Introdução às Redes de Petri e Aplicações

por

Paulo Romero Martins Maciel
Rafael Dueire Lins
Paulo Roberto Freire Cunha

Departamento de Informática
Universidade Federal de Pernambuco

X ESCOLA DE COMPUTAÇÃO
CAMPINAS-SP, julho de 1996.

Prefácio

Redes de Petri é uma técnica de especificação formal bem estabelecida, largamente difundida e adequada para a modelagem de sistemas que tenham atividades paralelas, concorrentes, assíncronas e não-determinísticas. Vários autores têm mostrado a abrangência e a aplicabilidade das redes de Petri nas mais diversas áreas, tais como: na ciência da computação, engenharias eletrônica e química, administração de empresas etc. No entanto, tem sido mais explorada na ciência da computação e na engenharia eletrônica, desde o estudo para a especificação de sistemas de hardware ou software, avaliação de desempenho, especificação de protocolos de comunicação, diagnóstico de falhas e no projeto de software/hardware, entre outros.

Este texto oferece uma introdução às redes de Petri e aplicações. Inicialmente, são apresentados os diversos tratamentos dados as redes de Petri. De maneira uniforme, apresentamos os conceitos, propriedades e metodologias de análise e aplicações. Um grande número de exemplos de modelagem são utilizados, possibilitando a percepção do poder das redes de Petri e tornando o leitor apto a modelar problemas através desta técnica.

Esta monografia está estruturada em seis capítulos, tendo um capítulo de introdução, onde é apresentado um breve histórico das redes de Petri, situando o leitor nas mais diversas áreas de pesquisa em que redes de Petri vêm sendo utilizadas. São apresentados os conceitos básicos tais como: estrutura, grafo associado, o conceito de marcação, regras de execução e alguns exemplos para ilustrar estes conceitos. O segundo capítulo inicia por apresentar os modelos básicos de redes de Petri que possibilitam a modelagem de sistemas mais complexos. Posteriormente, é apresentado um vasto número de exemplos, que ilustram ao leitor o poder de modelagem das redes de Petri através da especificação de problemas clássicos e novos exemplos descritos pelos autores. O terceiro capítulo discute as propriedades das re-

des de Petri, bem como diversos métodos de análise para verificação destas propriedades. No quarto capítulo, apresentam-se algumas extensões às redes de Petri. Estas extensões elevam o poder de modelagem das redes de Petri ao da máquina de *Turing*. Vêem-se, neste capítulo, as redes de Petri com arco inibidor, redes de Petri coloridas, redes hierárquicas e redes de Petri temporizadas determinísticas. No quinto capítulo apresentamos duas aplicações das redes de Petri, na primeira tratamos da tradução de programa escritos em linguagem concorrente para as redes de Petri; a segunda, apresenta a modelagem e diagnóstico de falhas em sistemas de controle industrial. No capítulo das considerações finais fazemos um balanço das contribuições desta monografia e apresentamos algumas linhas de pesquisas atuais. Três apêndices também compõem este trabalho. No primeiro deles é apresentada uma breve introdução a teoria *bag*, aqui utilizada na formalização das redes de Petri. O apêndice B revisa os aspectos mais importantes da álgebra matricial. Visando tornar o conteúdo do material apresentado mais agradável ao leitor, as provas dos teoremas fundamentais da teoria das redes de Petri foram omitidas dos capítulos e agrupadas no apêndice C.

Agradecimentos

Em primeiro lugar gostaríamos de expressar a nossa gratidão às nossas famílias pelo apóio e compreensão durante todas as etapas do desenvolvimento desta monografia.

Este trabalho foi enriquecido com os comentários de David Simonetti Barbalho e Grzegorz Rozenberg e pela revisão gramatical do professor Gilberto Medeiros.

Paulo Maciel agradece a Universidade de Pernambuco (UPE) a sua liberação para o programa de doutoramento na UFPE com suporte financeiro da CAPES.

Rafael Lins e Paulo Cunha são financiados pelo CNPq através de bolsas individuais de pesquisa.

Os autores

Conteúdo

1 Introdução **1**

1.1 Estrutura das Redes de Petri 6

1.1.1 Estrutura Definida em *Bag* 6

1.1.2 Estrutura Definida em Matriz 9

1.1.3 Estrutura Definida por Relações 10

1.2 Rede de Petri Marcada 12

1.3 Regras de Execução 14

1.4 Matriz de Incidência 17

1.5 Rede Pura e *Self-Loop* 18

1.6 Seqüências Disparáveis 20

1.7 Função de Próximo Estado 20

1.8 Equação Fundamental 21

1.9 Grafo das Marcações Acessíveis 22

1.10 Condições Externas 25

1.11 Associação de Rótulos a Transições 26

1.12 Linguagens das Redes de Petri 27

1.13 Classes das Redes de Petri 29

1.14 Redes de Petri Ordinárias 29

1.15 Redes de Alto Nível 31

2 Modelagem com Redes de Petri e Sub-Classes **33**

2.1 Redes Elementares 33
2.1.1 Seqüenciamento 33
2.1.2 Distribuição 34
2.1.3 Junção 34
2.1.4 Escolha Não-Determinística 35
2.1.5 Atribuição 37
2.2 *Confusion* 37
2.3 Modelagem de Processos 38
2.3.1 Processos Paralelos 38
2.3.2 Exclusão Mútua 40
2.3.3 Máquinas de Estado Finito 42
2.3.4 Computação Dataflow 45
2.3.5 Sistemas Pipeline 47
2.3.6 Protocolo de Comunicação 49
2.3.7 Produtor/Consumidor 52
2.3.8 Jantar dos Filósofos 54
2.3.9 Fluxo de Controle de Algoritmo 55
2.3.10 Computação-Fraca 60
2.4 Subclasses das Redes de Petri 60
2.4.1 Redes de Petri Máquina de Estado 62
2.4.2 Redes de Petri Grafo Marcado 62
2.4.3 Redes de Petri Escolha Livre 63

3 Análise das Redes de Petri **67**

3.1 Propriedades Comportamentais 67
3.1.1 Alcançabilidade (*Reachability*) 68
3.1.2 Limitação (*Boundedness*) 69
3.1.3 Segurança (*Safeness*) 70
3.1.4 *Liveness* 74

3.1.5 Cobertura (*Coverability*) 76
3.1.6 Persistência 76
3.1.7 Reversibilidade 77
3.1.8 Justiça (*Fairness*) 78
3.2 Propriedades Estruturais 80
3.2.1 Limitação Estrutural 81
3.2.2 Conservação 81
3.2.3 Repetitividade 85
3.2.4 Consistência 86
3.3 Análise das Redes de Petri 87
3.3.1 Árvore de Cobertura 87
3.3.2 Análise Sobre a *EFRP* 92
3.3.3 Invariantes de Transição 94
3.3.4 Invariantes de Lugar 95
3.3.5 Invariantes por Justaposição e por Concatenação . . . 97
3.3.6 Regras de Redução para Análise 104
3.4 Complexidade e Decidibilidade 108

4 Extensões às Redes de Petri **111**
4.1 Rede de Petri com Arco Inibidor 111
4.2 Rede de Petri Colorida 113
4.3 Redes Hierárquicas 121
4.4 Rede de Petri Temporizada Determinística 125

5 Aplicações **131**
5.1 Tradução *OCCAM*-Redes de Petri 132
5.1.1 Modelagem dos Processos Primitivos 133
5.1.2 Modelagem dos Combinadores 137
5.2 Diagnóstico de Falhas em Sistemas de Controle Industrial . . 148
5.2.1 As Especificações Funcionais 150

5.2.2 As Funções de Estado e Excitação 150
5.2.3 As Regras Operacionais 152
5.2.4 Obtenção das Redes de Petri 152
5.2.5 Um Exemplo 153
5.2.6 Método para Obtenção das Regras Operacionais . . . 154
5.2.7 Modelagem e Validação 158
5.2.8 Elemento de Falha de Vetor Sintoma 159
5.3 Algumas Considerações 165

6 Considerações Finais **167**

A Teoria Bag **179**

B Álgebra Matricial **181**

C Teoremas **185**

Lista de Figuras

1.1 Elementos Básicos 3
1.2 Grafo 3
1.3 Arcos Valorados 4
1.4 Períodos do Dia 4
1.5 Linha de Produção 5
1.6 Redes Marcadas $R_1 e$ R_2 13
1.7 Disparo de Transições 15
1.8 Rede $R_1 Marcada$ 16
1.9 Transição Fonte (*Source*) antes e após o Disparo 17
1.10 Transição de Absorção (*"Sink"*) antes e após o Disparo 17
1.11 Transformação de uma Rede Impura em Pura 19
1.12 Rede N_1 22
1.13 Grafo das Marcações Acessíveis 23
1.14 Grafo das Marcações Acessíveis Infinito 24
1.15 Condições Externas Associadas às Transições 26
1.16 Rede Etiquetada 27
1.17 Rede Rotulada 28
1.18 Rede Binária e não-Ponderada 30

2.1 Seqüenciamento 34
2.2 Distribuição 34
2.3 Junção 35

2.4 Escolha Não-Determinística 36
2.5 Conflito Estrutural e Efetivo 36
2.6 Atribuição 37
2.7 *Confusion* Simétrica e Assimétrica 38
2.8 Atividades Paralelas 40
2.9 Exclusão Mútua 41
2.10 Interface usando Lugares 43
2.11 Máquina de Estado - Detector de Seqüência 44
2.12 Detector de Seqüência 45
2.13 Computação *Dataflow* 46
2.14 Unidades Funcionais em *Pipeline* 48
2.15 *Pipeline* 48
2.16 Comunicação entre Processos com Reconhecimento e por *Buffer* 50
2.17 Comunicação entre Processos 50
2.18 Protocolo de Comunicação 51
2.19 Produtor/Consumidor 52
2.20 Produtor/Consumidor com Buffer Limitado 53
2.21 O Jantar dos Filósofos 55
2.22 Computação Simples 56
2.23 Fusão dos Lugares 57
2.24 If a then B else C 57
2.25 While a do B 58
2.26 For a n do B 59
2.27 Computação-Fraca da Adição 61
2.28 Computação-Fraca da Multiplicação 61
2.29 Subclasse *Máquina de Estado* 63
2.30 Subclasse *Grafo de Eventos* 64
2.31 Subclasse *Escolha Livre* 65

3.1 Rede N 69
3.2 Rede Limitada (*Bounded*) 70
3.3 Rede Segura (*Safe*) 71
3.4 Transformação de Rede Não-Segura(*Unsafe*) em Segura(*Safe*) 72
3.5 Rede Não-Segura 73
3.6 Rede Segura 73
3.7 Redes *Live* e *Non-Live* 74
3.8 Níveis de *Liveness* 75
3.9 Persistência 77
3.10 Reversibilidade 79
3.11 *Bounded Fairness* 79
3.12 *Fairness Incondicional* 80
3.13 Estritamente Conservativa 82
3.14 Conservação 83
3.15 Rede não Conservativa 84
3.16 Repetitividade 85
3.17 Árvore de Cobertura 88
3.18 Grafo de Cobertura 89
3.19 Árvore de Cobertura - Verificação de Propriedades 90
3.20 Grafo de Cobertura - Verificação de Propriedades 90
3.21 Problemas com a Alcançabilidade 91
3.22 Problemas com *Liveness* 91
3.23 Alcançabilidade de uma Marcação 93
3.24 Não-Alcançabilidade 94
3.25 Fusão de Elementos 98
3.26 Sub-redes $R_1 e \mathrm{R}_2$ 101
3.27 Rede R 102
3.28 Reduções - Fusão de Lugares Série 105
3.29 Reduções - Fusão de Transições Série 106

3.30 Reduções - Fusão de Lugares Paralelos 106
3.31 Reduções - Fusão de Transições Paralelas 107
3.32 Reduções - Eliminação de Self-loop 107

4.1 Teste de Zero Marcas em Lugares Limitados 112
4.2 Teste de Zero Marcas 113
4.3 Redes Coloridas - Estudos Iniciais 114
4.4 Modelo da Linha de Manufatura Usando Redes Ordinárias . . 115
4.5 Modelo da Linha de Manufatura Usando Redes Coloridas . . 117
4.6 Jantar dos Filósofos - Modelo com Redes *Place/Transition* . . 118
4.7 Jantar dos Filósofos - Modelo com Redes Coloridas 119
4.8 Superpágina 122
4.9 Subpágina 123
4.10 Superpágina - Aquisição e Transferência de Informações . . . 124
4.11 Subpáginas - Aquisição e Transferência de Informações 124
4.12 Transição com Disparo Atômico 126
4.13 Marcação Habilitando Duas Transições 127
4.14 Sistema de Contagem 128
4.15 Relógio 128
4.16 Modelo 129

5.1 Atribuição 134
5.2 *Input* 134
5.3 *Output* 135
5.4 *Skip* 136
5.5 *Stop* 137
5.6 *Sequence* 138
5.7 *Conditional* 139
5.8 *Loop* 140
5.9 *Parallel* 141

5.10 *Alternation* 142
5.11 *Alternation* - Regulador de Fluxo 144
5.12 Rede dos Processos Comunicantes 146
5.13 Processo P_1 147
5.14 Processo P_3 147
5.15 Página K_3 148
5.16 Processo P_4 149
5.17 Página K_5 149
5.18 Sistema de Controle Seqüencial 150
5.19 Linha de Produção de Bebidas 154
5.20 Diagrama Elétrico 154
5.21 Modelos Elementares 159
5.22 Modelo Global 160
5.23 Estado $S0$ 163
5.24 Estado Se 163
5.25 Sintoma 164
5.26 Desabilitação de *l3* 165

Capítulo 1

Introdução

A teoria inicial das redes de Petri foi apresentada na tese de doutoramento *Kommunikation mit Automaten* do Dr. C. A. Petri em 1962 na faculdade de Matemática e Física da Universidade de Darmstadt na então Alemanha Ocidental. O trabalho de Petri atraiu a atenção de A.W Holt que, em conjunto com outros pesquisadores, desenvolveu muito da teoria, notação e representação das redes de Petri. De 1970 a 1975, o grupo de estrutura da computação do MIT foi o mais ativo na na condução da pesquisa sobre redes de Petri. Em 1975, houve uma Conferência sobre redes de Petri, no entanto não houve publicação dos anais [19]. Em 1979, pesquisadores de vários países europeus reuniram-se em Hamburgo para um curso avançado sobre a *Teoria Geral das Redes de Processos e Sistemas* [54]. Seguiram-se diversos outros trabalhos propondo alterações ao modelo original, tais como redes com arco inibidor, e redes temporizadas determinísticas e estocásticas. Hoje em dia, redes de Petri é considerada uma técnica para especificação de sistemas concorrentes consolidada. Grupos de pesquisa em todo o mundo têm redes de Petri como tema, desenvolvendo estudos sobre seus aspectos teóricos e suas aplicações.

Diversas técnicas de modelagem matemática de sistemas em diversas áreas da ciência têm sido propostas. Barroca e McDermid [50] apresenta a seguinte classificação:

1. Técnicas Baseadas em Modelos:

   Fornece uma descrição abstrata explícita sobre estados e operações que transformam os estados, no entanto não oferece meios explícitos para

especificar concorrência. Ex.: Z [55].

2. Técnicas Baseadas em Álgebra de Processos.

   Estas técnicas fornecem meios explícitos para especificar concorrência. O comportamento dos processos é representado através de comunicações observáveis. Ex.: CCS [49], CSP [56] e LOTOS [57, 86].

3. Técnicas Baseadas em Lógica.

   Uma grande variedade de técnicas baseadas em lógica tem sido propostas, onde se analisam as relações causais e aspectos relacionados à temporização. Ex.: Lógica Modal de Ações [58].

4. Técnicas Baseadas em Redes.

   Estas técnicas modelam concorrência, através de mecanismos implícitos de fluxo de *tokens* na rede. Este fluxo é controlado por condições que habilitam a realização de tarefas (eventos). Ex.: redes de Petri.

Redes de Petri é uma técnica de especificação de sistemas que possibilita uma representação matemática e possui mecanismos de análise poderosos, que permitem a verificação de propriedades e a verificação da *corretude* do sistema especificado. Usando-se redes de Petri pode-se modelar sistemas paralelos, concorrentes, assíncronos e não-determinísticos [3]. Algumas abordagens matemáticas têm sido propostas; entre estas uma que trata as redes de Petri sob o ponto de vista da álgebra matricial [1, 2], outra conforme a teoria *bag* [4] e uma terceira baseada em relações. Neste trabalho, definiremos e apresentaremos os principais conceitos das redes de Petri sob esses pontos de vista, mostrando quando cada um desses tratamentos é mais conveniente. Neste trabalho, apresentamos um grande número de modelos de problemas clássicos, bem como modelagem, propriedades, métodos de análise úteis na verificação dessas propriedades e extensões às redes de Petri.

A representação gráfica das redes de Petri(ver figura 1.2) tem se mostrado muito útil, pois permite a visualização dos processos e a comunicação entre eles. As redes de Petri são formadas por dois tipos de componentes: um ativo denominado de *transição* e outro passivo denominado *lugar*. Os lugares correspondem às variáveis de estado e as transições às ações (eventos) realizadas pelo sistemas. A realização de uma ação está associada a algumas pré-condições (condição de alguma variáveis de estado), ou seja, existe uma relação entre os lugares e as transições, que possibilita a realização de uma ação. De forma semelhante, após a realização de uma ação, alguns

lugares terão suas informações alteradas (pós-condições). Graficamente, os lugares são representados por círculos e as transições por traços ou barras (ver figura 1.1).

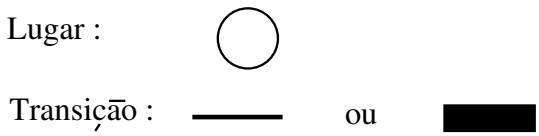

Figura 1.1: Elementos Básicos

Estes dois elementos são os vértices do grafo associado às redes de Petri. Os vértices são interligados por arcos dirigidos. Os arcos que interligam lugares às transições correspondem à relação entre as condições verdadeiras, que em um dado momento, possibilitam a execução das ações, enquanto os arcos que interligam transições aos lugares representam a relação entre as ações e as condições que se tornam verdadeiras com a execução das ações.

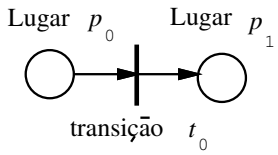

Figura 1.2: Grafo

Os vértices de uma grafo associada a uma rede de Petri podem ser interligados por múltiplos arcos. Por exemplo, um lugar pode ser conectado a uma transição através de diversos arcos ou vice-versa. Por conveniência, podemos substituir os múltiplos arcos por um único arco valorado, onde o numeral associado ao arco corresponde ao número de arcos que interligam os vértices (ver figura 1.3).

Exemplo 1

Ilustramos os conceitos informalmente apresentados descrevendo o ciclo repetitivo dos turnos (períodos) de um dia. Dividamos o dia em três

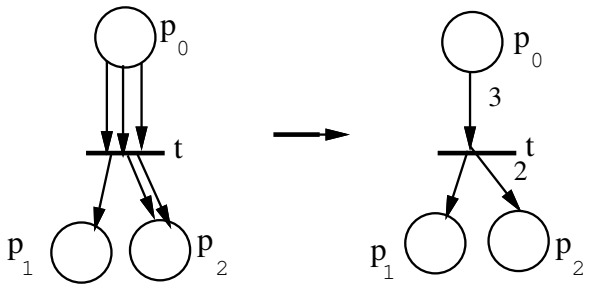

Figura 1.3: Arcos Valorados

períodos: *manhã*, *tarde* e *noite*, ou seja, há três condições. A transição de uma dessas condições para uma outra, por exemplo - amanhecer (*noite* $\rightarrow$ *manhã*), são os eventos. O modelo que representa o ciclo operacional desse sistema é formado pelas três condições, representadas por três variáveis de estado (lugares), e por três eventos (transições): *amanhecer*, *entardecer* e *anoitecer*. Para representar a situação atual, ou seja, em que condição encontra-se o sistema modelado, usamos uma marca grafada (um ponto) no lugar que corresponde a essa situação, por exemplo: a condição atual é *manhã*. Na figura 1.4.a temos o modelo que representa este sistema e a sua situação atual.

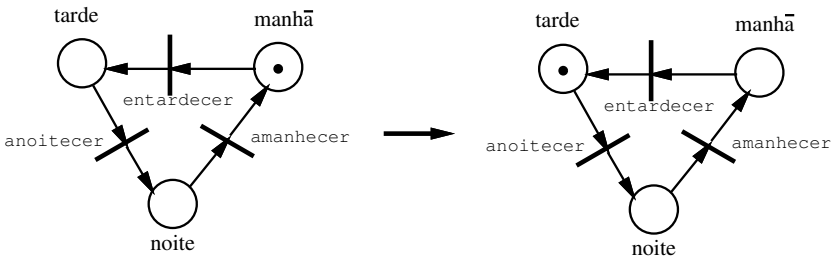

Figura 1.4: Períodos do Dia

Nesse modelo, temos a condição atual representada pela marca no lugar *manhã*. Estando nesta condição o único evento que poderá ocorrer é *entardecer*, que é representado pela transição de mesmo nome. Na ocorrência desse

evento, teremos uma nova situação atual, ou seja: *tarde*, que é representada graficamente, na figura 1.4.b, por uma marca no lugar *tarde*.

Exemplo 2

A seguir, apresentamos um outro modelo em redes de Petri que representa algumas das tarefas de uma linha de montagem. Neste modelo, descrevemos o empacotamento e envio ao setor de expedição de conjuntos de *parafusos* e *porcas*, contudo, não representamos as tarefas da manufatura dos parafusos e porcas. Este sistema forma conjuntos de três parafusos e três porcas. Um novo conjunto só poderá ser empacotado quando o conjunto atualmente produzido for enviado à fase seguinte do processo. Após os pacotes (conjuntos) serem formados, estes são enviados ao setor de expedição. A figura 1.5 apresenta o modelo que representa esta linha de montagem.

