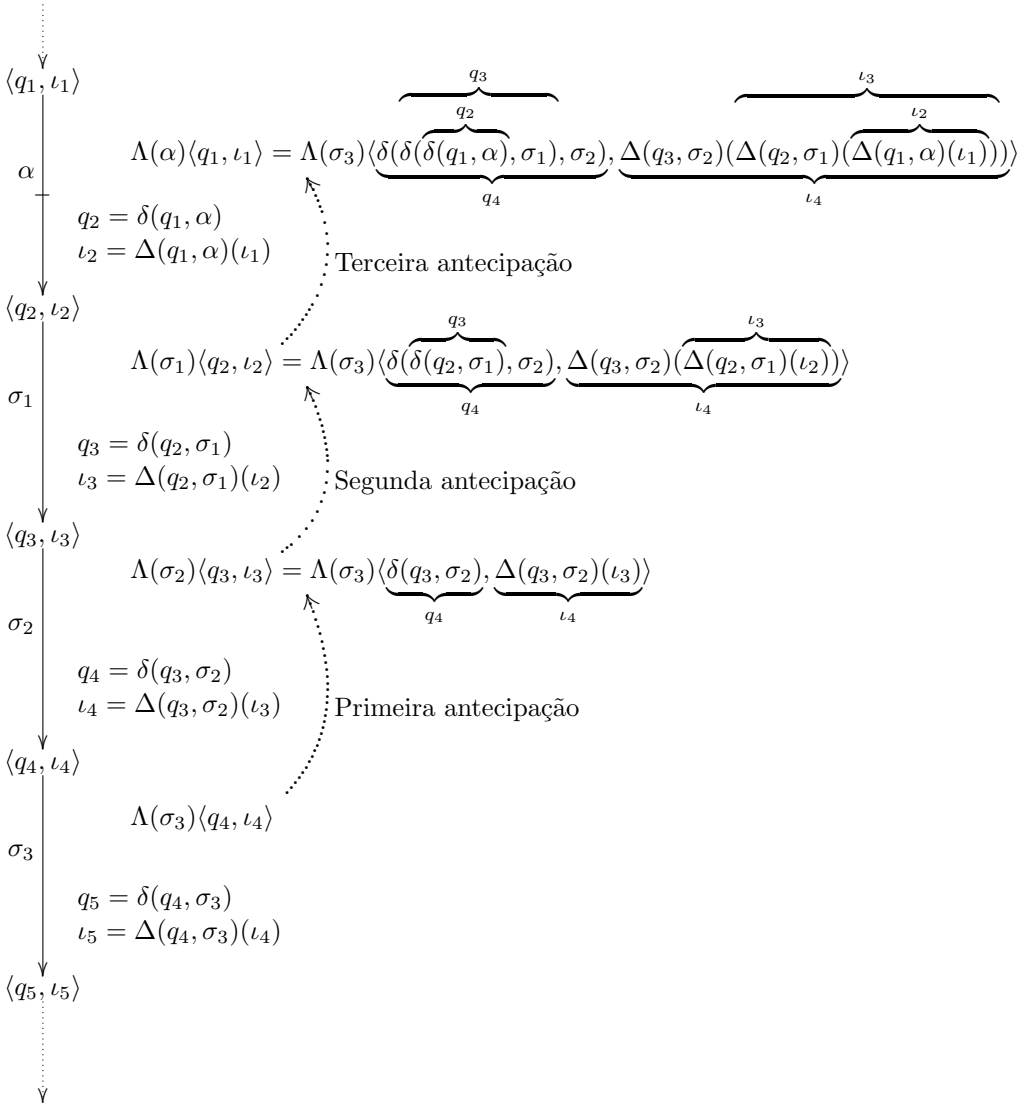

Figura 4.1: Sucessivas antecipações.

Numa segunda antecipação, a pré-condição da transição σ_2 obtida anteriormente é antecipada para sua transição predecessora σ_1. Novamente, observa-se que a configuração $\langle q_4, \iota_4 \rangle$ é igual à configuração $\langle \delta(\delta(q_2, \sigma_1), \sigma_2), \Delta(q_3, \sigma_2)(\Delta(q_2, \sigma_1)(\iota_2)) \rangle$, ou seja, $\langle \delta^*(q_2, \sigma_1\sigma_2), \Delta^*(q_2, \sigma_1\sigma_2)(\iota_2) \rangle$. Finalmente, a pré-condição $\Lambda(\sigma_1)\langle q_2, \iota_2 \rangle$ é antecipada para a transição controlável α da mesma maneira. A pré-condição:

$$\Lambda(\alpha)\langle q_1, \iota_1 \rangle = \Lambda(\sigma_3)\langle \delta(\delta(\delta(q_1, \alpha), \sigma_1), \sigma_2), \Delta(q_3, \sigma_2)(\Delta(q_2, \sigma_1)(\Delta(q_1, \alpha)(\iota_1))) \rangle$$

é equivalente a $\Lambda(\alpha)\langle q_1, \iota_1\rangle = \Lambda(\sigma_3)\langle\delta^*(q_1, \alpha\sigma_1\sigma_2),\ \Delta^*(q_1, \alpha\sigma_1\sigma_2)(\iota_1)\rangle$.

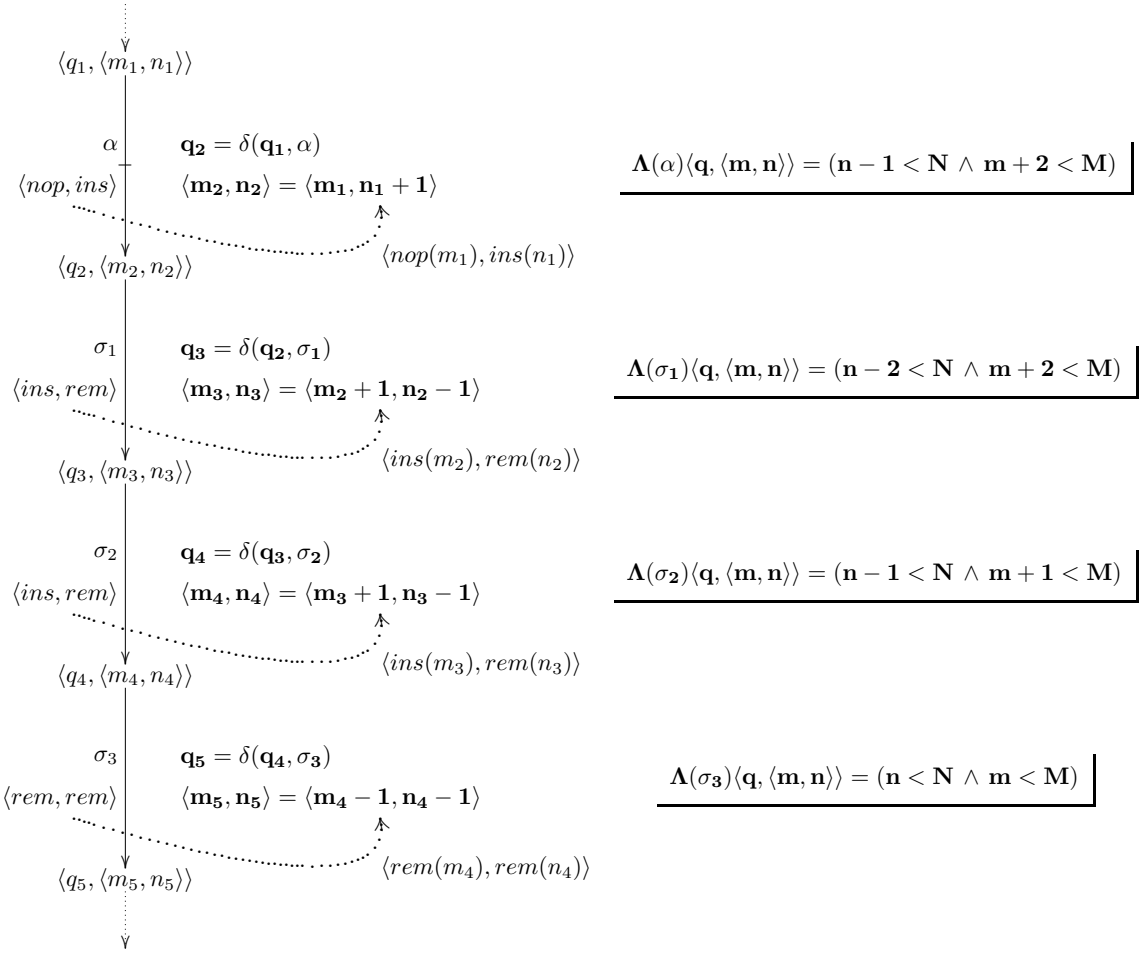

Figura 4.2: Exemplo de sucessivas antecipações.

EXEMPLO 4.3.1 (SUCESSIVAS ANTECIPAÇÕES). *Supondo-se uma coleção de dados igual àquela introduzida no Exemplo 3.1.4, a Figura 4.2 ilustra parcialmente um caminho $\omega = \phi_1\alpha\phi_2$, onde $\alpha\phi_2 = \alpha\sigma_1\sigma_2$, até a transição (q_4, σ_3), pertencente a um sistema fictício. Na figura, a guarda a ser antecipada é $\Lambda(\sigma_3)\langle q, \langle m, n\rangle\rangle = (n < N \wedge m < M)$. Do lado esquerdo dos arcos representando as transições são indicados o evento e a lista de operações associados a cada transição. Do lado direito, são representadas as equações usadas para a determinação das configurações alcançadas após a ocorrência das transições. Nas equações e pré-condições, n e m denotam variáveis e n_i e m_j são instanciações de n e m, respectivamente. Todas as equações relacionando instanciações da coleção*

de dados são relativas à instanciação $\langle m_4, n_4\rangle$ *no estado* q_4*. Por exemplo, com relação à instanciação* $\langle m_4, n_4\rangle$*, a instanciação* $\langle m_2, n_2\rangle$*, no estado* q_2*, é igual a* $\langle m_4+2, n_4-2\rangle$ *e a instanciação* $\langle m_5, n_5\rangle$*, no estado* q_5*, é igual a* $\langle m_4-1, n_4-1\rangle$*. Mais especificamente, se no estado* q_4*,* $n_4 = 4$ *e* $m_4 = 5$*, no estado* q_2*,* $n_2 = 6$ *e* $m_2 = 3$*; no estado* q_5*,* $n_5 = 3$ *e* $m_5 = 4$*.*

Na primeira antecipação, da transição σ_3 *para a transição* σ_2*, a pré-condição associada à transição* σ_2 *é* $\Lambda(\sigma_2)\langle q, \langle m, n\rangle\rangle = (n-1 < N \wedge m+1 < M)$*. Instanciando-se* n *com* n_3 *e* m *com* m_3*, a pré-condição de* σ_2 *seria:*

$$\begin{aligned}\Lambda(\sigma_2)\langle q_3, \langle m_3, n_3\rangle\rangle &= (n_3 - 1 < N \wedge m_3 + 1 < M)\\ &= (n_4 < N \wedge m_4 < M).\end{aligned}$$

Na segunda antecipação, a pré-condição $\Lambda(\sigma_2)\langle q, \langle m, n\rangle\rangle$ *é antecipada para a transição* σ_1*. Esta antecipação resulta a pré-condição* $\Lambda(\sigma_1)\langle q, \langle m, n\rangle\rangle = (n - 2 < N \wedge m + 2 < M)$*. Novamente, instanciando-se* n *com* n_2 *e* m *com* m_2*, a pré-condição de* σ_1 *seria:*

$$\begin{aligned}\Lambda(\sigma_1)\langle q_2, \langle m_2, n_2\rangle\rangle &= (n_2 - 2 < N \wedge m_2 + 2 < M)\\ &= (n_3 - 1 < N \wedge m_3 + 1 < M)\\ &= (n_4 < N \wedge m_4 < M).\end{aligned}$$

Finalmente, na terceira e última antecipação, a pré-condição de σ_1 *é antecipada para a transição controlável* α*. A pré-condição resultante desta antecipação é* $\Lambda(\alpha)\langle q, \langle m, n\rangle\rangle = (n-1 < N \wedge m+2 < M)$*. Instanciando-se* n *com* n_1 *e* m *com* m_1*, a pré-condição de* α *seria:*

$$\begin{aligned}\Lambda(\alpha)\langle q_2, \langle m_2, n_2\rangle\rangle &= (n_1 - 1 < N \wedge m_1 + 2 < M)\\ &= (n_2 - 2 < N \wedge m_2 + 2 < M)\\ &= (n_3 - 1 < N \wedge m_3 + 1 < M)\\ &= (n_4 < N \wedge m_4 < M).\end{aligned}$$

Observa-se que $n_4 = n_3 - 1 = n_2 - 2 = n_1 - 1$ *e* $m_4 = m_3 + 1 = m_2 + 2 = m_1 + 2$*. Logo, a fórmula* $(n_1 - 1 < N \wedge m_1 + 2 < M)$*, definida sobre a instanciação* $\langle m_1, n_1\rangle$*, implica logicamente a fórmula original* $(n_4 < N \wedge m_4 < M)$*, definida sobre a instanciação* $\langle m_4, n_4\rangle$*. Estendendo-se esta observação para qualquer estado* q *e quaisquer instanciações* n *e* m*, conclui-se que* $\Lambda(\alpha)\langle q, \langle m, n\rangle\rangle$*,*

testada durante a habilitação de α, implica logicamente $\Lambda(\sigma_3)\langle q, \langle m, n\rangle\rangle$.

Por questões de clareza, nas explanações anteriores foi omitida a possibilidade das transições α, σ_1 e σ_2 também possuírem pré-condições. Nestes casos, as pré-condições antecipadas e as pré-condições originais devem ser combinadas com o operador meet. O procedimento completo, o qual realiza as antecipações de todas as pré-condições associadas às transições não controláveis para as respectivas transições controláveis predecessoras é apresentado no Algoritmo 4.3.1.

Algoritmo 4.3.1 Síntese da máxima especificação controlável.

Dado o STExCD $\mathbb{S} = \langle G, \mathbf{M}_n, \Delta, \mathcal{P}_m\rangle$ e uma especificação Λ^{org}, possivelmente não controlável, os passos a seguir realizam sucessivas antecipações de pré-condições até que todas sejam associadas a transições controláveis. A especificação Λ é local e a especificação Λ^{ctl} resultante é controlável. A variável done é local e booleana.

$\Lambda \leftarrow \Lambda^{org}$
done $\leftarrow$ *false*
/ Laço principal */*
Enquanto $\neg$*done* **Faça**
  done $\leftarrow$ *true*
  / Seleção das pré-condições a serem antecipadas */*
  Para cada transição da forma $\delta(q_\sigma, \sigma)$ **Faça**
    Se $\sigma \in \Sigma_u \wedge \Lambda(\sigma)\langle q_\sigma, \iota_\sigma\rangle \neq true$ **então**
      / Antecipações da pré-condição $\Lambda(\sigma)\langle q_\sigma, \iota_\sigma\rangle$ */*
      Para cada transição da forma $\delta(q_\alpha, \alpha)$ **Faça**
        / α é predecessor de σ? */*
        Se $\delta(q_\alpha, \alpha) = q_\sigma$ **então**
          / Antecipação de $\Lambda(\sigma)$ para $\alpha \in Pred(\sigma)$ */*
          $\Lambda(\alpha)\langle q_\alpha, \iota_\alpha\rangle \leftarrow \Lambda(\alpha)\langle q_\alpha, \iota_\alpha\rangle \wedge \Lambda(\sigma)\langle\delta(q_\alpha, \alpha), \Delta(q_\alpha, \alpha)(\iota_\alpha)\rangle$
          done $\leftarrow$ *false*
        Fim Se
      Fim Para
      / A restrição sobre σ é irrelevante */*
      $\Lambda(\sigma)\langle q_\sigma, \iota_\sigma\rangle \leftarrow true$
    Fim Se
  Fim Para
Fim Enquanto
$\Lambda^{ctl} \leftarrow \Lambda$

No algoritmo, a variável *done* é usada para sinalizar a convergência das antecipações. Inicial-

mente, a especificação original Λ^{org} é copiada para a especificação Λ, a ser modificada, e a variável *done* é inicializada com *false*. O laço principal testa a convergência das antecipações através da variável *done*: quando *done* assume o valor verdade, as antecipações convergiram e nenhuma nova iteração é necessária. Dentro do laço principal, o primeiro comando **for** e o primeiro comando **if** selecionam a pré-condição a ser antecipada: $\Lambda(\sigma)\langle q_\sigma, \iota_\sigma\rangle$. Somente pré-condições não controláveis e diferentes de *true* são selecionadas. O segundo comando **for** juntamente com o segundo comando **if** selecionam as transições (α) predecessoras da transição σ. A atribuição $\Lambda(\alpha)\langle q_\alpha, \iota_\alpha\rangle \leftarrow \Lambda(\alpha)\langle q_\alpha, \iota_\alpha\rangle \wedge \Lambda(\sigma)\langle \delta(q_\alpha, \alpha), \Delta(q_\alpha, \alpha)(\iota_\alpha)\rangle$ realiza a antecipação. Após cada antecipação, a necessidade de uma nova iteração é sinalizada pela atribuição *done* $\leftarrow$ *false*. Finalmente, ao final de todas as antecipações da pré-condição associada à transição σ, esta é marcada como *true*.

Em termos mais formais, seja σ uma transição não controlável, $Pred(\sigma)$ o conjunto de todas as transições predecessoras da transição σ, $\alpha \in Pred(\sigma)$, Λ_i a especificação Λ na i-ésima iteração do laço principal, $\Lambda_i(\sigma)\langle q_\sigma, \iota_\sigma\rangle$ a pré-condição sobre a transição σ e $\langle q_\alpha, \iota_\alpha\rangle$ a configuração anterior à configuração $\langle q_\sigma, \iota_\sigma\rangle$, tal que $q_\sigma = \delta(q_\alpha, \alpha)$ e $\iota_\sigma = \Delta(q_\alpha, \alpha)(\iota_\alpha)$. Então, o efeito da execução completa do comando **for** mais interno do Algoritmo 4.3.1 é dado por:

$$\left(\bigsqcup_{\alpha \in Pred(\sigma)} \Lambda_i(\alpha)\langle q_\alpha, \iota_\alpha\rangle \wedge \overbrace{\Lambda_i(\sigma)\langle \underbrace{\delta(q_\alpha, \alpha)}_{q_\sigma}, \underbrace{\Delta(q_\alpha, \alpha)(\iota_\alpha)}_{\iota_\sigma}\rangle}^{\text{Antecipação de } \Lambda_i(\sigma)\langle q_\sigma, \iota_\sigma\rangle \text{ para } \alpha} \right) \Rightarrow \Lambda_i(\sigma)\langle q_\sigma, \iota_\sigma\rangle. \tag{4.1}$$

Em palavras, a execução completa do comando **for** mais interno produz um conjunto de pré-condições, associadas às transições predecessoras da transição σ, que, combinadas disjuntivamente, implicam a pré-condição da transição σ. Portanto, após a execução completa do comando **for** mais interno, a pré-condição da transição σ é irrelevante e pode ser eliminada.

Claramente, $\Lambda_i(\alpha)\langle q_\alpha, \iota_\alpha\rangle \wedge \Lambda_i(\sigma)\langle \delta(q_\alpha, \alpha), \Delta(q_\alpha, \alpha)(\iota_\alpha)\rangle$ é a pré-condição da transição α na $(i+1)$-ésima iteração. Isto permite que a Implicação 4.1 seja reescrita como:

$$\left(\bigsqcup_{\alpha \in Pred(\sigma)} \Lambda_{i+1}(\alpha)\langle q_\alpha, \iota_\alpha\rangle \right) \Rightarrow \Lambda_i(\sigma)\langle q_\sigma, \iota_\sigma\rangle, \tag{4.2}$$

onde $\Lambda_{i+1}(\alpha)\langle q_\alpha, \iota_\alpha\rangle = \Lambda_i(\alpha)\langle q_\alpha, \iota_\alpha\rangle \wedge \Lambda_i(\sigma)\langle \delta(q_\alpha, \alpha), \Delta(q_\alpha, \alpha)(\iota_\alpha)\rangle$.

Estendendo-se a Implicação 4.2 a todas as transições não controláveis e diferentes de *true*, pode-se expressar os efeitos da execução completa do comando **for** mais externo (ou seja, de cada iteração

do laço principal) através da implicação:

$$\Lambda_{i+1} \Rightarrow \Lambda_i. \tag{4.3}$$

Nota-se que entre a especificação Λ_i e a especificação Λ_{i+1} da iteração principal subseqüente, existem outras especificações intermediárias (λ) associadas às iterações do comando **for** mais externo. Assim,

$$(\Lambda_{i+1} = \lambda_p \sqcup \lambda_{p-1} \sqcup \ldots \sqcup \lambda_j \sqcup \ldots \sqcup \lambda_1) \Rightarrow \Lambda_i, \tag{4.4}$$

onde p é o número de transições não controláveis de Λ_i e cada λ_j, para $1 \leq j \leq p$, é a especificação intermediária resultante da antecipação de uma única pré-condição de Λ_i para suas respectivas transições predecessoras. Nota-se também, que para cada especificação intermediária λ_j, tem-se $\lambda_j \Rightarrow \Lambda_i$; além disso, se cada λ_j do join generalizado anterior é controlável, então Λ_{i+1} também é controlável.

Como não existem laços fechados contendo apenas eventos não controláveis e todos os traços do sistema iniciam com as transições controláveis ξ, o número de iterações do algoritmo é finito, isto é, o algoritmo termina. Quando uma especificação Λ associa uma pré-condição diferente de *true* a alguma transição não controlável, a qual não é precedida por nenhuma transição controlável diferente de ξ, a pré-condição é antecipada para todas as transições ξ e a especificação resultante, dependendo das restrições e das configurações iniciais, pode ser $\Lambda^{ctl} = \bot \Rightarrow \Lambda$. Como o supervisor foi definido apenas sobre especificações próprias, o resultado $\Lambda^{ctl} = \bot$ indica que não existe um supervisor capaz de atender os requisitos de Λ.

Estendendo-se a implicação 4.3 a todas as $n+1$ iterações do laço principal do algoritmo, pode-se escrever:

$$\underbrace{(\Lambda_{n+1} \Leftrightarrow \Lambda_n)}_{\text{Equivalência}} \Rightarrow \Lambda_{n-1} \Rightarrow \cdots \Rightarrow \Lambda_{i+1} \Rightarrow \Lambda_i \Rightarrow \cdots \Rightarrow (\Lambda_0 \Leftrightarrow \Lambda^{org}). \tag{4.5}$$

A equivalência $\Lambda_{n+1} \Leftrightarrow \Lambda_n$ indica que na $n+1$-ésima iteração do laço principal nenhuma antecipação foi realizada. De acordo com o algoritmo, isto acontece somente quando Λ_n é controlável.