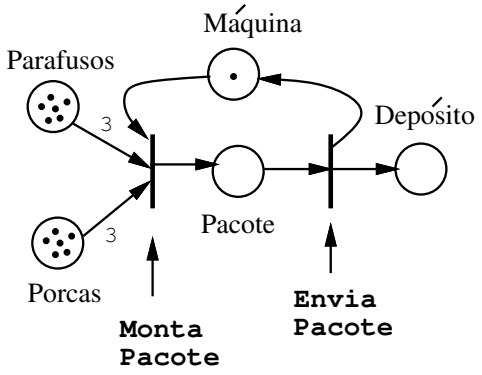

Figura 1.5: Linha de Produção

Após a manufatura dos parafusos, estes são depositados no lugar *Parafusos*. De forma semelhante, as porcas são armazenadas no lugar *Porcas*. A transição *Monta Pacote*, que representa a tarefa de formação dos conjuntos (porcas e parafusos), só estará apta a ser disparada quando houver no mínimo três parafusos e três porcas nos lugares *Parafusos* e *Porcas*, respectivamente. Sendo satisfeita esta condição, o disparo de *Monta Pacote* retira três marcas do lugar *Parafusos* e três marcas do lugar *Porcas* e deposita uma marca no lugar *Pacote*, que indica a montagem de um conjunto. Tendo sido produzido um pacote (marca no lugar *Pacote*), é possível colocarmos este

pacote no depósito de saída da seção (disparando-se *Envia Pacote*), para que esteja disponível ao setor de expedição. O pacote, ao ser colocado no depósito, possibilita à *Máquina* montar um outro conjunto, desde que haja parafusos e porcas suficientes, nos respectivos depósitos de entrada.

A seguir, são apresentados os conceitos básicos das redes de Petri, sua estrutura, o grafo associado, o conceitos de marcação e suas regras de execução. São apresentados, ainda, exemplos de modelagem de aplicações; no entanto, exemplos mais elaborados serão descritos no próximo capítulo e no capítulo 5.

1.1 Estrutura das Redes de Petri

Nesta seção, serão apresentadas três definições formais para as redes de Petri. Uma sobre o ponto de vista da teoria *bag* [4], que apresenta mapeamentos das transições para os *bags* de lugares. Um *bag* é uma generalização do conceito de conjunto que admite a repetição de elementos. Uma introdução sumária à teoria *bag* é apresentada no apêndice 1. Uma segunda definição usa a notação matricial [1]. E a terceira, define as redes de Petri através de relações e pesos associados a estas relações [5].

1.1.1 Estrutura Definida em *Bag*

A estrutura das redes de Petri, segundo a teoria *bag* [4] é composta por cinco partes: o conjunto de lugares **P**, o conjunto de transições **T**, o *bag* de entrada **I**, o *bag* de saída **O** e a capacidade associada a cada lugar. Para cada transição existe uma função de entrada, que é um mapeamento de uma transição t_j em um *bag* de lugares It_j. De forma semelhante, as funções de saída mapeiam uma transição t_j em um *bag* de lugares Ot_j. Para denotarmos conjuntos, utilizamos - { } e para os *bags* - [ ].

Definição 1.1.1 - Estrutura das Redes de Petri em bag: *define-se a estrutura de uma rede de Petri* **R**, *como uma quíntupla* $R = (P, T, I, O, K)$, *onde* $P = \{p_1, p_2, ..., p_n\}$ *é um conjunto finito não-vazio de lugares,* $T = \{t_1, t_2, ..., t_m\}$ *é um conjunto finito não-vazio de transições.* $I : T \rightarrow P^{\infty}$ *é um conjunto de bags que representa o mapeamento de transições para lugares de entrada.* $O : T \rightarrow P^{\infty}$ *é um conjunto de bags que representa o mapeamento de transições para lugares de saída.* $K : P \rightarrow N \bigcup \{\omega\}$, *é o conjunto*

das capacidade associadas a cada lugar, podendo assumir um valor infinito.

Exemplo 1.1

A rede dos *períodos do dia* (ver figura 1.4) é formada por três lugares e três transições, que representam respectivamente as variáveis de estados e as ações realizadas pelo sistema. As pré-condições para a realização das ações são representadas pelos *bags* $I(t_i)$, ou seja, para que o $Amanhecer$ possa acontecer, é necessário que a condição $Noite$ seja verdadeira, dado que $I(Amanhecer) = \{Noite\}$. Os lugares que são pós-condições das transições t_i são representados pelos *bags* $O(t_i)$. A transição $Amanhecer$ tem como lugar de saída $O(Amanhecer) = \{Manhã\}$.

$R_{Periodos\_do\_Dia} = (P, T, I, O, K)$

onde

o conjunto de lugares P é

$P = \{Manhã, Tarde, Noite\}$,

o conjunto de transições T é

$T = \{Amanhecer, Entardecer, Anoitecer\}$,

o conjunto de *bags* de entrada I é

$$I = \{\ I(Amanhecer) = [Noite],\quad I(Entardecer) = [Manhã],$$
$$I(Anoitecer) = [Tarde],\ \},$$

o conjunto de *bags* de saída O é

$$O = \{\ O(Amanhecer) = [Manhã],\quad O(Entardecer) = [Tarde],$$
$$O(Anoitecer) = [Noite]\ \},$$

e o conjunto de capacidades dos lugares k é

$k = \{k_{Manhã} = 1, k_{tarde} = 1, k_{Noite} = 1\}$.

No exemplo acima, não está caracterizada a necessidade do uso da teoria *bag*, pois nenhuma transição tem como entrada ou saída mais de um lugar.

Exemplo 2.1

A rede do exemplo *Linha de Produção*(ver figura 1.5) possui estrutura que faz uso da teoria *bag*:

$R_{Linha\_de\_Produção} = (P, T, I, O, K)$

onde

o conjunto de lugares P é

$P = \{Parafusos, Porcas, Pacote,\ Máquina,\ Depósito\ \}$,

o conjunto de transições T é

$T = \{MontaPacote, EnviaPacote\}$,

o conjunto de *bags* de entrada I é

$$I = \{\ I(MontaPacote),\quad I(EnviaPacote)\ \},$$

$$I(MontaPacote) = [\ Parafusos,\quad Parafusos,\quad Parafusos,$$
$$Porcas,\quad Porcas,$$
$$Porcas,\quad Máquina\ ],$$

$$I(EnviaPacote) = [\ Pacote\ ],$$

o conjunto de *bags* de saída O é:

$$O = \{O(MontaPacote), O(EnviaPacote)$$

$$O(MontaPacote) = [Pacote],$$

$$O(EnviaPacote) = [Máquina, Depósito],$$

e o conjunto de capacidades dos lugares k é:

$K = \{k_{Parafusos} = \omega, k_{Porcas} = \omega, k_{Pacotes} = 1,$
$k_{Máquina} = 1, k_{Depósito} = \omega\}$.

Observe-se que os *bags* de entrada da transição $Monta\_Pacote$ possui três elementos *Parafusos* e três elementos *Porcas*.

1.1.2 Estrutura Definida em Matriz

Devido aos resultados existentes nos estudos da álgebra matricial, as redes de Petri utilizam este ferramental para a formalização de sua teoria, possibilitando a análise de propriedades comportamentais e estrutrais, que serão adequadamente apresentadas no capítulo 3.

Segundo o ponto de vista matricial, a estrutura das redes de Petri é representada por uma quíntupla formada pelo conjunto de lugares, o conjunto de transições, a matriz de entrada das transições, a matriz de saída das transições e a capacidade de cada lugar, formalmente representado a seguir:

Definição 1.1.2 - Estrutura das Redes de Petri em Matrizes*: a estrutura de uma rede é uma quíntupla* $R = (P, T, I, O, K)$*, onde* P *é um conjunto finito de lugares.* T *é um conjunto finito de transições.* $I : P \times T \rightarrow \mathbb{N}$ *é a matriz de pré-condições.* $O : P \times T \rightarrow \mathbb{N}$ *é a matriz de pós-condições.* K *é o vetor das capacidades associadas aos lugares (*$K : P \rightarrow \mathbb{N} \bigcup \{\omega\}$*).*

Se o conjunto de lugares ou o conjunto de transições é vazio, a rede é dita degenerada.

Exemplo 1.2

Aqui apresentamos a estrutura da rede do exemplo *períodos do dia*, segundo a representação matricial. O conjunto de lugares e transições são representados respectivamente por P e T. As matrizes I e O representam as pré-condições e as pós-condições, respectivamente, de todas as transições das rede. A coluna *Amanhecer* da matriz I mostra que o lugar *Noite* é pré-condição de *Amanhecer*. De forma semelhante, observamos na coluna *Amanhecer* da matriz O que o lugar *Manhã* é pós-condição da transição *Amanhecer*. Estas duas matrizes representam a estrutura do modelo.

$$I = \left|\begin{array}{ccc|c} Amanhecer & Entardecer & Anoitecer & \\ 0 & 1 & 0 & Manhã \\ 0 & 0 & 1 & Tarde \\ 1 & 0 & 0 & Noite \end{array}\right|$$

$$O = \left|\begin{array}{ccc|c} Amanhecer & Entardecer & Anoitecer & \\ 1 & 0 & 0 & Manhã \\ 0 & 1 & 0 & Tarde \\ 0 & 0 & 1 & Noite \end{array}\right|$$

Exemplo 2.2

Apresentamos as matrizes de entrada (I) e de saída (O) do exemplo da montagem de conjunto de parafusos.Observamos a coluna $Monta\_Pacote$, na qual temos representados os lugares de entrada desta transição pelos respectivos pesos dos arcos, por exemplo, o arco que liga o lugar parafuso à esta transição tem peso três, sendo representado no componente $I(0,0) = 3$.

$$I = \left| \begin{array}{cc|l} MontaPacote & EviaPacote & \\ 3 & 0 & Parafusos \\ 3 & 0 & Porcas \\ 0 & 1 & Pacotes \\ 1 & 0 & Máquinas \\ 0 & 0 & Depósitos \end{array} \right.$$

$$O = \left| \begin{array}{cc|l} MontaPacote & EnviaPacote & \\ 0 & 0 & Parafusos \\ 0 & 0 & Porcas \\ 1 & 0 & Pacotes \\ 0 & 1 & Máquinas \\ 0 & 1 & Depósitos \end{array} \right.$$

1.1.3 Estrutura Definida por Relações

É possível definirmos a estrutura das redes de Petri usando relações [19]. A estrutura é formada por uma quíntupla *(P, T, A, V, K)*, composta pelo conjunto de lugares P, o conjunto de transições T, o conjunto dos arcos que interligam lugares às transições ou transições aos lugares, a valoração ou peso dos arcos representada por V e o conjunto das capacidades dos lugares K. Formalmente esta definição é apresentada a seguir:

Definição 1.1.3 - Estrutura das Redes de Petri por Relações*: define-se a estrutura das redes de Petri por uma quíntupla* (P, T, A, V, K)*, onde* T *é o conjunto de transições,* P *o conjunto de lugares,* A *o conjunto de arcos e* V *o conjunto de valorações dos arcos. Os elementos de* A *são arcos que conectam transições a lugares ou lugares a transições (*$A \subseteq (P \times T) \bigcup (T \times P)$*).*

Comos vimos na definição acima, podemos dividir os elementos de A em dois sub-conjuntos - o conjunto das entradas às transições e de saída das transições, $I = \{(p_i, t_j)\}$ e $O = \{(t_j, p_i)\}$, respectivamente.

Exemplo 1.3

A estrutura da rede de Petri do exemplo períodos do dia e da linha de produção de parafusos é

$R_{Periodos\_do\_Dia} = (P, T, I, O, K)$, onde

o conjunto de lugares P é:

$P = \{\ Manh\tilde{a}, Tarde, Noite\}$,

o conjunto de transições T é

$T = \{Amanhecer, Entardecer, Anoitecer\}$,

o conjunto de arcos A é

$$A = \{\ (Manh\tilde{a}, Entardecer),\quad (Entardecer, Tarde),$$
$$(Tarde, Anoitecer),\quad (Anoitecer, Noite),$$
$$(Noite, Amanhecer),\quad (Amanhecer, Manh\tilde{a})\ \},$$

o conjunto de valorações dos arcos V é

$V = \{1, 1, 1, 1, 1, 1\}$

e o conjunto de capacidades dos lugares k é

$k = \{k_{Manh\tilde{a}} = 1, k_{Tarde} = 1, k_{Noite} = 1\}$.

Exemplo 2.3

$R_{Linha\_de\_Produção} = (P, T, I, O, K)$, onde

o conjunto de lugares P é

$P = \{Parafusos, Porcas, Pacote, Máquina, Depósito\}$,

o conjunto de transições T é

$T = \{MontaPacote, EnviaPacote\}$,

o conjunto de arcos A é

$$A = \{\ (Parafusos, MontaPacote),\quad (Porcas, MontaPacote),$$
$$(Máquina, MontaPacote),\quad (MontaPacote, Pacote),$$
$$(Pacote, EnviaPacote),\quad (EnviaPacote, Máquina),$$
$$(EnviaPacote, Depósito)\ \},$$

o conjunto de valorações dos arcos V é

$V = \{3, 3, 1, 1, 1, 1\}$

e o conjunto de capacidades dos lugares k é

$K = \{k_{Parafusos} = \omega, k_{Porcas} = \omega, k_{Pacote} = 1, k_{Máquina} = 1, k_{Depósito} = \omega\}$.

Neste trabalho, tratamos as redes de Petri segundo o ponto de vista matricial e, quando conveniente, através do estudo dos *bags*, pois para modelos de grandes dimensões, as matrizes não possibilitam uma representação compacta, dado a sua esparcidade característica.

1.2 Rede de Petri Marcada

Uma marca (também denominada *ficha* ou *token*) é um conceito primitivo em redes de Petri, tal qual lugar e transição. As marcas são informações atribuídas aos lugares. O número e a distribuição de marcas nos lugares correspondem à marcação (estado) da rede em um determinado momento. Nesta seção apresentamos a definição formal de marcação, assim como a definição de uma rede de Petri marcada.

Definição 1.2.1 - Marcação: *seja* P *o conjunto de lugares de uma rede* R. *Define-se formalmente marcação como uma função que mapeia o conjunto de lugares* **P** *a inteiros não-negativos* $M : P \rightarrow N$.

Uma outra definição da marcação das redes de Petri pode ser apresentada na forma vetorial.

Definição 1.2.2 - Vetor Marcação: *seja* P *o conjunto de lugares de uma rede* R. *A marcação pode ser definida como um vetor* $M = (M(p_1), ..., M(p_n))$, *onde* $n = \#P$, *para todo* $p_i \in P$ *tal que* $M(p_i) \in \mathbb{N}$.

Em geral, a presença de marcas em um lugar pode ser interpretada como a presença de um recurso de um determinado tipo. Vale ressaltar, que o conjunto e a distribuição de marcas em uma rede é a marcação desta rede e esta representa o estado do modelo. Graficamente, as marcas são representadas por *pontos* colocados no interior dos lugares. É possível designarmos o número de marcas através de um numeral no interior dos lugares. Na figura 1.6 apresentamos as redes marcadas R_1 e R_2, cujos vetores marcação inicial são representados respectivamente pelos vetores $M_0^{R_1}$ e $M_0^{R_2}$.

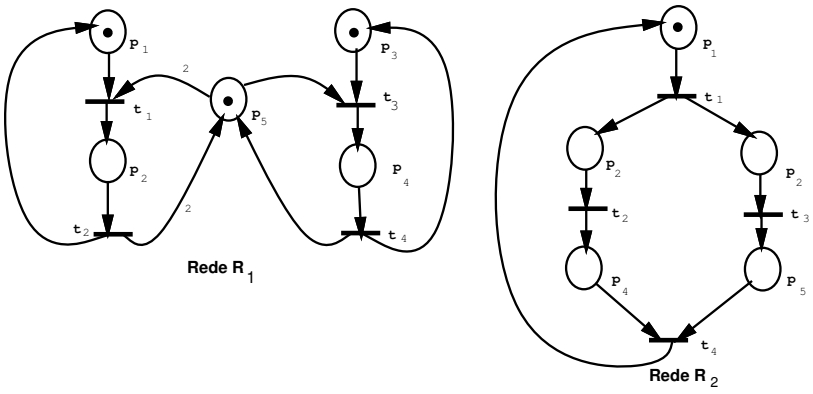

Figura 1.6: Redes Marcadas R_1 e R_2

$$M_0^{R_1} = \left|\begin{array}{c|l} 1 & p_1 \\ 0 & p_2 \\ 1 & p_3 \\ 0 & p_4 \\ 1 & p_5 \end{array}\right.$$

$$M_0^{R_2} = \left| \begin{array}{c|l} 1 & p_1 \\ 0 & p_2 \\ 0 & p_3 \\ 0 & p_4 \\ 0 & p_5 \end{array} \right.$$

Uma rede de Petri marcada pode ser definida por uma dupla formada pela estrutura da rede (R) e uma marcação inicial (M_0) associada à rede. A seguir apresentamos a definição formal das redes de Petri marcadas.

Definição 1.2.3 - Rede Marcada: *define-se uma rede de Petri marcada pela dupla $RM = (R; M_0)$, onde R é a estrutura da rede e M_0 a marcação inicial.*

O comportamento dos sistemas pode ser descrito em função dos seus estados e suas alterações. Para simular o comportamento dinâmico dos sistemas a marcação da rede de Petri é modificada, a cada ação realizada (transição disparada), segundo algumas regras de execução. Estas regras são apresentadas na seção seguinte.

1.3 Regras de Execução

O disparo de transições (execução das ações) é controlado pelo número e distribuição de marcas nos lugares. Uma transição t está habilitada se, e somente se, todos os lugares de entrada ($p_i \in P$) de t têm marcação $M(p_i) \geq I(p_i, t)$. Denota-se a habilitação de uma transição t para uma marcação M' por - $M'[t>$. Vale ressaltar que a marcação habilita o disparo de uma transição, contudo não obriga o seu disparo (não-determinismo). No exemplo da linha de produção, a existência de parafusos e porcas não obriga a montagem dos conjuntos [8]. Disparando-se uma transição t, retira-se um número de marcas, igual ao peso do arco de entrada, de todos os lugares de entrada p_i de t, e são criadas marcas nos lugares de saída. O número de marcas criadas em cada lugar de saída da transição t é igual ao peso do arco de saída. Denota-se a acessibilidade da marcação M'' a partir de M' pelo disparo da transição t por - $M'[t > M"$.

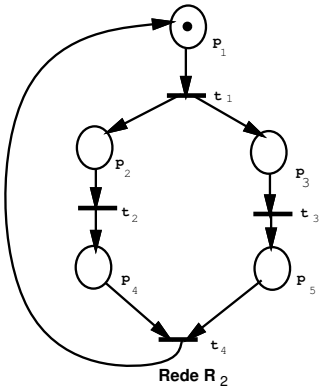

Figura 1.7: Disparo de Transições

A seguir, mostramos a alteração da marcação inicial $M_0 = (1, 0, 0, 0, 0)$ da rede marcada R_2, apresentada na seção anterior, devido ao disparo da transição t_1.

Se $M_0[t_j > M'$, então

$$M'(p) = M_0(p) - I(p, t_j) + O(p, t_j), \quad \forall p \in P$$

$$\begin{vmatrix} x \\ y \\ u \\ w \\ z \end{vmatrix} = \begin{vmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{vmatrix} - \begin{vmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{vmatrix} + \begin{vmatrix} 0 & p_1 \\ 1 & p_2 \\ 1 & p_3 \\ 0 & p_4 \\ 0 & p_5 \end{vmatrix}$$

$$M_{R_2'} = \begin{vmatrix} 0 & p_1 \\ 1 & p_2 \\ 1 & p_3 \\ 0 & p_4 \\ 0 & p_5 \end{vmatrix}$$

O disparo da transição t_1 altera a marcação, retirando uma marca do lugar p_1 e armazenando uma marca no lugar p_2 e p_3. A nova marcação

M' pode ser obtida subtraindo-se o vetor marcação atual M_0 pela coluna correspondente à transição disparada da matriz de entrada (I) e somando-se a coluna correspondente à transição disparada da matriz de saída (O), conforme mostrado na figura 1.7. Formalmente, apresentamos as regras de execução das redes de Petri a seguir:

Definição 1.3.1 - Transição Disparável: *uma transição* t *é disparável para uma marcação* M ($M[t>$) *se, e somente se,* $M(p) \geq I(p,t)$ *para todo* $p_i \in P$. *Se* t *é disparável para uma marcação* M_0, *então com o disparo de* t *obtem-se uma nova marcação* M' ($M[t > M'$), *tal que:* $M'(p) = M(p) - I(p,t) + O(p,t)$, *para todo* $p \in P$.

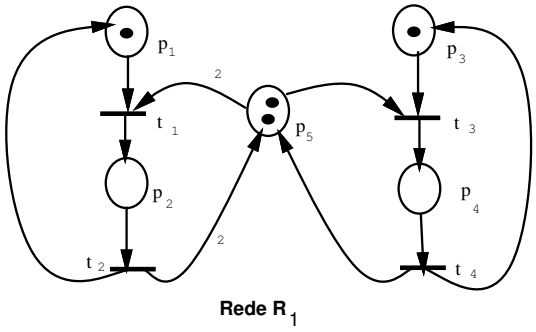

Figura 1.8: Rede R_1 Marcada

Apresentamos na figura 1.8 as marcações acessíveis, da rede R_1, através dos disparos das transições t_1 e t_3. A marcação inicial $M_0 = (1,0,1,0,2)$ habilita o disparo das transições t_1 e t_3. Com o disparo de t_1 obtemos a marcação $M_1 = (0,1,1,0,0)$; contudo, ao dispararmos, a partir de M_0, a transição t_3, atingimos a marcação $M_2 = (1,0,0,1,1)$.

A seguir, apresentamos dois casos particulares de configuração, que têm regras de disparo distintas da apresentada. O primeiro caso, corresponde a transições que não possuem lugares de entrada. Estas transições são denominadas transições fonte (*source*). A outra situação especial, é aquela que apresenta transições sem lugares de saída. Uma transição sem lugar de saída é chamada de transição de absorção (*sink*). Uma transição fonte está sempre habilitada (ver figura 1.9) e o disparo de uma transição de absorção

consome marcas nos lugares de entrada, porém não cria outros recursos (ver figura 1.10). A seguir definimos formalmente as transições de absorção e fonte.

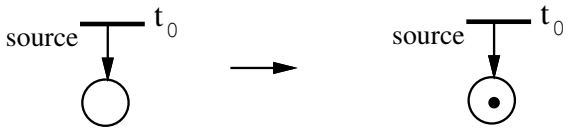

Figura 1.9: Transição Fonte (*Source*) antes e após o Disparo

Definição 1.3.2 - Transição Source: *a transição t_i é definida como "Source" se, e somente se, $I(p,t) = 0$, para todo $p \in P$.*

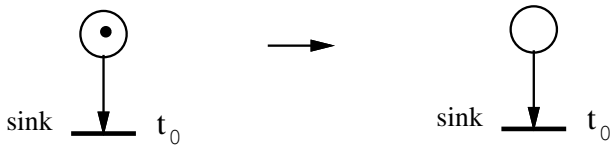

Figura 1.10: Transição de Absorção (*"Sink"*) antes e após o Disparo

Definição 1.3.3 - Transição Sink: *a transição t_i é definida como sink se, e somente se, $O(p,t) = 0$, para todo $p \in P$.*

1.4 Matriz de Incidência

Utilizando-se a representação matricial, a estrutura das redes de Petri é representada pelo conjunto de lugares, conjunto de transições, matriz de entrada e matriz de saída. Essas matrizes expressam as interconexões entre lugares e transições dos modelos, dado que a matriz de entrada apresenta os lugares de entrada (pré-condições) e a matriz de saída os lugares de saída (pós-condições) das transições, assim como a valoração destas interconexões

(arcos). Quando uma transição t dispara, a diferença entre as marcações M_1 e M_0 é igual à diferença dada por $O(p,t) - I(p,t)$, para todo lugar $p \in P$. Denomina-se $C = O - I$ a matriz de incidência, dado que esta fornece a incidência de arcos de entrada e saída em cada transição dos modelos. Esta matriz representa a estrutura dos sistemas modelados. A seguir, apresentamos a definição da matriz de incidência.

Definição 1.4.1 - Matriz de Incidência: *seja a rede* $R = (P, T, I, O, K)$, *onde* P *é o conjunto de lugares,* T *o conjunto de transições,* I *a matriz de entrada,* O *a matriz de saída e* K *a capacidade dos lugares. A matriz de incidência* C *das redes de Petri representa a relação* $P \times T \rightarrow Z$ *definida por:* $\forall p \in P, \quad \forall t \in T | C(p,t) = O(p,t) - I(p,t)$.

Embora a estrutura da rede seja representada pelas matrizes de entrada e saída e, conseqüentemente, pela matriz de incidência, em alguns casos a matriz de incidência não possibilita a representação desta estrutura completamente.

1.5 Rede Pura e *Self-Loop*

Os conceitos de *self-loop* e de rede pura são de particular importância, dado que a matriz de incidência de redes *impuras* não representa a estrutura da rede.

Um par (p_i, t_j) formado por um lugar p_i e uma transição t_j é denominado *self-loop*, se p_i é, ao mesmo tempo, pré-condição e pós condição de t_j. Uma rede de Petri é dita pura se não tem *self-loops*.

Definição 1.5.1 - Rede Pura: *uma rede* R_k *é dita pura se, e somente se, não contém nenhum self-loop, ou seja,* $I(p_j, t_i) * O(p_j, t_i) = 0$, *para todos* $t_i \in T, \quad e \quad p_j \in P$.

Uma rede com *self-loops* (impura) pode ser transformada (refinada) em uma rede pura através da introdução de pares *dummy*. O par *dummy* corresponde a um par formado por um lugar e uma transição que refinam um *self-loop*. Este refinamento pode ser observado na figura 1.11, onde temos a substituição do arco de entrada do lugar p_1 $(I(p_1) = t_1)$ pelo par *dummy* (p_2, t_2), de forma que a transição t_1 passa a ser entrada do lugar p_2 e este

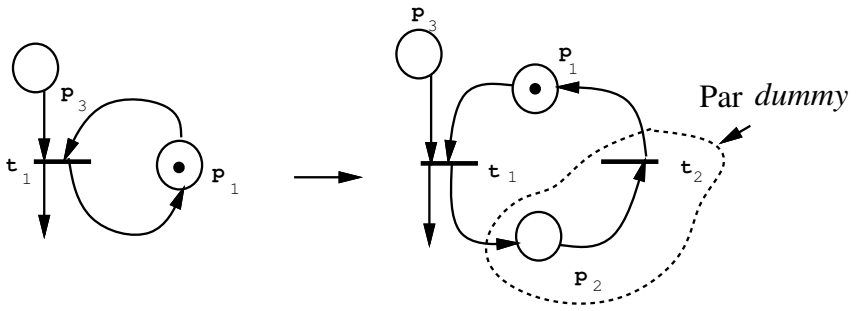

Figura 1.11: Transformação de uma Rede Impura em Pura

entrada da transição t_2. A transição t_2 é então associada como entrada do lugar p_1.

$$\left|\begin{array}{c|c} t_1 & \\ \hline 0 & p_1 \\ -1 & p_3 \\ . & . \end{array}\right|$$

$$\left|\begin{array}{cc|c} t_1 & t_2 & \\ \hline -1 & 1 & p_1 \\ 1 & -1 & p_2 \\ -1 & 0 & p_3 \\ . & 0 & . \end{array}\right|$$

Como mostrado anteriormente, a matriz de incidência é a representação compacta das matrizes de pré e pós-condições (I e O), ou seja, a matriz de incidência representa a estrutura do modelo. No entanto, observamos que uma rede impura (redes com *self-loop*) possui lugares que são entrada e saída de uma mesma transição. A estrutura dessas redes não é completamente representada pela matriz de incidência, muito embora o comportamento do modelo continue sendo representado adequadamente pelas regras de execução.

1.6 Seqüências Disparáveis

Se uma transição t_1 está habilitada para uma marcação M e uma segunda transição t_2 está habilitada para a marcação M_1, obtida após o disparo de t_1, dizemos que a seqüência $sq = t_1; t_2$ está habilitada para M. Ou seja, se $M[t_1 > M_1$ e $M_1[t_2 > M_2$ então $M[t_1; t_2 > M_2$. Desta forma, designamos o disparo de uma seqüência $sq \in T$ por $M[sq > M'$.

Definição 1.6.1 - Seqüências Disparáveis: *a seqüência* sq *está habilitada, possibilitando a obtenção de uma marcação* M'', $(M[sq > M'')$ *se, e somente se, ocorre um dos casos abaixo:*

- $sq = \lambda$, *onde* λ *é a seqüência vazia, tal que* $M'' = M$;
- $sq = sq't$, *onde* sq *é uma seqüência de transições* $t \in T$ *e existe* M' *tal que* $M[sq' > M'$ *e* $M'[t > M''$.

Se $M[sq > M''$, diz-se que sq é uma seqüência disparável para M.

1.7 Função de Próximo Estado

Seja R uma rede marcada e M uma marcação que habilita uma transição t_j, denominamos função de próximo estado aquela que possibilita a obtenção de uma nova marcação M' com o disparo de uma transição t_j da rede R, ou seja, se t_j estiver habilitada temos $\delta(M; t_j) = M'$.

Definição 1.7.1 - Função de Próximo Estado: *a função de próximo estado de uma rede* $R = (P, T, I, O, M, K)$ *é definida se, e somente se,* $M(p_i) \geq I(p_i, t_j)$, *para todo* $p_i \in P$. *Sendo esta condição satisfeita,* $\delta(M; t_j) = M'(\delta : \mathbb{N}^n \times T \rightarrow \mathbb{N}^n)$, *onde* $M'(p_i) = M(p_i) - I(p_i, t_j) + O(p_i, t_j)$, *para todo* $p_i \in P$ *e* $n = \#P$.

É conveniente estender a definição de função de próximo estado para mapear a marcação e uma seqüência de transições sq em uma nova marcação. Tenhamos uma uma seqüência de transições $t_1; t_2; t_3; ...; t_k$ e uma marcação M. A marcação $M = \delta(M, t_1; t_2; t_3; ...; t_k)$ é resultado do disparo da transição t_1, seguido de t_2 e assim por diante até o disparo de t_k, desde que o disparo de cada uma dessas transições gere uma marcação que habilite a transição seguinte.

Definição 1.7.2 - Função de Próximo Estado Estendida: *a função de próximo estado estendida para uma marcação* M *é uma seqüência* sq *de transições disparáveis* t_i *definida por:* $\delta(M, t_j; sq) = \delta(\delta((M, t_j), sq)$ *e* $\delta(M, \lambda) = M$.

1.8 Equação Fundamental

A equação fundamental das redes de Petri (equação de estados ou de marcações) descreve o comportamento das redes, bem como possibilita a análise de propriedades comportamentais e estruturais que serão apresentadas no capítulo 3.

Como visto na seção anterior, o disparo de uma transição t_0, habilitada por uma marcação M_0, gera uma nova marcação $M_1(M_0[t_0 > M_1)$. Esta alteração pode ser representada através da função de próximo estado $M_1 = \delta(M_0, t_0)$.

M_1 habilita a transição t_1 e que disparada, habilitará uma nova transição. Se após o disparo de uma seqüência de transições $sq = t_0; t_1; ...; t_k$ obtivermos uma marcação M, podemos representar esse fato através da função de próximo estado, ou seja $M = \delta(M_0, sq) = \delta(M_0, t_0; t1; ...; t_k)$.

Segundo a abordagem matricial das redes de Petri, uma transição t_j é representada por um vetor $s(t_j)$ com dimensão igual ao número de transições da rede, onde todos os componentes desse vetor são zero, exceto o j-ésimo componente que tem valor unitário ($s(t_j)$). Portanto,

$$M'(p) = M_0(p) - I(p, t_j) + O(p, t_j), \quad \forall p_i \in P.$$

pode ser representado por

$$M'(p) = M_0(p) - I.s(t_j)^T + O.s(t_j)^T = M_0(p) + (O - I).s(t_j)^T, \quad \forall p_i \in P$$

Representando as matrizes de pré e pós-condições (I e O) através da matriz de incidência C ($C = O - I$), temos:

$$M(p)' = M_0(p) + C.s(t_j)^T, \quad \forall p_i \in P$$

.

Aplicando a seqüência de transições sq na equação acima, obtemos:

$$M'(p_j) = \delta(M_0(p_j), sq) = M_0(p_j) + C.[s(t_0)^T + s(t_1)^T + ... + s(t_k)^T],\ \forall p_i \in P.]$$

O vetor $[s(t_0)^T + s(t_1)^T + ... + s(t_k)^T]$ é denominado vetor característico ou vetor de *Parikh*, e é representado por $\overline{s}$.

Definição 1.8.1 - Vetor Característico: *o vetor característico $\overline{s}$ de uma rede $R = (P, T, I, O, K)$ é um vetor de dimensão igual a $\#T$, onde os componentes deste vetor representam o número de disparos de cada transição.*

A equação $M'(p) = M_0(p) + C.\overline{s},\quad \forall p_i \in P$ é denominada Equação Fundamental das Redes de Petri ou equação de estados.

1.9 Grafo das Marcações Acessíveis

O grafo de marcações acessíveis é uma representação gráfica do conjunto das marcações que podem ser alcançadas para uma dada rede de Petri. O disparo de uma transição modifica a marcação, conforme a marcação atual e a estrutura da rede. Essas marcações obtidas após os disparos das transições são as marcações acessíveis de uma rede para uma determinada marcação inicial. Na figura 1.12, apresentamos uma rede marcada ($N_1 = (R_1, M_0)$).

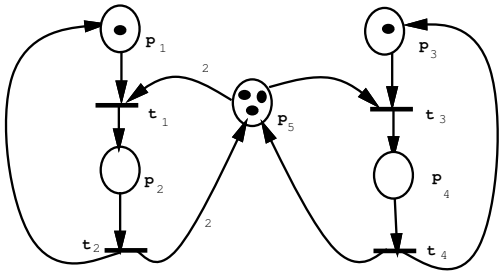

Figura 1.12: Rede N_1

O vetor M_k representa a k-ésima marcação da rede R_1($M_k = (m(p_1), m(p_2), m(p_3), m(p_4), m(p_5))$ e $A(R; M_0) = \{M_0, M_1, M_2, M_3\}$ o conjunto das marcações acessíveis obtidas a partir da marcação inicial M_0, onde:

$$M_0 = (1, 0, 1, 0, 3))$$

$$M_1 = (0, 1, 1, 0, 1))$$

$$M_2 = (0, 1, 0, 1, 0))$$

$$M_3 = (1, 0, 0, 1, 2))$$

Constrói-se o grafo das marcações acessíveis através da verificação e disparo das transições disparáveis para uma dada marcação inicial e repetindo-se esse processo para as marcações obtidas com esses disparos. Os vértices desse grafo são as marcações e os arcos que interconectam esses vértices representam o disparo de cada transição.

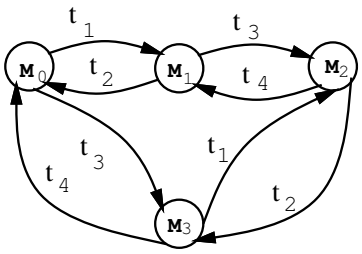

Figura 1.13: Grafo das Marcações Acessíveis

Na rede marcada da figura 1.12, podemos disparar a transição t_1 e a transição t_3. O disparo de t_1 leva a rede para a marcação M_1 e o disparo de t_3 para a marcação M_3. Na marcação M_1 podemos disparar t_2 e t_3. O disparo de t_2 retorna a marcação inicial da rede. O disparo de t_3, a partir de M_1 faz com que a rede atinja a marcação M_2. Na marcação M_2, as transições t_2 e t_4 estão habilitadas. Disparando-se t_4 retorna-se a marcação M_1 e disparando-se t_2 atinge-se M_3. Em M_3 é possível retornar-se a marcação M_2, pelo disparo de t_1 ou a marcação inicial pelo disparo de t_4.

Definição 1.9.1 - Conjunto das Marcações Acessíveis: *seja uma rede marcada $N = (R; M_0)$; define-se conjunto das marcações acessíveis $A(R; M_0)$ pelo conjunto de marcações obtidas a partir de uma marcação inicial M_0 pelo disparo de todas as possíveis seqüências de transições habilitadas, ou seja,*

$A(R;M_0) = \{M_i \in \mathbb{N}^m\}$, *tal que existe uma seqüência de transições* sq, *que* $M_0[sq > M_i$, *onde* $m = \#P$ *e* M_i *só é alcançável se, e somente se,* $M_i \in A(R;M)$.

Na figura 1.13, apresentamos o grafo das marcações acessíveis (GA) da rede R_1. O grafo das marcações acessíveis pode ser definido por uma dupla $\mathrm{GA}(R;M_0) = (M_i, a_{(i,j)})$, onde $M_i \in A(R;M_0)$ são os vértices do grafo e $a_{(i,j)}$ os arcos. Os arcos representam a alteração da marcação pelo disparo das transições. Apresentamos, a seguir, a definição do grafo das marcações acessíveis (grafo de acessibilidade) a estas marcações:

Definição 1.9.2 - Grafo das Marcações Acessíveis*: seja uma rede marcada* $N = (R;M_0)$*; define-se grafo das marcações acessíveis GA, por um par* $GA(R;M_0) = (M_i, a_{(i,j)})$, *onde* $M_i \in A(R;M_0)$ *são os vértices do grafo e* $a_{(i,j)}$ *os arcos. Os arcos representam* $M'[> M''$ *se, e somente se, existe uma transição* t *tal que* $M'[t > M''$ *(mudança de marcação).*

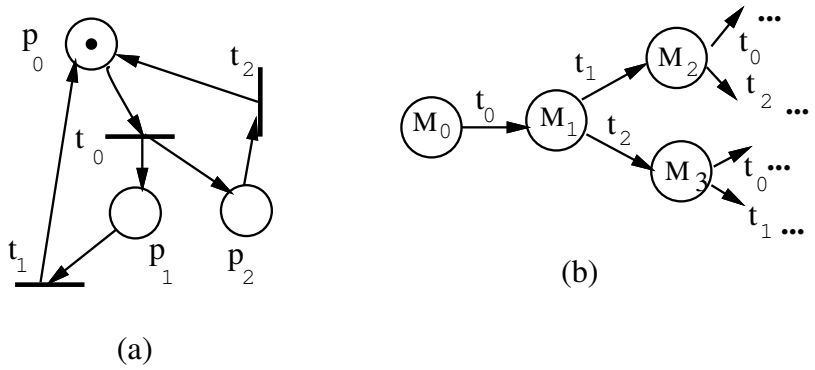

Figura 1.14: Grafo das Marcações Acessíveis Infinito

O vetor $M_k = (m(p_0), m(p_1), m(p_2))$ representa o vetor marcação da rede da figura 1.14, ou seja, o primeiro componente desse vetor representa a marcação do lugar p_0 para a k-ésima marcação. O segundo componente representa a marcação do lugar p_1 e o terceiro componente, a marcação do lugar p_2.

$$M_0 = (1, 0, 0))$$
$$M_1 = (0, 1, 1))$$
$$M_2 = (1, 0, 1))$$
$$M_3 = (1, 1, 0))$$

Para a rede da figura 1.12, observamos que o número de marcacões acessíveis é finito, portanto é possível representar, através do grafo de marcações acessíveis, todas as marcações obtidas com os disparos das transições. No entanto, um grande número de modelos pode ter um número infinito de marcações, o que torna impossível a representação dessas marcações nesse grafo. Para exemplificar esta situações observemos a rede da figura 1.14.a. A marcação inicial desta rede ($M_0 = \{1, 0, 0\}$) habilita o disparo da transição t_0, alterando a marcação da rede para $M_1 = \{0, 1, 1\}$. A marcação M_1 possibilita tanto o disparo de t_1 quanto o disparo de t_2. Observe que, tanto disparando a transição t_1 quanto disparando t_2, teremos uma nova marca no lugar p_0, o que possibilita o disparo de t_0, de tal forma que teremos um acúmulo de marcas nos lugares da rede e conseqüentemente um número infinito de marcações. Nesse caso, a rede pode sempre atingir um estado diferente dos já alcançados, fazendo com que o grafo das marcações acessíveis seja infinito, impossibilitando a representação gráfica de todas as marcações acessíveis (figura 1.14.b) do modelo. No capítulo 3, apresentamos duas representações gráficas que são utilizadas para representar, finitamente, um número infinito de marcações.

1.10 Condições Externas

Como já apresentado, as transições de uma rede de Petri representam as ações. A execução de uma ação pelo sistema, pode não depender de nenhuma condição externa ao sistema (entrada), contudo é possível que a execução de uma ação pelo sistema dependa não só de condições internas (marcas em lugares), como também de condições externas ao sistema modelado. Para possibilitar esta necessidade de representação, podemos associar estas condições às transições, ou seja, para que ocorra o disparo da transição é necessário que também a condição externa associada à transição seja verdadeira. A figura 1.15 mostra uma rede onde temos associadas condições externas às transições t_2 e t_3.

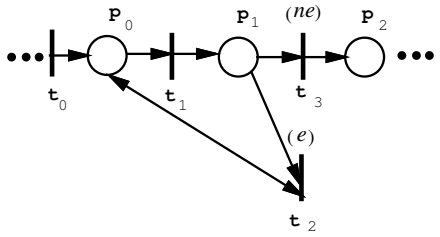

Figura 1.15: Condições Externas Associadas às Transições

Imaginemos as atividades de um programador, que, após encerrar a codificação de um programa, compila os módulos fontes separadamente e posteriormente efetua a ligação dos módulos obtidos. A rede da figura 1.15 apresenta abstratamente parte desse processo. Uma marca no lugar p_0 indica que há um programa disponível, permitindo, portanto, que ele seja compilado (disparo de t_1). As transições t_2 e t_3 estão associadas às condições externas, *(e)* e *(ne)*. Estas condições indicam se houve erro ou não no processo de compilação. Caso ocorra algum erro na compilação, a transição t_3 está impossibilitada de disparo e a transição t_2 pode ser disparada (possibilita uma nova compilação, pois uma marca será depositada no lugar p_0). Quando não ocorrer nenhum erro na compilação, é possível disparar t_3, possibilitando a execução das ações subseqüentes.

1.11 Associação de Rótulos a Transições

Em algumas oportunidades, no processo de modelagem de sistemas, desejamos não tomar conhecimento ou mesmo confundir ocorrências de ações (disparo de transições). Para tal, rotulamos transições distintas com o mesmo nome, podendo inclusive usar-se λ, a ação vazia.

Devemos definir um alfabeto $\sum$ e associá-lo à rede. Realizamos esta associação através da *rotulação* de transições da rede com símbolos de $\sum$.

Definição 1.11.1 - Função de Nomeação: *seja uma uma rede* $R = (P, T, I, O, K)$ *e um alfabeto* Σ. *A função* $\sigma : T \rightarrow \Sigma$ *é definida como função de nomeação.*

Na figura 1.16 apresentamos uma rede onde todas as transições estão rotuladas (etiquetadas).