Exemplo 4.3.2 (Antecipações de pré-condições – LPS). *Aplicando-se o Algoritmo 4.3.1 sobre a especificação Λ da linha de produção simples (Exemplo 3.2.5), obtém-se o supervisor ilustrado na Tabela 4.3. Nota-se que o número de estados discretos do supervisor é igual ao número de estados*

do STE modelando a planta, isto é, o número de estados discretos do supervisor não depende das especificações por pré-condições.

	s_1	f_1	s_2	f_2	m_2	m_1	s_3	f_3
i_1i_3	*true*	—	—	—	*true*	—	$b > 0$	—
i_1w_3	*true*	—	—	—	*true*	—	—	*true*
w_1i_3	—	*true*	—	—	—	—	$b > 0$	—
w_1w_3	—	*true*	—	—	—	—	—	*true*
i_2i_3	—	—	$a > 0 \wedge b < N$	—	—	$a = 0$	$b > 0$	—
i_2w_3	—	—	$a > 0 \wedge b < N$	—	—	$a = 0$	—	*true*
w_2i_3	—	—	—	*true*	—	—	$b > 0 \wedge b - 1 < N$	—
w_2w_3	—	—	—	*true*	—	—	—	*true*

Tabela 4.3: Supervisor da linha de produção simples do Exemplo 3.1.5 e especificação Λ do Exemplo 3.2.6.

Na especificação Λ descrita no Exemplo 3.2.5, as pré-condições associadas às transições não controláveis são $\Lambda(f_2)\langle w_2i_3, \langle a, b\rangle\rangle = (b < N)$ e $\Lambda(f_2)\langle w_2w_3, \langle a, b\rangle\rangle = (b < N)$ (ver Figura 3.3).

Na primeira iteração do Algoritmo 4.3.1, a pré-condição $\Lambda(f_2)\langle w_2i_3, \langle a, b\rangle\rangle = (b < N)$ é antecipada para as transições (i_2i_3, s_2) e (w_2w_3, f_3) e, a pré-condição $\Lambda(f_2)\langle w_2w_3, \langle a, b\rangle\rangle = (b < N)$ é antecipada para as transições (i_2w_3, s_2) e (w_2i_3, s_3). O resultado destas antecipações é:

$$\begin{aligned}
\Lambda(s_2)\langle i_2i_3, \langle a, b\rangle\rangle &= (a > 0 \wedge b < N),\\
\Lambda(f_3)\langle w_2w_3, \langle a, b\rangle\rangle &= (b < N),\\
\Lambda(s_2)\langle i_2w_3, \langle a, b\rangle\rangle &= (a > 0 \wedge b < N),\\
\Lambda(s_3)\langle w_2i_3, \langle a, b\rangle\rangle &= (b > 0 \wedge b - 1 < N).
\end{aligned}$$

A especificação resultante da primeira iteração não é controlável devido à pré-condição $\Lambda(f_3)\langle w_2w_3, \langle a, b\rangle\rangle = (b < N)$.

Na segunda iteração do Algoritmo 4.3.1, a pré-condição $\Lambda(f_3)\langle w_2w_3, \langle a, b\rangle\rangle = (b < N)$ é antecipada para as transições (i_2w_3, s_2) e (w_2i_3, s_3). O resultado destas duas antecipações é:

$$\begin{aligned}
\Lambda(s_2)\langle i_2w_3, \langle a, b\rangle\rangle &= (a > 0 \wedge b < N \wedge b < N),\\
\Lambda(s_3)\langle w_2i_3, \langle a, b\rangle\rangle &= (b > 0 \wedge b - 1 < N \wedge b - 1 < N).
\end{aligned}$$

A especificação resultante da segunda iteração é controlável. Na terceira iteração, nenhuma anteci-

pação é realizada e o Algoritmo 4.3.1 termina.

O principal resultado desta seção é apresentado na forma da seguinte proposição:

Proposição 4.3.4 (Resultado do Algoritmo 4.3.1). *A especificação Λ^{ctl} produzida pelo Algoritmo 4.3.1 é igual à máxima especificação controlável $\Lambda^{\uparrow}$.*

Demonstração. *Para a demonstração da Proposição 4.3.4 será usado o método $\triangle$ introduzido em [ZKW99]. Este método define uma relação $\triangle \subseteq \Lambda(\mathbb{S}) \times \Lambda(\mathbb{S})$, reflexiva com relação a um conjunto de pontos fixos $\Lambda_{\triangle}(\mathbb{S})$ de um operador monotônico $\Psi : \Lambda(\mathbb{S}) \to \Lambda(\mathbb{S})$. Em seguida, a especificação controlável minimamente restritiva é computada como o maior ponto fixo [Tar55] do operador Ψ. Para a aplicação do método, é requerido que $\Lambda(\mathbb{S})$ seja um reticulado completo e $\Lambda_{\triangle}(\mathbb{S})$ um semi-reticulado superior com relação à operação join de $\Lambda(\mathbb{S})$.*

Aplicação do método $\triangle$ [ZKW99]

Seja $\triangle \subseteq \Lambda(\mathbb{S}) \times \Lambda(\mathbb{S})$ a relação em $\Lambda(\mathbb{S})$ definida como:

$$\triangle = \{ (\lambda, \Lambda) \mid \lambda,\ \Lambda \in \Lambda(\mathbb{S}) \text{ e } \lambda \Rightarrow \Lambda \}.$$

Nota-se que, além de outros pares, a relação $\triangle$ inclui os pares (λ_j, Λ_i) da Implicação 4.4.

No método $\triangle$, o operador supremo *$\Psi : \Lambda(\mathbb{S}) \to \Lambda(\mathbb{S})$, onde $\Lambda(\mathbb{S})$ é um reticulado completo, é definido como:*

$$\Psi(\Lambda_i) = \bigsqcup_{j=1}^{p} \{ \lambda_j \mid (\lambda_j, \Lambda_i) \in \triangle \} = \Lambda_{i+1}.$$

Comparando-se esta definição com a Implicação 4.4, conclui-se que $\Psi(\Lambda_i)$ realiza a antecipação de todas as pré-condições não controláveis de Λ_i para as respectivas transições predecessoras. No Algoritmo 4.3.1, o operador Ψ é implementado pelos dois comandos **for** *aninhados.*

De acordo com a Proposição 3 de [ZKW99], o conjunto de pontos fixos de Ψ é $\Lambda_{\triangle}(\mathbb{S}) = \{ \lambda \in \Lambda(\mathbb{S}) \mid \lambda \Rightarrow \Psi(\Lambda) \}$. Para cada $\lambda \in \Lambda(\mathbb{S})$, o par (λ, λ) está em $\triangle$ se, e somente se, $\lambda \in \mathbb{C}(\Lambda)$. Portanto, a relação $\triangle$ é reflexiva em $\Lambda_{\triangle}(\mathbb{S}) = \mathbb{C}(\Lambda)$ e,

$$\Lambda^{\uparrow} = \bigsqcup \{ \lambda \mid \lambda \in \Lambda_{\triangle}(\mathbb{S}),\ \lambda \Rightarrow \Lambda^{org} \}.$$

De acordo com o Teorema 4.3.1, para o conjunto $\{(\lambda^\alpha, \Lambda) \in \triangle \mid \alpha \in A\}$*, onde* A *é algum conjunto de índices, tem-se:*

$$\left(\bigsqcup_\alpha \lambda^\alpha\right) \Rightarrow \Lambda.$$

Portanto, $(\sqcup_\alpha \lambda^\alpha, \Lambda) \in \triangle$*. Além disso, se* $(\Lambda_2', \Lambda_2) \in \triangle$ *e* $\Lambda_2' \Rightarrow \Lambda_2$*, então* $\Lambda_2' \Rightarrow \Lambda_2 \Rightarrow \Lambda_1$*, ou seja,* $(\Lambda_2', \Lambda_1) \in \triangle$*. De acordo com a Proposição 4 de [ZKW99], se* $(\Lambda_2, \Lambda_1) \in \triangle$ *e* $\Lambda_1 \Rightarrow \Lambda_1'$ *implica* $(\Lambda_2, \Lambda_1') \in \triangle$*,* Ψ *deve ser monotônico. Facilmente estas condições podem ser verificadas e, portanto, o operador* Ψ *definido anteriormente é monotônico.*

De acordo com a Definição 4 de [ZKW99], para $\Lambda_i \in \Lambda(\mathbb{S})$*,*

$$\Psi(\Lambda_i) = \bigsqcup_{j=1}^{p} \{\lambda_j \mid \lambda_j \Rightarrow \Lambda_i\}.$$

A iteração do Teorema 3 de [ZKW99] se torna:

$$\begin{aligned} \bot &= \Lambda^{org} \\ \Lambda_{i+1} &= \Psi(\Lambda_i). \end{aligned} \tag{4.6}$$

Claramente, esta iteração gera a seqüência de Implicações 4.5 e, portanto, o Algoritmo 4.3.1 é uma implementação particular do Teorema 3 de [ZKW99].

Estabelecida a equivalência do Algoritmo 4.3.1 com o Teorema 3 de [ZKW99], pode-se concluir que o resultado do Algoritmo 4.3.1 (ou das Iterações 4.6), é:

$$\Lambda^\uparrow = \bigsqcup \{\lambda \mid \lambda \in \mathbb{C}(\Lambda),\ \lambda \Rightarrow \Lambda^{org}\} = \Lambda_n.$$

■

Logo, nossa solução para o Problema 4.3.1 é ótima.

Exemplo 4.3.3 (Antecipações de pré-condições – LR). *Aplicando-se o Algoritmo 4.3.1 à especificação do Exemplo 3.2.6 para a linha realimentada, são realizadas as seguintes antecipações:*
A pré-condição do evento r *cuja origem é* iiw*, é antecipada para os eventos:*

- t *partindo do estado* iii*, isto é,* $\Lambda(t)\langle iii, \iota\rangle = \Lambda(r)\langle iiw, \Delta^*(iii, t)(\iota)\rangle$

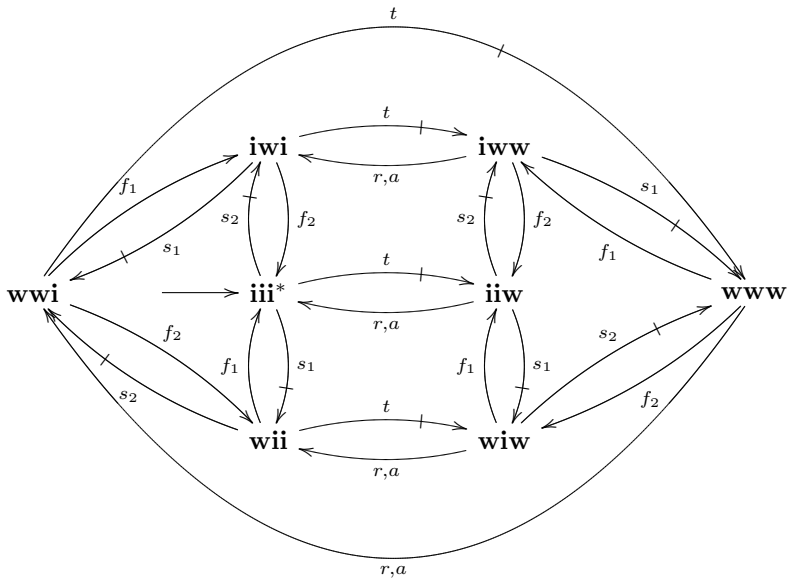

Figura 4.3: STE da linha realimentada.

- s_1 *partindo do estado* iiw, *isto é,* $\Lambda(s_1)\langle iiw, \iota\rangle = \Lambda(r)\langle iiw, \Delta^*(iiw, s_1 f_1)(\iota)\rangle$
- s_2 *partindo do estado* iiw, *isto é,* $\Lambda(s_2)\langle iiw, \iota\rangle = \Lambda(r)\langle iiw, \Delta^*(iiw, s_2 f_2)(\iota)\rangle$
- t *partindo do estado* iwi, *isto é,* $\Lambda(t)\langle iwi, \iota\rangle = \Lambda(r)\langle iiw, \Delta^*(iwi, t f_2)(\iota)\rangle$
- s_1 *partindo do estado* iww, *isto é,* $\Lambda(s_1)\langle iww, \iota\rangle = \Lambda(r)\langle iiw, \Delta^*(iww, s_1 f_1 f_2 + s_1 f_2 f_1)(\iota)\rangle$
- t *partindo do estado* wii, *isto é,* $\Lambda(t)\langle wii, \iota\rangle = \Lambda(r)\langle iiw, \Delta^*(wii, t f_1)(\iota)\rangle$
- s_2 *partindo do estado* wiw, *isto é,* $\Lambda(s_2)\langle wiw, \iota\rangle = \Lambda(r)\langle iiw, \Delta^*(wiw, s_2 f_1 f_2 + s_2 f_2 f_1)(\iota)\rangle$
- t *partindo do estado* wwi, *isto é,* $\Lambda(t)\langle wwi, \iota\rangle = \Lambda(r)\langle iiw, \Delta^*(wwi, t f_1 f_2 + t f_2 f_1)(\iota)\rangle$

A pré-condição do evento a *cuja origem é* iiw, *é antecipada para os eventos:*

- t *partindo do estado* iii, *isto é,* $\Lambda(t)\langle iii, \iota\rangle = \Lambda(a)\langle iiw, \Delta^*(iii, t)(\iota)\rangle$
- s_1 *partindo do estado* iiw, *isto é,* $\Lambda(s_1)\langle iiw, \iota\rangle = \Lambda(a)\langle iiw, \Delta^*(iiw, s_1 f_1)(\iota)\rangle$
- s_2 *partindo do estado* iiw, *isto é,* $\Lambda(s_2)\langle iiw, \iota\rangle = \Lambda(a)\langle iiw, \Delta^*(iiw, s_2 f_2)(\iota)\rangle$
- t *partindo do estado* iwi, *isto é,* $\Lambda(t)\langle iwi, \iota\rangle = \Lambda(a)\langle iiw, \Delta^*(iwi, t f_2)(\iota)\rangle$

- s_1 *partindo do estado* iww*, isto é,* $\Lambda(s_1)\langle iww, \iota\rangle = \Lambda(a)\langle iiw, \Delta^*(iww, s_1f_1f_2 + s_1f_2f_1)(\iota)\rangle$
- t *partindo do estado* wii*, isto é,* $\Lambda(t)\langle wii, \iota\rangle = \Lambda(a)\langle iiw, \Delta^*(wii, tf_1)(\iota)\rangle$
- s_2 *partindo do estado* wiw*, isto é,* $\Lambda(s_2)\langle wiw, \iota\rangle = \Lambda(a)\langle iiw, \Delta^*(wiw, s_2f_1f_2 + s_2f_2f_1)(\iota)\rangle$
- t *partindo do estado* wwi*, isto é,* $\Lambda(t)\langle wwi, \iota\rangle = \Lambda(a)\langle iiw, \Delta^*(wwi, tf_1f_2 + tf_2f_1)(\iota)\rangle$

Os detalhes destas e de todas as demais antecipações para a LR *são apresentados no Apêndice A.1. O resultado final destas antecipações é apresentado na Tabela 4.4. Na tabela, as fórmulas apresentadas resultam de simplificações realizadas manualmente.*

Estado	$\Lambda^{ctl}(s_1)$	$\Lambda^{ctl}(s_2)$	$\Lambda^{ctl}(t)$
iii	$a_1 < N$	$a_1 > 0 \wedge a_2 < M$	$a_2 > 0 \wedge a_1 < N$
iiw	$a_1 + 1 < N$	$a_1 > 0 \wedge a_2 < M$	—
iwi	$a_1 < N \wedge a_2 < M$	—	$a_2 > 0 \wedge a_1 < N$
iww	$a_1 + 1 < N \wedge a_2 < M$	—	—
wii	—	$a_1 > 0 \wedge a_2 < M$	$a_2 > 0 \wedge a_1 + 1 < N$
wiw	—	$a_1 > 0 \wedge a_1 < N \wedge a_2 < M$	—
wwi	—	—	$a_2 > 0 \wedge a_1 + 1 < N$
www	—	—	—

Tabela 4.4: Resultado do Algoritmo 4.3.1 para a linha realimentada.

4.3.3 Complexidade

Computacionalmente, pré-condições e operações simbólicas são implementados por árvores. Alguns exemplos são ilustrados na Figura 4.4. Na figura, as árvores ilustradas são as representações computacionais de (a) $m > 0$, (b) $n - 1$, (c) $m > 0 \wedge n < N$ e (d) $m > 0 \wedge n - 1 < N$.

O processamento representado por $\Lambda(\alpha)\langle q_\alpha, \iota_\alpha\rangle \wedge \Lambda(\sigma)\langle\delta(q_\alpha, \alpha), \Delta(q_\alpha, \alpha)(\iota_\alpha)\rangle$ consiste da construção de um novo nó, representando a operação $\wedge$, cuja sub-árvore da esquerda é uma cópia da árvore $\Lambda(\alpha)\langle q_\alpha, \iota_\alpha\rangle$ e cuja sub-árvore da direita é uma cópia da árvore $\Lambda(\sigma)\langle q_\sigma, \iota_\sigma\rangle$, com cada folha q_σ substituída por $\delta(q_\alpha, \alpha)$ e cada folha ι_σ substituída por uma cópia da árvore $\Delta(q_\alpha, \alpha)(\iota_\alpha)$. Claramente, o custo computacional da realização de uma única antecipação é alto e não pode ser ignorado na determinação da complexidade do Algoritmo 4.3.1. Este custo depende do número de componentes da coleção de dados, da complexidade das pré-condições e das operações associadas aos eventos. No

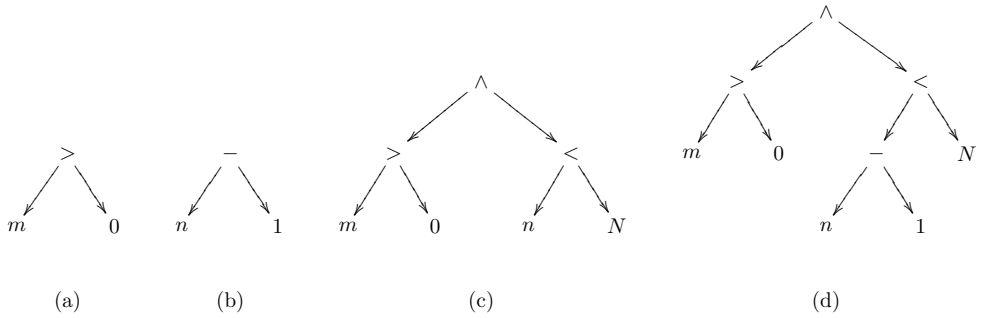

Figura 4.4: Implementação de fórmulas através de árvores.

que segue, apenas a complexidade das pré-condições será considerada e as seguintes suposições serão assumidas:

1. A coleção de dados contém apenas um componente, isto é, define apenas uma variável.

2. As pré-condições iniciais são da forma $n \# N$, onde n é a variável, $\#$ é um operador relacional ou de igualdade e N é uma constante.

3. Expressões do tipo $n \# N$ são denominadas *átomos*. Uma conjunção do tipo $n \#_1 N_1 \wedge n \#_2 N_2 \wedge \ldots \wedge n \#_k N_k$ é dita para ter k átomos.

Com estas suposições, o cálculo da complexidade desejada se resume ao cálculo do número de antecipações de átomos realizadas pelo Algoritmo 4.3.1.

Um sistema crítico com $n = 4$ estados discretos e $m = 4$ eventos (incluindo ξ), é ilustrado na Figura 4.5. Na figura, todos os eventos, a exceção de ξ, são não controláveis. Como o STE é determinístico e não existem laços fechados de eventos não controláveis, o tamanho máximo das subcadeias de eventos não controláveis é determinado pelo número de estados. Assim, o número máximo de iterações realizadas pelo comando **while** do algoritmo é n. O número máximo de transições não controláveis que podem existir neste caso é dado por $(n-1)\,(m-1)$ e o número de transições controláveis (ξ) é dado por n.

A primeira iteração do algoritmo manipula pré-condições consistindo de um único átomo. Nesta primeira iteração, as 3 pré-condições das transições com origem em q_0 são antecipadas para as 4 transições ξ. O total de antecipações de átomos assim realizadas é 12, ou num caso geral, $n\,(m-1)$. O resultado destas antecipações são pré-condições consistindo da conjunção de 3 átomos, ou no caso

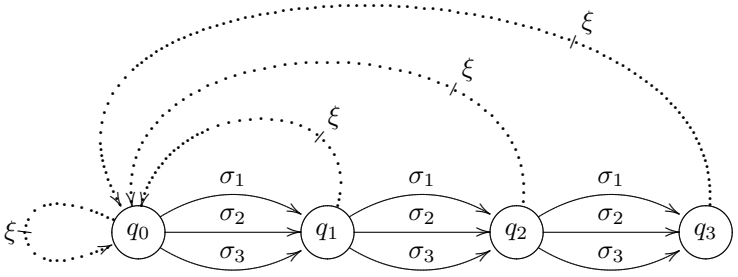

Figura 4.5: Um caso crítico.

geral, $(m-1)$ átomos [1].

Ainda na primeira iteração, as 3 pré-condições das transições com origem em q_1 são antecipadas para as 3 transições predecessoras cujo destino é q_1. O total de antecipações de átomos assim realizadas é 9, ou num caso geral, $(m-1)^2$. Como antes, o resultado destas antecipações são 3 pré-condições consistindo da conjunção de $(m-1)$ átomos. O mesmo pode ser dito sobre as antecipações dos átomos das pré-condições associadas às transições com origem em q_2. Assim o total de antecipações de átomos realizadas na primeira iteração do algoritmo é:

$$\underbrace{n\,(m-1)}_{q_0} + \underbrace{\underbrace{(m-1)^2}_{q_1} + \underbrace{(m-1)^2}_{q_2} + \ldots + \underbrace{(m-1)^2}_{q_{n-1}}}_{n-2} = n\,(m-1) + (n-2)\,(m-1)^2.$$

Ao final da primeira iteração, todas as pré-condições são da forma $a_1 \wedge a_2 \wedge \ldots \wedge a_{m-1}$, onde a_i denota um átomo, isto é, consistem de $(m-1)$ átomos. Além disso, o número de pré-condições diferentes de *true* foi reduzido de $(m-1)$.

A segunda iteração do algoritmo manipula pré-condições consistindo de 3 átomos. Nesta iteração, o total de antecipações de átomos das transições com origem em q_0 para as transições ξ é dado pela multiplicação do número de antecipações de pré-condições pelo número de átomos das pré-condições, ou seja, $n\,(m-1)\,(m-1) = n\,(m-1)^2$. O resultado destas antecipações são n pré-condições consistindo da conjunção de 9 átomos, ou no caso geral, $(m-1)^2$ átomos. Com relação às transições com origem no estado q_1, o número de antecipações de átomos realizadas é igual a $27 = 3^3$, ou, $(m-1)\,(m-1)^2$.

[1] Tudo isto, considerando-se que as antecipações são realizadas na ordem das transições mais próximas ao estado q_0 para as transições mais afastadas do estado q_0.