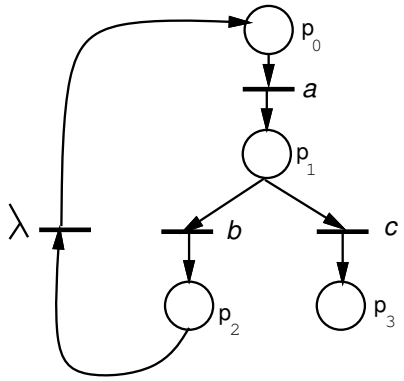

Figura 1.16: Rede Etiquetada

Nesta rede temos as transições rotuladas as ações do tipo *a, b, c* e a ação vazia λ. Uma transição rotulada como uma ação vazia, ou seja $\sigma(t_j) = \lambda$ não aparece em qualquer seqüência de transições disparadas.

Para que não haja confusão entre condições e rótulos nas transições, adotaremos a convenção de grafar as condições sempre entre parênteses.

1.12 Linguagens das Redes de Petri

As redes de Petri possibilitam a especificação, modelagem e verificação de propriedades de sistemas. Como já observamos, as transições modelam as ações, a ocorrência de uma ação é representada pelo disparo de uma transição e as seqüências de ações são modeladas por seqüências de transições. Essas seqüências de transições são extremamente importantes quando desejamos observar a equivalência entre modelos. Duas redes são ditas equivalentes se todas as seqüências de transições fornecidas são iguais. O conjunto de *strings* (seqüências de transições) gerados por todas as possíveis seqüências de disparo definem uma linguagem formal denominada *Linguagem da rede de Petri.*

Usando-se os conceitos aplicados para a obtenção de liguagens das máquinas

de estados finitos, desenvolveu-se a teoria das linguagens das redes de Petri. Na seção anterior, apresentamos a *rotulação* (nomeação) de transições das redes de Petri a partir de um alfabeto Σ.

A figura 1.17 apresenta uma rede que tem suas transições rotuladas conforme o alfabeto $\Sigma = \{a, b\}$. Observe-se contudo, que uma transição é rotulada como ação vazia (λ), que corresponde ao *string* vazio ϵ, e obviamente não tem influência na linguagem gerada.

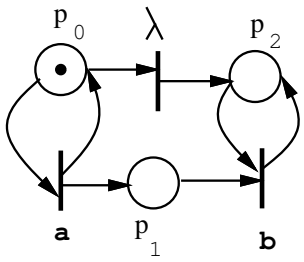

Figura 1.17: Rede Rotulada

Ainda na rede da figura 1.17, a transição rotulada com a pode ser disparada um número infinito de vezes e o lugar p_1 acumula um número de marcas exatamente igual ao número de disparo desta transição. No entanto, quando dispararmos a transição rotulada com uma ação vazia, a transição rotulada com a é impedida de ser disparada, pois a marca do lugar p_0 é consumida. O disparo da transição rotulada com λ deposita uma marca no lugar p_2, o que habilita a transição rotulada com b. Essa transição poderá ser disparada o mesmo número de vezes que foi disparada a transição rotulada com o símbolo a. A linguagem gerada pela rede da figura é $\mathrm{L}(R, M_0) = a^n b^n, \quad n \geq 0$. Um trabalho mais detalhado sobre este tema pode ser encontrado em [4].

É importante salientar que a expressividade da rede de Petri supera o formalismo das gramáticas livre de contexto limitadas e expressões regulares, uma vez que, face a inexistência de mecanismos de contagem (memória) nesses formalismos, a linguagem $\mathrm{L}(R, M_0) = a^n b^n, \quad n \geq 0$ não pode ser expressa diretamente, tendo-se que recorrer ao artifício de representar a linguagem como sendo $\mathrm{L}= (ab)^n$, impondo-se a geração casada dos símbolos.

1.13 Classes das Redes de Petri

Algumas classes de problemas têm características que devem ser explicitadas ou mesmo representadas de forma mais compacta, na especificação. Apresentamos algumas dessas classes de problemas, bem como apontamos para algumas extensões às redes de Petri que os solucionam.

Em algumas fases do processo de desenvolvimento de um sistema, necessitamos representar o sistema em evolução com um maior ou menor detalhamento, de maneira compacta. Dependendo das dimensões do sistema, o modelo representativo pode ter dimensões não práticas e não elucidativas. Em outra fase do desenvolvimento pode ser necessário o esclarecimento de determinados aspectos, que em fases anteriores não se faziam necessários. Tendo em vista esses aspectos, a literatura fornece diferentes classificações para as redes de Petri.

- Redes de Petri Ordinárias:

  RdPs Binárias ou Condição-Evento

  Redes de Petri *Place-Transition*

  RdPs Não-Ponderadas

  RdPs Ponderadas

- Redes de Petri Não-Ordinárias ou de Alto Nível:

  RdPs Predicado-Evento

  RdPs Coloridas

  RdPs Hierárquicas

Aqui seguimos a classificação apresentada em [1], embora existam outras classificações na literatura.

1.14 Redes de Petri Ordinárias

As redes de Petri ordinárias caracterizam-se pelo tipo de suas *marcas*. Nessas classes de rede, o tipo das marcas é inteiro não negativo, enquanto as redes de Petri não-ordinárias possuem marcas de tipos particulares.

A classe binária (condição/evento) representa as redes mais simples entre todas as classes. Nessa classe, os lugares podem conter no máximo uma ficha e todos os arcos têm peso unitário. A figura 1.18.a apresenta uma rede dessa classe.

Nas redes *Place-Transition* os lugares podem acumular marcas, assim como os arcos podem ser valorados. Alguns autores, contudo, fazem distinção entre as redes em que os arcos têm valores difrentes de um e as redes com arcos de peso igual a um.

As redes de Petri não-ponderadas são uma classe em que os lugares podem ter mais do que uma marca e a valoração dos arcos é unitária. A figura 1.18.a apresenta uma rede binária, que descreve um processo que pode utilizar dois recursos (r_1, r_2) e uma rede não ponderada (figura 1.18.b) que representa o mesmo processo, de forma mais compacta, com apenas um lugar representando os recursos. O número de marcas no lugar r indica a quantidade de recursos.

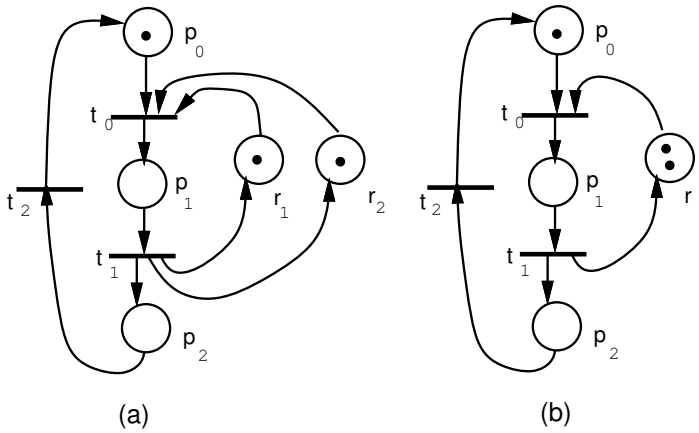

Figura 1.18: Rede Binária e não-Ponderada

Nas redes ponderadas, os pesos dos arcos não estão limitados a 1. Nessa classe de rede podemos ter arcos múltiplos, que podem ser representados por um único arco e associado a este, uma valoração. A rede R_1 da figura 1.12 apresenta uma rede dessa classe.

1.15 Redes de Alto Nível

As redes não-ordinárias são caracterizadas pelos tipos de suas marcas que não são mais elementos do tipo inteiro positivo. Nessas classes de redes, as marcas são diferenciadas com parâmetros de *cor*, que permite a individualização da marca (a especificação de um objeto por sua cor) ou mesmo a marca representada não por um único objeto mas por um conjunto de objetos. Essas classes de redes serão estudadas no capítulo 4.

Todas as classes de redes que apresentamos têm o mesmo poder de computacional, no entanto algumas dessas classes possibilitam um mais alto ou mais baixo nível de abstração dos modelos. Dependendo da aplicação ou mesmo do estágio de desenvolvimento, faz sentido a escolha de uma determinada classe de rede para a modelagem do problema. No entanto, nenhuma destas classes possibilita o *teste a zero* (testar se um lugar não tem marca) de lugares com capacidade ilimitada ($k = \infty$). No capítulo 4 apresentamos algumas extensões às redes de Petri, que possibilitam esse teste e fornecem meios para a especificação de outras propriedades dos sistemas computacionais, tais como tempo e prioridade. As extensões aqui elucidadas são as redes de Petri com arco inibidor, redes de Petri temporizadas determinísticas, estocásticas e com prioridades. Infelizmente, essas extensões reduzem o poder de decisão das redes de Petri. Alguns trabalhos investigam o uso de subclasses de redes de Petri que possibilitem aumentar o poder de decisão, sem contudo reduzir em demasia o seu poder de modelagem.

No capítulo seguinte, apresentamos a modelagem com redes de Petri, descrevendo uma série de problemas clássicos, que possibilitam a percepção do poder de modelagem das redes de Petri.

Capítulo 2

Modelagem com Redes de Petri e Sub-Classes

Este capítulo apresenta inicialmente algumas redes básicas e essenciais para a modelagem de sistemas em geral [2]. Redes a partir das quais, torna-se possível a modelagem de sistemas mais complexos [18, 17, 16, 12, 10, 46] em função destas construções elementares. Posteriormente, apresentamos uma série de exemplos clássicos, onde são apresentados os seus modelos. Estes exemplos têm aspectos importantes a serem observados, e a sua modelagem [25] serve também como um exercício para uma posterior especificação de sistemas mais complexos.

2.1 Redes Elementares

As redes elementares [4] [2] são utilizadas como blocos básicos que possibilitam a especificação de aplicações mais complexas. Nesta seção veremos o modelo de seqüenciamento, distribuição, junção, escolha não-determinística e atribuição.

2.1.1 Seqüenciamento

O seqüenciamento é uma rede que representa a execução de ação, desde que uma condição seja verdadeira. Após a execução desta ação temos uma nova condição que poderá possibilitar a execução de uma nova ação [2].

Na figura 2.1 apresentamos este modelo. Uma marca no lugar p_0 habilita a transição t_0 e com o disparo desta transição é estabelecida uma nova condição (p_1 é marcado). Esta nova condição pode permitir o disparo de uma outra transição que esteja asociada ao lugar p_1.

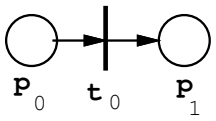

Figura 2.1: Seqüenciamento

2.1.2 Distribuição

A rede que representa a distribuição é apresentada na figura 2.2. Esta rede possibilita a criação de processos paralelos a partir de um processo pai [2]. Nesta rede o disparo da transição t_0, retira uma marca do lugar p_0 e coloca uma marca nos lugares p_1 e p_2. Estas novas condições estabelecidas, permitem a execução de outras tarefas paralelamente, ou seja, p_1 é pré-condição para a execução de uma tarefa e p_2 pré-condição para a execução de uma outra tarefa.

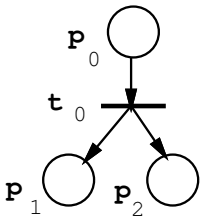

Figura 2.2: Distribuição

2.1.3 Junção

A rede que modela o sincronismo ou junção é apresentada na figura 2.3. Esta rede possibilita a sincronização entre processos. Este modelo tem

fundamental importância, pois na modelagem de atividades concorrentes freqüentemente é necessária a sincronização entre tarefas. Esta rede recombina duas tarefas, pemitindo que, por exemplo, um processo continue sua execução apenas após o término de outros processos específicos [2].

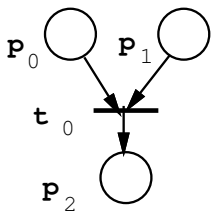

Figura 2.3: Junção

O disparo da transição t_0 só é possível quando existir uma ficha nos lugares p_0 e p_1. Esta condição sendo satifeita, é possível disparar-se t_0. Efetuando-se o disparo, retira-se uma ficha dos lugares p_0 e p_1 e coloca-se uma ficha no lugar p_2. Esta nova condição estabelecida pode possibilitar a execução de outras ações do sistema.

2.1.4 Escolha Não-Determinística

Nesta seção apresentamos uma rede elementar que dependendo da aplicação, é denominada conflito, escolha ou decisão. Definiremos também os conceitos de conflito estrutural e conflito efetivo [2].

A rede elementar da figura 2.4 representa a escolha não-determinística do disparo de transições. Neste modelo, o disparo de uma transição inabilita o disparo da outra transição.

Alguns autores, no entanto, enfatizam uma distinção entre o que denominam conflito (escolha, decisão) estrutural e conflito efetivo.

Definição 2.1.1 - Conflito Estrutural: *duas transições t_0 e t_1 estão em conflito estrutural se, e somente se, têm um lugar p comum como entrada, ou seja* $\forall p \in P \quad I(p,t_0) \times I(p,t_1) \neq 0, \quad \forall p \in P$.

Na figura 2.5.a mostramos as transições t_1 e t_2 em conflito estrutural. O conceito de conflito efetivo está relacionado à estrutura da rede, como no

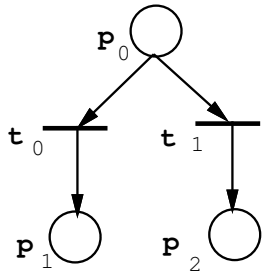

Figura 2.4: Escolha Não-Determinística

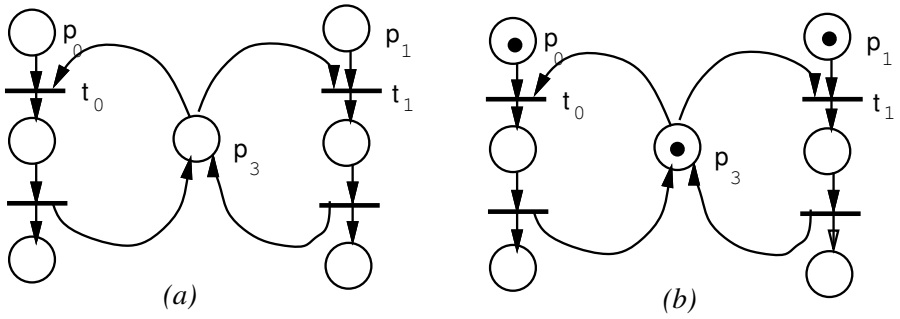

Figura 2.5: Conflito Estrutural e Efetivo

conflito estrutural, no entanto este está associado também à marcação. Se duas transições em conflito estrutural; e o disparo de uma dessas transições inabilita o disparo da outra transição, dizemos que estas transições estão em conflito efetivo (transição t_0 e t_1 da figura 2.5.b).

Definição 2.1.2 - Conflito Efetivo: *Duas transições t_0 e t_1 estão em conflito efetivo para uma marcação M se e somente se estas estão em conflito estrutural em relação ao lugar* p *e $M[t_0>, M[t_1>, M(p) < I(p,t_0)+I(p,t_1)$.*

2.1.5 Atribuição

A atribuição é uma rede elementar[2] que permite que dois ou mais processos possibilitem o disparo de uma mesma transição(ver figura 2.6). Nesta rede o disparo das transições t_0 e t_1 são independentes, modificando, contudo, a marcação do lugar p_2 que é pós-condição tanto de uma transição como da outra. Portanto, após o disparo de qualquer uma dessas transições, cria-se uma condição (marca no lugar de saída) que possibilita a disparo de uma outra transição.

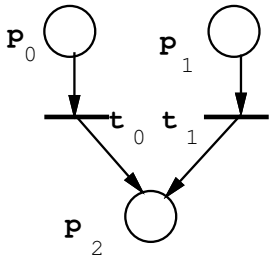

Figura 2.6: Atribuição

2.2 *Confusion*

Nesta seção apresentamos a modelagem de atividades conflitantes em que há também um envolvimento com atividades concorrentes. Como vimos, dois eventos são conflitantes se um ou outro pode ocorrer de forma mutuamente exclusiva. Eventos concorrentes podem ocorrer em qualquer ordem sem

conflito. A situação onde dois eventos estão ao mesmo tempo em conflito e em concorrência é denominada *confusion*.

Na figura 2.7 apresentamos dois tipos de *confusion*. No modelo da figura 2.7.a apresentamos um tipo de confusão denominado *confusion* simétrica. Neste caso as transições t_0 e t_2 são concorrentes, no entanto cada uma destas transições (t_0 e t_2) está em conflito efetivo com t_1, pois o disparo de t_1 impossibilita o disparo de t_0 e t_2.

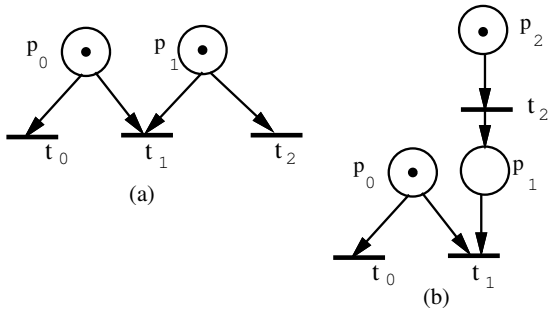

Figura 2.7: *Confusion* Simétrica e Assimétrica

Na figura 2.7.b apresentamos o outro tipo de *confusion*, a *confusion* assimétrica. Nessa rede as transições t_0 e t_2 são concorrentes, no entanto se a transição t_2 disparar antes da transição t_0, teremos um conflito efetivo entre t_0 e t_1.

2.3 Modelagem de Processos

Apresentamos nesta seção diversos problemas clássicos e seus respectivos modelos em redes de Petri [27] [28] [46] [47] [24] [23]. Estes modelos são representados em função das redes elementares apresentadas na seção anterior.

2.3.1 Processos Paralelos

Na modelagem de processos paralelos [37] [38] [31] [42], o modelo do processo global é obtido pela união (composição) de modelos representativos de cada

tarefa que compõem o processo global e os modelos básicos de distribuição e sincronização.

Vejamos, como exemplo, uma linha de produção de canetas esferográficas, e que neste nível da especificação estamos interessados apenas em modelar a produção do invólucro das canetas, das cargas e a montagem destas duas partes. A rede da figura 2.8 representa esta atividade. A transição t_0 representa o início das atividades, e uma marca em p_i indica que as condições para o início dessas atividades são adequadas. O lugar p_0 representa o depósito de matéria prima. Ao disparar-se a transição t_0, uma marca do lugar p_0 e de p_i é consumida e criam-se duas marcas nos lugares de saída, ou seja, p_1 e p_2. Esses lugares representam as condições necessárias (matéria prima, disponibilidade de máquinas e mão de obra etc) para a fabricação da carga e do invólucro, respectivamente. Uma marca no lugar p_1 habilita a transição t_1, assim como a transição t_2 encontra-se habilitada, dado que há uma marca no lugar p_2. A transição t_1 representa a manufatura do invólucro, enquanto a transição t_2 representa a o processo de fabricação da carga. Essas transições podem ser disparadas independentemente uma da outra, dado que a manufatura do invólucro independe da produção da carga, ou seja, nenhuma pré-condição de uma transição é pré-condição da outra transição. Após a manufatura das duas partes que compõem a caneta, é necessário representar a atividade de montagem destas partes. Dado que, só é possível colocar a carga dentro do invólucro (parte externa), quando ambas as partes estiverem prontas, a transição que representa esta montagem deve ter estas duas condições (carga pronta, invólucro pronto) como pré-condições. Representamos esta tarefa através da transição t_3. O disparo da transição t_3 depende das duas pré-condições, ou seja, que haja uma marca no lugar p_3 (invólucro pronto) e p_4 (carga pronta). Após o disparo dessa transição, deposita-se uma marca em p_5 (depósito de canetas) e se restabelece a condição para a manufatura de uma novo produto (marca no lugar p_i), desde que haja matéria prima (marcas em p_0).

Observamos, portanto, que, através das redes elementares (distribuição e junção), foi possível modelar atividades paralelas independentes. Notemos também o carater hierárquico das redes de Petri, que permite um alto nível de abstração, possibilitando o encapsulamento de um número de ações e condições em lugares ou transições. A depender da conveniência, novas subredes podem detalhar partes da rede original.

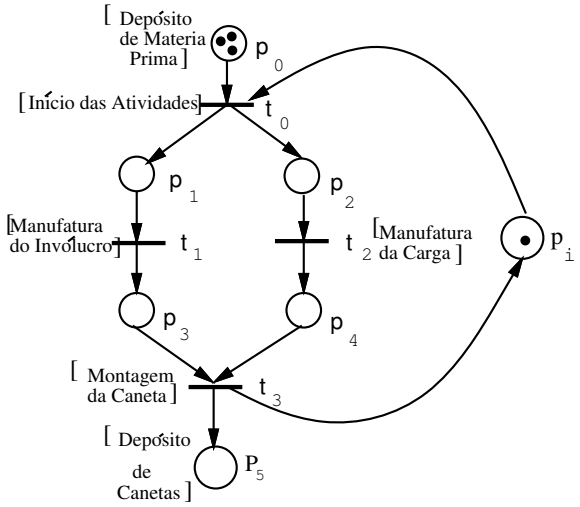

Figura 2.8: Atividades Paralelas

2.3.2 Exclusão Mútua

Para tratarmos completamente os aspectos relativos a resolução de problemas através de processos que tenham atividades paralelas, é necessário estudar os casos onde componentes do sistema cooperam para a obtenção de uma solução conjunta[39]. Isso requer o compartilhamento de recursos entre esses componentes do sistema. Esse compartilhamento deve ser controlado para todo o sistema de forma confiável.