Já, com relação às transições com origem no estado q_1, nenhuma antecipação é realizada, pois após a primeira iteração, as pré-condições associadas a estas transições são todas iguais a *true*. O total de antecipações de átomos realizadas nesta segunda iteração é:

$$\underbrace{n\,(m-1)^2}_{q_0} + \underbrace{\underbrace{(m-1)\,(m-1)^2}_{q_1} + \underbrace{(m-1)\,(m-1)^2}_{q_2} + \ldots + \underbrace{(m-1)\,(m-1)^2}_{q_{n-2}}}_{n-3}$$

$$= n\,(m-1)^2 + (n-3)\,(m-1)^3.$$

Somando-se o número de antecipações de átomos das duas primeiras iterações, obtém-se:

$$\underbrace{n\,(m-1) + n\,(m-1)^2}_{\text{Número de antecipações sobre } q_0} \quad + \quad \underbrace{(n-2)\,(m-1)^2 + (n-3)\,(m-1)^3}_{\text{Número de antecipações sobre } q_1 \text{ e } q_2}.$$

Este raciocínio pode ser estendido para as demais iterações. Como o número máximo de iterações que efetivamente realizam antecipações é igual ao número de estados (n) menos 1, o número máximo de antecipações de átomos que se pode realizar num caso geral é dado por:

$$n \sum_{k=1}^{n-1} (m-1)^k \quad + \quad (m-1)^2 \sum_{i=1}^{n-2} \sum_{j=0}^{i-1} (m-1)^j =$$

$$\frac{n\,(m-1)^n - n - n\,(m-2)}{m-2} \quad + \quad \left(\frac{m-1}{m-2}\right)^2 \left[\,(m-1)^{n-1} - n\,(m-2) + m - 3\,\right] =$$

$$\frac{n}{m-2}\left(\sum_{k=0}^{n} \frac{n!}{k!\,(n-k)!}\,\mathbf{m^k}\,(-1)^{n-k}\right) - \frac{n}{m-2} - n \;+$$

$$\left(\frac{m-1}{m-2}\right)^2 \left[\left(\sum_{k=0}^{n-1} \frac{(n-1)!}{k!\,(n-1-k)!}\,m^k\,(-1)^{n-1-k}\right) - n\,(m-2) + m - 3\right]. \quad (4.7)$$

A Tabela 4.5 e a Figura 4.6 apresentam o número de antecipações de átomos realizadas para algumas combinações de n e m. Como se pode ver, a complexidade cresce polinomialmente com o número de transições (não controláveis) e exponencialmente com o número de estados. Assim, a complexidade pode ser expressa, na notação O [CLR99], como $O(m^n)$.

n - Número de estados	m - Número de eventos							
	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18
3	22	45	76	115	162	217	280	351
4	72	201	432	795	1320	2037	2976	4167
5	194	762	2132	4850	9606	17234	28712	45162
6	476	2700	9976	28280	67140	140476	267440	473256

Tabela 4.5: Número de antecipações de átomos realizadas para alguns valores de n e m.

4.4 Observações

A síntese de controladores consiste basicamente da transformação de uma especificação não controlável em uma especificação controlável. Uma especificação não controlável descreve pelo menos uma ação de controle que não pode ser implementada, uma vez que ela impõe restrições sobre a ocorrência de um evento não controlável. Para tornar uma ação não controlável $\Lambda(\beta)$ em uma ação "indiretamente" controlável, algumas questões devem ser respondidas.

1. Existe algum conjunto finito de ações controláveis $\{\Lambda(\alpha_i) \mid 1 \leq i \leq n \text{ e } \alpha_i \in \Sigma_c\}$ que implique $\Lambda(\beta)$, isto é, $\Lambda(\alpha_1) \sqcup \Lambda(\alpha_2) \sqcup \ldots \sqcup \Lambda(\alpha_n) \Rightarrow \Lambda(\beta)$?

2. Quais são os eventos α_i?

3. Quais são as restrições $\Lambda(\alpha_i)$?

Graças ao evento ξ, a resposta da primeira questão é sempre "sim". A resposta da segunda questão é "os eventos α_i são os eventos controláveis que podem ocorrer mais tarde e antes da ocorrência do evento β em alguma cadeia de $L(\mathbb{S})$". Finalmente, a última questão só pode ser respondida para a classe dos sistemas que não possuem laços fechados de eventos não controláveis. Esta restrição é grave no sentido de restringir fortemente a classe dos problemas tratáveis e, no futuro, soluções para sua eliminação devem ser pesquisadas.

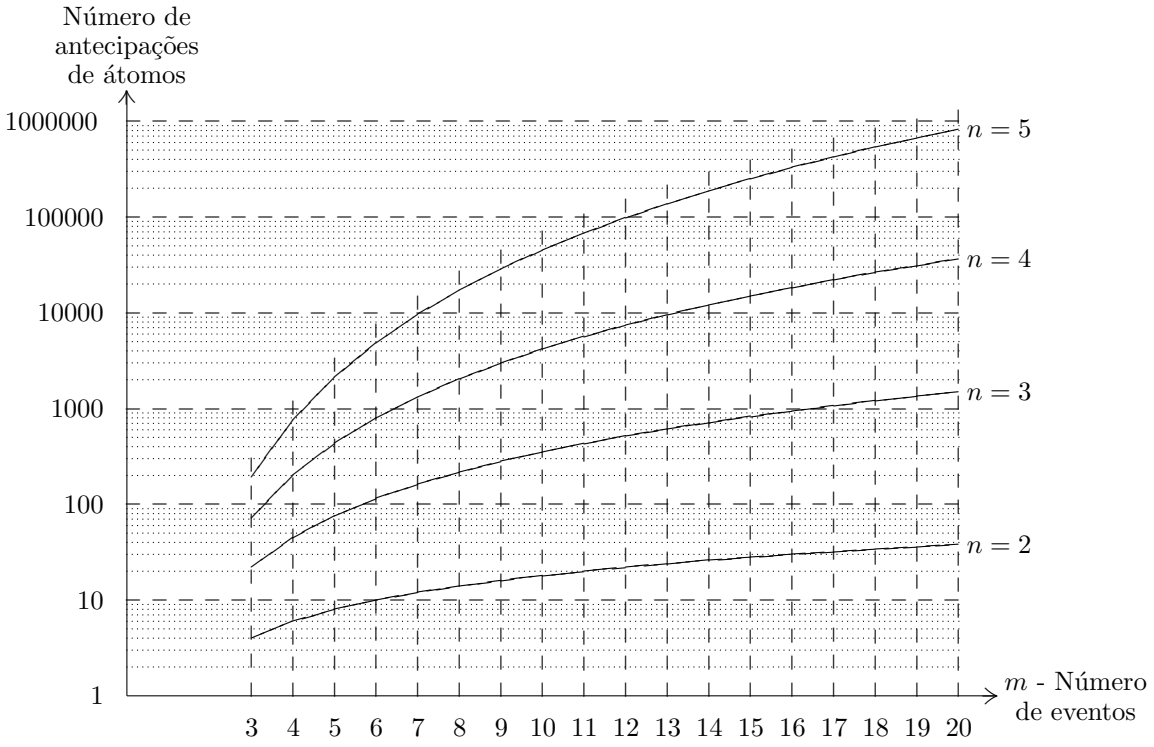

Figura 4.6: Gráfico das antecipações de átomos realizadas para alguns valores de n e m.

Capítulo 5

Supervisores não bloqueantes

Este capítulo resume um estudo relacionado com a síntese de supervisores não bloqueantes para STE-PC. Este problema não está completamente resolvido devido principalmente ao "problema de decisão" da lógica simbólica de primeira ordem (Entscheidungsproblem, em alemão [HA86]) o qual consiste da busca por um algoritmo geral que decida se uma sentença é universalmente válida ou não. Alonzo Church [Chu36] e, independentemente, Alan Turing [Tur36] mostraram que isto é impossível. Entretanto, quando o espaço de configurações é finito e enumerável, uma solução pode ser encontrada automaticamente. Para os demais casos, a metodologia descrita neste capítulo requer intervenções humanas para a resolução do problema de decisão comentado anteriormente e para a análise da acessibilidade e co-acessibilidade dos sistemas.

Nos STE-PC, a marcação representa um objetivo a ser atingido ou uma tarefa a ser realizada. Este objetivo é descrito pelo predicado marcador $\mathcal{P}_m : Q \times \mathbf{I}_n \rightarrow \mathbb{B}$, o qual define um subconjunto de configurações marcadas. Se o predicado marcador é falso para todas as configurações do sistema, assume-se que o sistema deva evoluir continuamente. Neste caso, se o sistema não puder mais evoluir a partir de uma configuração qualquer, diz-se que o sistema é bloqueante. Se, por outro lado, o predicado marcador define um subconjunto não vazio de configurações marcadas, o sistema pode ou não evoluir até que uma nova configuração marcada seja alcançada. Após alcançar uma configuração marcada e dependendo da especificação dos requisitos o sistema pode parar ou evoluir em direção à realização de uma outra tarefa. Nestes casos, se o sistema não puder mais evoluir a partir de uma configuração não marcada, ou se o sistema continuar a evoluir continuamente sem nunca alcançar uma configuração marcada, diz-se também que o sistema é bloqueante.

Se o objetivo de um sistema é parar após a realização de alguma tarefa, nenhum evento deve

ser habilitado após a realização da tarefa. Portanto, no projeto dos predicados marcadores deve-se observar que, com relação aos estados discretos q das configurações marcadas, os eventos fisicamente possíveis a partir de q (se houver algum) devem ser todos controláveis, isto é, $\Sigma(q) \subseteq \Sigma_c$. Com relação às instanciações ι das configurações marcadas, nenhum cuidado particular deve ser levado em consideração, a não ser é claro, a possibilidade física da realização das tarefas correspondentes.

É importante observar que este conceito de marcação não é comum na área dos SED. Na teoria clássica, por exemplo, quando o objetivo da supervisão não é especificado através de linguagens marcadas, todos os estados do sistema são considerados marcados. Como o problema da síntese de supervisores não bloqueantes para STExCD não está completamente resolvido, no futuro o conceito de marcação utilizado nesta seção poderá ser revisto.

Na literatura [AW03] são tratados dois tipos de bloqueio, os quais impedem o sistema de evoluir ou completar suas tarefas. O deadlock ocorre quando o sistema alcança um estado não marcado a partir do qual ele não pode continuar com suas atividades e o livelock (Figura 5.1) ocorre quando o sistema alcança um ciclo infinito que impede a terminação de uma tarefa. Neste trabalho, somente bloqueios causados por deadlocks serão tratados [1].

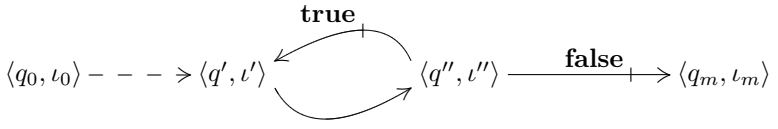

Figura 5.1: Livelock.

5.1 Supervisores livre de deadlocks

O projeto de uma especificação livre de deadlocks, geralmente, não é uma tarefa trivial. Entretanto, a partir da análise do comportamento em malha fechada $L(V_\Lambda/\mathbb{S})$, os deadlocks podem ser teoricamente detectados e, a partir deles, uma nova especificação para evitar aqueles deadlocks pode ser derivada. Isto sugere um método prático para se encontrar um supervisor livre de deadlocks. A parte de qualquer método prático, do ponto de vista formal, devem ser mostradas as condições a serem verificadas para garantir a existência de um supervisor livre de deadlocks.

[1] Há de se observar que se os sistemas estão livres de livelock, como considerado, e livres de deadlocks, os estados marcados, se existirem, são sempre acessíveis.

Definição 5.1.1 (Especificações livres de deadlocks). *Seja* $\mathbb{S} = \langle S, \mathbf{M}_n, \Delta, \mathcal{P}_m \rangle$ *um STExCD não-bloqueante e* Λ *uma especificação por pré-condições para* $\mathbb{S}$*. Diz-se que* Λ *está* livre de deadlocks *com relação ao predicado marcador* $\mathcal{P}_m$ *se toda configuração acessível não marcada satisfazendo* Λ *é seguida por alguma outra configuração, a qual também satisfaz* Λ*. Em termos formais,* Λ *está livre de deadlocks se:*

$$\forall \langle q, \iota \rangle \in \mathbb{D}_\Lambda, \quad \neg \mathcal{P}_m \langle q, \iota \rangle \Rightarrow \left(\bigvee_{\sigma \in \Sigma(q)} \Lambda(\sigma) \langle q, \iota \rangle \right). \tag{5.1}$$

Assim, se o predicado marcador é falso para todas as configurações satisfazendo Λ, então o lado direito da implicação deve ser sempre verdade, isto é, para toda configuração $\langle q, \iota \rangle \in \mathbb{D}_\Lambda$, a pré-condição associada a pelo menos um evento $\sigma \in \Sigma(q)$ deve ser verdade. Se o predicado marcador é verdade para uma configuração $\langle q, \iota \rangle \in \mathbb{D}_\Lambda$, então, dependendo da especificação Λ, o lado da direita da implicação pode ou não ser verdade, isto é, pode existir ou não uma próxima configuração $\langle \delta(q, \sigma), \Delta(q, \sigma)(\iota) \rangle$, a qual também satisfaz Λ.

As configurações que causam deadlocks, isto é, as configurações que não satisfazem a Implicação 5.1, satisfazem:

$$\neg \mathcal{P}_m \langle q, \iota \rangle \wedge \left(\bigwedge_{\sigma \in \Sigma(q)} \neg \Lambda(\sigma) \langle q, \iota \rangle \right). \tag{5.2}$$

Evidentemente, as configurações que não causam deadlocks satisfazem a Fórmula 5.2 negada, isto é:

$$\neg \left(\neg \mathcal{P}_m \langle q, \iota \rangle \wedge \left(\bigwedge_{\sigma \in \Sigma(q)} \neg \Lambda(\sigma) \langle q, \iota \rangle \right) \right). \tag{5.3}$$

Exemplo 5.1.1 (Deadlocks da linha realimentada). *A especificação controlável evitando os overflows e underflows na linha realimentada foi descrita no Exemplo 4.2.1. Como comentado naquele exemplo, a especificação controlável* Λ^{ctl} *(Tabela 4.2) apresenta configurações que causam deadlocks. Isto pode ser comprovado, aplicando-se a Equação 5.1 à configuração* $\langle iii, \langle N, M \rangle \rangle$*,*

$$\underbrace{\langle iii, \langle N, M\rangle\rangle \in \mathbb{D}_\Lambda,}_{true} \quad \underbrace{\neg \mathcal{P}_m \langle iii, \langle N, M\rangle\rangle}_{true} \Rightarrow \left(\bigvee_{\sigma \in \{s_1, s_2, t\}} \Lambda(\sigma)\langle iii, \langle N, M\rangle\rangle \right).$$

Desenvolvendo-se a disjunção generalizada,

$$\Lambda(s_1)\langle iii, \langle N, M\rangle\rangle \vee \Lambda(s_2)\langle iii, \langle N, M\rangle\rangle \vee \Lambda(t)\langle iii, \langle N, M\rangle\rangle =$$
$$\underbrace{(a_1 < N)}_{false} \vee (a_1 > 0 \wedge \underbrace{a_2 < M)}_{false} \vee (a_2 > 0 \wedge \underbrace{a_1 < N)}_{false} = false.$$

Portanto, para a configuração $\langle iii, \langle N, M\rangle\rangle$ a fórmula da Equação 5.1 é falsa e, conseqüentemente, o sistema não está livre de deadlocks.

Neste exemplo, o conjunto de configurações causando deadlocks *é determinado de acordo com a Fórmula 5.2. As configurações que podem causar* deadlocks *são aquelas associadas ao estado iii ilustrado na Figura 5.2.*

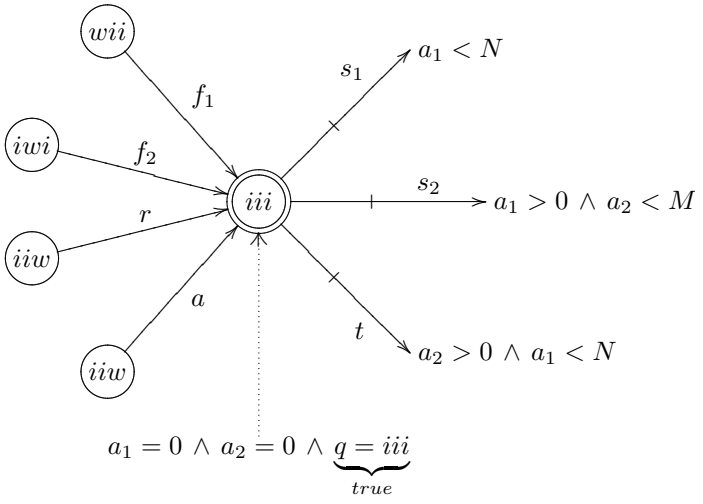

Figura 5.2: Estado iii.

De acordo com a Fórmula 5.2, as configurações causando deadlocks *satisfazem* $\neg \mathcal{P}_m\langle iii, \iota\rangle \wedge$

$\neg\Lambda^{ctl}(s_1)\langle iii, \iota\rangle \wedge \neg\Lambda^{ctl}(s_2)\langle iii, \iota\rangle \wedge \neg\Lambda^{ctl}(t)\langle iii, \iota\rangle$, *isto é, para* $q = iii$,

$$\neg(a_1 = 0 \wedge a_2 = 0) \wedge \neg(a_1 < N) \wedge \neg(a_1 > 0 \wedge a_2 < M) \wedge \neg(a_2 > 0 \wedge a_1 < N) = \\ (a_1 \neq 0 \vee a_2 \neq 0) \wedge (a_1 \geq N) \wedge (a_1 \leq 0 \vee a_2 \geq M) \wedge (a_2 \leq 0 \vee a_1 \geq N).$$

De acordo com a propriedade distributiva da conjunção, pode-se reescrever a fórmula anterior como:

$$(a_1 \neq 0 \wedge \underbrace{a_1 \geq N \wedge a_1 \leq 0}_{false} \wedge a_2 \leq 0) \vee (a_1 \neq 0 \wedge \underbrace{a_1 \geq N \wedge a_1 \leq 0}_{false} \wedge a_1 \geq N) \vee \\ (a_1 \neq 0 \wedge a_1 \geq N \wedge \underbrace{a_2 \geq M \wedge a_2 \leq 0}_{false}) \vee (a_1 \neq 0 \wedge a_1 \geq N \wedge a_2 \geq M \wedge a_1 \geq N) \vee \\ (a_2 \neq 0 \wedge \underbrace{a_1 \geq N \wedge a_1 \leq 0}_{false} \wedge a_2 \leq 0) \vee (a_2 \neq 0 \wedge \underbrace{a_1 \geq N \wedge a_1 \leq 0}_{false} \wedge a_1 \geq N) \vee \\ (a_2 \neq 0 \wedge a_1 \geq N \wedge \underbrace{a_2 \geq M \wedge a_2 \leq 0}_{false}) \vee (a_2 \neq 0 \wedge a_1 \geq N \wedge a_2 \geq M \wedge a_1 \geq N).$$

Após as simplificações sugeridas, a fórmula identificando as configurações causando **deadlocks** *é reduzida para:*

$$(a_1 \neq 0 \wedge a_1 \geq N \wedge a_2 \geq M \wedge a_1 \geq N) \vee (a_2 \neq 0 \wedge a_1 \geq N \wedge a_2 \geq M \wedge a_1 \geq N) \\ = (a_1 \geq N \wedge a_2 \geq M). \quad (5.4)$$

Exemplo 5.1.2 (Eliminação dos deadlocks – LR). *As configurações causando* **deadlocks** *na linha realimentada sob supervisão podem ser evitadas. Para este fim, acrescenta-se aos requisitos do sistema uma especificação por pré-condições que evita as configurações causando* **deadlocks**. *Esta especificação, a qual será denotada por* Λ_{nb}, *pode ser computada a partir da especificação por invariantes dada pela Fórmula 5.3. No caso da linha realimentada, esta especificação por invariantes consiste da fórmula:*

$$Inv_{nb}\langle iii, \langle a_1, a_2\rangle\rangle = \neg(a_1 \geq N \wedge a_2 \geq M) = (a_1 < N \vee a_2 < M),$$

o qual deve ser verificado no estado iii.

Como visto na Seção 3.2.3, a especificação por pré-condições equivalente ao invariante anterior consiste das seguintes pré-condições:

$$Inv_{nb}\langle iii, \langle a_1, a_2\rangle\rangle = \begin{cases} \Lambda_{nb}(f_1)\langle wii, \langle a_1, a_2\rangle\rangle = (a_1 + 1 < N \ \vee \ a_2 < M) \\ \Lambda_{nb}(f_2)\langle iwi, \langle a_1, a_2\rangle\rangle = (a_1 < N \ \vee \ a_2 + 1 < M) \\ \Lambda_{nb}(r)\langle iiw, \langle a_1, a_2\rangle\rangle = (a_1 + 1 < N \ \vee \ a_2 < M) \\ \Lambda_{nb}(a)\langle iiw, \langle a_1, a_2\rangle\rangle = (a_1 < N \ \vee \ a_2 < M). \end{cases}$$

Antecipando-se as pré-condições anteriores para os eventos controláveis precedentes, obtém-se a especificação controlável Λ_{nb}^{ctl} consistindo das seguintes pré-condições:

$$\begin{cases} \Lambda_{nb}^{ctl}(s_1)\langle iii, \langle a_1, a_2\rangle\rangle = \underbrace{(a_1 + 1 < N \ \vee \ a_2 < M)}_{\text{veio de } \Lambda_{nb}(f_1)\langle wii, a_1, a_2\rangle} \\ \Lambda_{nb}^{ctl}(s_2)\langle iii, \langle a_1, a_2\rangle\rangle = \underbrace{(a_1 - 1 < N \ \vee \ a_2 + 1 < M)}_{\text{veio de } \Lambda_{nb}(f_2)\langle iwi, a_1, a_2\rangle} \\ \Lambda_{nb}^{ctl}(t)\langle iii, \langle a_1, a_2\rangle\rangle = \underbrace{(a_1 + 1 < N \ \vee \ a_2 - 1 < M)}_{\text{veio de } \Lambda_{nb}(r)\langle iiw, a_1, a_2\rangle} \wedge \underbrace{(a_1 < N \ \vee \ a_2 - 1 < M)}_{\text{veio de } \Lambda_{nb}(a)\langle iiw, a_1, a_2\rangle}. \end{cases}$$

Considerando-se o contexto do sistema supervisionado, isto é, as configurações acessíveis sob supervisão, sabe-se que $a_1 \leq N$ e $a_2 \leq M$. Portanto, as restrições de Λ_{nb}^{ctl} sobre a ocorrência dos eventos s_2 e t podem ser simplificadas para true. Procedendo-se assim, a especificação Λ_{nb}^{ctl} se resume à restrição $(a_1 + 1 < N \ \vee \ a_2 < M)$ imposta sobre a ocorrência do evento s_1 a partir do estado iii.

Compondo-se conjuntivamente a especificação controlável Λ^{ctl}, evitando os overflows e underflows nos armazéns (Exemplo 4.2.1), com a especificação Λ_{nb}^{ctl}, evitando as configurações que causam deadlocks, obtêm-se as pré-condições da Tabela 5.1. A restrição sobre s_1 no estado iii foi obtida da

conjunção:

$$\overbrace{(a_1 < N)}^{\Lambda^{ctl}(s_1)} \wedge \overbrace{(a_1 + 1 < N \vee a_2 < M)}^{\Lambda^{ctl}_{nb}(s_1)} =$$

$$(\underbrace{a_1 < N \wedge a_1 + 1 < N}_{a_1+1<N} \vee a_1 < N \wedge a_2 < M) =$$

$$\underbrace{(a_1 + 1 < N \vee a_1 < N \wedge a_2 < M)}_{\text{restrição final}}.$$

	$\Lambda^{ctl}_{nb}(s_1)$	$\Lambda^{ctl}_{nb}(s_2)$	$\Lambda^{ctl}_{nb}(t)$
iii	$a_1 + 1 < N \vee a_1 < N \wedge a_2 < M$	$a_1 > 0 \wedge a_2 < M$	$a_1 < N \wedge a_2 > 0$
iiw	$a_1 + 1 < N$	$a_1 > 0 \wedge a_2 < M$	—
iwi	$a_1 < N \wedge a_2 < M$	—	$a_2 > 0 \wedge a_1 < N$
iww	$a_1 + 1 < N \wedge a_2 < M$	—	—
wii	—	$a_1 > 0 \wedge a_2 < M$	$a_2 > 0 \wedge a_1 + 1 < N$
wiw	—	$a_1 > 0 \wedge a_1 < N \wedge a_2 < M$	—
wwi	—	—	$a_2 > 0 \wedge a_1 + 1 < N$
www	—	—	—

Tabela 5.1: Especificação controlável e livre de deadlocks para a linha realimentada.