Imaginemos que tenhamos uma máquina única para a produção de cargas e invólucros de canetas esferográficas e que sua utilização em cada uma dessas tarefas impossibilita a outra. Portanto, quando essa máquina estiver sendo utilizada na produção da carga, a linha de produção do invólucro deverá aguardar para que possa utilizar a mesma máquina e vice-versa.

A figura 2.9 apresenta a rede de Petri que representa esse processo. O lugar p_0 representa o depositório de matéria prima, o lugar p_i a habilitação do sistema para a manufatura. Os lugares p_1 e p_2 representam os depositórios de matéria prima para a fabricação dos invólucros e das cargas, respectivamente. Os lugares p_3 e p_4 representam, respectivamente, a manufatura dos invólucros e das cargas. O lugar p_7 representa a máquina compartilhada

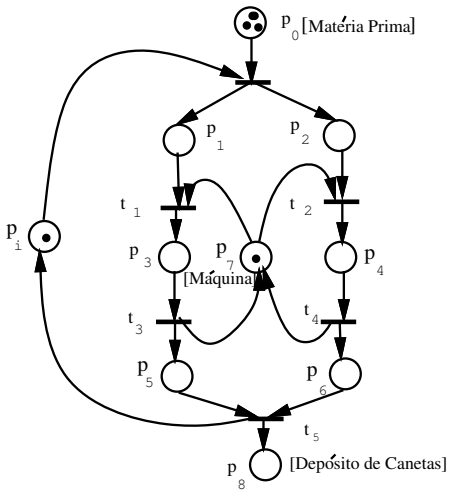

Figura 2.9: Exclusão Mútua

nas linhas de produção, neste caso. Uma marca nesse lugar representa uma máquina está disponível. A ausência de marca neste lugar informa a indisponibilidade da máquina, ou seja, uma das linhas de produção está utilizando essa máquina. Os lugares p_5 e p_6 são os depositórios de invólucros e cargas e o lugar p_8 o depositório de canetas.

A transição t_0 representa o início da manufatura de canetas. Esta primeira tarefa, da linha de produção, corresponde à distribuição de matéria prima para os depositórios p_1 e p_2 para a fabricação de cargas e invólucros. As transições t_1 e t_2 representam o início do processo de manufatura dos invólucros e das cargas, respectivamente. Observe-se que a marcação da figura 2.9 possibilita o disparo de uma das transições, pois p_7 possui apenas uma marca. Com o disparo de uma dessas transições, por exemplo a transição t_1, fica impossibilitado o disparo da transição t_2 (conflito efetivo), pois a ocorrência do disparo de t_1 consome uma marca de p_1 e de p_7 e produz-se uma marca no lugar p_3. Dado que p_7 é entrada da transição t_2, e que este encontra-se sem marca, o disparo de t_2 torna-se impossibilitado. Após o disparo de t_1, apenas a transição t_3 é habilitada. A transição t_3 representa o término de manufatura de um invólucro e a liberação da máquina compartilhada, enquanto a transição t_4 representa o término da fabricação de uma

carga, bem como a liberação da máquina compartilhada. O disparo de t_3 consome uma marca do lugar p_3 e produz uma marca no lugar p_5 e no lugar p_7. De forma semelhante, o disparo de t_4 consome uma marca do lugar p_4 e produz uma marca nos lugares p_6 e p_7. A marca armazenada no lugar p_7 informa a liberação da máquina compartilhada. A transição t_5 representa a montagem das canetas. Observe-se que esta transição consome uma marca do lugar p_5 (invólucro) e uma marca do lugar p_6 (carga) e deposita uma única marca no lugar p_8 e no lugar p_i, que habilita a manufatura de outra caneta, desde que haja matéria prima.

2.3.3 Máquinas de Estado Finito

Uma técnica bastante difundida para a especificação de sistemas de software e hardware são as máquinas de estado finito. Nesta seção, apresentamos a modelagem das máquinas de estado finito através das redes de Petri e suas vantagens [4, 44, 1, 22].

Definição 2.3.1 *Uma Máquina de Estados Finitos pode ser definida pela quíntupla* $(S, A_i, A_o, \delta, \Gamma)$*, onde:*

- S *é um conjunto finito de estados,*
- A_i *é o alfabeto de entrada,*
- A_o *é o alfabeto de saída,*
- $\delta : S \times A_i \rightarrow S$ *é a função que mapeia o estado atual e a entrada no próximo estado (função de próximo estado),*
- $\Gamma : S \times A_i \rightarrow A_o$ *é uma função que mapeia o estado atual e a entrada em uma saída.*

Para modelarmos as Máquinas de Estado Finitos através de redes de Petri, temos que observar a comunicação entre a rede e o exterior. A troca de informações entre o sistema modelado e o exterior pode ser feita de três maneiras: através da associação das condições externas às transições (conforme visto no capítulo anterior), usando-se lugares, ou usando-se transições.

Se utilizarmos transições para representar a comunicação com o meio exterior, a indicação de um símbolo de entrada é representada pelo disparo de

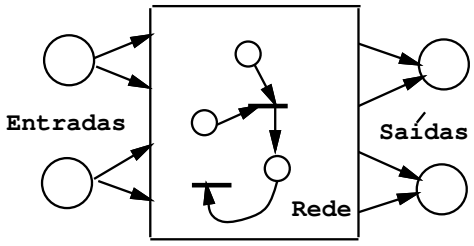

Figura 2.10: Interface usando Lugares

uma transição. As transições de entrada não têm lugares como entrada, assim como as transições de saída não têm lugares de saída. Se representarmos esta comunicação com o exterior através de lugares, cada símbolo de entrada será representado por um lugar. O mundo exterior deposita as marcas nesses lugares. Após a simulação, quando as marcas forem depositados nos lugares de saída, essas serão removidas destes lugares pelo mundo exterior. A figura 2.10 apresenta um modelo que representa esse tipo de comunicação com o mundo exterior.

Para transformarmos uma máquina de estado em uma rede de Petri equivalente, associa-se a cada estado da máquina um lugar na rede e a cada arco da máquina uma transição na rede. Cada arco da máquina de estado representa uma mudança de estado, portanto o estado atual (representado na rede por um lugar) é pré-condição da transição (rede) que representa a mudança de estado. O próximo estado é pós-condição dessa transição. A mudança de estado, bem como o próximo estado atingido, depende tanto do estado atual quanto das entradas do sistema. Dessa forma, se representarmos as entradas através da associação de condições externas às transições, veremos que cada transição da rede terá apenas um lugar de entrada e apenas um lugar de saída.

Como mostrado no início desta seção, é possível representar as entradas e saídas através de lugares, assim como os estados da máquina. O lugar que representa o estado atual é marcado enquanto os demais não estão marcados.

Na figura 2.11, apresentamos uma máquina de estados que representa um sistema que detecta uma seqüência de dois ou mais símbolos 0 de um alfabeto. Representamos o alfabeto de entrada como $A_i = \{0, 1\}$. O estado inicial da máquina é o A. Se um caracter 1 chegar à entrada, a máquina

continuará no estado A e a saída será 0. Caso chegue à entrada um caracter 1, a máquina vai para o estado B e a saída continua com zero. Caso a máquina esteja no estado B, e chegue à entrada um caracter 0, a máquina continuará neste estado e a saída irá para um, indicando a seqüência de dois ou mais caracteres 0's. No entanto, se a máquina estiver no estado B e chegar à entrada um caracter 1, a máquina retorna ao estado A e a saída será 0.

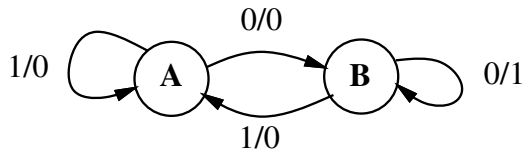

Figura 2.11: Máquina de Estado - Detector de Seqüência

Na figura 2.12 apresentamos a rede correspondente a esta máquina, ou seja, uma rede que detecta a chegada de dois ou mais símbolos 0 colocando a saída em 1. Caso contrário, a saída pemanece em 0.

Podemos, portanto, definir a máquina de estados finito $(S, A_i, A_o, \delta, \Gamma)$ em função das redes de Petri (P, T, I, O, K) da seguinte forma:

Definição 2.3.2 -Máquina de Estados Finitos em Redes de Petri*: seja o conjunto de lugares P os estados, o alfabeto de entrada A_i e o alfabeto de saída $P = S \bigcup A_i \bigcup A_o$. O conjunto de transições representa as ações. Estas ações dependem do estado atual e do alfabeto de entrada $T = \{t_{s,a} | s \in S, a \in A_i\}$. $I(t_{s,a}) = \{s, a\}$ representa as pré-condições para a execução das ações, ou seja, o estado atual e os símbolos do alfabeto de entrada que influenciam na execução de uma ação. $O(t_{s,a}) = \{\delta(s, a), \Gamma(s, a)\}$ representa o próximo estado e o símbolo do alfabeto de saída obtidos após o disparo da transição.*

Se os alfabetos de entrada e saída não forem representados por lugares, como mostrado na rede da figura 2.12, ou seja, as pré-condições e pós-condições afetadas, quando do disparo de uma transição, são representadas através da associação destas às transições(ver seção 1.10), o conjunto de lugares P representa somente os estados.

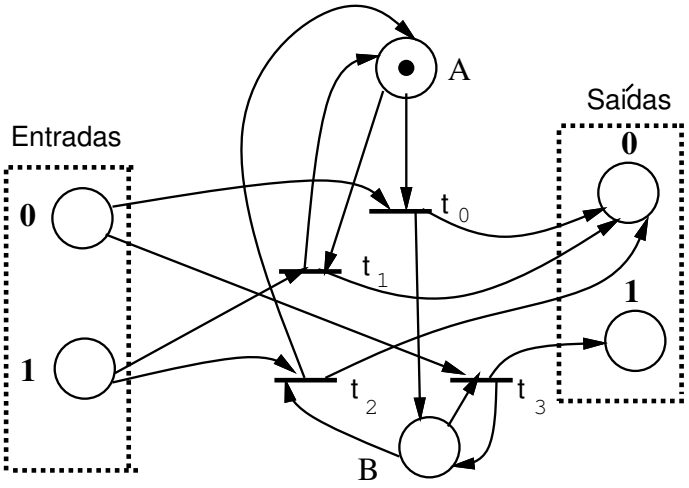

Figura 2.12: Detector de Seqüência

Embora a representação por redes de Petri das máquinas de estado seja menos compreensível, as redes de Petri possibilitam a composição seqüencial e paralela de redes para a construção de modelos mais complexos, através da fusão de lugares de entrada e saída.

2.3.4 Computação Dataflow

As redes de Petri são utilizadas normalmente para modelar fluxo de controle; no entanto possibilitam também a modelagem de fluxo de dados (*dataflow*) [19, 51]. Nesta seção apresentamos a modelagem de computação *dataflow* por redes de Petri.

Em um modelo *dataflow* não há a noção de fluxo de controle seqüencial como em sistemas de computação convencional. Diversas unidades funcionais podem executar suas tarefas ao mesmo tempo, no momento em que os operandos tornam-se disponíveis. Algoritmos projetados com essas características possibilitam um alto grau de paralelismo em máquinas *dataflow*.

A figura 2.13 modela a computação *dataflow* para a expressão $z = e^{(x+y)/y}$. As marcas nas redes representam para a computação *dataflow*

valores ou mesmo a disponibilidades de dados.

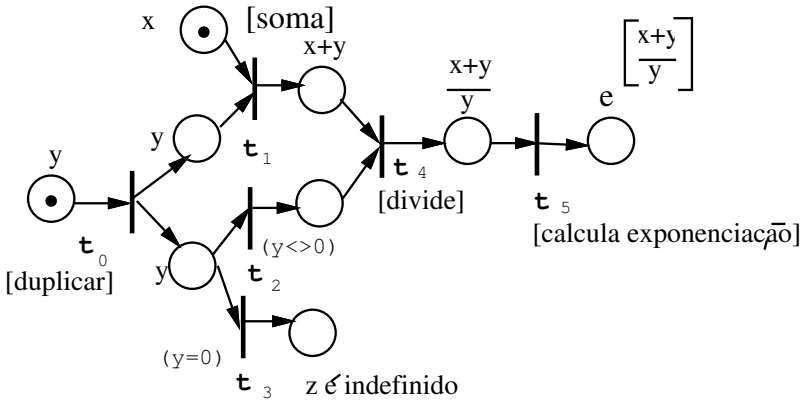

Figura 2.13: Computação *Dataflow*

No exemplo mostrado, temos inicialmente disponíveis os valores x e y. O disparo da transição t_0 provoca a duplicação de y, o que permite a execução concorrente da transição t_1 e t_2 ou t_3, dado que estas transições (t_2, t_3) estão em conflito (escolha). No entanto, embora estas transições estejam em conflito, as duas estão rotuladas com as condições externas $y_1 \neq 0$ e $y = 0$, respectivamente. Caso o valor de y seja zero, dispara-se a transição t_3 que colocará no lugar "*z é indefinido*" uma marca. Se y for diferente de zero, dispara-se a transição t_2. O disparo da transição t_2 armazena uma marca no lugar de saída. Uma marca nesse lugar informa a disponibilidade de um valor diferente de zero para y. O disparo de t_1 representa a realização da operação de adição entre x e y. O disparo de t_4 representa a divisão entre os termos $x + y$ e y. Este disparo deposita uma marca no lugar $(x + y)/y$, o que possibilta o disparo de t_5 (exponenciação), fornecendo uma marca ao lugar $z = e^{(x+y)/y}$, representando que o cálculo desejado está disponível.

Esse estilo de computação possibilita a exploração do paralelismo do sistema. Como observamos, quando as operações são realizadas e há a disponibilidade dos novos operandos, desde que haja unidades funcionais disponíveis, outras operações podem ser realizadas paralelamente. Esta classe de problemas pode ser modelada perfeitamente por redes de Petri, dado que esta técnica possibilita a descrição de sistemas concorrentes e não-determinísticos.

Um ponto de grande importância para a computação *dataflow* são os modelos de arquitetura *pipeline*. Nessas arquiteturas os dados são processados por estágios sucessivos de computação, de tal forma que cada estágio esteja ocupado em cada ciclo de operação.

2.3.5 Sistemas Pipeline

Existem muitas formas para classificar computação paralela. A computação paralela pode ser caracterizada segundo duas perspectivas [34]. No primeiro ponto de vista observamos o particionamento e distribuição das tarefas, no segundo a forma de execução. Observando-se o primeiro dos pontos de vista, podemos caracterizar este particionamento e distribuição de duas formas: a computação *function-parallel* e *data-parallel*. Na computação *function-parallel* decomponhe-se um programa em módulos de diferentes funcionalidades que podem ser executados em um ou vários processadores. Este tipo de paralelismo é adequado para aplicações que podem ser desenvolvidas utilizando-se diversas módulos independentes. Na computação *data-parallel* os dados são divididos entre os processadores. Os processadores podem executar o mesmo programa, no entanto manipulam diferentes subconjuntos de dados.

Se observarmos agora a segunda perspectiva de caracterização dos processos paralelos (execução), podemos classificar as computações como concorrentes e *pipelined*. A computação concorrente explora o paralelismo espacial, ou seja utiliza diversos processadores para a execução de múltiplas tarefas independentes simultaneamente. Estas tarefas podem ser *data-parallel* ou *function-parallel*.

O paralelismo *pipelined* explora os aspectos temporais. Cada processador manipula apenas os dados fornecidos à sua entrada e passa estes dados transformados para o próximo processador (estágio). Uma noção importante nesse tipo de processamento é o fluxo de dados, pois os dados são passados de estágio para estágio, ficando a comunicação entre processadores restrita a estágios vizinhos.

Um sistema *pipeline* é composto por um número de estágios que podem estar em execução simultaneamente. Quando um estágio i encerra sua atividade, este transfere o resultado obtido para o próximo estágio e aguarda por novas informações provenientes de estágios anteriores.

Nesta seção modelamos um sistema *pipeline* com duas unidades funcio-

nais (figuara 2.14). Neste sistema as informações são fornecidas ao sistema através de uma entrada da unidade funcional A.

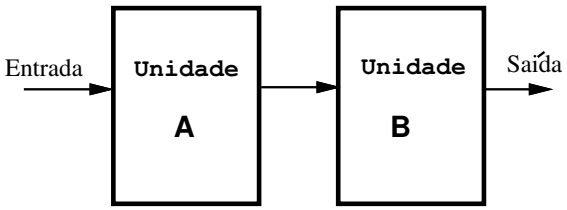

Figura 2.14: Unidades Funcionais em *Pipeline*

Informações só serão aceitas se a entrada estiver desocupada. Após ler os dados da entrada, a unidade funcional A trata estas informações e então tranfere para a entrada da unidade funcional B. A unidade A pode então receber uma nova informação. De forma semelhante a unidade B lê as informações enviadas pela unidade A e então trata as informações, posteriormente fornecendo os resultados à saída do sistema. Na figura 2.15 apresentamos uma rede que descreve as tarefas executadas pelo diagrama em bloco da figura 2.14.

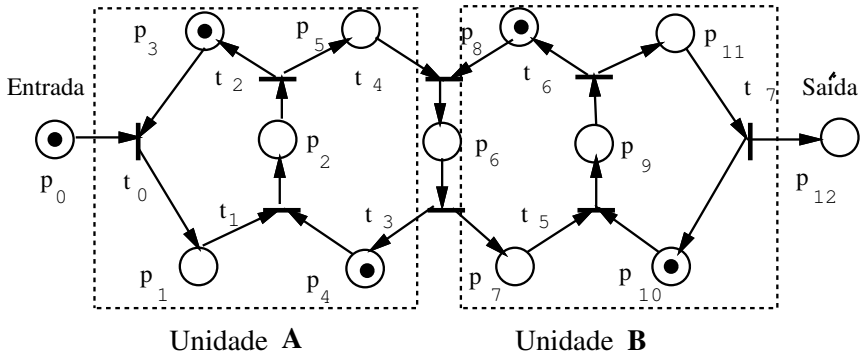

Figura 2.15: *Pipeline*

A entrada do sistema é representada pelo lugar p_0, portanto uma marca nesse lugar indica que há informação a ser tratada pelo sistema. Os lugares

p_1 e p_3, respectivamente, representam a entrada da unidade A com alguma informação para ser tratada e esta entrada vazia. A leitura da entrada da unidade A (lugar p_0) corresponde ao disparo da transição t_0, que só está habilitada quando houver informação disponível e esta unidade estiver habilitada para efetuar a leitura ($M(p_0) \geq 1$ e $M(p_3) = 1$). Os lugares p_4 e p_5 representam a saída da unidade A habilitada (vazia) e com algum dado (bloqueada), respectivamente. A transição t_1, que representa o início das operações a serem realizadas pela unidade A, só pode ser disparada se houver dado na entrada da unidade (p_1) e se a saída desta unidade estiver habilitada (p_4). Uma marca no lugar p_2 representa que a unidade A está em operação. Este lugar pode estar sintetizando um rede complexa que pode representar, em detalhes, as operações e condições desta unidade. A transição t_2 representa a escrita de um resultado na saída da unidade A.

Marcas nos lugar p_7 e p_8 representam que a entrada da unidade B possui dado e está habilitada, respectivamente. A transição t_4 representa a operação inicial para a transferência da informação tratada pela unidade A para a unidade B. O lugar p_6 resume a seqüência de operações relativa à transferência de informações entre a unidade A e a unidade B(uma rede que representa os detalhes desta atividade). A transição t_3 representa a operação final desse processo de transferência.

De forma semelhante, as operações e condições modeladas para a unidade A, a transição t_5 representa o início das operações realizadas pela unidade B. O início dessas operações está condicionado à disponibilidade de dados na entrada da unidade B (marca no lugar p_7) e que a saída desta unidade esteja vazia (lugar p_{10} marcado). O lugar p_9 sintetiza as operações realizadas pela unidade B, a transição t_6 representa a operação de escrita na saída da unidade B. Um dado na saída da unidade funcional B é representado por uma marca no lugar p_{11}. A transição t_7 modela a operação de escrita na saída do sistema (p_{12}), assim como a liberação da entrada entrada da unidade B (marca no lugar p_{10}).

2.3.6 Protocolo de Comunicação

Protocolo de comunição é uma outra área onde as redes de Petri têm sido utilizadas para representar características essenciais, bem como análise de propriedades [14, 30]. Nesta seção apresentamos a modelagem de um protocolo de comunicação bastante simples, no entanto protocolos mais sofisticados podem ser modelados naturalmente usando-se as redes de Petri.

A modelagem de entidades comunicantes, pode ser representada através de uma única transição (figura 2.16.a), ou mesmo explicitando-se a informação por um lugar (figura 2.16.b), através de lugares compartilhados, que representam a informação e o reconhecimento dessa informação (*acknowledgement*) (figura 2.17), ou ainda através de um *buffer* (figura 2.17).

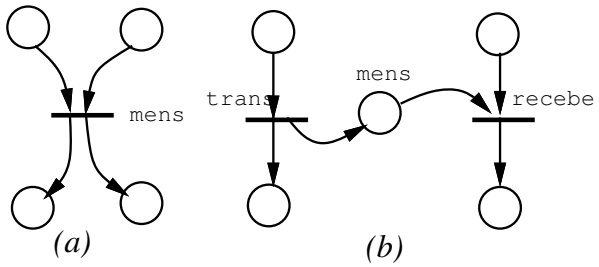

Figura 2.16: Comunicação entre Processos com Reconhecimento e por *Buffer*

A seguir, descrevemos um sistema comunicante composto por três processos, um transmissor e dois receptores. Os receptores recebem mensagens enviadas pelo transmissor. A recepção da mensagem é efetuada de forma não-determinística por um dos sistemas receptores, que então envia o reconhecimento da recepção da informação o que possibilita a transmissão de um nova informação.