Após a eliminação de um conjunto de configurações causando deadlocks, um novo conjunto de pré-condições é obtido. Este novo conjunto de pré-condições pode implicar um novo conjunto de configurações causando deadlocks. Deste modo, a computação da fórmula definindo as configurações que causam deadlocks deve ser refeita até que uma das fórmulas seja logicamente equivalente à fórmula anterior. No caso do exemplo anterior, as novas configurações bloqueantes satisfazem:

$$\neg(a_1 = 0 \wedge a_2 = 0) \wedge \neg(a_1 + 1 < N \vee a_1 < N \wedge a_2 < M) \wedge$$
$$\neg(a_1 > 0 \wedge a_2 < M) \wedge \neg(a_1 < N \wedge a_2 > 0),$$

ou seja,

$$(a_1 \neq 0 \ \vee \ a_2 \neq 0) \ \wedge \ (\overbrace{a_1 + 1 \geq N \ \vee \ a_1 \geq N}^{a_1 \geq N} \vee a_2 \geq M) \wedge$$
$$(a_1 \leq 0 \ \vee \ a_2 \geq M) \ \wedge \ (a_1 \geq N \ \vee \ a_2 \leq 0)$$
$$= (a_1 \neq 0 \ \vee \ a_2 \neq 0) \ \wedge \ (a_1 \geq N \ \vee \ a_2 \geq M) \wedge$$
$$(a_1 \leq 0 \ \vee \ a_2 \geq M) \ \wedge \ (a_1 \geq N \ \vee \ a_2 \leq 0).$$

Distribuindo-se a fórmula anterior em relação às conjunções, tem-se:

$$\overbrace{(a_1 \neq 0 \ \wedge \ \underline{a_1 \geq N} \ \wedge \ \underline{a_1 \leq 0} \ \wedge \ a_1 \geq N)}^{false} \ \vee \ \overbrace{(a_1 \neq 0 \ \wedge \ \underline{a_1 \geq N} \ \wedge \ \underline{a_1 \leq 0} \ \wedge \ a_2 \leq 0)}^{false} \vee$$
$$\overbrace{(a_1 \neq 0 \ \wedge \ a_1 \geq N \ \wedge \ a_2 \geq M \ \wedge \ a_1 \geq N)}^{a_1 \geq N \wedge a_2 \geq M} \ \vee \ \overbrace{(a_1 \neq 0 \ \wedge \ a_1 \geq N \ \wedge \ \underline{a_2 \geq M} \ \wedge \ \underline{a_2 \leq 0})}^{false} \vee$$
$$\overbrace{(a_1 \neq 0 \ \wedge \ a_2 \geq M \ \wedge \ \underline{a_1 \leq 0} \ \wedge \ \underline{a_1 \geq N})}^{false} \ \vee \ \overbrace{(a_1 \neq 0 \ \wedge \ \underline{a_2 \geq M} \ \wedge \ a_1 \leq 0 \ \wedge \ \underline{a_2 \leq 0})}^{false} \vee$$
$$\overbrace{(a_1 \neq 0 \ \wedge \ a_2 \geq M \ \wedge \ a_2 \geq M \ \wedge \ a_1 \geq N)}^{a_2 \geq M \wedge a_1 \geq N} \ \vee \ \overbrace{(a_1 \neq 0 \ \wedge \ a_2 \geq M \ \wedge \ \underline{a_2 \geq M} \ \wedge \ \underline{a_2 \leq 0})}^{false} \vee$$
$$\overbrace{(a_2 \neq 0 \ \wedge \ \underline{a_1 \geq N} \ \wedge \ \underline{a_1 \leq 0} \ \wedge \ a_1 \geq N)}^{false} \ \vee \ \overbrace{(a_2 \neq 0 \ \wedge \ \underline{a_1 \geq N} \ \wedge \ \underline{a_1 \leq 0} \ \wedge \ a_2 \leq 0)}^{false} \vee$$
$$\overbrace{(a_2 \neq 0 \ \wedge \ a_1 \geq N \ \wedge \ a_2 \geq M \ \wedge \ a_1 \geq N)}^{a_1 \geq N \wedge a_2 \geq M} \ \vee \ \overbrace{(a_2 \neq 0 \ \wedge \ a_1 \geq N \ \wedge \ \underline{a_2 \geq M} \ \wedge \ \underline{a_2 \leq 0})}^{false} \vee$$
$$\overbrace{(a_2 \neq 0 \ \wedge \ a_2 \geq M \ \wedge \ \underline{a_1 \leq 0} \ \wedge \ \underline{a_1 \geq N})}^{false} \ \vee \ \overbrace{(a_2 \neq 0 \ \wedge \ \underline{a_2 \geq M} \ \wedge \ a_1 \leq 0 \ \wedge \ \underline{a_2 \leq 0})}^{false} \vee$$
$$\overbrace{(a_2 \neq 0 \ \wedge \ a_2 \geq M \ \wedge \ a_2 \geq M \ \wedge \ a_1 \geq N)}^{a_2 \geq M \wedge a_1 \geq N} \ \vee \ \overbrace{(a_2 \neq 0 \ \wedge \ a_2 \geq M \ \wedge \ \underline{a_2 \geq M} \ \wedge \ \underline{a_2 \leq 0})}^{false}.$$

Finalmente, após as simplificações sugeridas, a fórmula identificando as configurações causando deadlocks fica:

$$a_1 \geq N \wedge a_2 \geq M,$$

a qual é igual (e, portanto, logicamente equivalente) à Fórmula 5.4. Conseqüentemente, a especificação descrita na Tabela 5.1 está livre de deadlocks.

5.2 Passos para a eliminação de bloqueios

A fim de formalizar a metodologia para a detecção e eliminação de deadlocks descrita na seção anterior, é apresentado a seguir, o procedimento "Livre de deadlocks".

Proposição 5.2.1 (Especificação controlável máxima livre de deadlocks). *A especificação Λ^{ctl} resultante do Procedimento 5.2.1 é a especificação controlável minimamente restritiva, livre de deadlocks, tal que $\Lambda^{\uparrow} = \Lambda^{ctl} \Rightarrow \Lambda^{org}$.*

Demonstração. *Supondo-se a existência de um supervisor livre de deadlocks, uma descrição formal dos efeitos do Procedimento 5.2.1 é*

$$\Lambda^{\uparrow} = (\ldots((\Lambda^{ctr} \sqcap_1 \Lambda^{ctr}_{nb}) \sqcap_2 \Lambda^{ctr}_{nb}) \ldots \sqcap_k \Lambda^{ctr}_{nb}),$$

onde $\sqcap_i$ é a i-ésima aplicação do operador meet. Como meet sempre produz uma especificação controlável, $\Lambda^{\uparrow}$ é controlável.

Como todas as especificações locais Λ^{ctr} e Λ^{ctr}_{nb} são produzidas pelo Algoritmo 4.3.1, as conjunções $\cdots \sqcap_i \Lambda^{ctr}_{nb}$ são minimamente restritivas. Portanto, $\Lambda^{\uparrow}$ é minimamente restritiva.

Agora, como $\Lambda^{\uparrow}$ está livre de deadlocks e, supostamente, livre de livelocks, a partir de uma configuração não marcada, o sistema sempre pode evoluir.

■

5.3 Observações

O Procedimento 5.2.1 apresenta os seguintes problemas:

Procedimento 5.2.1 Eliminação de deadlocks.

Dado o STE-PC $V_{\Lambda^{org}}/\mathbb{S}$, onde $\mathbb{S} = \langle G, \mathbf{M}_n, \Delta, \mathcal{P}_m \rangle$ é um STExCD e Λ^{org} é a especificação original, possivelmente não controlável, os passos a seguir realizam as antecipações das pré-condições não controláveis para os respectivos eventos controláveis, detectam e eliminam bloqueios causados por deadlocks. A especificação Λ^{ctl} resultante é controlável e livre de deadlocks. Inv_1 e Inv_2 são especificações locais por invariantes e Λ_{nb} e Λ_{nb}^{ctl} são especificações locais por pré-condições.

Converter Λ^{org} para Λ^{ctl} através do Algoritmo 4.3.1

Calcular a especificação por invariantes Inv_2 evitando os deadlocks em $V_{\Lambda^{ctl}}/\mathbb{S}$

Repita

$Inv_1 \leftarrow Inv_2$

Converter a especificação Inv_2 anterior na especificação por pré-condições Λ_{nb}

Converter Λ_{nb} para Λ_{nb}^{ctl} através do Algoritmo 4.3.1

$\Lambda^{ctl} \leftarrow \Lambda^{ctl} \sqcap \Lambda_{nb}^{ctl}$

Calcular a especificação por invariantes Inv_2 evitando os deadlocks em $V_{\Lambda^{ctl}}/\mathbb{S}$

Até que Inv_1 seja logicamente equivalente a Inv_2

A solução é Λ^{ctl}

1. *Complexidade computacional decorrente da não simplificação automática de fórmulas.* Mesmo existindo regras sintáticas bem definidas pela lógica de primeira ordem e pela aritmética para a simplificação de fórmulas, a aplicação destas regras muitas vezes requer o conhecimento do domínio acessível das variáveis usadas. Tais domínios são definidos pela acessibilidade dos sistemas realimentados, a qual, em geral, não é computável.

2. *Problema da decisão da lógica de primeira ordem.* Este problema aparece na verificação da equivalência entre Inv_1 e Inv_2.

3. *Convergência.* No caso de domínios infinitos de instanciações das variáveis, se não existir um supervisor livre de deadlocks, o procedimento não converge.

4. O Procedimento 5.2.1 não garante a realização das tarefas. Devido à impossibilidade da com-

putação de $Reach^{QI}(V_\Lambda/\mathbb{S})$ e $CoReach^{QI}(V_\Lambda/\mathbb{S})$, o sistema supervisionado resultante pode estar livre de deadlocks (e livelocks) e, mesmo assim, pode evoluir continuamente sem nunca alcançar uma configuração marcada.

O Procedimento 5.2.1 é decidível e converge nos casos em que todas as pré-condições consistem de proposições sobre conjuntos de instanciações discretos, tais como as especificações da Seção 3.3 [2]. Para estes casos, entretanto, a solução produzida (a qual pode ser reduzida automaticamente para um conjunto de pré-condições *true* e *false*) é equivalente à solução produzida pela teoria clássica e, portanto, não oferece vantagens significativas em relação a esta última. Conclui-se assim, que o método para a síntese de supervisores não bloqueantes descrito anteriormente é mais um procedimento a ser realizado manualmente pelos engenheiros de controle do que o esboço de um possível algoritmo de síntese.

[2] Ver também os Experimentos 6.1 e 6.2.

Capítulo 6

Experimentação

Para a experimentação dos tópicos explorados ao longo deste texto, serão usados alguns exemplos, cada qual com uma característica própria. Embora todos os exemplos sejam simples, eles foram propositadamente escolhidos para explorarem a maioria dos detalhes, potencialidades e deficiências dos métodos introduzidos anteriormente. O exemplo "mesa giratória" (Seção 6.1) será usado principalmente para exemplificar como as especificações de comportamentos desejáveis são (ou podem ser) construídas. O exemplo "o gato e o rato" (Seção 6.2) é particularmente interessante na exploração dos tópicos relacionados com a eliminação dos bloqueios. A planta "máquinas com vários modos de operação" (Seção 6.3) explora mais detalhadamente os algoritmos para a síntese de supervisores. Neste último experimento, também podem ser observados os problemas decorrentes da não simplificação de fórmulas.

Todos os experimentos foram resolvidos de acordo com o Algoritmo 4.3.1. No entanto, a fim de permitir uma maior clareza dos resultados, a apresentação dos resultados obtidos foi alterada através de simplificações manuais. A falta de clareza dos resultados produzidos pelo protótipo decorre da "impossibilidade" de simplificações automáticas de fórmulas lógicas.

6.1 Mesa giratória

Este exemplo apareceu a primeira vez em [dQSC01]. Na linha de produção de uma indústria automobilística, existe uma mesa giratória com quatro slots onde são efetuadas operações de furo e teste de peças metálicas (Figura 6.1). Os componentes identificados na mesa giratória são a esteira E, a furadeira F, o testador T, o manipulador robótico R e a mesa giratória M.

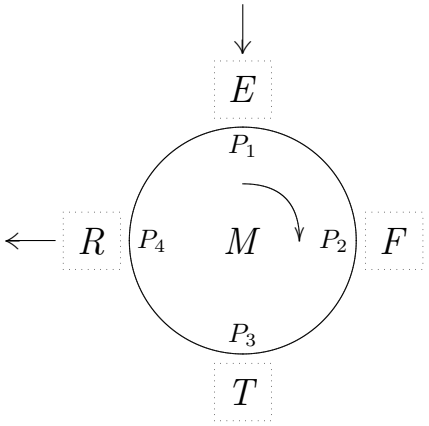

Figura 6.1: Mesa giratória.

Na mesa giratória, o processamento de uma única peça consiste da seguinte seqüência de passos:

1. A esteira gira até que uma peça seja posicionada em P_1.
2. A mesa gira 90°.
3. A peça no slot P_2 é furada.
4. A mesa gira 90°.
5. A peça no slot P_3 é testada.
6. A mesa gira 90°.
7. O manipulador robótico retira a peça do slot P_4.

Supõem-se que não existem sensores indicando a presença de peças nos slots da mesa. Os eventos diponíveis são descritos na Tabela 6.1.

Considerando-se o processamento concorrente de até quatro peças, o primeiro objetivo deste exemplo é a definição de uma lógica de controle que restrinja o comportamento da planta àqueles satisfazendo:

- a mesa não deve girar sem ao menos uma peça na mesa ou enquanto houver uma peça no slot P_4;

Equipamento	Evento	Descrição
Esteira	e_s	Associado ao início do depósito de uma peça no slot P_1.
	e_f	Associado ao final da operação da esteira automática. (Não controlável.)
Mesa giratória	m_s	Associado ao início de um giro de 90° da mesa.
	m_f	Associado ao final da operação da mesa giratória. (Não controlável.)
Furadeira	f_s	Associado ao início da furação da peça que estiver no slot P_2.
	f_f	Associado ao final da operação da furadeira automática. (Não controlável.)
Testador	t_s	Associado ao início do teste de peça que estiver no slot P_3.
	t_f	Associado ao final da operação do teste automático. (Não controlável.)
Manipulador Robótico	r_s	Associado ao início da retirada de uma peça no slot P_4.
	r_f	Associado ao final da operação do manipulador robótico. (Não controlável.)

Tabela 6.1: Eventos da mesa giratória.

- a mesa não deve girar enquanto a esteira, a furadeira, o testador ou o manipulador estiverem em operação;
- a mesa não deve girar se no slot P_2 houver uma peça não furada ou se no slot P_3 houver uma peça não testada;
- a esteira não deve depositar uma peça em P_1 se neste slot já existir uma peça;
- a esteira, a furadeira, o testador e o manipulador não devem operar enquanto a mesa estiver girando;
- a furadeira, o testador e o manipulador não devem operar sem uma peça nos slots P_2, P_3 e P_4, respectivamente; e
- uma peça deve ser furada e testada (exatamente, uma única vez).

6.1.1 Modelo da planta

Os sistemas de transição de estados da esteira, da mesa giratória, da furadeira, do testador e do manipulador robótico são ilustrados na Figura 6.2. A composição síncrona entre estes subsistemas é um sistema de transição com 32 estados da forma $\langle f, e, m, r, t \rangle$, onde f, e, m, r e $t \in \{i, w\}$ denotam, respectivamente, os estados locais da furadeira, da esteira, da mesa, do manipulador robótico e do testador [1]. O estado inicial da planta é $\langle i, i, i, i, i \rangle$ e o predicado marcador é falso para todas as configurações.

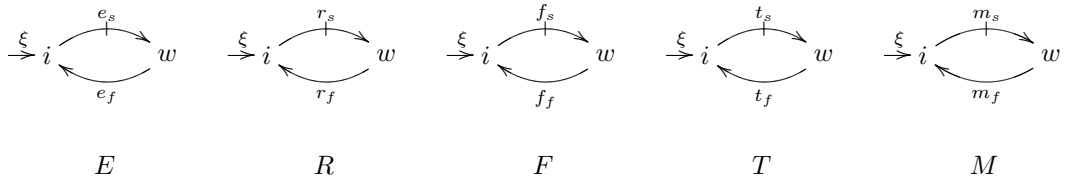

Figura 6.2: Componentes da mesa giratória.

A coleção de dados para a mesa giratória consiste dos monóides de operações $\mathbf{M}_3 = \langle X, Y, P \rangle$, onde:

1. $X = M_{\mathbb{B}}$ é um monóide booleano. A instanciação $true \in \mathbb{B}$ indica que a peça no slot P_2 foi furada.

2. $Y = M_{\mathbb{B}}$ é um monóide booleano. A instanciação $true \in \mathbb{B}$ indica que a peça no slot P_3 foi testada.

   As operações de interesse em X e Y são *set* e *reset*, definidas como:

$$\forall\, x \in \mathbb{B}, \begin{cases} set(x) = true & \text{marca a peça no slot } P_2 \text{ como furada} \\ reset(x) = false & \text{desmarca a peça no slot } P_2 \text{ como furada} \end{cases}$$

   e

$$\forall\, y \in \mathbb{B}, \begin{cases} set(y) = true & \text{marca a peça no slot } P_3 \text{ como testada} \\ reset(y) = false & \text{desmarca a peça no slot } P_3 \text{ como testada.} \end{cases}$$

[1] A composição síncrona não é ilustrada devido à sua complexidade.

3. $P = M_{\mathbb{Z}}$ é um monóide de operações sobre o conjunto dos inteiros. Representados pelas suas notações binárias, cada bit das instanciações de P reflete a ocupação de um slot da mesa. O bit menos significativo de uma instanciação reflete a ocupação do slot P_1, o segundo e o terceiro bits refletem a ocupação dos slots P_2 e P_3, respectivamente, e o quarto bit, o mais significativo dos quatro, reflete a ocupação do slot P_4. Apenas estes quatro bits são usados. Assim, a instanciação 1010_2 indica que os slots P_4 e P_2 estão ocupados e os demais, livres [2]. As operações de interesse em P são *rotate*, *insert* e *remove* definidas como:

$$\forall\, p \in \mathbb{Z},\ \begin{cases} insert(p) = p + 1 & \text{inserção de uma peça no slot } P_1 \\ rotate(p) = 2 * p & \text{rotação de } 90^\circ \text{ da mesa} \\ remove(p) = p - 8 & \text{remoção de uma peça do slot } P_4. \end{cases}$$

A multiplicação $2 * p$ produz um deslocamento de bits à esquerda e reflete um giro de 90° da mesa. A adição $p + 1$ indica que a posição P_1 foi ocupada por uma peça e a subtração $p - 8$, indica a retirada de uma peça da posição P_4. Nestes últimos dois casos, condições anormais de overflow em P_1 e underflow em P_4, podem ocorrer. Tais condições devem ser evitadas como indicam os requisitos descritos anteriormente.

Uma instanciação genérica da coleção de dados $\mathbf{M}_3$ é denotada $\langle x, y, p\rangle \in \mathbb{B} \times \mathbb{B} \times \mathbb{Z}$, onde x, y e p são, respectivamente, instanciações de X, Y e P. A associação de operações aos eventos é:

1. $\forall\, \langle f, e, m, r, t\rangle \in Q, \quad \Delta(\langle f, e, m, r, t\rangle, \xi) = \langle init_X, init_Y, init_P\rangle$,
2. $\forall\, \langle f, e, m, r, t\rangle \in Q, \quad \Delta(\langle f, e, m, r, t\rangle, e_s) = \langle nop, nop, nop\rangle$,
3. $\forall\, \langle f, e, m, r, t\rangle \in Q, \quad \Delta(\langle f, e, m, r, t\rangle, e_f) = \langle nop, nop, insert\rangle$,
4. $\forall\, \langle f, e, m, r, t\rangle \in Q, \quad \Delta(\langle f, e, m, r, t\rangle, f_s) = \langle nop, nop, nop\rangle$,
5. $\forall\, \langle f, e, m, r, t\rangle \in Q, \quad \Delta(\langle f, e, m, r, t\rangle, f_f) = \langle set, nop, nop\rangle$,
6. $\forall\, \langle f, e, m, r, t\rangle \in Q, \quad \Delta(\langle f, e, m, r, t\rangle, t_s) = \langle nop, nop, nop\rangle$,
7. $\forall\, \langle f, e, m, r, t\rangle \in Q, \quad \Delta(\langle f, e, m, r, t\rangle, t_f) = \langle nop, set, nop\rangle$,

[2] O índice 2, em 1010_2, diz que 1010 é a representação binária de um inteiro. A representação decimal não é enfatizada.

8. $\forall\ \langle f, e, m, r, t\rangle \in Q, \quad \Delta(\langle f, e, m, r, t\rangle, r_s) = \langle nop, nop, remove\rangle$,
9. $\forall\ \langle f, e, m, r, t\rangle \in Q, \quad \Delta(\langle f, e, m, r, t\rangle, r_f) = \langle nop, nop, nop\rangle$,
10. $\forall\ \langle f, e, m, r, t\rangle \in Q, \quad \Delta(\langle f, e, m, r, t\rangle, m_s) = \langle nop, nop, nop\rangle$ e
11. $\forall\ \langle f, e, m, r, t\rangle \in Q, \quad \Delta(\langle f, e, m, r, t\rangle, m_f) = \langle reset, reset, rotate\rangle$,

onde Q é o conjunto de estados discretos da planta e, para todo $x \in \mathbb{B}$, $y \in \mathbb{B}$ e $p \in \mathbb{Z}$, $init_X(x) = false$, $init_Y(y) = false$ e $init_P(p) = 0$.

6.1.2 Especificação

As pré-condições restringindo o comportamento da planta àqueles satisfazendo os requisitos descritos anteriormente são enumeradas em seguida. Nas pré-condições, as divisões são consideradas inteiras e os operadores %, como em $a\%b$, representam os restos das divisões de a por b. Assim, $p\%2 = 1$ é verdade se o bit menos significativo de p é 1, $p/2\%2 = 1$ é verdade se o segundo bit menos significativo de p é 1 e, assim, sucessivamente.