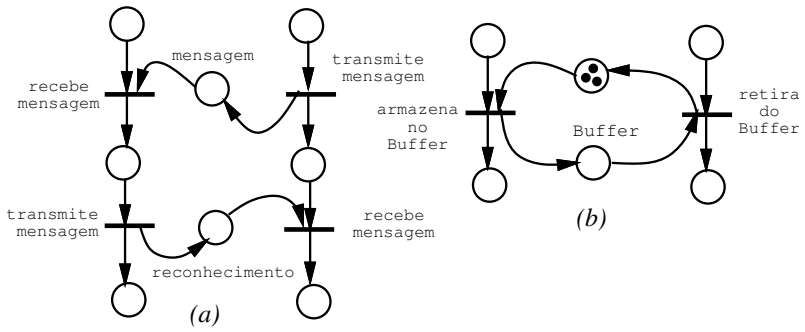

Figura 2.17: Comunicação entre Processos

Na figura 2.18 apresentamos uma rede que representa o protocolo acima descrito. A marca no lugar p_0 habilita a transmissão da mensagem (disparo de t_0). O disparo de t_0 retira a marca do lugar p_0 e deposita uma marca nos lugares p_1 e p_6. A marca no lugar p_6 representa o envio da informação. Esta informação pode ser recebida por um dos processos receptores, de forma mutuamente exclusiva e não-determinística, ou seja, o lugar p_6 marcado possibilita tanto o disparo de t_2 (recepção efetuada pelo $Receptor_1$) quanto o disparo de t_4 (Recepção efetuada pelo $Receptor_2$), pois $M[t_2 >, M[t_4 >$ $e \quad M(p_6) < I(p_6, t_2) + I(p_6, t_4)$, ou seja, a marcação em p_6 não é suficiente para habilitar t_2 e t_4.

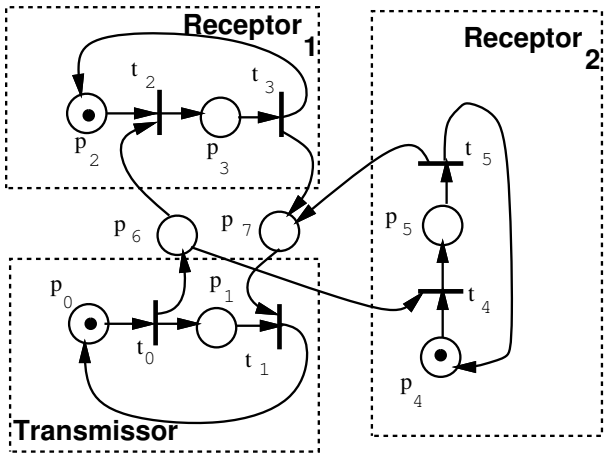

Figura 2.18: Protocolo de Comunicação

O disparo de t_2 retira a marca dos lugares p_2 e p_6 e armazena uma marca no lugar p_3. O reconhecimento da recepção da mensagem é realizado através do disparo da transição t_3. Disparando-se esta transição retira-se a marca do lugar p_3 e deposita-se uma marca nos lugares p_2 e p_7. A marca no lugar p_7 representa a informação de reconhecimento da mesagem enviada. A marca no lugar p_2 restaura a situação inicial do processo $Receptor_1$. A recepção da mensagem pelo processo $Receptor_2$ é realizada de forma similar. A marca no lugar p_7 possibilita o disparo de t_1, dado que há uma marca no lugar p_1. Disparando-se t_1 restaura-se a situação inicial do Transmissor (lugar p_0 marcado), o que possibilita a transmissão de novas informações.

2.3.7 Produtor/Consumidor

Nesta seção apresentamos o problema Produtor/Consumidor [5], onde são observados também aspectos relativos ao compartilhamento de dados. O processo produtor cria objetos que são armazenados em um *buffer*. O consumidor aguarda até que haja objetos disponíveis nesse *buffer* para que os possa consumir.

A rede que modela este problema é esquematizada na figura 2.19, onde podemos observar o sistema produtor, o *buffer* e o consumidor. O número de marcas nos lugares p_0 e p_2 representa o número de processos produtores e consumidores, respectivamente, ou seja, se tivermos, nesses lugares, apenas uma marca em cada, teremos apenas um produtor e um consumidor.

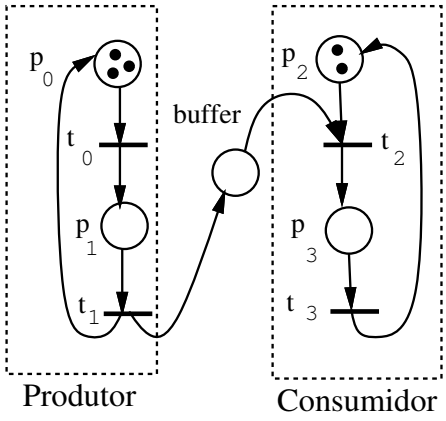

Figura 2.19: Produtor/Consumidor

A transição t_0 representa a produção dos itens e a transição t_1 o armazenamento dos itens no *buffer*. A transição t_2 representa a remoção, pelo consumidor, do i tem do *buffer* e finalmente a transição t_3 representa o consumo do item. Note-se que o disparo de t_1 e t_3 armazena uma marca nos lugares p_0 e p_2, respectivamente. Isso significa que o produtor ou o consumidor está pronto para uma nova execução de suas tarefas. Uma forma alternativa para representar o problema com múltiplos produtores/consumidores seria repetir as sub-redes que os representam, no entanto o modelo teria larga dimensão.

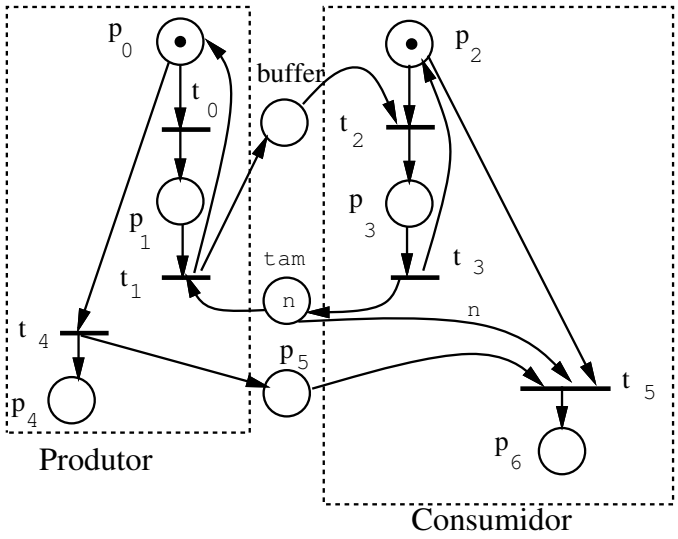

Figura 2.20: Produtor/Consumidor com Buffer Limitado

2.3.7.1 Sistema com *Buffer* Limitado

Apresentamos agora uma variante do problema Produtor/Consumidor, onde temos o *buffer* com tamanho limitado e ainda especificando um estado terminal, ou seja possibilitando aos processos (produtores e consumidores) o encerramento de suas atividades. Na figura 2.20 apresentamos uma rede que modela este problema. Observe-se que este modelo contém a estrutura do exemplo anterior, no entanto um outro lugar é adicionado ao modelo (*tam*). O número de marcas inicialmente depositadas neste lugar representa o tamanho do *buffer*. Quando disparamos a transição t_1, o número de marcas do lugar *tam* é decrementado, pois o número de posições livres no *buffer* decresce. No entanto, quando disparamos a transição t_3 (remoção), o número de marcas no lugar tam é incrementado. Uma marca no lugar p_0 possibilita a produção de um novo item, no entanto também possibilita o término da produção, através do disparo da transição t_4. As transições t_0 e t_4 são rotulados com as condições de *não-término* (*nt*) e *término* (*t*) da produção, respectivamente. O término das operações de consumo só pode ser realizado (disparo de t_5) quando houver uma marca no lugar p_2, o processo produtor tiver encerrado suas operações (lugar p_5 marcado) e o *buffer* estiver vazio

(lugar *tam* com n marcas).

2.3.8 Jantar dos Filósofos

Apresentamos nesta seção a modelagem por redes de Petri do problema do jantar dos filósofos [26] proposto por Dijkstra. Este problema descreve uma situação que consiste de filósofos que podem estar comendo, pensando ou com fome. Os filósofos estão sentados em volta de uma mesa. Cada filósofo tem um *garfo* e um prato de comida a sua frente, no entanto, para que cada filósofo possa comer, é necessário que haja dois *garfos*, ou seja, o seu e o do seu vizinho. O problema em discussão é que, se todos os filósofos pegarem em um mesmo instante o garfo da direita e aguardarem a liberação do garfo a sua esquerda, o sistema entrará em *deadlock* (impasse).

Na figura 2.21 apresentamos um modelo que representa a solução deste problema. Nesta solução representamos os recursos (garfos) por marcas nos lugares $garfo_1$, $garfo_2$ e $garfo_3$. Os estados de cada filósofo são representados pelos lugares, pp_i, pcf_i e pc_i, que identificam os estados *pensando, com fome* e *comendo*, respectivamente. O evento *começar-a-pensar* é representado pela transição tcp_i, assim como *tem-fome* e *começa-a-comer* são representados respectivamente por ttf_i e tcc_i.

Na figura 2.21 vemos que o *filósofo*$_3$ está pensando (lugar pp_3 marcado), que o *filósofo*$_2$ está com fome (lugar pcf_2 marcado) e que o *filósofo*$_1$ está comendo (lugar pc_1 marcado). Quando o *filósofo*$_1$ encerra a refeição, este libera os *garfos*, depositando-os nos lugares $garfo_1$ e $garfo_3$ e vai para o estado *pensando* (marca no lugar pp_1). O *filósofo*$_2$ que está *com fome* pode agora *começar a comer* (disparar a transição tcc_2), pois esta encontra-se habilitada. Ao disparar-se esta transição, são retiradas as marcas dos lugares $garfo_1$ e $garfo_2$, passando este filósofo para o estado *comendo*. O *filósofo*$_1$ e *filósofo*$_3$ que estão pensando, podem então ter fome (disparo de ttc_1 e ttf_3). Ocorrendo isto, é retirada a marca dos lugares pp_1 e pp_3 e armazenadas marcas nos lugares pcf_1 e pc_3. Observe que, neste estado, tanto o *filósofo*$_1$ como o *filósofo*$_3$ podem começar a comer, no entanto se um o fizer o outro fica impossibilitado de o fazer. Note-se, contudo, que todos os filósofos têm a possibilidade de fazer a refeição. Podemos também verificar a inexistência de *deadlock* na rede.

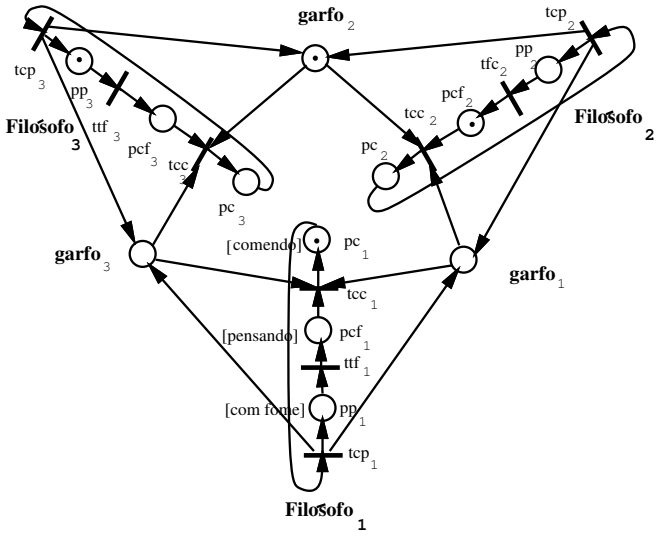

Figura 2.21: O Jantar dos Filósofos

2.3.9 Fluxo de Controle de Algoritmo

Apresentamos nesta seção os modelos básicos (redes) que possibilitam a modelagem por redes de Petri de programas escritos em linguagens seqüenciais [35] [17] [37] [29] [40].

Podemos representar um programa segundo dois aspectos distintos: o controle e a computação. Na computação referimo-nos as operações de atribuição, lógicas e aritméticas, leitura e escrita em dispositivos periféricos e memória. Por outro lado, o controle consiste da ordenação das tarefas, não importando o que cada computação realiza. Sendo as redes de Petri mais adequadas para representar os aspectos de controle, adotamos as redes de Petri para a modelagem dos aspectos de controle de programas seqüenciais. Assim como em um fluxograma, um modelo em redes de Petri é uma abstração do programa e reflete apenas a estrutura, não especificando as computações realizadas.

As linguagens de programação imperativas, de uma forma geral, possuem alguns construtores básicos. Nessas linguagens temos as computações simples, tais como atribuição, operações aritméticas, lógicas e também as

decisões e laços. Nesta seção, apresentamos modelos que representam essas estruturas, bem como o uso dessas estruturas associadas, para a especificação de programas.

2.3.9.1 Computação Seqüencial Simples

Representamos a computações por uma transição, não importando o que esta computação realize.

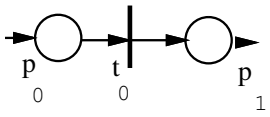

Figura 2.22: Computação Simples

Observando a figura 2.22, verificamos uma transição t_0 que tem como pré-condição o lugar p_0 e como pós-condição o lugar p_1. Na figura 2.23 temos duas instruções de um programa. A execução dessas instruções será de forma seqüencial, ou seja, a instrução $a := e + 1$ é executada antes da instrução $b =: c - 3$, onde após a execução da primeria instrução a segunda será executada. O controle dessas instruções é especificado de forma similar. O seqüencimento das duas intruções é representado pela rede da figura 2.23. Para obtermos o modelo que representa o seqüenciamento dessas instruções, realizamos a fusão dos lugares que são pós-condição da primeira instrução e o lugar que é pré-condição na rede que representa a segunda instrução, ou seja, dos lugares p_{i+1} e p_j.

2.3.9.2 If-then-else

O outro modelo básico na modelagem de fluxo de controle é a decisão (*if-then-else*). Para modelar esta instrução, utilizamos a rede elementar que modela a escolha não-determinística.

As transições são rotuladas com as condições de teste. Na figura 2.24 apresentamos uma rede que modela adequadamente este construtor. Observe-se que uma marca no lugar p_0 habilitaria ambas as transições, no entanto estas transições estão associadas as condições geradas na execução do pro-

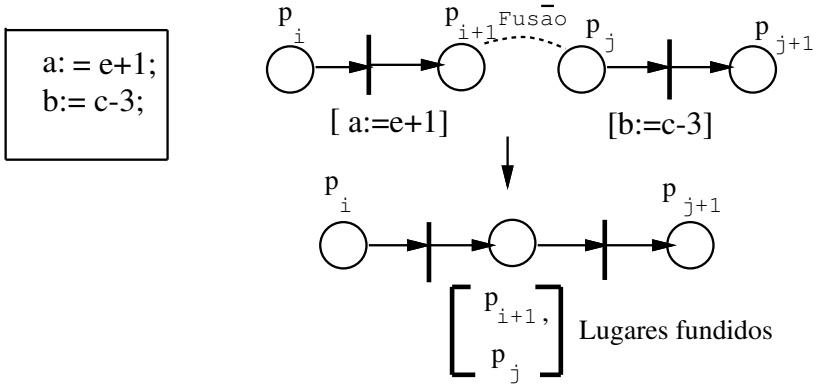

Figura 2.23: Fusão dos Lugares

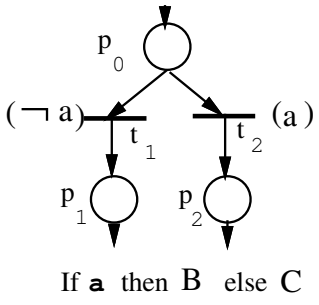

Figura 2.24: If a then B else C

grama. As transições são associadas as condições $\neg a$ e a, ou seja, quando uma destas condições é verdadeira a outra é falsa. Assim sendo, apenas uma dessa transições estará habilitada em um dado instante. Ao disparar-se uma transição, impossibilita-se o disparo da outra, pois retira-se a marca do lugar p_0 e deposita-se uma marca no lugar p_1 ou p_2 dependendo de qual transição tenha sido disparada.

2.3.9.3 While-do

Na figura 2.25 apresentamos a rede que modela o laço *while-do*. Esta modelo é construído a partir da rede anterior (figura 2.24). O corpo do laço é uma rede que tem topologia adequada ao programa. Uma marca no lugar p_0 possibilita o disparo das transições t_1 e t_2. Enquanto a condição a for verdadeira a transição $t2$ será executada. O disparo de t_2 possibilita a execução da rede que representa o corpo do laço.

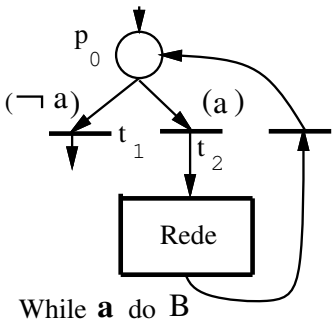

Figura 2.25: While a do B

Após a execução da rede (corpo do laço) dispara-se a transição t_3, que restaura a marca no lugar p_0. Quando a condição $\neg a$ tornar-se verdadeira, a transição t_1 estará habilitada (desde que $M(p_0) \geq I(p_0, t_1)$). O disparo da transição t_1 representa o fim da execução do laço.

2.3.9.4 For-do

O outro tipo de laço que modelamos através de redes de Petri foi o *for-do*. Para modelarmos este laço, fizemos uso de lugares duais de tal forma a

possibilitar o controle do número de iterações a ser efetuado. Na figura 2.26 temos um modelo que representa este laço.

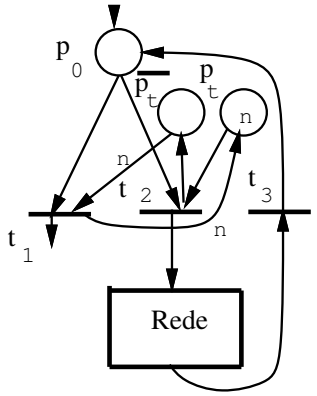

Figura 2.26: For *a n* do *B*

Observe-se que este laço tem estrutura semelhante ao do modelo anterior. No entanto, o controle, na rede elementar que representa a escolha, não é efetuado através de condições externas associadas às transições e sim através do par de lugares (p_t e $\overline{p_t}$), que possibilitam o disparo de uma das transições (t_1 e t_2), exclusivamente. Inicialmente o lugar p_t é marcado com o número de iterações do laço. Quando uma marca chegar ao lugar p_0 a transição t_2 torna-se habilitada. Disparando-se t_2, retira-se uma marca do lugar p_0 e do lugar p_t e depositam-se uma marca no lugar $\overline{p_t}$ e uma outra que hablita a execução da rede que modela o corpo do laço. Quando a execução do corpo do laço termina, a transição t_3 é disparada, restaurando a marca no lugar p_0. Mais uma vez a transição t_2 encontra-se habilitada. Desta forma, enquanto houver marcas no lugar p_t, a transição t_2 será habilitada (desde que haja uma marca no lugar p_0). No entanto, após n iterações não haverá nenhuma marca no lugar p_t, o que impossibilita o disparo de t_2. Para esta marcação a transição t_1 encontra-se habilitada, pois $M(p_0) \geq I(p_0, t_1)$ *e* $M(\overline{p_t}) \geq I(\overline{p_t}, t_1)$. Com o disparo da transição t_1, restaura-se o número de marcas do lugar p_t, ou seja, n marcas, retiram-se n marcas do lugar $\overline{p_t}$ e encerra-se a execução do laço.

2.3.10 Computação-Fraca

A noção de computabilidade fraca em redes de Petri significa que o valor computado por uma rede que calcula uma determinada função, não excede $f(x_1, ..., x_r)$, ou seja, após a execução da rede, o valor fornecido por essa, não será superior a $f(x_1, ..., x_r)$. Essa idéia é importante devido à natureza não-determinística do disparo de transições.

A noção de computabilidade-fraca por redes é apresentada em [59], onde:

Definição 2.3.3 Computação Fraca por Redes de Petri: *Seja uma rede* $R = (P, T, I, O, K)$, *que possui* r *lugares de entrada* (IN_i), *um lugar de saída* (Out), *um lugar de início* (On), *um lugar de finalização* (Off) *e um conjunto finito de lugares* p_i. *R realiza uma computação fraca para um função* $f : N^r \rightarrow N$ *se, e somente se, para cada vetor de entrada* $X \in N^r$ $(X = (x_1, ..., x_i, ..., X_r))$ *e uma marcação inicial* $M_0 \in N$ *temos:*

- $M_0(On) = 1 \quad e \quad M(IN_i) = x_i \quad para \quad 1 \leq i \leq r$,
- $M_0(Out) = M_0(Off) = M_0(p_i) = 0, \quad \forall p_i \quad interno$,
- $\forall M \in A(R, M_0), \quad onde \quad M \neq M_0$, *tem-se* $M(On) = 0 \quad e \quad 0 \leq M(Off) \leq 1 \quad e \quad M(Out) \leq f(x_1, ..., x_r)$,
- $\forall M \in A(R, M_0)$ *tal que* $M(Off) = 1$ *implica a marcação* M *ser morta, ou seja, nenhuma transição está habilitada nessa marcação,*
- $\forall k$, *onde* $0 \leq k \leq f(x_1, ..., x_r)$, *existe uma marcação* $M \in A(R, M_0)$ *tal que* $M(Out) = k$ *e* $M(Off) = 1$.

As redes das figura 2.27 e figura 2.28 provam que as redes de Petri podem realizar a computação fraca de funções. Essas rede computam a adição e multiplicação das variáveis x_1 e x_2, respectivamente.