- A mesa não deve girar sem ao menos uma peça na mesa.

$$\Lambda(m_s)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = (p > 0).$$

- A mesa não deve girar enquanto houver uma peça no slot P_4.

$$\Lambda(m_s)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = (p < 8).$$

- A mesa não deve girar enquanto a esteira, a furadeira, o testador ou o manipulador estiverem em operação.

$$\Lambda(m_s)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = (e = i \wedge f = i \wedge t = i \wedge r = i).$$

- A mesa não deve girar se no slot P_2 houver uma peça não furada.

$$\Lambda(m_s)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = (p/2\%2 \neq 1 \vee x).$$

- A mesa não deve girar se no slot P_3 houver uma peça não testada.

$$\Lambda(m_s)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = (p/4\%2 \neq 1 \ \vee \ y).$$

- A esteira não deve depositar uma peça em P_1 se neste slot já existir uma peça.

$$\Lambda(e_f)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = (p\%2 = 0).$$

- A esteira, a furadeira, o testador e o manipulador robótico não devem operar enquanto a mesa estiver girando.

$$\Lambda(e_s)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = (m = i),$$
$$\Lambda(f_s)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = (m = i),$$
$$\Lambda(t_s)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = (m = i) \text{ e}$$
$$\Lambda(r_s)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = (m = i).$$

- A furadeira não deve operar sem uma peça no slot P_2.

$$\Lambda(f_s)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = (p/2\%2 = 1).$$

- O testador não deve operar sem uma peça no slot P_3.

$$\Lambda(t_s)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = (p/4\%2 = 1).$$

- O manipulador não deve operar sem uma peça no slot P_4.

$$\Lambda(r_s)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = (p/8\%2 = 1).$$

- Uma peça deve ser furada uma única vez.

$$\Lambda(f_s)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = (\neg x).$$

- Uma peça deve ser testada uma única vez.

$$\Lambda(t_s)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = (\neg y).$$

Compondo-se as especificações parciais anteriores, obtém-se a especificação global consistindo das seguintes pré-condições:

$$\begin{aligned}
&\Lambda(r_s)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = (m = i \wedge p/8\%2 = 1),\\
&\Lambda(r_f)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = true,\\
&\Lambda(t_s)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = (m = i \wedge \neg y \wedge p/4\%2 = 1),\\
&\Lambda(t_f)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = true,\\
&\Lambda(f_s)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = (m = i \wedge \neg x \wedge p/2\%2 = 1),\\
&\Lambda(f_f)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = true,\\
&\Lambda(e_s)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = (m = i),\\
&\Lambda(e_f)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = (p\%2 = 0),\\
&\Lambda(m_s)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = ((p/2\%2 \neq 1 \vee x) \wedge (p/4\%2 \neq 1 \vee y) \wedge\\
&\qquad\qquad (p > 0) \wedge (p < 8) \wedge (e = i \wedge f = i \wedge t = i \wedge r = i)),\\
&\Lambda(m_f)\langle\langle f, e, m, r, t\rangle, \langle x, y, p\rangle\rangle = true.
\end{aligned}$$

6.1.3 Supervisor

Na mesa giratória, a única pré-condição a ser antecipada é aquela associada ao evento não controlável e_f. Dependendo do estado da planta, esta pré-condição deve ser antecipada para os eventos e_s, f_s e t_s. O resultado final de tais antecipações é apresentado na Tabela 6.2.

Na tabela, um traço da forma — indica que a transição não é definida. As entradas *false* indicam que a transição é definida, mas permanentemente desabilitada; tais entradas, basicamente, ou impedem o início de operação de algum equipamento quando a mesa está girando, ou impedem o início do giro da mesa quando algum equipamento está em operação (coluna m_s).

Nos estados $\langle -, i, i, -, -\rangle$, isto é, quando a esteira não está em operação e a mesa não está girando, a pré-condição $\Lambda(e_s)\langle q, x, y, p\rangle = (p/1\,\%\,2 = 0)$ restringe o início de operação da esteira às condições a partir das quais não ocorre o overflow no slot P_1.

Quando a esteira está em operação, o evento e_f não pode ser impedido. Nestes casos, os termos $p/1\,\%\,2 = 0$ adicionados às pré-condições dos eventos f_s e t_s garantem que a habilitação destes eventos não acarretarão overflows no slot P_1. É importante notar que este termo é redundante, pois, na prática, isto nunca acontece; sua eliminação, entretanto, depende do contexto de todo o sistema e por esta razão, difícil de ser realizada automaticamente.

As entradas da coluna t_s podem ser interpretadas de forma semelhante às entradas da coluna f_s, substituindo-se apenas a operação de furação pela operação de teste. Na coluna r_s, os termos $(p-8)\,\%\,2 = 0$ nas entradas associadas aos estados nos quais a esteira está em operação, têm papéis semelhantes aos termos $p/1\,\%\,2 = 0$ das colunas f_s e t_s, isto é, garantir o não overflow no slot P_1.

O único estado sujeito ao bloqueio é o estado $\langle i,i,i,i,i\rangle$. Neste estado o invariante definindo as configurações não bloqueantes é dado por:

$$\overbrace{p/1\,\%\,2 = 0}^{C_1} \vee \overbrace{p > 0 \wedge p < 8 \wedge (p/2\,\%\,2 \neq 1 \vee x) \wedge (p/4\,\%\,2 \neq 1 \vee y)}^{C_2} \vee$$
$$\underbrace{\neg x \wedge p/2\,\%\,2 = 1}_{C_3} \vee \underbrace{\neg y \wedge p/4\,\%\,2 = 1}_{C_4} \vee \underbrace{p/8\,\%\,2 = 1}_{C_5}.$$

A Tabela 6.3 resume a análise deste invariante em cada instanciação associada ao estado $\langle i,i,i,i,i\rangle$. Nas colunas "Termos verdade" [3] são apresentados os termos da disjunção anterior que, avaliados na instanciação correspondente, resultam verdades. Como em cada instanciação pelo menos um termo da disjunção resulta verdade, conclui-se que o sistema está livre de deadlocks.

6.1.4 Observações

Neste exemplo se pode perceber toda a flexibilidade das especificações por pré-condições. Esta flexibilidade só está limitada à capacidade do supervisor avaliar as expressões e fórmulas simbólicas.

[3] Nestas colunas, C_1 corresponde ao termo $p/1\,\%\,2 = 0$, C_2 corresponde ao termo $p > 0 \wedge p < 8 \wedge (p/2\,\%\,2 \neq 1 \vee x) \wedge (p/4\,\%\,2 \neq 1 \vee y)$, C_3 corresponde ao termo $\neg x \wedge p/2\,\%\,2 = 1$, C_4 corresponde ao termo $\neg y \wedge p/4\,\%\,2 = 1$ e, finalmente, C_5 corresponde ao termo $p/8\,\%\,2 = 1$.

$\langle f, e, m, r, t \rangle$	e_s	m_s	f_s	t_s	r_s
$\langle i, i, i, i, i \rangle$	$p\,\%\,2 = 0$	$p > 0 \wedge p < 8 \wedge$ $(p/2\,\%\,2 \neq 1 \vee x) \wedge$ $(p/4\,\%\,2 \neq 1 \vee y)$	$\neg x \wedge p/2\,\%\,2 = 1$	$y \wedge p/4\,\%\,2 = 1$	$p/8\,\%\,2 = 1$
$\langle i, i, i, i, w \rangle$	$p\,\%\,2 = 0$	$false$	$\neg x \wedge p/2\,\%\,2 = 1$	—	$p/8\,\%\,2 = 1$
$\langle i, i, i, w, i \rangle$	$p\,\%\,2 = 0$	$false$	$\neg x \wedge p/2\,\%\,2 = 1$	$\neg y \wedge p/4\,\%\,2 = 1$	—
$\langle i, i, i, w, w \rangle$	$p\,\%\,2 = 0$	$false$	$\neg x \wedge p/2\,\%\,2 = 1$	—	—
$\langle i, i, w, i, i \rangle$	$false$	—	$false$	$false$	$false$
$\langle i, i, w, i, w \rangle$	$false$	—	$false$	—	$false$
$\langle i, i, w, w, i \rangle$	$false$	—	$false$	$false$	—
$\langle i, i, w, w, w \rangle$	$false$	—	$false$	—	—
$\langle i, w, i, i, i \rangle$	—	$false$	$\neg x \wedge p/2\,\%\,2 = 1 \wedge p\,\%\,2 = 0$	$\neg y \wedge p/4\,\%\,2 = 1 \wedge p\,\%\,2 = 0$	$p/8\,\%\,2 = 1 \wedge (p-8)\,\%\,2 = 0$
$\langle i, w, i, i, w \rangle$	—	$false$	$\neg x \wedge p/2\,\%\,2 = 1 \wedge p\,\%\,2 = 0$	—	$p/8\,\%\,2 = 1 \wedge (p-8)\,\%\,2 = 0$
$\langle i, w, i, w, i \rangle$	—	$false$	$\neg x \wedge p/2\,\%\,2 = 1 \wedge p\,\%\,2 = 0$	$\neg y \wedge p/4\,\%\,2 = 1 \wedge p\,\%\,2 = 0$	—
$\langle i, w, i, w, w \rangle$	—	$false$	$\neg x \wedge p/2\,\%\,2 = 1 \wedge p\,\%\,2 = 0$	—	—
$\langle i, w, w, i, i \rangle$	—	—	$false$	$false$	$false$
$\langle i, w, w, i, w \rangle$	—	—	$false$	—	$false$
$\langle i, w, w, w, i \rangle$	—	—	$false$	$false$	—
$\langle i, w, w, w, w \rangle$	—	—	$false$	—	—
$\langle w, i, i, i, i \rangle$	$p\,\%\,2 = 0$	$false$	—	$\neg y \wedge p/4\,\%\,2 = 1$	$p/8\,\%\,2 = 1$
$\langle w, i, i, i, w \rangle$	$p\,\%\,2 = 0$	$false$	—	—	$p/8\,\%\,2 = 1$
$\langle w, i, i, w, i \rangle$	$p\,\%\,2 = 0$	$false$	—	$\neg y \wedge p/4\,\%\,2 = 1$	—
$\langle w, i, i, w, w \rangle$	$p\,\%\,2 = 0$	$false$	—	—	—
$\langle w, i, w, i, i \rangle$	$false$	—	—	$false$	$false$
$\langle w, i, w, i, w \rangle$	$false$	—	—	—	$false$
$\langle w, i, w, w, i \rangle$	$false$	—	—	$false$	—
$\langle w, i, w, w, w \rangle$	$false$	—	—	—	—
$\langle w, w, i, i, i \rangle$	—	$false$	—	$\neg y \wedge p/4\,\%\,2 = 1 \wedge p\,\%\,2 = 0$	$p/8\,\%\,2 = 1 \wedge (p-8)\,\%\,2 = 0$
$\langle w, w, i, i, w \rangle$	—	$false$	—	—	$p/8\,\%\,2 = 1 \wedge (p-8)\,\%\,2 = 0$
$\langle w, w, i, w, i \rangle$	—	$false$	—	$\neg y \wedge p/4\,\%\,2 = 1 \wedge p\,\%\,2 = 0$	—
$\langle w, w, i, w, w \rangle$	—	$false$	—	—	—
$\langle w, w, w, i, i \rangle$	—	—	—	$false$	$false$
$\langle w, w, w, i, w \rangle$	—	—	—	—	$false$
$\langle w, w, w, w, i \rangle$	—	—	—	$false$	—
$\langle w, w, w, w, w \rangle$	—	—	—	—	—

Tabela 6.2: Supervisor da mesa giratória.

p	x	y	Termos verdade	p	x	y	Termos verdade
0000_2	*true*	*true*	$C_1 \vee C_2$	1000_2	*true*	*true*	$C_1 \vee C_5$
0000_2	*true*	*false*	$C_1 \vee C_2$	1000_2	*true*	*false*	$C_1 \vee C_5$
0000_2	*false*	*true*	$C_1 \vee C_2$	1000_2	*false*	*true*	$C_1 \vee C_5$
0000_2	*false*	*false*	$C_1 \vee C_2$	1000_2	*false*	*false*	$C_1 \vee C_5$
0001_2	*true*	*true*	C_2	1001_2	*true*	*true*	C_5
0001_2	*true*	*false*	C_2	1001_2	*true*	*false*	C_5
0001_2	*false*	*true*	C_2	1001_2	*false*	*true*	C_5
0001_2	*false*	*false*	C_2	1001_2	*false*	*false*	C_5
0010_2	*true*	*true*	$C_1 \vee C_2$	1010_2	*true*	*true*	$C_1 \vee C_5$
0010_2	*true*	*false*	$C_1 \vee C_2$	1010_2	*true*	*false*	$C_1 \vee C_5$
0010_2	*false*	*true*	$C_1 \vee C_3$	1010_2	*false*	*true*	$C_1 \vee C_3 \vee C_5$
0010_2	*false*	*false*	$C_1 \vee C_3$	1010_2	*false*	*false*	$C_1 \vee C_3 \vee C_5$
0011_2	*true*	*true*	C_2	1011_2	*true*	*true*	C_5
0011_2	*true*	*false*	C_2	1011_2	*true*	*false*	C_5
0011_2	*false*	*true*	C_3	1011_2	*false*	*true*	$C_3 \vee C_5$
0011_2	*false*	*false*	C_3	1011_2	*false*	*false*	$C_3 \vee C_5$
0100_2	*true*	*true*	$C_1 \vee C_2$	1100_2	*true*	*true*	$C_1 \vee C_5$
0100_2	*true*	*false*	$C_1 \vee C_4$	1100_2	*true*	*false*	$C_1 \vee C_4 \vee C_5$
0100_2	*false*	*true*	$C_1 \vee C_2$	1100_2	*false*	*true*	$C_1 \vee C_5$
0100_2	*false*	*false*	$C_1 \vee C_4$	1100_2	*false*	*false*	$C_1 \vee C_4 \vee C_5$
0101_2	*true*	*true*	C_2	1101_2	*true*	*true*	C_5
0101_2	*true*	*false*	C_4	1101_2	*true*	*false*	$C_4 \vee C_5$
0101_2	*false*	*true*	C_2	1101_2	*false*	*true*	C_5
0101_2	*false*	*false*	C_4	1101_2	*false*	*false*	$C_4 \vee C_5$
0110_2	*true*	*true*	$C_1 \vee C_2$	1110_2	*true*	*true*	$C_1 \vee C_5$
0110_2	*true*	*false*	$C_1 \vee C_4$	1110_2	*true*	*false*	$C_1 \vee C_4 \vee C_5$
0110_2	*false*	*true*	$C_1 \vee C_3$	1110_2	*false*	*true*	$C_1 \vee C_3 \vee C_5$
0110_2	*false*	*false*	$C_1 \vee C_3 \vee C_4$	1110_2	*false*	*false*	$C_1 \vee C_3 \vee C_4 \vee C_5$
0111_2	*true*	*true*	C_2	1111_2	*true*	*true*	C_5
0111_2	*true*	*false*	C_4	1111_2	*true*	*false*	$C_4 \vee C_5$
0111_2	*false*	*true*	C_3	1111_2	*false*	*true*	$C_3 \vee C_5$
0111_2	*false*	*false*	$C_3 \vee C_4$	1111_2	*false*	*false*	$C_3 \vee C_4 \vee C_5$

Tabela 6.3: Análise do invariante evitando deadlocks da mesa giratória.

6.2 O gato e o rato

A Figura 6.3 ilustra a planta "o gato e o rato". Na planta existem 4 quartos que se conectam entre si através das portas m_i e c_i. Na configuração inicial, encontra-se no quarto número 2 um gato e no quarto número 4, um rato. O gato pode passar de um quarto a outro através das portas c_i e o rato pode passar de um quarto a outro através das portas m_i. Considerando-se que a abertura (e o fechamento) de todas as portas pode ser controlada, o objetivo deste exemplo é o projeto de um supervisor não bloqueante que restrinja minimamente o fechamento das portas e que impeça que o gato e o rato se encontrem no mesmo quarto.

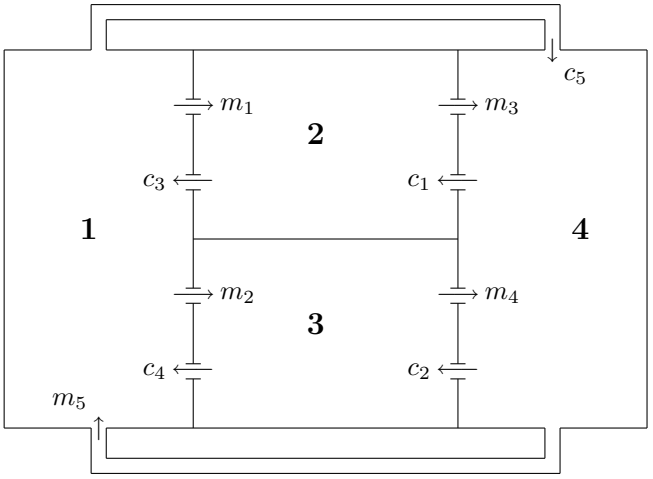

Figura 6.3: O gato e o rato.

6.2.1 Modelo da planta

As informações necessárias para restringir o comportamento do sistema àqueles descritos anteriormente, são simplesmente as posições do gato e do rato em cada instante. Tais informações estão associadas aos estados discretos do STE obtido a partir da composição síncrona entre os subsistemas descrevendo os comportamentos do gato e do rato. Os estados discretos da planta serão denotados $\langle c, m \rangle$, onde c e m representam respectivamente, as posições do gato e do rato. A função de transição de estados deste sistema é ilustrada na Tabela 6.4.

Estado $\langle c, m\rangle$	c_1	c_2	c_3	c_4	c_5	m_1	m_2	m_3	m_4	m_5
$\langle 1,1\rangle$	—	—	—	—	$\langle 4,1\rangle$	$\langle 1,2\rangle$	$\langle 1,3\rangle$	—	—	—
$\langle 1,2\rangle$	—	—	—	—	$\langle 4,2\rangle$	—	—	$\langle 1,4\rangle$	—	—
$\langle 1,3\rangle$	—	—	—	—	$\langle 4,3\rangle$	—	—	—	$\langle 1,4\rangle$	—
$\langle 1,4\rangle$	—	—	—	—	$\underline{\langle 4,4\rangle}$	—	—	—	—	$\underline{\langle 1,1\rangle}$
$\langle 2,1\rangle$	—	—	$\underline{\langle 1,1\rangle}$	—	—	$\underline{\langle 2,2\rangle}$	$\langle 2,3\rangle$	—	—	—
$\langle 2,2\rangle$	—	—	$\langle 1,2\rangle$	—	—	—	—	$\langle 2,4\rangle$	—	—
$\langle 2,3\rangle$	—	—	$\langle 1,3\rangle$	—	—	—	—	—	$\langle 2,4\rangle$	—
$\langle 2,4\rangle$	—	—	$\langle 1,4\rangle$	—	—	—	—	—	—	$\langle 2,1\rangle$
$\langle 3,1\rangle$	—	—	—	$\underline{\langle 1,1\rangle}$	—	$\langle 3,2\rangle$	$\underline{\langle 3,3\rangle}$	—	—	—
$\langle 3,2\rangle$	—	—	—	$\langle 1,2\rangle$	—	—	—	$\langle 3,4\rangle$	—	—
$\langle 3,3\rangle$	—	—	—	$\langle 1,3\rangle$	—	—	—	—	$\langle 3,4\rangle$	—
$\langle 3,4\rangle$	—	—	—	$\langle 1,4\rangle$	—	—	—	—	—	$\langle 3,1\rangle$
$\langle 4,1\rangle$	$\langle 2,1\rangle$	$\langle 3,1\rangle$	—	—	—	$\langle 4,2\rangle$	$\langle 4,3\rangle$	—	—	—
$\langle 4,2\rangle$	$\underline{\langle 2,2\rangle}$	$\langle 3,2\rangle$	—	—	—	—	—	$\underline{\langle 4,4\rangle}$	—	—
$\langle 4,3\rangle$	$\langle 2,3\rangle$	$\underline{\langle 3,3\rangle}$	—	—	—	—	—	—	$\underline{\langle 4,4\rangle}$	—
$\langle 4,4\rangle$	$\langle 2,4\rangle$	$\langle 3,4\rangle$	—	—	—	—	—	—	—	$\langle 4,1\rangle$

Tabela 6.4: STE da planta "o gato e o rato".

6.2.2 Especificação

As transições a serem evitadas são aquelas sublinhadas na Tabela 6.4. Claramente, a lógica para a restrição dos eventos depende exclusivamente do estado da planta. Portanto, a coleção de dadospara este exemplo pode ser vazia. A especificação requerida inclui as seguintes pré-condições:

$$\Lambda(m_1)\langle c,m\rangle = (c \neq 2), \quad \Lambda(m_2)\langle c,m\rangle = (c \neq 3), \quad \Lambda(m_3)\langle c,m\rangle = (c \neq 4),$$
$$\Lambda(m_4)\langle c,m\rangle = (c \neq 4), \quad \Lambda(m_5)\langle c,m\rangle = (c \neq 1),$$
$$\Lambda(c_1)\langle c,m\rangle = (m \neq 2), \quad \Lambda(c_2)\langle c,m\rangle = (m \neq 3), \quad \Lambda(c_3)\langle c,m\rangle = (m \neq 1),$$
$$\Lambda(c_4)\langle c,m\rangle = (m \neq 1), \quad \Lambda(c_5)\langle c,m\rangle = (m \neq 4).$$

6.2.3 Supervisor

Como todos os eventos são controláveis, a especificação anterior é controlável. O supervisor da Tabela 6.5 restringe os comportamentos da planta àqueles em que o gato e o rato nunca ocupam quartos iguais. Entretanto, o sistema supervisionado bloqueia no estado $\langle 1,4\rangle$. Isto pode ser observado na Tabela 6.6 a qual resume a computação dos invariantes definindo as configurações bloqueantes em cada estado.

Estado $\langle c,m\rangle$	c_1	c_2	c_3	c_4	c_5	m_1	m_2	m_3	m_4	m_5
$\langle 1,1\rangle$	—	—	—	—	$m \neq 4$	$c \neq 2$	$c \neq 3$	—	—	—
$\langle 1,2\rangle$	—	—	—	—	$m \neq 4$	—	—	$c \neq 4$	—	—
$\langle 1,3\rangle$	—	—	—	—	$m \neq 4$	—	—	—	$c \neq 4$	—
$\langle 1,4\rangle$	—	—	—	—	$m \neq 4$	—	—	—	—	$c \neq 1$
$\langle 2,1\rangle$	—	—	$m \neq 1$	—	—	$c \neq 2$	$c \neq 3$	—	—	—
$\langle 2,2\rangle$	—	—	$m \neq 1$	—	—	—	—	$c \neq 4$	—	—
$\langle 2,3\rangle$	—	—	$m \neq 1$	—	—	—	—	—	$c \neq 4$	—
$\langle 2,4\rangle$	—	—	$m \neq 1$	—	—	—	—	—	—	$c \neq 1$
$\langle 3,1\rangle$	—	—	—	$m \neq 1$	—	$c \neq 2$	$c \neq 3$	—	—	—
$\langle 3,2\rangle$	—	—	—	$m \neq 1$	—	—	—	$c \neq 4$	—	—
$\langle 3,3\rangle$	—	—	—	$m \neq 1$	—	—	—	—	$c \neq 4$	—
$\langle 3,4\rangle$	—	—	—	$m \neq 1$	—	—	—	—	—	$c \neq 1$
$\langle 4,1\rangle$	$m \neq 2$	$m \neq 3$	—	—	—	$c \neq 2$	$c \neq 3$	—	—	—
$\langle 4,2\rangle$	$m \neq 2$	$m \neq 3$	—	—	—	—	—	$c \neq 4$	—	—
$\langle 4,3\rangle$	$m \neq 2$	$m \neq 3$	—	—	—	—	—	—	$c \neq 4$	—
$\langle 4,4\rangle$	$m \neq 2$	$m \neq 3$	—	—	—	—	—	—	—	$c \neq 1$

Tabela 6.5: Supervisor (bloqueante) para a planta "o gato e o rato".

Como pode ser visto na Tabela 6.4, os eventos conduzindo o sistema ao estado $\langle 1,4\rangle$ são:

- m_3 a partir do estado $\langle 1,2\rangle$,
- m_4 a partir do estado $\langle 1,3\rangle$,
- c_3 a partir dos estados $\langle 2,4\rangle$,
- c_4 a partir dos estados $\langle 3,4\rangle$.

A transformação do invariante falso do estado $\langle 1,4\rangle$, impede a ocorrência dos eventos m_3, m_4, c_3 e c_4 nas condições anteriores. O STE não bloqueante é ilustrado na Tabela 6.7 e Figura 6.4.

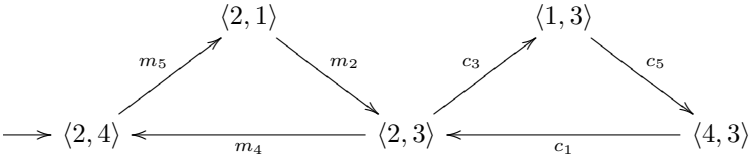

Figura 6.4: Supervisor não bloqueante para a planta "o gato e o rato".

Estado $\langle c, m \rangle$	Invariante	
$\langle 1,1 \rangle$	$m \neq 4 \vee c \neq 2 \vee c \neq 3$	*true*
$\langle 1,2 \rangle$	$m \neq 4 \vee c \neq 4 \vee m \neq 4 \wedge (m \neq 2 \vee m \neq 3) \vee c \neq 4 \wedge c \neq 1$	*true*
$\langle 1,3 \rangle$	$m \neq 4 \vee c \neq 4 \vee m \neq 4 \wedge (m \neq 2 \vee m \neq 3) \vee c \neq 4 \wedge c \neq 1$	*true*
$\langle 1,4 \rangle$	$m \neq 4 \vee c \neq 1$	*false*
$\langle 2,1 \rangle$	$m \neq 1 \vee c \neq 2 \vee c \neq 3$	*true*
$\langle 2,2 \rangle$	$m \neq 1 \vee c \neq 4$	*true*
$\langle 2,3 \rangle$	$m \neq 1 \vee c \neq 4$	*true*
$\langle 2,4 \rangle$	$m \neq 1 \vee c \neq 1 \vee m \neq 1 \wedge m \neq 4 \vee c \neq 1 \wedge (c \neq 2 \vee c \neq 3)$	*true*
$\langle 3,1 \rangle$	$m \neq 1 \vee c \neq 2 \vee c \neq 3$	*true*
$\langle 3,2 \rangle$	$m \neq 1 \vee c \neq 4$	*true*
$\langle 3,3 \rangle$	$m \neq 1 \vee c \neq 4$	*true*
$\langle 3,4 \rangle$	$m \neq 1 \vee c \neq 1 \vee m \neq 1 \wedge m \neq 4 \vee c \neq 1 \wedge (c \neq 2 \vee c \neq 3)$	*true*
$\langle 4,1 \rangle$	$m \neq 2 \vee m \neq 3 \vee c \neq 2 \vee c \neq 3$	*true*
$\langle 4,2 \rangle$	$m \neq 2 \vee m \neq 3 \vee c \neq 4$	*true*
$\langle 4,3 \rangle$	$m \neq 2 \vee m \neq 3 \vee c \neq 4$	*true*
$\langle 4,4 \rangle$	$m \neq 2 \vee m \neq 3 \vee c \neq 1$	*true*

Tabela 6.6: Invariantes da planta “o gato e o rato”.

6.2.4 Observações

Neste exemplo, todas as pré-condições são proposições definidas sobre as variáveis c e m, cujos domínios de instanciações são finitos e enumeráveis. Além disso, não existem variáveis livres, isto é, parâmetros. Para casos como este (ver Seção 3.3), o Procedimento 5.2.1 é decidível e converge. O conjunto de pré-condições do supervisor não bloqueante pode ser resolvido off-line e o resultado produzido é um supervisor semelhante ao da teoria clássica, ou seja, um gerador.

$\langle c, m \rangle$	c_1	c_2	c_3	c_4	c_5	m_1	m_2	m_3	m_4	m_5
$\langle 1,1 \rangle$	—	—	—	—	$m \neq 4$	$c \neq 2$	$c \neq 3$	—	—	—
					$\langle 4,1 \rangle$	$\langle 1,2 \rangle$	$\langle 1,3 \rangle$	$c \neq 4 \wedge c \neq 1$		
$\langle 1,2 \rangle$	—	—	—	—	$m \neq 4 \wedge (m \neq 2 \vee m \neq 3)$	—	—	$\langle 1,4 \rangle$	—	—
					$\langle 4,2 \rangle$				$c \neq 4 \wedge c \neq 1$	
$\langle 1,3 \rangle$	—	—	—	—	$m \neq 4 \wedge (m \neq 2 \vee m \neq 3)$	—	—	—	$\langle 1,4 \rangle$	—
					$\langle 4,3 \rangle$					
$\langle 1,4 \rangle$	—	—	—	—	$m \neq 4$	—	—	—	—	$c \neq 1$
					$\langle 4,4 \rangle$					$\langle 1,1 \rangle$
$\langle 2,1 \rangle$	—	—	$m \neq 1$	—	—	$c \neq 2$	$c \neq 3$	—	—	—
			$\langle 1,1 \rangle$			$\langle 2,2 \rangle$	$\langle 2,3 \rangle$			
$\langle 2,2 \rangle$	—	—	$m \neq 1$	—	—	—	—	$c \neq 4$	—	—
			$\langle 1,2 \rangle$					$\langle 2,4 \rangle$		
$\langle 2,3 \rangle$	—	—	$m \neq 1$	—	—	—	—	—	$c \neq 4$	—
			$\langle 1,3 \rangle$						$\langle 2,4 \rangle$	
$\langle 2,4 \rangle$	—	—	$m \neq 1 \wedge m \neq 4$	—	—	—	—	—	—	$c \neq 1 \wedge (c \neq 2 \vee c \neq 3)$
			$\langle 1,4 \rangle$							$\langle 2,1 \rangle$
$\langle 3,1 \rangle$	—	—	—	$m \neq 1$	—	$c \neq 2$	$c \neq 3$	—	—	—
				$\langle 1,1 \rangle$		$\langle 3,2 \rangle$	$\langle 3,3 \rangle$			
$\langle 3,2 \rangle$	—	—	—	$m \neq 1$	—	—	—	$c \neq 4$	—	—
				$\langle 1,2 \rangle$				$\langle 3,4 \rangle$		
$\langle 3,3 \rangle$	—	—	—	$m \neq 1$	—	—	—	—	$c \neq 4$	—
				$\langle 1,3 \rangle$					$\langle 3,4 \rangle$	
$\langle 3,4 \rangle$	—	—	—	$m \neq 1 \wedge m \neq 4$	—	—	—	—	—	$c \neq 1 \wedge (c \neq 2 \vee c \neq 3)$
				$\langle 1,4 \rangle$						$\langle 3,1 \rangle$
$\langle 4,1 \rangle$	$m \neq 2$	$m \neq 3$	—	—	—	$c \neq 2$	$c \neq 3$	—	—	—
	$\langle 2,1 \rangle$	$\langle 3,1 \rangle$				$\langle 4,2 \rangle$	$\langle 4,3 \rangle$			
$\langle 4,2 \rangle$	$m \neq 2$	$m \neq 3$	—	—	—	—	—	$c \neq 4$	—	—
	$\langle 2,2 \rangle$	$\langle 3,2 \rangle$						$\langle 4,4 \rangle$		
$\langle 4,3 \rangle$	$m \neq 2$	$m \neq 3$	—	—	—	—	—	—	$c \neq 4$	—
	$\langle 2,3 \rangle$	$\langle 3,3 \rangle$							$\langle 4,4 \rangle$	
$\langle 4,4 \rangle$	$m \neq 2$	$m \neq 3$	—	—	—	—	—	—	—	$c \neq 1$
	$\langle 2,4 \rangle$	$\langle 3,4 \rangle$								$\langle 4,1 \rangle$

Tabela 6.7: Supervisor não bloqueante para a planta “o gato e o rato”.

6.3 Máquinas com vários modos de operação

A máquina da Figura 6.5 possui vários modos de operação. Em I a máquina está em repouso e em W_i, $1 \leq i \leq 7$, a máquina está trabalhando no modo i. As setas rotuladas β_j, $1 \leq j \leq 3$, indicam mudanças controláveis dos modos de operação e as setas rotuladas γ_k, $1 \leq k \leq 5$, e α_1 indicam mudanças não-controláveis.

A planta "máquinas com vários modos de operação" consiste de duas máquinas com vários modos de operação, M_1 e M_2, que compartilham um mesmo armazém A (ver Exemplo 3.1.2) com a capacidade de armazenar de 0 a N produtos (onde N é um parâmetro do projeto). Na planta, sempre que alguma transição β_j ocorre, ocorre também a inserção de um produto no armazém e, sempre que alguma transição γ_k ocorre, ocorre também a remoção de um produto do armazém. As transições α_1 não implicam alterações no armazém.

O que se deseja é a construção de um supervisor que atue sobre as mudanças controláveis nos modos de operação das máquinas de modo a permiti-las somente quando a partir delas não exista a possibilidade de ocorrer o overflow ou o underflow no armazém.

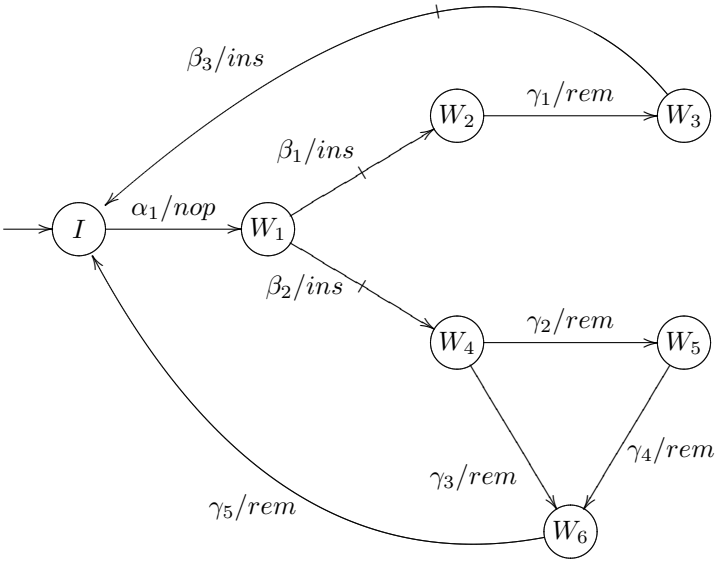

Figura 6.5: Máquina com vários modos de operação.

6.3.1 Modelo da planta

Devido à sua complexidade, o produto síncrono entre as máquinas não é ilustrado. Como as máquinas são iguais, nas notações a seguir, acentos serão utilizados para relacionar os elementos da composição com as máquinas de origem. Os elementos relacionados com a máquina M_1 serão denotados x^1, e os elementos relacionados com a máquina M_2 serão denotados x^2. Portanto, γ_1^1 é um evento de M_1 e γ_1^2 é um evento de M_2. O estado da composição $M_1||M_2$ serão denotados por $\langle s^1, s^2\rangle$, onde s^1 denota o estado da máquina M_1 e s^2 denota o estado de M_2. A coleção de dados necessária inclui apenas um monóide modelando o armazém A. A variável introduzida por este monóide será denotada por n. Os eventos β_j são associados à operações de inserção de itens em A e os eventos γ_k são associados à operações de remoção de itens de A.

6.3.2 Especificação

A especificação evitando underflow em A restringe as transições γ_k a $n > 0$ e a especificação evitando overflow em A restringe as transições β_k a $n < N$.

6.3.3 Supervisor

A Tabela 6.8 ilustra o supervisor requerido neste exemplo. O único estado sujeito ao bloqueio é o estado $\langle W_1^1, W_1^2\rangle$. O invariante definindo as configurações que não causam deadlocks neste estado é:

$$n < N. \tag{6.1}$$

Como no estado $\langle W_1^1, W_1^2\rangle$ a instanciação $n = N$ é acessível (e isto foi determinado empiricamente), o supervisor da Tabela 6.8 é bloqueante.

Com a finalidade de eliminar o bloqueio anterior, faz-se necessário restringir os eventos α_1^1 e α_1^2 com a pré-condição $n < N$. Antecipando-se estas pré-condições, obtém-se o novo conjunto de pré-condições da Tabela 6.9. É importante observar que durante estas antecipações, o evento ξ recebeu a pré-condição $n < N$. Esta pré-condição fornece informações para o instanciamento do parâmetro N.

Calculando-se novamente o invariante no estado $\langle W_1^1, W_1^2\rangle$, obtém-se:

$$n < N, \tag{6.2}$$

o qual é igual ao invariante anterior. Como no supervisor da Tabela 6.9 o valor de n no estado $\langle W_1^1, W_1^2 \rangle$ nunca é igual a N, o novo supervisor não bloqueia.

6.3.4 Observações

Os invariantes das Equações 6.1 e 6.2 e as pré-condições das Tabelas 6.8 e 6.9 foram simplificados de acordo com regras lógicas e de acordo com o contexto do sistema (acessibilidade). Computacionalmente, tais simplificações, geralmente, não são realizáveis. Por exemplo, sem simplificações as Equações 6.1 e 6.2 seriam, respectivamente:

$$n < N \wedge n + 1 > 0 \vee n < N \wedge n + 1 > 0 \wedge n > 0 \wedge n - 1 > 0$$

e

$$\begin{aligned} n < N \wedge n + 1 > 0 \vee n < N \wedge n + 1 > 0 \wedge n > 0 \wedge n - 1 > 0 \vee \\ n < N \wedge n + 1 > 0 \wedge \\ (n < N \wedge n + 1 > 0 \vee n < N \wedge n + 1 > 0 \wedge n > 0 \wedge n - 1 > 0 \vee n < N) \vee \\ n < N \wedge n + 1 > 0 \wedge n > 0 \wedge n - 1 > 0 \wedge \\ (n - 1 < N \wedge n > 0 \vee n - 1 < N \wedge n > 0 \wedge n - 1 > 0 \wedge n - 2 > 0) \wedge \\ (n - 2 < N \wedge n - 1 > 0 \vee n - 2 < N \wedge n - 1 > 0 \wedge n - 2 > 0 \wedge n - 3 > 0). \end{aligned}$$

A dificuldade de se realizar simplificações automáticas torna os algoritmos para a síntese de supervisores, detecção e eliminação de bloqueios muito complexos computacionalmente.

	β_1^1	β_1^2	β_2^1	β_2^2	β_3^1	β_3^2
$\langle I^1, W_1^2\rangle$	—	$n < N$	—	$n < N \wedge n - 1 > 0$	—	—
$\langle I^1, W_3^2\rangle$	—	—	—	—	—	$n < N$
$\langle W_1^1, I^2\rangle$	$n < N$	—	$n < N \wedge n - 1 > 0$	—	—	—
$\langle W_3^1, I^2\rangle$	—	—	—	—	$n < N$	—
$\langle W_1^1, W_1^2\rangle$	$n < N$	$n < N$	$n < N \wedge n - 1 > 0$	$n < N \wedge n - 1 > 0$	—	—
$\langle W_1^1, W_2^2\rangle$	$n < N \wedge n > 0$	—	$n < N \wedge n - 2 > 0$	—	—	—
$\langle W_1^1, W_3^2\rangle$	$n < N$	—	$n < N \wedge n - 1 > 0$	—	—	$n < N$
$\langle W_1^1, W_4^2\rangle$	$n < N \wedge n - 2 > 0$	—	$n < N \wedge n - 4 > 0$	—	—	—
$\langle W_1^1, W_5^2\rangle$	$n < N \wedge n - 1 > 0$	—	$n < N \wedge n - 3 > 0$	—	—	—
$\langle W_1^1, W_6^2\rangle$	$n < N \wedge n > 0$	—	$n < N \wedge n - 2 > 0$	—	—	—
$\langle W_2^1, W_1^2\rangle$	—	$n < N \wedge n > 0$	—	$n < N \wedge n - 2 > 0$	—	—
$\langle W_2^1, W_3^2\rangle$	—	—	—	—	—	$n < N$
$\langle W_3^1, W_1^2\rangle$	—	$n < N$	—	$n < N \wedge n - 1 > 0$	$n < N$	—
$\langle W_3^1, W_2^2\rangle$	—	—	—	—	$n < N$	—
$\langle W_3^1, W_3^2\rangle$	—	—	—	—	$n < N$	—
$\langle W_3^1, W_4^2\rangle$	—	—	—	—	$n < N \wedge n - 1 > 0$	—
$\langle W_3^1, W_5^2\rangle$	—	—	—	—	$n < N \wedge n > 0$	—
$\langle W_3^1, W_6^2\rangle$	—	—	—	—	$n < N$	—
$\langle W_4^1, W_1^2\rangle$	—	$n < N \wedge n - 2 > 0$	—	$n < N \wedge n - 4 > 0$	—	—
$\langle W_4^1, W_3^2\rangle$	—	—	—	—	—	$n < N \wedge n - 1 > 0$
$\langle W_5^1, W_1^2\rangle$	—	$n < N \wedge n - 1 > 0$	—	$n < N \wedge n - 3 > 0$	—	—
$\langle W_5^1, W_3^2\rangle$	—	—	—	—	—	$n < N \wedge n > 0$
$\langle W_6^1, W_1^2\rangle$	—	$n < N \wedge n > 0$	—	$n < N \wedge n - 2 > 0$	—	—
$\langle W_6^1, W_3^2\rangle$	—	—	—	—	—	$n < N$

Tabela 6.8: Supervisor para as máquina com múltiplos modos de operação.

	β_1^1	β_1^2	β_2^1	β_2^2	β_3^1	β_3^2
$\langle I^1, W_1^2\rangle$	—	$n < N$	—	$n < N \wedge n - 1 > 0$	—	—
$\langle I^1, W_3^2\rangle$	—	—	—	—	—	$n + 1 < N$
$\langle W_1^1, I^2\rangle$	$n < N$	—	$n < N \wedge n - 1 > 0$	—	—	—
$\langle W_3^1, I^2\rangle$	—	—	—	—	$n + 1 < N$	—
$\langle W_1^1, W_1^2\rangle$	$n < N$	$n < N$	$n < N \wedge n - 1 > 0$	$n < N \wedge n - 1 > 0$	—	—
$\langle W_1^1, W_2^2\rangle$	$n < N \wedge n > 0$	—	$n < N \wedge n - 2 > 0$	—	—	—
$\langle W_1^1, W_3^2\rangle$	$n < N$	—	$n < N \wedge n - 1 > 0$	—	—	$n + 1 < N$
$\langle W_1^1, W_4^2\rangle$	$n < N \wedge n - 2 > 0$	—	$n < N \wedge n - 4 > 0$	—	—	—
$\langle W_1^1, W_5^2\rangle$	$n < N \wedge n - 1 > 0$	—	$n < N \wedge n - 3 > 0$	—	—	—
$\langle W_1^1, W_6^2\rangle$	$n < N \wedge n > 0$	—	$n < N \wedge n - 2 > 0$	—	—	—
$\langle W_2^1, W_1^2\rangle$	—	$n < N \wedge n > 0$	—	$n < N \wedge n - 2 > 0$	—	—
$\langle W_2^1, W_3^2\rangle$	—	—	—	—	—	$n < N$
$\langle W_3^1, W_1^2\rangle$	—	$n < N$	—	$n < N \wedge n - 1 > 0$	$n + 1 < N$	—
$\langle W_3^1, W_2^2\rangle$	—	—	—	—	$n < N$	—
$\langle W_3^1, W_3^2\rangle$	—	—	—	—	$n + 1 < N$	—
$\langle W_3^1, W_4^2\rangle$	—	—	—	—	$n < N \wedge n - 1 > 0$	$n + 1 < N$
$\langle W_3^1, W_5^2\rangle$	—	—	—	—	$n < N \wedge n > 0$	—
$\langle W_3^1, W_6^2\rangle$	—	—	—	—	$n < N \wedge n < N$	—
$\langle W_4^1, W_1^2\rangle$	—	$n < N \wedge n - 2 > 0$	—	$n < N \wedge n - 4 > 0$	—	—
$\langle W_4^1, W_3^2\rangle$	—	—	—	—	—	$n < N \wedge n - 1 > 0$
$\langle W_5^1, W_1^2\rangle$	—	$n < N \wedge n - 1 > 0$	—	$n < N \wedge n - 3 > 0$	—	—
$\langle W_5^1, W_3^2\rangle$	—	—	—	—	—	$n < N \wedge n > 0$
$\langle W_6^1, W_1^2\rangle$	—	$n < N \wedge n > 0$	—	$n < N \wedge n - 2 > 0$	—	—
$\langle W_6^1, W_3^2\rangle$	—	—	—	—	—	$n < N \wedge n < N$

Tabela 6.9: Supervisor não bloqueante para as máquina com múltiplos modos de operação.

Capítulo 7

Conclusões e trabalhos futuros

Neste documento foi descrito um modelo para SED no qual a planta é composta por um STE e uma coleção de dados. O STE é usado como âncora para a definição dos comportamentos do sistema, os quais são descritos como seqüências de configurações da forma $\xrightarrow{\xi} \langle q_0, \iota_0 \rangle \xrightarrow{\sigma_1} \langle q_1, \iota_1 \rangle \xrightarrow{\sigma_2} \ldots \xrightarrow{\sigma_n} \langle q_n, \iota_n \rangle$. Na configuração $\langle q, \iota \rangle$, q denota um estado discreto do STE e ι, uma instanciação da coleção de dados. A inclusão de uma coleção de dados ao modelo tem a finalidade de fornecer informações para que os requisitos desejáveis do sistema controlado possam ser descritos por fórmulas lógicas. A habilitação dos eventos controláveis condicionada à avaliação de fórmulas confere grande flexibilidade ao processo de supervisão além de permitir a construção de supervisores parametrizados.

7.1 Conclusões

Em seguida, são apresentadas algumas conclusões referentes aos requisitos descritos no Capítulo 1.

1. *A classe dos problemas tratados deve ser igual ou maior do que a classe dos problemas tratados pela teoria clássica.*

   Na Seção 3.3 foi mostrado que toda especificação dada em termos de geradores pode ser transformada em uma especificação por pré-condições. O exemplo da linha de produção simples desenvolvida ao longo dos Capítulos 3 e 4 ilustra um caso cujo comportamento não é regular, o qual não pode ser resolvido (de forma razoável) pela teoria clássica. Portanto, excluindo-se os sistemas contendo laços fechados de eventos não controláveis, a classe dos sistemas tratados pelos STE-PC inclui os problemas tratados pela teoria clássica e outros que, pelo menos

do ponto de vista prático, seriam muito difíceis de serem tratados pela teoria clássica. Neste sentido, este objetivo pode ser considerado parcialmente atingido.

2. *O modelo deve permitir que sistemas complexos sejam construídos a partir de subsistemas mais simples.*

   Nos STE-PC, o STE modelando a planta é obtido pela composição síncrona entre subsistemas mais simples. A coleção de dados acoplada a este STE é uma coleção de dados(monóides de operações) mais simples, projetados isoladamente. Tanto a composição síncrona quanto a construção da coleção de dados, a qual é construída sem a necessidade de operações especiais, podem ser facilmente automatizadas. Isto permite que os sistemas sejam descritos a partir de seus componentes mais simples. Portanto, o objetivo de permitir a construção do sistema a partir de subsistemas mais simples foi plenamente atingido. Entretanto, há de se observar que em comparação com a teoria clássica, nos STE-PC o modelo é mais complexo, uma vez que a coleção de dados e as pré-condições também devem ser representadas.