2.4 Subclasses das Redes de Petri

Nas seções anteriores, apresentamos diversos exemplos de problemas clássicos e seus modelos em redes de Petri. Mostramos, dessa forma, o poder de modelagem das redes de Petri observando diversos aspectos dos sistemas concorrentes. No entanto, as redes de Petri não possibilitam o *teste a zero*

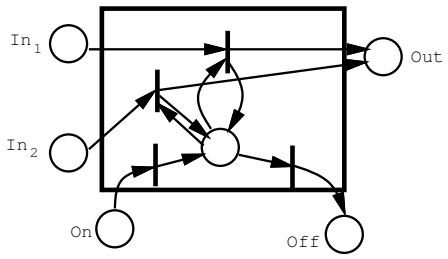

Figura 2.27: Computação-Fraca da Adição

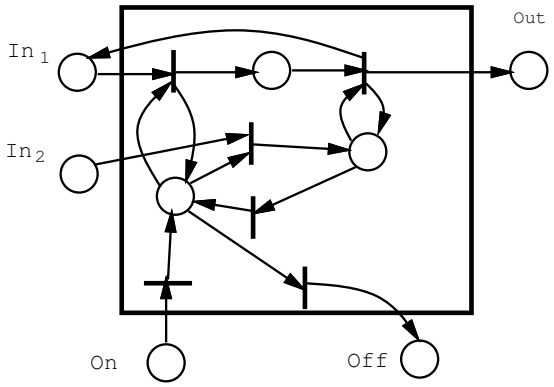

Figura 2.28: Computação-Fraca da Multiplicação

(testar se um lugar não tem marca) de lugares com capacidade ilimitada ($k = \infty$). No capítulo 4 apresentamos algumas extensões às redes de Petri, que possibilitam este teste e fornecem meios para a especificação de outras propriedades dos sistemas computacionais, tais como tempo e prioridade. As extensões que apresentamos são as redes de Petri com arco inibidor, redes de Petri temporizadas determinísticas, redes coloridas e redes hierárquicas. Infelizmente, essas extensões reduzem o poder de decisão sobre as redes de Petri. Alguns trabalhos investigam o uso de subclasses de redes de Petri que possibilitem aumentar estes poder de decisão, sem contudo reduzir em demasia o seu poder de modelagem.

Nesta seção apresentamos as principais subclasses ([1] [4]) das redes de Petri. Nestas subclasses são efetuadas algumas restrições estruturais às redes de Petri, de forma a possibilitar o desenvolvimento de modelos onde o poder de decisão é maior, sem contudo reduzir em demasia o poder de modelagem das redes de Petri.

2.4.1 Redes de Petri Máquina de Estado

A subclasse das redes de Petri denominada *máquinas de estado*, caracteriza-se por restrições ao número de arcos que podem ser entrada e saída de uma transição. Cada transição da rede só pode ter um arco como entrada e um arco como saída.

Definição 2.4.1 - Rede de Petri Máquina de Estado: *seja uma rede* $R_1 = (P, T, I, O, K)$. R_1 *é classificada como máquina de estado se e somente se* $|I(t_j)| = 1$ *e* $|O(t_j)| = 1, \quad \forall t_j \in T$.

Na figura 2.29 temos uma sub-rede que possui as características estruturais de uma rede *máquina de estado*. Observe-se que todas as transições do modelo têm apenas um arco de entrada e um arco de saída. Neste modelo, representamos o conflito e a atribuição, no entanto esta subclasse não possibilita a modelagem de sistemas paralelos e a sincronização.

2.4.2 Redes de Petri Grafo Marcado

A subclasse das redes de Petri denominada *grafo marcado* ou *grafo de eventos* restringe o número de arcos de entrada e saída dos lugares da rede. Cada lugar da rede só pode ter um arco como entrada e um arco como saída.

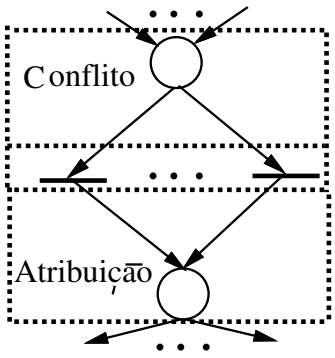

Figura 2.29: Subclasse *Máquina de Estado*

Definição 2.4.2 - Rede de Petri Grafo Marcado: *seja uma rede* $R_1 = (P, T, I, O, K)$. *R_1 é classificada como grafo marcado se, e somente se,* $|I(p_i)| = 1$ *e* $|O(p_i)| = 1, \quad \forall p_i \in P$.

Na figura 2.30 apresentamos uma sub-rede que tem as propriedades estruturais de uma rede *grafo de eventos*. Nessa rede, os lugares possuem apenas uma transição de entrada e uma transição de saída. Obviamente, dado que não podemos ter lugares com dois arcos de saída, esta classe de redes não possibilita a expressão da *escolha*. No entanto, esta classe permite a modelagem de processos paralelos e a sincronização entre processos.

A rede de Petri *grafo marcado* é a representação dual da classe *máquinas de estado*, pois enquanto na classe *máquina de estado* cada transição tem um arco como entrada e um arco como saída, no *grafo marcado* temos o inverso, ou seja, cada lugar tem um arco de entrada e um arco de saída.

2.4.3 Redes de Petri Escolha Livre

A subclasse das redes de Petri denominada *escolha livre*. possibilita a modelagem do conflito (modelado pela *máquina de estado*), do paralelismo e a sincronização (modelados pelo *grafo marcado*), porém de uma forma mais restrita que o modelo geral das redes de Petri. Na literatura temos duas definições distintas de redes *escolha livre*. A primeira delas aparece no trabalho de *Hack* [52] e a segunda delas foi apresentada por *Commoner*, a qual

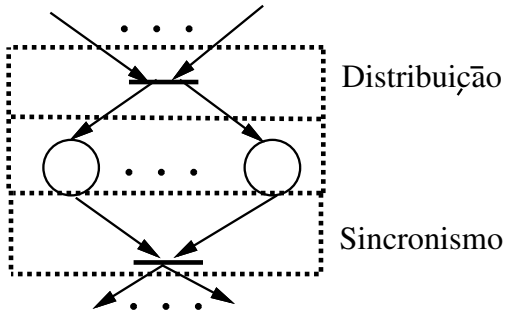

Figura 2.30: Subclasse *Grafo de Eventos*

nos referimos por rede de Petri *escolha livre estendida* [53].

Definição 2.4.3 - Escolha Livre: *seja uma rede* $R_1 = (P, T; I, O, K)$. *Esta rede é classificada como uma rede escolha livre se, e somente se,* $I(t_j) = \{p_i\}$ *ou* $O(p_i) = \{t_j\}$, $\forall t_j \in T$ *e* $p_i \in I(t_j)$.

Esta classe de rede possibilita o controle de eventos em conflitos, pois quando um lugar é entrada de diversas transições, este lugar é a única entrada destas transições. Desta forma, todas as transições estarão habilitadas ou nenhuma estará, possibilitando a escolha da realização do evento livremente.

Definição 2.4.4 - Escolha Livre Estendida: *seja uma rede* $R_1 = (P, T; I, O, K)$. *Esta rede é classificada como uma rede escolha livre se, e somente se, para todo par de lugares* $(p_i, p_k) \in P^2$, $O(p_i) \bigcap O(p_k) \neq 0$ *então* $O(p_i) = O(p_k)$.

Na figura 2.31 temos alguns modelos simples que são classificados conforme as definições apresentadas nesta seção.

Essas formas restritas de conflito definidas apresentam condições necessárias e suficientes para que uma rede marcada dessa classe seja *live* e *segura*(*safe*). Conforme as definições, se uma destas transições está habilitada todas as outras também estão, ou seja, podem ser livremente disparadas.

Neste capítulo vimos os modelos básicos que possibilitam a descrição do seqüenciamento, escolha não-deterministica, paralelismo e sincronismo,

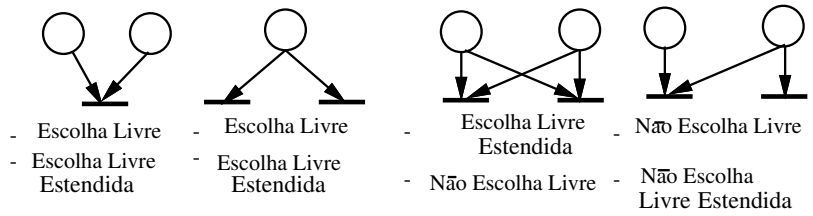

Figura 2.31: Subclasse *Escolha Livre*

assim como a modelagem por redes de Petri de diversos problemas clássicos da ciência da computação, demonstrando assim o poder de modelagem das redes de Petri.

Capítulo 3

Análise das Redes de Petri

Nos capítulos anteriores mostramos o poder de modelagem das redes de Petri. No entanto, podemos ainda formular a seguinte questão: "*O que faremos com esses modelos ?* ". Redes de Petri, não se restringem apenas a uma ferramenta que possibilita a modelagem de problemas que tenham atividades concorrentes. Foi desenvolvida, em paralelo, uma série de métodos que permitem a análise de um grande número de propriedades em sistemas [4] [19] [2]. As propriedades das redes de Petri podem ser divididas em dois grandes grupos: propriedades dependentes da marcação e as não dependentes da marcação, denominadas propriedades comportamentais e estruturais, respectivamente. Neste capítulo apresentamos essas propriedades, bem como uma série de métodos de análise para permitir a verificação das redes de Petri.

Inicialmente, apresentamos as propriedades comportamentais, em seguida as propriedades estruturais e finalizamos o capítulo apresentando alguns métodos de análise.

3.1 Propriedades Comportamentais

Na literatura sobre redes de Petri algumas propriedades têm sido consideradas em profundidade. Aqui apresentamos um estudo sobre as propriedades comportamentais e as propriedades estruturais. Em seguida, serão descritas algumas técnicas de análise utilizadas, para verificação dessas propriedades nos modelos propostos.

3.1.1 Alcançabilidade (*Reachability*)

Alcançabilidade ou *Reachability* é fundamental para o estudo de propriedades dinâmicas de qualquer sistema. A alcançabilidade indica a possibilidade de atingirmos uma determinada marcação pelo disparo de um número finito de transições, a partir de uma marcação inicial. Nesta seção apresentamos este conceito no contexto das redes de Petri [4, 19, 13].

Para a rede de Petri marcada $N = (R, M_0)$, representada na figura 3.1, o disparo de uma transição t_i altera a marcação da rede, conforme as regras descritas no capítulo 1. Uma marcação M' é dita acessível de M_0 se existe uma seqüência de transições que, disparadas, levam a marcação M' (ver seção 1.6). Ou seja, se a marcação M_0 habilita a transição t_0, disparando-se esta transição atinge-se a marcação M_1. A marcação M_1 habilita t_1 a qual sendo disparada atinge-se a marcação M_2 e assim por diante até a obtenção da marcação M'. Definindo-se formalmente, temos:

Definição 3.1.1 - Alcançabilidade: *seja* $M_i[t_j > M_k$ *e* $M_k[t_h > M_l$ *então* $M_i[t_j t_h > M_l$. *Por recorrência designamos o disparo de uma seqüência* $s \in T$ *por* $M[s > M'$. *O conjunto de todas as possíveis marcações obtidas a partir da marcação* M_0 *na rede* $N = (R, M_0)$ *é denotado por* $A(R; M_0) = \{M' \in \mathbb{N}^m | \ \exists \ \ s \ \ | \ \ M_0[s > M'\}$.*(ver seção 1.9), onde m é a cardinalidade do conjunto de lugares.*

O problema da análise desta propriedade consiste em determinarmos se uma dada marcação $M' \in A(R, M_0)$ da rede marcada N. Na figura 3.1 desejamos, por exemplo, saber se a marcação $M' = (0, 0, 0, 1)$ (lugar p_3 marcado) é acessível a partir da marcação inicial apresentada ($M = (1, 0, 0, 0)$) (apenas p_0 com uma marca). Observe-se que atingimos esta marcação tanto executando a seqüência $s' = t_0 t_1 t_3$ como também executando a seqüência $s'' = t_2 t_3$, ou seja, esta marcação pode ser obtida a partir da marcação inicial.

Em alguns casos, desejamos observar apenas alguns lugares específicos da rede em estudo. Estes problema é denominado *submarcação alcançável* (*submarking reachability*). Aqui analisamos se uma dada submarcação $M' \in A(R, M_0)$ da rede marcada N, onde M' é qualquer marcação restrita a um subconjunto de lugares de P.

Muitos outros problemas de análise podem ser observados em termos do problema de *alcançabilidade*. Por exemplo, se uma rede fica em *deadlock*

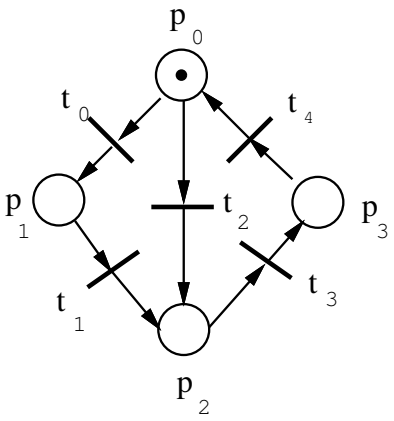

Figura 3.1: Rede N

em uma determinada marcação, podemos querer saber se essa marcação é acessível.

3.1.2 Limitação (*Boundedness*)

Nesta seção apresentamos o conceito de limitação (*boundedness*) nas redes de Petri e sua importância na verificação de uma especificação de sistemas [19].

Definição 3.1.2 - Limitação: *seja um lugar $p_i \in P$, de uma rede de Petri marcada $N_1 = (R, M_0)$. Este lugar são ditos k-limitados (k-bounded) ($k \in \mathbb{N}$) ou simplesmente limitados se para toda marcação acessível $M \in A(R, M_0)$, $M(p_i) \leq k$.*

Caso a propriedade da definição acima não seja observada, o lugar é dito não-limitado. Note que o limite k é o número de marcas que um lugar pode acumular. O lugar p_4 da figura 3.2 é *2-limitado* enquanto o lugar p_3 é *4-limitado*. Uma rede de Petri marcada $N = (R, M_0)$ é dita *k-limitada* se o número de marcas de cada lugar de N não excede k em qualquer marcação acessível de N ($M(p) \leq k$, $\forall M \in GA(R, M_0)$, $\forall p \in P$).

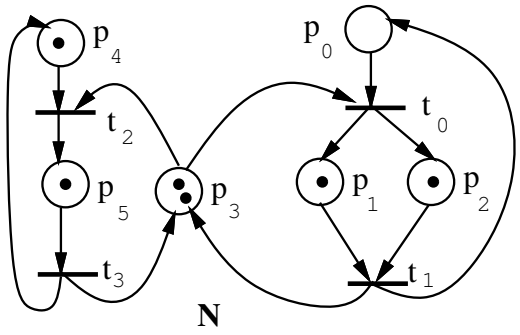

Figura 3.2: Rede Limitada (*Bounded*)

Definição 3.1.3 - Rede Limitada: *dizemos que uma rede* $N_1 = (R; M_0)$ *é limitada (bounded) se* $k(p_i) \leq \infty$ *para todo* $p_i \in P$.

Dado que todos os lugares da rede da figura 3.2 são limitados, então essa rede é classificada como limitada. Ainda podemos concluir que essa rede é $4 - limitada$, pois nenhum lugar acumula mais que 4 marcas em qualquer marcação acessível de N.

3.1.3 Segurança (*Safeness*)

Nesta seção apresentamos a propriedade de segurança (*safeness*) das rede de Petri. O conceito de *safeness* é uma particularização do conceito de *limitação* [4]. Esta propriedade é uma das mais importantes quando na especificação de sistemas digitais.

Como vimos na seção anterior um lugar p_i é dito *k-limitado* se o número de marcas que este lugar pode acumular estiver limitado ao número k. Um lugar que é *1-limitado* pode ser simplesmente ser chamado de seguro (*safe*).

Definição 3.1.4 - Lugar Seguro: *seja um lugar* $p_i \in P$ *de uma rede marcada* $N_1 = (R, M_0)$, *onde* $R = (P, T, I, O, K)$, p_i *é seguro se para toda marcação* $M' \in A(R, M_0)$ *tivermos* $M(p_i) \leq 1$.

Portanto, um lugar pode ser usado para representar uma porta ou mesmo um flip-flop. Dizemos que a rede $N_1 = (R, M_0)$ é segura (binária) se todos

os lugares dessa rede são seguros, ou seja, todos os lugares desta rede podem conter uma ou nenhuma marca.

Definição 3.1.5 - Rede Segura: *uma rede* $N_1 = (R, M_0)$ *é definida como segura se* $M(p_i) \leq 1$, *para todo* $p_i \in P$.

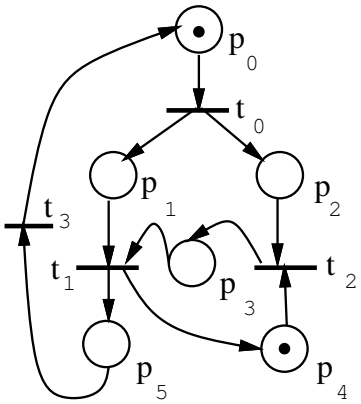

Figura 3.3: Rede Segura (*Safe*)

Na figura 3.3 temos uma rede segura, pois todos os lugares desta rede não acumulam mais do que uma marca.

Em uma rede que tem um lugar ou mesmo uma transição rotulada a uma condição externa especificando a entrada do sistema, conforme a rede da figura 3.6.a, podemos ter lugares que acumulam marcas. A entrada deste modelo é representada pela transição t_0 associada a condição X. Esta transição pode disparar a qualquer momento, à medida em que houver informação (X for verdadeira) para ser tratada. Desta forma, observamos que os lugares p_0, p_1, p_2 e p_4 podem acumular mais do que uma marca, ou seja, esta rede não é segura. Dede que a rede só possua arcos de peso unitário, podemos transformá-la em uma rede segura(*safe*), pela introdução de lugares complementares (duais). Esta transformação é realizada observando os seguintes aspectos:

- Se $p_k \in O(t_i)$ e $p_k \notin I(t_i)$, então cria-se o lugar p'_k, onde $p'_k \in I(t_i)$ e $M(p'_k) = 1$.

Se $p_k \in I(t_j)$, então o lugar $p'_k \in O(t_j)$.

Este novo lugar criado (p'_k) representa o lugar p_k vazio, ou seja, estes lugares são complementares. Quando há uma marca no lugar p_k o lugar p'_k não possui marcas e vice-versa. Dessa maneira conseguimos controlar o número de marcas em cada lugar, pois os lugares duais agem como "travas", impossibilitando o incremento de marcas nos lugares originais.

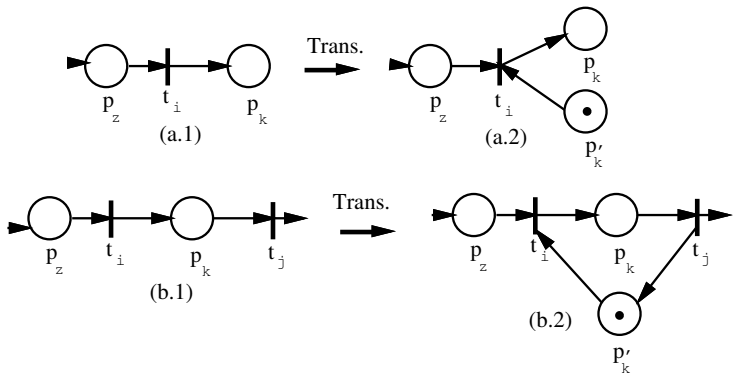

Figura 3.4: Transformação de Rede Não-Segura(*Unsafe*) em Segura(*Safe*)

Dado que o lugar t_i da sub-rede da figura 3.4.a1, potencialmente, pode ser disparada um número infinito de vezes, podemos concluir que o lugar p_k é não-seguro, pois o disparo de t_i incrementa o número de marcas de p_k e como p_k não é entrada de nenhuma transição, a remoção de suas marcas é impossível. Essa rede pode ser convertida em uma rede segura, pela introdução do lugar p'_k, sendo este lugar entrada da transição t_i, onde $M(p'_k) = 1$, conforme apresentamos na figura 3.4.a.2.

De forma semelhante, a rede da figura 3.4.b.1 também é uma rede não-segura, pois o disparo de um número potencialmente infinito de t_i possibilita o acúmulo de um número infinito de marcas em p_k. Para tornar essa rede segura, criamos o lugar p'_k, onde esse lugar é entrada da transição t_i e saída da transição t_j, dado que esta transição é saída de p_k. Como no caso anterior, o lugar p'_k deve ser mono-marcado. Observe-se que os lugares p_k e p'_k são complementares, ou seja, quando há uma marca em p_k não há marca em pk' e vice-versa.

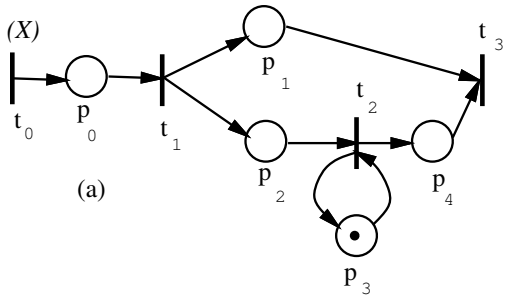

Figura 3.5: Rede Não-Segura

Na figura 3.6 apresentamos a uma rede segura obtida a partir da rede não-segura da figura 3.5 pela introdução dos lugares duais p'_0, p'_1, p'_2 e p_4, marcados com uma marca cada. Observe-se que com disparo de t_0, a marca do lugar p'_0 é retirada, o que desabilita esta transição e deposita uma marca no lugar p_0. A transição t_0 só poderá ser disparada após o disparo da transição t_1, pois com o disparo desta, uma marca é restaurada no lugar p_0. Observe-se, portanto, o comportamento semelhante entre os lugares p_i e seus respectivos lugares duais p'_i do modelo.