3. *A linguagem para a descrição dos comportamentos desejáveis deve permitir que especificações complexas possam ser construídas a partir de especificações mais simples.*

   Na Seção 3.2.4 foram definidos dois operadores para a composição de especificações. O operador meet ($\sqcap$) é usado na composição conjuntiva de especificações e o operador join ($\sqcup$) é usado na composição disjuntiva de especificações. A composição conjuntiva $\Lambda_1 \sqcap \Lambda_2$ produz uma especificação satisfazendo tanto os requisitos de Λ_1 quanto os de Λ_2. A composição disjuntiva $\Lambda_1 \sqcup \Lambda_2$ produz uma especificação satisfazendo ou os requisitos de Λ_1 ou os requisitos de Λ_2. Composições envolvendo combinações conjuntivas e disjuntivas simultaneamente, também podem ser usadas. Portanto, este item foi totalmente atingido.

4. *A linguagem para a descrição dos comportamentos desejáveis deve ser intuitiva e flexível.*

   Uma das grandes preocupações quando da proposição dos STE-PC diz respeito à praticidade e facilidade de uso da linguagem de especificação. Nos STE-PC, o projeto de uma especificação é dividida em duas etapas: (a) o projeto da coleção de dados e (b) a descrição dos requisitos em termos de pré-condições. Portanto, a complexidade de interpretação das pré-condições de uma especificação está intimamente relacionada à complexidade de interpretação da coleção de dados (ver Exemplo 6.1). Deste modo, não se pode avaliar a clareza das pré-condições isoladamente sem avaliar a clareza da coleção de dados correspondente. A separação do projeto da

coleção de dados do projeto dos requisitos (aparentemente) facilita a escrita e a interpretação das especificações. No entanto, as questões relativas à praticidade e facilidade da linguagem usada na especificação não podem ser respondidas até que os STE-PC sejam efetivamente aplicados a casos reais.

Com relação à flexibilidade na descrição dos comportamentos desejáveis, observa-se que:

(a) Tantos componentes podem ser adicionados à coleção de dados quantos forem necessários mesmo depois de alguns requisitos já terem sido incorporados na especificação.

(b) Um mesmo componente pode ser descrito de diferentes formas (ver Exemplos 3.1.2 e 3.1.3 – página 42).

(c) Nenhuma restrição foi colocada sobre os conjuntos de instanciações.

(d) A única restrição sobre as operações (funções totais) é que elas sejam computáveis.

Estas observações, os exemplos vistos, a grande flexibilidade na construção da coleção de dados, o grande poder de expressão da lógica de primeira ordem e as diferentes formas de composição de especificações, permitem que se conclua que as especificações são extremamente flexíveis.

5. *A linguagem para a descrição dos comportamentos desejáveis deve permitir a construção de especificações parametrizadas, isto é, classes de especificações.*

   O projeto e a manipulação de classes de especificações constituem uma das principais contribuições proporcionadas pelos STE-PC. Ao longo desta tese, a parametrização das especificações foi explorada em vários exemplos. Portanto, este requisito foi plenamente atingido.

6. *O modelo do sistema e a linguagem para a descrição dos comportamentos desejáveis devem ser expansíveis, isto é, devem permitir que no futuro uma classe maior de problemas possam ser tratados (tal como sistemas temporizados).*

   Supondo-se a existência de um "relógio" capaz de gerar eventos tick, uma variável tempo pode ser adicionada à coleção de dados. Então, pré-condições podem ser facilmente definidas para capturar os limites inferiores e superiores para a ocorrência dos eventos. Conseqüentemente, a possibilidade da extensão dos STE-PC para sistemas temporizados existe e deverá ser explorada no futuro.

   Quanto aos sistemas híbridos, sugere-se adicionar aos estados discretos do STE, dinâmicas contínuas definidas sobre a coleção de dados. Esta inclusão produziria uma representação dos

sistemas semelhante aos autômatos híbridos [ACHH93]. A questão a ser respondida futuramente é se os algoritmos propostos por esta tese podem ser adaptados a este novo modelo.

Em resumo, intuitivamente, percebe-se uma real possibilidade de extensão dos STE-PC tanto para a classe dos sistemas temporizados quanto para a classe dos sistemas híbridos. No entanto, não se pode afirmar, até o presente momento, que tais expansões são viáveis.

7. *Os procedimentos algorítmicos para a síntese de controladores devem tratar especificações parametrizadas. Nestes casos, os resultados produzidos devem ser soluções parametrizadas, isto é, classes de soluções.*

   Os procedimentos algorítmicos vistos independem da especificação ser ou não parametrizada. Caso a especificação seja parametrizada, o supervisor resultante também é parametrizado. Neste caso, a solução encontrada é denominada offline e é válida para um subconjunto de instanciações dos parâmetros. A implementação online [1] do supervisor requer o instanciamento de todos os parâmetros. Este requisito foi parcialmente atingido, uma vez que o procedimento para síntese de supervisores não 'bloqueantes não foi totalmente viabilizado.

8. *As soluções produzidas pelos procedimentos algorítmicos para a síntese de controladores devem ser reduzidas.*

   A implementação online do supervisor requer a representação do sistema de transição de estados, a representação da coleção de dados e a representação das pré-condições associadas aos eventos. Com relação a estas representações, observa-se o seguinte:

   (a) O número de estados discretos do supervisor é no máximo igual ao número de estados da planta.

   (b) O código necessário para a manutenção da coleção de dados depende da complexidade das suas operações.

   (c) As pré-condições offline produzidas pelos algoritmos não são minimizadas (ver exemplo da Seção 6.3).

   Com relação ao item 8a, o número de estados discretos dos STE-PC é, na maioria das vezes (mas nem sempre), consideravelmente menor do que o número de estados de um supervisor equivalente produzido pela teoria clássica.

[1] Uma implementação em um controlador lógico programável.

Com relação ao item 8b, uma escolha cuidadosa da coleção de dados pode reduzir significativamente o tamanho do supervisor. É claro, isto é um problema de engenharia.

Finalmente, com relação ao item 8c, a minimização automática de fórmulas simbólicas constituem um problema até o momento não resolvido.

9. *As soluções produzidas pelos procedimentos algorítmicos para a síntese de controladores devem ser ótimas, isto é, minimamente restritivas.*

   Demonstrado nas Proposições 4.3.3 e 4.3.4.

10. *Os procedimentos algorítmicos para a síntese de controladores devem ser computáveis e computacionalmente eficientes.*

    O Algoritmo 4.3.1 sempre termina e produz o resultado desejado. O Procedimento 5.2.1 pode não convergir e requer a verificação da equivalência entre as especificações, a qual, em geral, é indecidível (Entscheidungsproblem).

7.2 Trabalhos futuros

O futuro dos STE-PC depende basicamente do retorno que a comunidade interessada dará em relação ao trabalho aqui desenvolvido. Não haverá futuro para os STE-PC se os conceitos aqui apresentados não forem divulgados. Portanto, na continuidade, os primeiros e mais importantes trabalhos a serem feitos são publicações. Os retornos destas publicações, naturalmente, poderão (a) contribuir para a melhoria dos conceitos aqui desenvolvidos, (b) identificar limitações teóricas e práticas até aqui não observadas e (c) sugerir possíveis soluções para os problemas ainda não resolvidos.

Com relação à continuidade das pesquisas sobre STE-PC, sugerem-se os seguintes trabalhos futuros:

1. *Construção de um protótipo flexível.* O Algoritmo 2.1.1 e o Procedimento 5.2.1 foram implementados. No entanto, pouca ênfase foi dada às interfaces com os usuários ou ao desempenho dos algoritmos. Estas deficiências dificultam a utilização do software desenvolvido a um conjunto mais amplo de exemplos. Portanto, faz-se necessário o desenvolvimento de um protótipo mais bem elaborado.

2. *Simplificação automática de fórmulas lógicas simbólicas.* A manipulação de fórmulas lógicas simbólicas acarreta um custo computacional bastante elevado no algoritmo e procedimento descritos. Alguns exemplos típicos de fórmulas não simplificadas são:

   (a) $n < N \wedge n+1 > 0 \wedge n > 0 \wedge n-1 > 0 \wedge n-2 > 0 \wedge n-3 > 0 \wedge n-4 > 0$.

   (b) $m \neq 1 \vee c \neq 1 \vee m \neq 1 \wedge m \neq 4 \vee c \neq 1 \wedge (c \neq 2 \vee c \neq 3)$.

   (c) $p/8\,\%\,2 = 1 \wedge ((p-8)\,\%\,2 = 0 \vee p-8 > 0 \wedge p-8 < 8 \wedge (p-8/2\,\%\,2 \neq 1 \vee x) \wedge (p-8/4\,\%\,2 \neq 1 \vee y) \vee \neg x \wedge p-8/2\,\%\,2 = 1 \vee \neg y \wedge p-8/4\,\%\,2 = 1 \vee p-8/8\,\%\,2 = 1)$.

   Para reduzir custos computacionais e facilitar a interpretação dos resultados, fórmulas como as anteriores devem ser reduzidas. Este é um problema ainda em aberto e deve ser melhor explorado.

3. *Verificação automática da equivalência entre especificações.* A síntese de controladores não bloqueantes ainda não foi resolvida. O Procedimento 5.2.1 sugerido requer a verificação automática da equivalência entre as especificações Inv_1 e Inv_2, a qual, no caso da lógica de primeira ordem, é indecidível. Para certas classes da lógica de primeira ordem, entretanto, este problema de decisão pode ser tratado [WJ76, CLP04].

4. *Redução dos controladores.* As soluções produzidas pelos métodos descritos consistem de um sistema de transição de estados, cujas transições são logicamente condicionadas, e uma coleção de dados. O sistema de transição inclui todos os estados da planta. Como a ação do supervisor pode impedir que alguns destes estados sejam alcançados, o sistema de transição pode ser reduzido.

5. *Arquitetura da implementação de supervisores em CLP.* O objetivo final de uma ferramenta automatizada para a síntese de controladores é a geração automática do código a ser executado por um CLP (ou algum outro tipo de dispositivo controlador). A geração deste código depende do dispositivo controlador a ser utilizado e, portanto, difícil de ser generalizada. De qualquer forma, uma arquitetura geral independente de dispositivos controladores pode ser definida.

6. *Tratamento de sistemas temporizados.* Como sugerido na seção anterior, a extensão dos STE-PC à sistemas temporizados e híbridos pode ser explorada.

Apêndice A

Apêndices

A.1 Antecipações de pré-condições – LR

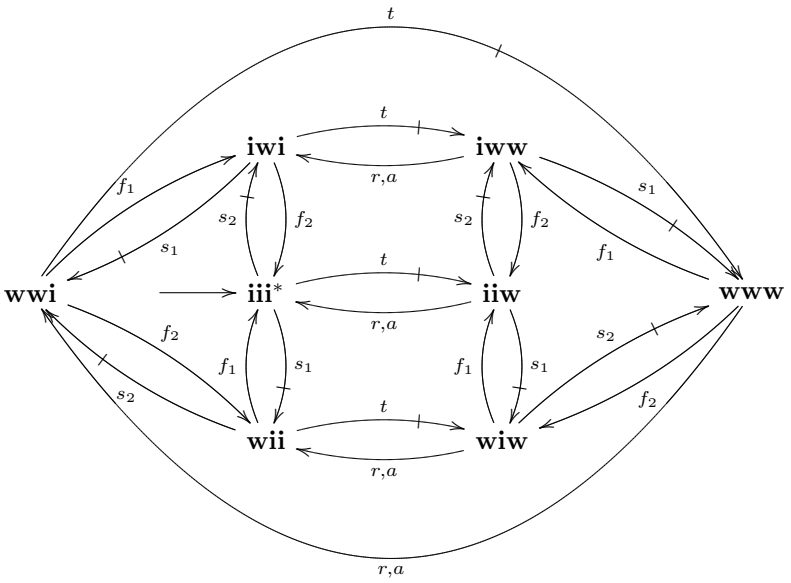

Figura A.1: STE da linha realimentada.

Aplicando-se o Algoritmo 4.3.1 à especificação do Exemplo 3.2.6 para a linha realimentada, são realizadas as seguintes antecipações:
A pré-condição do evento r cuja origem é iiw, é antecipada para os eventos:

- t partindo do estado iii, isto é, $\Lambda(t)\langle iii, \iota\rangle = \Lambda(r)\langle iiw, \Delta^*(iii, t)(\iota)\rangle$

- s_1 partindo do estado iiw, isto é, $\Lambda(s_1)\langle iiw, \iota\rangle = \Lambda(r)\langle iiw, \Delta^*(iiw, s_1 f_1)(\iota)\rangle$
- s_2 partindo do estado iiw, isto é, $\Lambda(s_2)\langle iiw, \iota\rangle = \Lambda(r)\langle iiw, \Delta^*(iiw, s_2 f_2)(\iota)\rangle$
- t partindo do estado iwi, isto é, $\Lambda(t)\langle iwi, \iota\rangle = \Lambda(r)\langle iiw, \Delta^*(iwi, t f_2)(\iota)\rangle$
- s_1 partindo do estado iww, isto é, $\Lambda(s_1)\langle iww, \iota\rangle = \Lambda(r)\langle iiw, \Delta^*(iww, s_1 f_1 f_2 + s_1 f_2 f_1)(\iota)\rangle$
- t partindo do estado wii, isto é, $\Lambda(t)\langle wii, \iota\rangle = \Lambda(r)\langle iiw, \Delta^*(wii, t f_1)(\iota)\rangle$
- s_2 partindo do estado wiw, isto é, $\Lambda(s_2)\langle wiw, \iota\rangle = \Lambda(r)\langle iiw, \Delta^*(wiw, s_2 f_1 f_2 + s_2 f_2 f_1)(\iota)\rangle$
- t partindo do estado wwi, isto é, $\Lambda(t)\langle wwi, \iota\rangle = \Lambda(r)\langle iiw, \Delta^*(wwi, t f_1 f_2 + t f_2 f_1)(\iota)\rangle$

A pré-condição do evento a cuja origem é iiw, é antecipada para os eventos:

- t partindo do estado iii, isto é, $\Lambda(t)\langle iii, \iota\rangle = \Lambda(a)\langle iiw, \Delta^*(iii, t)(\iota)\rangle$
- s_1 partindo do estado iiw, isto é, $\Lambda(s_1)\langle iiw, \iota\rangle = \Lambda(a)\langle iiw, \Delta^*(iiw, s_1 f_1)(\iota)\rangle$
- s_2 partindo do estado iiw, isto é, $\Lambda(s_2)\langle iiw, \iota\rangle = \Lambda(a)\langle iiw, \Delta^*(iiw, s_2 f_2)(\iota)\rangle$
- t partindo do estado iwi, isto é, $\Lambda(t)\langle iwi, \iota\rangle = \Lambda(a)\langle iiw, \Delta^*(iwi, t f_2)(\iota)\rangle$
- s_1 partindo do estado iww, isto é, $\Lambda(s_1)\langle iww, \iota\rangle = \Lambda(a)\langle iiw, \Delta^*(iww, s_1 f_1 f_2 + s_1 f_2 f_1)(\iota)\rangle$
- t partindo do estado wii, isto é, $\Lambda(t)\langle wii, \iota\rangle = \Lambda(a)\langle iiw, \Delta^*(wii, t f_1)(\iota)\rangle$
- s_2 partindo do estado wiw, isto é, $\Lambda(s_2)\langle wiw, \iota\rangle = \Lambda(a)\langle iiw, \Delta^*(wiw, s_2 f_1 f_2 + s_2 f_2 f_1)(\iota)\rangle$
- t partindo do estado wwi, isto é, $\Lambda(t)\langle wwi, \iota\rangle = \Lambda(a)\langle iiw, \Delta^*(wwi, t f_1 f_2 + t f_2 f_1)(\iota)\rangle$

A pré-condição do evento f_2 cuja origem é iwi, é antecipada para os eventos:

- s_2 partindo do estado iii, isto é, $\Lambda(s_2)\langle iii, \iota\rangle = \Lambda(f_2)\langle iwi, \Delta^*(iii, s_2)(\iota)\rangle$
- s_2 partindo do estado iiw, isto é, $\Lambda(s_2)\langle iiw, \iota\rangle = \Lambda(f_2)\langle iwi, \Delta^*(iiw, s_2 r + s_2 a)(\iota)\rangle$
- s_1 partindo do estado iwi, isto é, $\Lambda(s_1)\langle iwi, \iota\rangle = \Lambda(f_2)\langle iwi, \Delta^*(iwi, s_1 f_1)(\iota)\rangle$
- t partindo do estado iwi, isto é, $\Lambda(t)\langle iwi, \iota\rangle = \Lambda(f_2)\langle iwi, \Delta^*(iwi, tr + ta)(\iota)\rangle$

- s_1 partindo do estado iww, isto é, $\Lambda(s_1)\langle iww, \iota\rangle = \Lambda(f_2)\langle iwi, \Delta^*(iww, s_1f_1r + f_1a + s_1rf_1 + s_1af_1)(\iota)\rangle$
- s_2 partindo do estado wii, isto é, $\Lambda(s_2)\langle wii, \iota\rangle = \Lambda(f_2)\langle iwi, \Delta^*(wii, s_2f_1)(\iota)\rangle$
- s_2 partindo do estado wiw, isto é, $\Lambda(s_2)\langle wiw, \iota\rangle = \Lambda(f_2)\langle iwi, \Delta^*(wiw, s_2f_1r + f_1a + s_2rf_1 + s_2af_1)(\iota)\rangle$
- t partindo do estado wwi, isto é, $\Lambda(t)\langle wwi, \iota\rangle = \Lambda(f_2)\langle iwi, \Delta^*(wwi, tf_1r + f_1a + trf_1 + taf_1)(\iota)\rangle$

A pré-condição do evento f_2 cuja origem é iww, é antecipada para os eventos:

- s_2 partindo do estado iiw, isto é, $\Lambda(s_2)\langle iiw, \iota\rangle = \Lambda(f_2)\langle iww, \Delta^*(iiw, s_2)(\iota)\rangle$
- t partindo do estado iwi, isto é, $\Lambda(t)\langle iwi, \iota\rangle = \Lambda(f_2)\langle iww, \Delta^*(iwi, t)(\iota)\rangle$
- s_1 partindo do estado iww, isto é, $\Lambda(s_1)\langle iww, \iota\rangle = \Lambda(f_2)\langle iww, \Delta^*(iww, s_1f_1)(\iota)\rangle$
- s_2 partindo do estado wiw, isto é, $\Lambda(s_2)\langle wiw, \iota\rangle = \Lambda(f_2)\langle iww, \Delta^*(wiw, s_2f_1)(\iota)\rangle$
- t partindo do estado wwi, isto é, $\Lambda(t)\langle wwi, \iota\rangle = \Lambda(f_2)\langle iww, \Delta^*(wwi, tf_1)(\iota)\rangle$

A pré-condição do evento r cuja origem é iww, é antecipada para os eventos:

- s_2 partindo do estado iiw, isto é, $\Lambda(s_2)\langle iiw, \iota\rangle = \Lambda(r)\langle iww, \Delta^*(iiw, s_2)(\iota)\rangle$
- t partindo do estado iwi, isto é, $\Lambda(t)\langle iwi, \iota\rangle = \Lambda(r)\langle iww, \Delta^*(iwi, t)(\iota)\rangle$
- s_1 partindo do estado iww, isto é, $\Lambda(s_1)\langle iww, \iota\rangle = \Lambda(r)\langle iww, \Delta^*(iww, s_1f_1)(\iota)\rangle$
- s_2 partindo do estado wiw, isto é, $\Lambda(s_2)\langle wiw, \iota\rangle = \Lambda(r)\langle iww, \Delta^*(wiw, s_2f_1)(\iota)\rangle$
- t partindo do estado wwi, isto é, $\Lambda(t)\langle wwi, \iota\rangle = \Lambda(r)\langle iww, \Delta^*(wwi, tf_1)(\iota)\rangle$

A pré-condição do evento a cuja origem é iww, é antecipada para os eventos:

- s_2 partindo do estado iiw, isto é, $\Lambda(s_2)\langle iiw, \iota\rangle = \Lambda(a)\langle iww, \Delta^*(iiw, s_2)(\iota)\rangle$
- t partindo do estado iwi, isto é, $\Lambda(t)\langle iwi, \iota\rangle = \Lambda(a)\langle iww, \Delta^*(iwi, t)(\iota)\rangle$
- s_1 partindo do estado iww, isto é, $\Lambda(s_1)\langle iww, \iota\rangle = \Lambda(a)\langle iww, \Delta^*(iww, s_1f_1)(\iota)\rangle$
- s_2 partindo do estado wiw, isto é, $\Lambda(s_2)\langle wiw, \iota\rangle = \Lambda(a)\langle iww, \Delta^*(wiw, s_2f_1)(\iota)\rangle$

- t partindo do estado wwi, isto é, $\Lambda(t)\langle wwi, \iota\rangle = \Lambda(a)\langle iww, \Delta^*(wwi, tf_1)(\iota)\rangle$

A pré-condição do evento f_1 cuja origem é wii, é antecipada para os eventos:

- s_1 partindo do estado iii, isto é, $\Lambda(s_1)\langle iii, \iota\rangle = \Lambda(f_1)\langle wii, \Delta^*(iii, s_1)(\iota)\rangle$
- s_1 partindo do estado iiw, isto é, $\Lambda(s_1)\langle iiw, \iota\rangle = \Lambda(f_1)\langle wii, \Delta^*(iiw, s_1r + s_1a)(\iota)\rangle$
- s_1 partindo do estado iwi, isto é, $\Lambda(s_1)\langle iwi, \iota\rangle = \Lambda(f_1)\langle wii, \Delta^*(iwi, s_1f_2)(\iota)\rangle$
- s_1 partindo do estado iww, isto é, $\Lambda(s_1)\langle iww, \iota\rangle = \Lambda(f_1)\langle wii, \Delta^*(iww, s_1f_2r + f_2a + s_1rf_2 + s_1af_2)(\iota)\rangle$
- s_2 partindo do estado wii, isto é, $\Lambda(s_2)\langle wii, \iota\rangle = \Lambda(f_1)\langle wii, \Delta^*(wii, s_2f_2)(\iota)\rangle$
- t partindo do estado wii, isto é, $\Lambda(t)\langle wii, \iota\rangle = \Lambda(f_1)\langle wii, \Delta^*(wii, tr + ta)(\iota)\rangle$
- s_2 partindo do estado wiw, isto é, $\Lambda(s_2)\langle wiw, \iota\rangle = \Lambda(f_1)\langle wii, \Delta^*(wiw, s_2f_2r + f_2a + s_2rf_2 + s_2af_2)(\iota)\rangle$
- t partindo do estado wwi, isto é, $\Lambda(t)\langle wwi, \iota\rangle = \Lambda(f_1)\langle wii, \Delta^*(wwi, tf_2r + f_2a + trf_2 + taf_2)(\iota)\rangle$

A pré-condição do evento f_1 cuja origem é wiw, é antecipada para os eventos:

- s_1 partindo do estado iiw, isto é, $\Lambda(s_1)\langle iiw, \iota\rangle = \Lambda(f_1)\langle wiw, \Delta^*(iiw, s_1)(\iota)\rangle$
- s_1 partindo do estado iww, isto é, $\Lambda(s_1)\langle iww, \iota\rangle = \Lambda(f_1)\langle wiw, \Delta^*(iww, s_1f_2)(\iota)\rangle$
- t partindo do estado wii, isto é, $\Lambda(t)\langle wii, \iota\rangle = \Lambda(f_1)\langle wiw, \Delta^*(wii, t)(\iota)\rangle$
- s_2 partindo do estado wiw, isto é, $\Lambda(s_2)\langle wiw, \iota\rangle = \Lambda(f_1)\langle wiw, \Delta^*(wiw, s_2f_2)(\iota)\rangle$
- t partindo do estado wwi, isto é, $\Lambda(t)\langle wwi, \iota\rangle = \Lambda(f_1)\langle wiw, \Delta^*(wwi, tf_2)(\iota)\rangle$

A pré-condição do evento r cuja origem é wiw, é antecipada para os eventos:

- s_1 partindo do estado iiw, isto é, $\Lambda(s_1)\langle iiw, \iota\rangle = \Lambda(r)\langle wiw, \Delta^*(iiw, s_1)(\iota)\rangle$
- s_1 partindo do estado iww, isto é, $\Lambda(s_1)\langle iww, \iota\rangle = \Lambda(r)\langle wiw, \Delta^*(iww, s_1f_2)(\iota)\rangle$
- t partindo do estado wii, isto é, $\Lambda(t)\langle wii, \iota\rangle = \Lambda(r)\langle wiw, \Delta^*(wii, t)(\iota)\rangle$
- s_2 partindo do estado wiw, isto é, $\Lambda(s_2)\langle wiw, \iota\rangle = \Lambda(r)\langle wiw, \Delta^*(wiw, s_2f_2)(\iota)\rangle$

- t partindo do estado wwi, isto é, $\Lambda(t)\langle wwi, \iota\rangle = \Lambda(r)\langle wiw, \Delta^*(wwi, tf_2)(\iota)\rangle$

A pré-condição do evento a cuja origem é wiw, é antecipada para os eventos:

- s_1 partindo do estado iiw, isto é, $\Lambda(s_1)\langle iiw, \iota\rangle = \Lambda(a)\langle wiw, \Delta^*(iiw, s_1)(\iota)\rangle$
- s_1 partindo do estado iww, isto é, $\Lambda(s_1)\langle iww, \iota\rangle = \Lambda(a)\langle wiw, \Delta^*(iww, s_1f_2)(\iota)\rangle$
- t partindo do estado wii, isto é, $\Lambda(t)\langle wii, \iota\rangle = \Lambda(a)\langle wiw, \Delta^*(wii, t)(\iota)\rangle$
- s_2 partindo do estado wiw, isto é, $\Lambda(s_2)\langle wiw, \iota\rangle = \Lambda(a)\langle wiw, \Delta^*(wiw, s_2f_2)(\iota)\rangle$
- t partindo do estado wwi, isto é, $\Lambda(t)\langle wwi, \iota\rangle = \Lambda(a)\langle wiw, \Delta^*(wwi, tf_2)(\iota)\rangle$

A pré-condição do evento f_1 cuja origem é wwi, é antecipada para os eventos:

- s_1 partindo do estado iwi, isto é, $\Lambda(s_1)\langle iwi, \iota\rangle = \Lambda(f_1)\langle wwi, \Delta^*(iwi, s_1)(\iota)\rangle$
- s_1 partindo do estado iww, isto é, $\Lambda(s_1)\langle iww, \iota\rangle = \Lambda(f_1)\langle wwi, \Delta^*(iww, s_1r + s_1a)(\iota)\rangle$
- s_2 partindo do estado wii, isto é, $\Lambda(s_2)\langle wii, \iota\rangle = \Lambda(f_1)\langle wwi, \Delta^*(wii, s_2)(\iota)\rangle$
- s_2 partindo do estado wiw, isto é, $\Lambda(s_2)\langle wiw, \iota\rangle = \Lambda(f_1)\langle wwi, \Delta^*(wiw, s_2r + s_2a)(\iota)\rangle$
- t partindo do estado wwi, isto é, $\Lambda(t)\langle wwi, \iota\rangle = \Lambda(f_1)\langle wwi, \Delta^*(wwi, tr + ta)(\iota)\rangle$

A pré-condição do evento f_2 cuja origem é wwi, é antecipada para os eventos:

- s_1 partindo do estado iwi, isto é, $\Lambda(s_1)\langle iwi, \iota\rangle = \Lambda(f_2)\langle wwi, \Delta^*(iwi, s_1)(\iota)\rangle$
- s_1 partindo do estado iww, isto é, $\Lambda(s_1)\langle iww, \iota\rangle = \Lambda(f_2)\langle wwi, \Delta^*(iww, s_1r + s_1a)(\iota)\rangle$
- s_2 partindo do estado wii, isto é, $\Lambda(s_2)\langle wii, \iota\rangle = \Lambda(f_2)\langle wwi, \Delta^*(wii, s_2)(\iota)\rangle$
- s_2 partindo do estado wiw, isto é, $\Lambda(s_2)\langle wiw, \iota\rangle = \Lambda(f_2)\langle wwi, \Delta^*(wiw, s_2r + s_2a)(\iota)\rangle$
- t partindo do estado wwi, isto é, $\Lambda(t)\langle wwi, \iota\rangle = \Lambda(f_2)\langle wwi, \Delta^*(wwi, tr + ta)(\iota)\rangle$

A pré-condição do evento f_1 cuja origem é www, é antecipada para os eventos:

- s_1 partindo do estado iww, isto é, $\Lambda(s_1)\langle iww, \iota\rangle = \Lambda(f_1)\langle www, \Delta^*(iww, s_1)(\iota)\rangle$

- s_2 partindo do estado wiw, isto é, $\Lambda(s_2)\langle wiw, \iota\rangle = \Lambda(f_1)\langle www, \Delta^*(wiw, s_2)(\iota)\rangle$
- t partindo do estado wwi, isto é, $\Lambda(t)\langle wwi, \iota\rangle = \Lambda(f_1)\langle www, \Delta^*(wwi, t)(\iota)\rangle$

A pré-condição do evento f_2 cuja origem é www, é antecipada para os eventos:

- s_1 partindo do estado iww, isto é, $\Lambda(s_1)\langle iww, \iota\rangle = \Lambda(f_2)\langle www, \Delta^*(iww, s_1)(\iota)\rangle$
- s_2 partindo do estado wiw, isto é, $\Lambda(s_2)\langle wiw, \iota\rangle = \Lambda(f_2)\langle www, \Delta^*(wiw, s_2)(\iota)\rangle$
- t partindo do estado wwi, isto é, $\Lambda(t)\langle wwi, \iota\rangle = \Lambda(f_2)\langle www, \Delta^*(wwi, t)(\iota)\rangle$

A pré-condição do evento r cuja origem é www, é antecipada para os eventos:

- s_1 partindo do estado iww, isto é, $\Lambda(s_1)\langle iww, \iota\rangle = \Lambda(r)\langle www, \Delta^*(iww, s_1)(\iota)\rangle$
- s_2 partindo do estado wiw, isto é, $\Lambda(s_2)\langle wiw, \iota\rangle = \Lambda(r)\langle www, \Delta^*(wiw, s_2)(\iota)\rangle$
- t partindo do estado wwi, isto é, $\Lambda(t)\langle wwi, \iota\rangle = \Lambda(r)\langle www, \Delta^*(wwi, t)(\iota)\rangle$

A pré-condição do evento a cuja origem é www, é antecipada para os eventos:

- s_1 partindo do estado iww, isto é, $\Lambda(s_1)\langle iww, \iota\rangle = \Lambda(a)\langle www, \Delta^*(iww, s_1)(\iota)\rangle$
- s_2 partindo do estado wiw, isto é, $\Lambda(s_2)\langle wiw, \iota\rangle = \Lambda(a)\langle www, \Delta^*(wiw, s_2)(\iota)\rangle$
- t partindo do estado wwi, isto é, $\Lambda(t)\langle wwi, \iota\rangle = \Lambda(a)\langle www, \Delta^*(wwi, t)(\iota)\rangle$

Referências Bibliográficas

[ACHH93] Alur, R., C. Courcoubetis, T.A. Henzinger, e P.H. Ho: *Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems.* In Grossman, R.L., A. Nerode, A.P. Ravn, e H. Rischel (editores): *Hybrid Systems*, Lecture Notes in Computer Science, páginas 209–229, New York, 1993. Springer-Verlag. *(Citado na página 130.)*

[AW03] Abdelwahed, S. e W.M. Wonham: *Blocking detection in discrete event systems.* In *Proceeding of the American Control Conference*, páginas 1673–1678, Denver, CO, 2003. *(Citado na página 94.)*

[BL00] Barrett, G. e S. Lafortune: *Decentralized supervisory control with communicating controllers.* IEEE Transactions on Automatic Control, 45(9):1620–1638, 2000. *(Citado na página 21.)*

[Chu36] Church, A.: *An unsolvable problem of elementary number theory.* American Journal of Mathematics, 58:345–363, 1936. *(Citado na página 93.)*

[CL91] Chen, E. e S. Lafortune: *On nonconflicting languages that arise in supervisory control of discrete event systems.* Systems & Control Letters, 17(2):105–113, 1991. *(Citado na página 20.)*

[CL99] Cho, K.-H. e J.-T. Lim: *Mixed centralized/decentralized supervisory control of discrete event dynamic systems.* Automatica, 35:121–128, 1999. *(Citado na página 21.)*

[CL00] Chen, Y.L. e F. Lin: *Modeling of discrete event systems using finite state machines with parameters.* In *Proceedings of the 2000 IEEE – International Conference on Control Applications*, páginas 941–946, Anchorage, Alaska, USA, Setembro 25–27 2000. *(Citado nas páginas 3, 21 e 24.)*

[CL01a] Chen, Y.-L. e F. Lin: *Hierarchical modeling and abstraction of discrete event systems using finite state machines with parameters*. In *Proceedings of the 40th IEEE Conference on Decision and Control*, páginas 4110–4115, 2001. *(Citado na página 3.)*

[CL01b] Chen, Y.-L. e F. Lin: *Safety control of discrete event systems using finite state machines with parameters*. In *Proceedings of the 2001 American Control Conference*, páginas 975–980, 2001. *(Citado nas páginas 3, 25 e 31.)*

[CLL00] Chen, Y.-L., S. Lafortune, e F. Lin: *Design of nonblocking modular supervisors using event priority functions*. IEEE Transactions on Automatic Control, 45(3), Março 2000. *(Citado na página 20.)*

[CLP04] Caferra, R., A. Leitsch, e N. Peltier: *Automated Model Building*, volume 31 de *Applied Logic Series*. Kluwer Academics Publishers, 2004, ISBN 1-4020-2652-8. *(Citado na página 132.)*

[CLR99] Cormen, T.H., C.E. Leiserson, e R.L. Rivest: *Introduction to Algorithms*. MIT Press, 1999. *(Citado na página 91.)*

[CQ94] Cohen, G. e J. Quadrat (editores): *11th International Conference on Analysis and Optimization of Systems: Discrete Event Systems*, volume 199 de *Lecture Notes in Control and Information Sciences*. Springer Verlag, London, 1994. *(Citado na página 3.)*

[CR90] Cassandras, C. e P. J. Ramadge (editores): *IEEE Control Systems Magazine*, volume 10. IEEE press, Piscataway, NJ, Junho 1990. *(Citado na página 3.)*

[dQC00] Queiroz, M.H. de e J.E.R. Cury: *Modular supervisory control of large scale discrete event systems*. In *Proceedings of the Fifth Workshop on Discrete Event Systems (WODES 2000)*, Ghent, Belgium, Agosto 2000. *(Citado na página 20.)*

[dQSC01] Queiroz, M.H. de, E.A.P. Santos, e J.E.R. Cury: *Síntese modular do controle supervisório em diagrama escada para uma célula de manufatura*. In *V Simpósio Brasileiro de Automação Inteligente - V SBAI*, Canela, RS, BRASIL, Novembro, 7–9 2001. *(Citado na página 105.)*

[FK00] Fabian, M. e R. Kumar: *Mutually nonblocking supervisory control of discrete event systems*. Automatica, 36:1863–1869, 2000. *(Citado na página 20.)*

[GR87] Golaszewski, C.H. e P.J. Ramadge: *Control of discrete event processes with forced events.* In *Proceedings of 26th IEEE Conference on Decision and Control*, páginas 247–251, Los Angeles, CA, USA, Dezembro 1987. *(Citado na página 14.)*

[GW00] Gohari, P. e W.M. Wonham: *On the complexity of supervisory control design in the RW framework.* IEEE Trans. on Systems, Man and Cybernetics; Part B: Cybernetics. (Special Issue on Discrete Systems and Control), 30(5):643–652, Outubro 2000. *(Citado na página 21.)*

[HA86] Hilbert, D. e W. Ackermann: *Grundzüge der theoretischen Logik.* In Berka, K. e L. Kreiser (editores): *Logik-Texte: Kommentierte Auswahl zur Geschichte der Modernen Logik (vierte Auflage)*, páginas 117–123. Akademie-Verlag, Berlin, 1986. *(Citado na página 93.)*

[HC83] Ho, Y.-C. e X.-R. Cao: *Perturbation Analysis and Optimization of Queueing Networks.* J. Optimization Theory and Applications, 40:559–582, 1983. *(Citado na página 2.)*

[HMU01] Hopcroft, J.E., R. Motwani, e J.D. Ullman: *Introduction to automata theory, languages, and computation.* Addison-Wesley, Reading, MA, USA, 2001, ISBN 0-201-44124-1. *(Citado nas páginas 6, 20, 41 e 50.)*

[Hoa85] Hoare, C.A.R.: *Communicating sequential processes.* Prentice Hall, Inc, Englewood Cliffs, NJ, 1985. *(Citado na página 12.)*

[Hua91] Huang, N.: *Supervisory control of vector discrete-event systems.* Tese de Mestrado, Department of El. Eng., University of Toronto, Canada, Abril 1991. *(Citado na página 21.)*

[JCK01] Jiang, S., V. Chandra, e R. Kumar: *Decentralized supervisory control of discrete event systems with multiple local specifications.* In *Proceedings of 2001 American Control Conference*, Arlington, VA, Junho 2001. *(Citado na página 21.)*

[JK99] Jiang, S. e R. Kumar: *Decentralized supervisory control of concurrent discrete event systems with partial observations.* In *Proceedings of 1999 Annual Allerton Conference*, Urbana, IL, Setembro 1999. *(Citado na página 21.)*

[KG95] Kumar, R. e V.K. Garg: *Modeling and control of logical discrete event systems.* Kluwer International Series in Engineering and Computer Science. Norwell, MA, USA, Janeiro 1995, ISBN 0-7923-9538-7. *(Citado na página 17.)*

[KS97] Kumar, R. e M. A. Shayman: *Centralized and decentralized supervisory control of nondeterministic systems under partial observation*. SIAM Journal of Control and Optimization, 35(2):363–383, Março 1997. *(Citado na página 21.)*

[Lew92] Lewis, F.L.: *Introduction to Modern Control Theory*, capítulo 1 – Applied Optimal Control and Estimation. Prentice-Hall, 1992. *(Citado na página 1.)*

[Li97] Li, Y.: *On deadlock-free modular supervisory control of discrete-event systems*. IEEE Transactions on Automatic Control, 42(12):1705–1709, 1997. *(Citado nas páginas 20 e 21.)*

[LW88] Lin, F. e W. M. Wonham: *Decentralized supervisory control of discrete event systems*. Information Sciences, 44:199–224, 1988. *(Citado na página 21.)*

[LW89] Li, Y. e W.M. Wonham: *Composition and modular state-feedback control of vector discrete-event systems*. In *Proceedings of the 1989 Conference on Information Sciences and Systems*, páginas 103–110, 1989. *(Citado na página 21.)*

[LW93] Li, Y. e W. M. Wonham: *Control of vector discrete event systems I - the base model*. IEEE Transactions on Automatic Control, 38(8):1214–1227, Agosto 1993. *(Citado na página 21.)*

[LW94] Li, Y. e W. M. Wonham: *Control of vector discrete event systems II - controller synthesis*. IEEE Transactions on Automatic Control, 39(3):512–531, 1994. *(Citado na página 21.)*

[LW95] Li, Y. e W. M. Wonham: *Concurrent vector discrete event systems*. IEEE Transactions on Automatic Control, 40(4):628–638, 1995. *(Citado na página 21.)*

[MB99] MacLane, S. e G. Birkhoff: *Algebra*. American Mathematical Society, Junho 1999, ISBN 0821816462. *(Citado nas páginas 41, 57 e 58.)*

[OCK03a] Oliveira, C., J.E.R. Cury, e C.A.A. Kaestner: *Um modelo parametrizado para sistemas a eventos discretos com comportamentos não regulares (1ª parte)*. In *VI SBAI'2003*, páginas 209–235, Bauru, Brazil, Setembro 2003. *(Citado na página 6.)*

[OCK03b] Oliveira, C., J.E.R. Cury, e C.A.A. Kaestner: *Um modelo parametrizado para sistemas a eventos discretos com comportamentos não regulares (2ª parte)*. In *VI SBAI'2003*, Bauru, Brazil, Setembro 2003. *(Citado na página 6.)*

[OCK04] Oliveira, C., J.E.R. Cury, e C.A.A. Kaestner: *Discrete event systems with guards.* In *Proceedings of the 11th IFAC Symposium on Information Control Problems in Manufacturing*, 2004. *(Citado na página 7.)*

[OCK05] Oliveira, C., J.E.R. Cury, e C.A.A. Kaestner: *Supervisory Control Problem for Parameterized and Non-Regular Discrete Event Systems.* 2005. *(Citado na página 7.)*

[Oga82] Ogata, K.: *Engenharia de Controle Moderno.* Editora Prentice Hall do Brasil Ltda., Rio de Janeiro, RJ, 1982. *(Citado na página 1.)*

[OvS01] Overkamp, A. e J. H. van Schuppen: *Maximal solutions in decentralized supervisory control.* SIAM Journal on Control and Optimization, 39(2):492–511, 2001. *(Citado na página 21.)*

[RW87] Ramadge, P.J. e W.M. Wonham: *Supervisory control of a class of discrete event processes.* In *SIAM Journal of Control and Optimization*, volume 25(1), páginas 206–230, 1987. *(Citado nas páginas 3, 9, 37 e 50.)*

[RW89] Ramadge, P.J. e W.M. Wonham: *The control of discrete-event systems.* Proceedings of the IEEE; Special issue on Dynamics of Discrete Event Systems, 77(1):81–97, Janeiro 1989. *(Citado nas páginas 3, 14 e 37.)*

[RW92] Rudie, K. e W. M. Wonham: *Think globally, act locally: decentralized supervisory control.* IEEE Transactions on Automatic Control, 37(11):1692–1708, Novembro 1992. *(Citado na página 21.)*

[SSK96] Spathopoulos, M. P., R. Smedinga, e P. Kozak (editores): *International Workshop on Discrete Event Systems.* Institute of Electrical Engineers, UK, 1996. *(Citado na página 3.)*

[Tar55] Tarski, A.: *A lattice-theoretical fixpoint theorem and its application.* Pacific Journal of Mathematics, 5:285–309, 1955. *(Citado na página 84.)*

[TMHL97] Thistle, J. G., R. P. Malhame, H.-H. Hoang, e S. Lafortune: *Supervisory control of distributed systems part I: modeling, specification, and synthesis.* IEEE Transactions on Control Systems Technology, 1997. Submitted. *(Citado na página 20.)*

[Tur36] Turing, A.: *On computable numbers, with an application to the Entscheidungsproblem.* In *Proceedings of the London Mathematical Society*, volume 42 de *2*, páginas 230–265, 1936. Errata apareceu na Série 2, 43 (1937), páginas 544–546. *(Citado na página 93.)*

[TW94] Thistle, J.G. e W.M. Wonham: *Supervision of infinite behavior of discrete event systems.* SIAM J. Control Optimization, 32(4):1098–1113, Julho 1994. *(Citado na página 14.)*

[VK88] Varaiya, P. e A. B. Khurzhanski (editores): *Discrete event systems: models and applications*, volume 103 de *Lecture Notes in Control and Information Sciences*. Springer-Verlag New York, LLC, Março 1988, ISBN 0387186662. *(Citado na página 3.)*

[WJ76] W.H. Joiner, Jr.: *Resolution Strategies as Decision Procedures.* J. ACM, 23(3):398–417, 1976, ISSN 0004-5411. *(Citado na página 132.)*

[Won98] Wonham, W.: *Notes on control of discrete-event systems. Course notes for ECE 1636F/1637S*, 1998. *(Citado nas páginas 11, 17, 18 e 61.)*

[WR88] Wonham, W.M. e P.J. Ramadge: *Modular supervisory control of discrete event systems.* Mathematics of Control Signals and Systems, 1(1):13–30, 1988. *(Citado na página 20.)*

[YL00a] Yoo, T. e S. Lafortune: *A generalized framework for decentralized supervisory control of discrete event systems.* In *Proceedings of 2000 International Workshop on Discrete Event Systems*, Ghent, Belgium, 2000. *(Citado na página 21.)*

[YL00b] Yoo, T.S. e S. Lafortune: *New results on decentralized supervisory control of discrete-event systems.* In *Proceedings of 39st IEEE Conference on Decision and Control*, Sidney, Australia, 2000. *(Citado na página 21.)*

[YL02] Yoo, T.-S. e S. Lafortune: *A general architecture for decentralized supervisory control of discrete-event systems.* Discrete Event Dynamic Systems: Theory and Applications, 12(3):335–377, Julho 2002. ID: 406981. *(Citado na página 21.)*

[ZKW99] Zad, S.H., R.H. Kwong, e W.M. Wonham: *Supremum operators and computation of supremal elements in system theory.* SIAM Journal on Control and Optimization, 37(3):695–709, 1999. *(Citado nas páginas 84 e 85.)*

Versão e pacotes LaTeX importantes

Versão	: 10 de outubro de 2005
Editor	: WinEdt 5 Build: 20031209 (v. 5.4)
LaTeX	: Distribuição MikTeX 2.4 (LaTeX 2_ε)
Sistema	: Microsoft® Windows XP®

---

backrefx.sty	: Referência cruzada na bibliografia.
nomentbl.sty	: Confecção da nomenclatura.
babelbib.sty	: Bibliografia em português.
aeguill.sty	: Fonte do texto normal.
amsmath.sty	: Fonte de símbolos matemáticos.
amssymb.sty	: Fonte de símbolos matemáticos.

Autor : Claudio de Oliveira (claudio@ppgia.pucpr.br)

Endereço : Programa de Pós-Graduação em Informática Aplicada – PPGIA
PUCPR Pontifícia Universidade Católica do Paraná
Centro de Ciências Exatas e de Tecnologia - CCET
Bloco 2 – Parque Tecnológico – 2° andar
Rua Imaculada Conceição, 1155 – Prado Velho
80215-901 – Curitiba - PR
Telefone: (41) 3271-1669 / Fax: (41) 3271-2121

Orientador : José Eduardo Ribeiro Cury (cury@lcmi.ufsc.br)

Endereço : Departamento de Automação e Sistemas – CTC
UFSC Universidade Federal de Santa Catarina
Caixa Postal 476
88040-900 – Florianópolis - SC
Florianópolis - SC - Brasil
Fone/Fax: (48) 331-9934

Coorientador : Celso Antônio Alves Kaestner (kaestner@ppgia.pucpr.br)

Endereço : Programa de Pós-Graduação em Informática Aplicada – PPGIA
PUCPR Pontifícia Universidade Católica do Paraná
Centro de Ciências Exatas e de Tecnologia - CCET
Bloco 2 – Parque Tecnológico – 2° andar
Rua Imaculada Conceição, 1155 – Prado Velho
80215-901 – Curitiba - PR
Telefone: (41) 3271-1669 / Fax: (41) 3271-2121