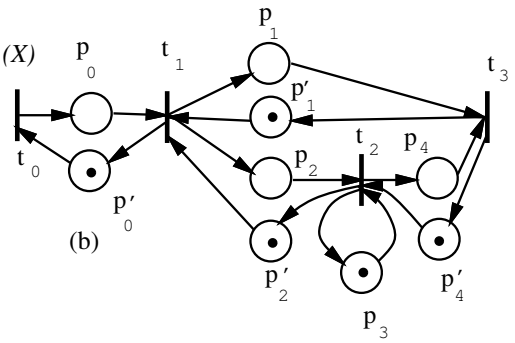

Figura 3.6: Rede Segura

3.1.4 *Liveness*

A ausência de *deadlock* (impasse) em sistemas está fortemente ligada ao conceito de *liveness* [1] [33]. Nesta seção, definiremos o conceito de *liveness* em função das possibilidades de disparo de transições.

Deadlock tem sido fortemente estudado na ciência da computação. Para analisarmos este conceito, observemos a rede de Petri da figura 3.7.a, onde representamos dois processos que compartilham dois recursos representados pelos lugares p_6 e p_7; no entanto se os dois processos precisam desses recursos em um mesmo momento, o sistema pode entrar em *deadlock*. As seguintes seqüências são possíveis, $t_0t_1t_2t_3t_4t_5$, $t_3t_4t_5t_0t_1t_2$, contudo se dispararmos a transição t_0 e em seguida dispararmos t_3 não temos marcas sufucientes para disparar as transições t_1 e/ou t_4, ou mesmo qualquer outra transição, ficando todo o sistema bloqueado ou em *deadlock*. *Deadlock* em uma rede Petri é a impossibilidade do disparo de qualquer transições da rede. A rede da figura 3.7.b não possui *deadlock*, dado que não existe uma marcação que impossibilite o disparo de uma outra transição do modelo.

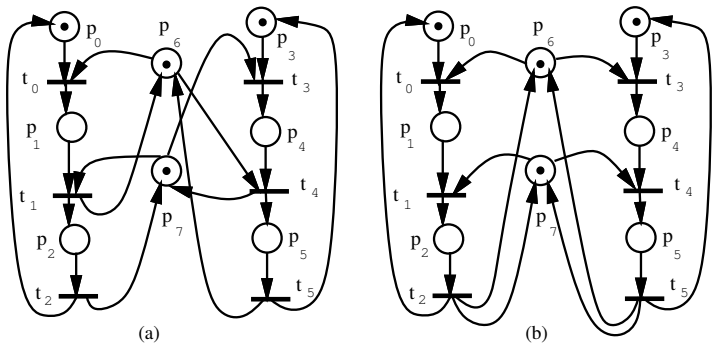

Figura 3.7: Redes *Live* e *Non-Live*

Definição 3.1.6 - Transição Potencialmente Disparável: *chamamos uma transição t_i potencialmente disparável em uma marcação M_0 se existe uma marcação $M' \in A(R, M_0)$ tal que t_i é habilitada para esta marcação.*

Denominamos uma transição t_i *live* em uma marcação M se t_i é potencialmente disparável para todas as marcações $M \in A(R, M)$, ou seja, uma transição é *live* se esta não é passível de *deadlock*.

Definição 3.1.7 - Rede Live: *uma rede $N_1 = (R; M_0)$ é dita live se para toda $M_i \in A(R; M_0)$ é possível disparar-se qualquer transição de N_1 através do disparo de alguma seqüência de transições de $L(R, M_0)$.*

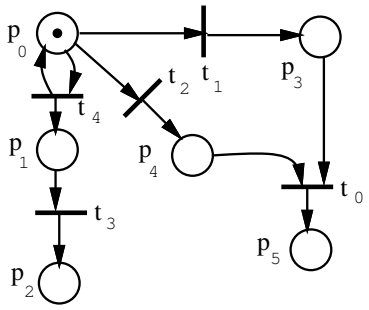

Figura 3.8: Níveis de *Liveness*

Liveness é uma propriedade fundamental para sistemas encontrados no mundo real, no entanto muitas vezes é muito *caro* observar esta propriedade em alguns sistemas de grande porte. Assim sendo, há uma tendência de relaxarmos esta condição e adotarmos níveis de *liveness* para as transições e, conseqüentemente, para as redes [53]. Uma transição t_i de uma rede $N_1 = (R; M_0)$ é denominada:

1. Morta (*dead*) ou nível (n_0), se esta transição nunca poderá ser disparada em qualquer seqüência de $L(R, M_0)$, ou seja, se não existe alguma marcação $M' \in A(R, M_0)$ tal que M' habilita t_i;

2. $N_1 - live$, se pode ser disparada pelo menos uma vez em alguma seqüência de $L(R, M_0)$;

3. $N_2 - live$, se t_i pode ser disparada pelo menos k vezes em alguma seqüência de disparo de $L(R, M_0)$;

4. $N_3 - live$, se t_i aparece um número infinito de vezes em alguma seqüência de disparo de $L(R, M_0)$;

5. $N_4 - live$ ou simplesmente *live*, se t_i é $n_1 - live$ para todas as marcações de $A(R, M_0)$.

Uma rede de Petri $N_1 = (R; M_0)$ é classificada como n_i-*live* se todas as transições na rede são n_i-*live*. Observe-se que uma rede n_4-*live*, corresponde à definição de rede *live* apresentada anteriormente. Ressaltamos ainda que se uma rede é n_4-*live* então esta rede também é n_3-*live* que implica ser n_2-*live* e n_1-*live*. Obviamente, este princípio não se aplica para o nível n_0.

3.1.5 Cobertura (*Coverability*)

A propriedade de cobertura(*coverability*) de uma rede de Petri está fortemente ligada ao conceito de alcançabilidade e de *liveness* apresentado nas seções anteriores.

Quando desejamos saber, por exemplo, se alguma marcação M_i pode ser obtida a partir de uma marcação específica M', temos o problema denominado cobertura de uma marcação.

Uma marcação M' em uma rede é dita coberta se existe uma marcação $M'' \geq M'$ para cada componente dos vetores M'' e M', ou seja, o número de marcas de cada lugar da rede é $M''(p_i) \geq M(p_i)$.

Definição 3.1.8 - Cobertura de uma Marcação: *seja a marcação M' em uma rede $N_1 = (R; M_0)$. M' é dita coberta(coverable) se existe uma marcação $M'' \in A(R, M_0)$ tal que $M''(p_i) \geq M(p_i), \forall p_i \in P$.*

Este conceito é ligado ao conceito de transição potencialmente disparável $(N_1 - live)$. Seja M' a marcação mínima necessária para o disparo da transição t_i, assim sendo, a transição t_i só será disparável$(N_1 - live)$ se M' for passível de cobertura (*coverable*).

Em alguns problemas desejamos apenas observar o comportamento de determinados lugares. Para isto restringimos a pesquisa apenas a estes lugares de particular interesse; neste caso, desejamos observar se uma determinada sub-marcação é passível de cobertura. Este problema é chamado por cobertura de uma submarcação(*submarking coverability*).

3.1.6 Persistência

Uma rede é dita *persistente* se, para qualquer par de transições habilitadas, o disparo de uma transição não desabilita o disparo da outra transição, ou seja, esta transição continua habilitada até o seu disparo [19].

Definição 3.1.9 - Persistência: *seja uma rede marcada* $N_1 = (R; M_0)$. N_1 *é dita persistente se para qualquer duas transições* $(t_j, t_l) \in T^2$, *existe* $M' \in A(R, M_0)$ *tal que* $M'[t_j >$ *e* $M'[t_l >$ *de tal forma que* $M'[t_j > M'' \mid M''[t_l >$ *e vice-versa.*

A noção de persistência é muito importante quando tratamos de sistemas paralelos e em circuitos digitais que tenham atividades assíncronas. A *persistência* está fortemente ligada a subclasse das redes de Petri denominadas *escolha livre*(*free choice nets*), que apresentamos no capítulo anterior.

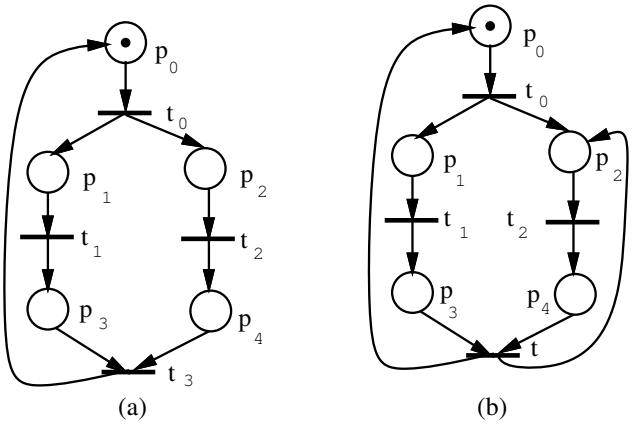

Figura 3.9: Persistência

Vale ressaltar que toda rede da classe *grafo marcado* é uma rede persistente (figura 3.9.a), pois esta subclasse não possui conflito. Contudo, nem toda rede persistente é uma rede *grafo marcado*, como podemos observar na figura 3.9.b. Nessa rede o lugar p_2 possui duas transições de entrada, o que viola a condição para que uma rede seja classificado como *grafo marcado*. Observe-se, contudo, que este modelo não é seguro(*safe*), pois os lugares p_2 e p_4 (figura 3.9.b) podem acumular marcas.

3.1.7 Reversibilidade

Em sistemas computacionais muitas das aplicações têm a propriedade do retorno ao estado inicial ou mesmo a algum grupo de estados [19].

Inicialmente, definimos uma rede como *reversível* se para cada marcação M_i no conjunto das marcações acessíveis a marcação inicial pode ser novamente alcançada. No entanto, em muitas aplicações não desejamos observar se a marcação inicial foi obtida, mas sim se alguma outra marcação específica foi alcançada. Desta forma, enfraquecemos a condição de *reversibilidade*, observando agora o retorno a uma marcação específica, não necessariamente à marcação inicial.

Definição 3.1.10 - Home-State: *seja uma marcação* $M_k \in A(R; M_0)$. M_k *é denominada home-state se* M_k *é acessível de* M_i, *para toda* $M_i \in A(R; M_0)$.

Definição 3.1.11 - Reversibilidade: *uma rede de Petri marcada* $N_1 = (R, M_0)$ *é reversível se existir uma marcação* M_k *acessível de* M_i, $\forall M_i \in A(R; M_0)$.

Na figura 3.10 apresentamos duas redes reversíveis (*a* e *b*) e uma irreversível. Note-se, contudo, que os modelos da figura 3.10.a e da figura 3.10.b mesmo sendo reversíveis não possuem *liveness*, ou seja, é possível retornar ao *home state*; no entanto algumas transições não podem ser disparadas. Na figura 3.10.a a transição t_j nunca será disparada, o mesmo ocorre com a transição t_k da figura 3.10.b. As redes das figura 3.10.c e da figura 3.9.b são *live*, contudo irreversíveis, dado que os lugares p_i e p_2, respectivamente, acumulam marcas, de forma que impossibilitam o retorno ao *home state*. A figura 3.9.a é *live* e reversível.

3.1.8 Justiça (*Fairness*)

Embora existam diferentes pontos de vista sobre o significado justiça (*fairness*) no contexto dos sistemas concorrentes, aqui apresentamos os dois principais conceitos.

Inicialmente, tratamos sobre justiça limitada (*bounded-fairness*) ou *B-fair*). Segundo este ponto de vista, duas transições t_i e t_j são classificadas como *B-fair*, se o número de vezes que uma delas dispara, enquanto a outra não dispara, é limitado.

Definição 3.1.12 - Rede com Justiça Limitada: *seja uma rede marcada* $N = (R; M_0)$. N *é dita B-fair se para todo par de transições* $(t_i, t_j) \in T^2$, *onde* $i \neq j$, *são B-fair.*

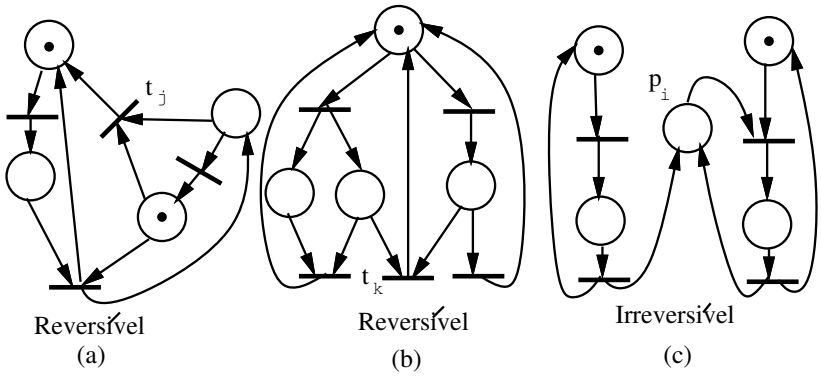

Figura 3.10: Reversibilidade

Na figura 3.11 temos uma rede *B-fair*, pois para cada disparo da transição t_0 dispara-se uma vez a transição t_1 e vice-versa. A mesma relação vale entre as transições t_2 e t_3. Observamos também que se tomarmos os pares de transições: (t_0, t_2), (t_0, t_3), (t_1, t_2) e (t_1, t_3) disparamos t_0 ou t_1 duas vezes para cada disparo de t_2 ou t_3, portanto verificamos que todas os pares de transições desta redes são *B-fair*.

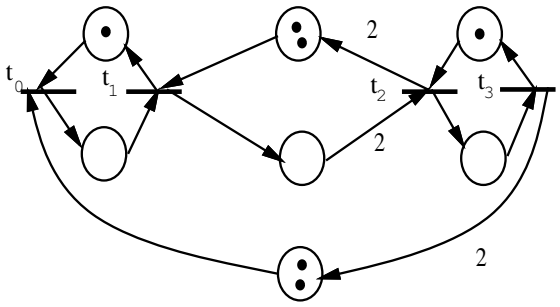

Figura 3.11: *Bounded Fairness*

Um outro conceito é conhecido como justiça(*fairness*) incondicional ou global). Sob este ponto de vista, uma seqüência de transições s_1 é classificada como *fair* incondicional se essa seqüência é finita ou se todas as transições

da rede aparecem um número infinito de vezes nessa seqüência.

Definição 3.1.13 - Rede Fair Incondicional: *seja uma rede marcada* $N = (R; M_0)$ *e* s_i *uma seqüência disparável de transições.* N *é dita Fair incondicional se para toda seqüência* s_i *a partir de* $M_i \in A(R; M_0)$ *é Fair incondicional*

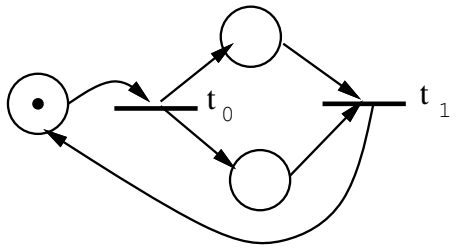

Figura 3.12: *Fairness Incondicional*

Na figura 3.12 temos uma rede *fair* incondicional, dado que as transições que compõem esta rede (t_0 e t_1) aparecem um número ilimitado de vezes na única seqüência de disparável de transições ($t_0, t_1, t_0, t_1, t_0, t_1, ...$).

Tendo apresentado as propriedades comportamentais, na seção seguinte apresentaremos algumas propriedades estruturais das redes de Petri, ou seja propriedades que refletem apenas características da estrutura dos modelos, não sendo influenciadas pela marcação.

3.2 Propriedades Estruturais

Muitas vezes estamos interessados em observar propriedades que estão apenas relacionadas com a estrutura dos modelos. As propriedades estruturais são aquelas que refletem características independentes da marcação.

Desde que as redes sejam puras, a estrutura da rede pode ser representada pela matriz de incidência. Nesta seção, consideramos todas as redes como puras (vimos na seção 1.5 que podemos transformar uma rede com *self-loops* numa rede pura). Serão vistos os conceitos de limitação estrutural, conservação, repetitividade e consistência.

3.2.1 Limitação Estrutural

Uma rede de Petri $R = (P, T, I, O, K)$ é classificada como estruturalmente limitada(*structural bounded*) se é limitada para qualquer marcação inicial.

Teorema 3.2.1 *Uma rede de Petri $R = (P, T, I, O, K)$ é estruturalmente limitada se, e somente se, existe um vetor de inteiros positivos W com dimensão $\#P$ tal que $W.C \leq 0$.*

Seja uma marcação $M \in A(R, M_0)$, de modo que, multiplicando-se o vetor W pela equação fundamental das redes de Petri, temos:

$$W.M = W.M_0 + W.C.\overline{s}$$

Sendo $W.C \leq 0$ e $\overline{s} \geq 0$, temos:

$$W.M \leq W.M_0$$

Então a marcação de cada lugar da rede é limitada a

$$M(p) \leq (W.M_0)/W(p)$$

onde $W(p)$ é p-ésimo componente do vetor W.

3.2.2 Conservação

A *conservação* é uma importante propriedade das redes de Petri permitindo, por exemplo, a verificação da não-destruição de recursos através da simples conservação de marcas [20] [21].

Na figura 3.13 apresentamos uma rede na qual observamos esta propriedade. O disparo de qualquer transição desta rede não altera o número de marcas, ou seja, recursos não são criados nem destruídos.

Definição 3.2.1 - Rede Estritamente Conservativa: *Seja uma rede marcada $N = (R; M_0)$. N é dita estritamente conservativa se temos $\Sigma M_i(p_k) = \Sigma M_0(p_k)$, $\forall p_k \in P$, $\forall M_i \in A(R; M_0)$.*

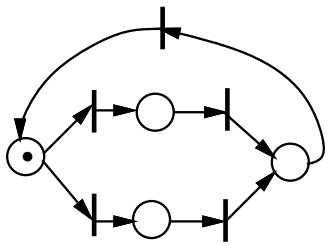

Figura 3.13: Estritamente Conservativa

Para que esta propriedade seja observada, é necessário que $\#I(t_j) = \#O(t_j)$, pois redes com esta estrutura não possibilitam alteração do número de marcas da rede.

Na rede da figura 3.14 apresentamos duas linhas de montagem que compartilham duas máquinas, representadas pelas marcas no lugar p_4. A linha de montagem A ao iniciar suas operações, requisita as duas máquinas (observar o peso do arco entre p_4 e t_1). Caso essas duas máquinas não estejam disponíveis, a linha de montagem aguardará até que esta condição seja satisfeita. Ao encerrar suas tarefas, as máquinas são liberadas. A linha B, no entanto, solicita apenas uma máquina, pois o peso do arco entre p_4 e t_2 é 1. De forma semelhante à linha A, quando a linha de montagem B encerra suas operações, esta libera a máquina requisitada.

Esta rede não pode ser classificada como uma rede *estritamente conservativa*, pois o número de marcas da rede é alterado quando disparamos as transições. Por exemplo, ao dispararmos a transição t_1 as marcas dos lugares p_0 e p_4 são consumidas e é produzida apenas uma marca no lugar p_2. Contudo, é possível transformar essa rede numa rede *estritamente conservativa*, através da alteração dos valores dos arcos da rede, de forma que a restrição acima ($\#I(t_j) = \#O(t_j)$) seja verdadeira. Na figura 3.14.b apresentamos a rede da figura 3.14.a transformada para atender a esta restrição. Isto significa que o número de arcos de saída de t_1 e t_2, respectivamente é alterado para 3 e 2, assim como os arcos de entrada de t_3 e t_4 para 3 e 2, respectivamente.

A *conservação* nas redes de Petri não se restringe, contudo, à manutenção de um somatório de marcas da rede ser constante. Como vimos anteriormente, dependendo da estrutura da rede, podemos converter uma rede que não seja estritamente conservativa numa rede *estritamenta conservativa*. Redes

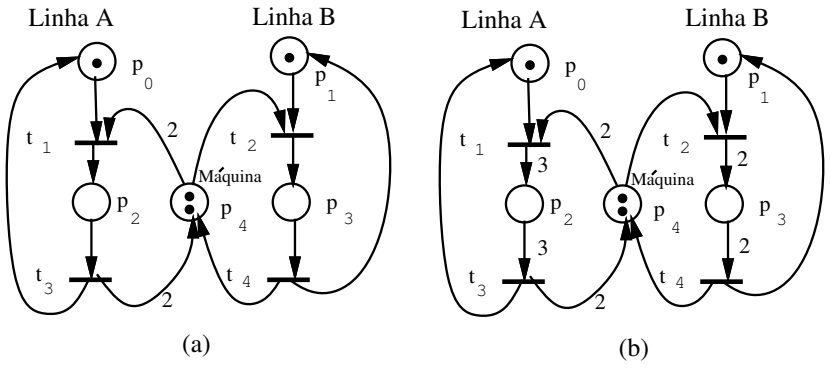

Figura 3.14: Conservação

com estas características, ou seja, que podem ser transformadas em redes estritamente conservativas, são denominadas *conservativas*. Uma marca em um lugar pode representar diversos recursos que pode ser usado posteriormente para criar diversas marcas através do disparo de transições. O que temos de fato é um somatório ponderado constante. Associamos a cada lugar da rede um peso (valor inteiro). Este peso é multiplicado pelo número de marcas de cada lugar e então somado; este somatório deve permanecer constante. São associados pesos zero a lugares que não são importantes. Dado que todas as redes são conservativas, se os pesos associados a todos lugares da rede forem zero, esta evidência não nos fornece informação relevante. Desta forma, uma rede é dita *conservativa* se esta é conservativa para algum vetor de peso de inteiros positivos.

Definição 3.2.2 - Rede Conservativa: *uma rede marcada* $N = (R; M_0)$ *é dita conservativa com relação a um vetor de pesos* $W = (w_1, w_2, ..., w_n)$, *se* $\Sigma w_i.M_k(p_i) = \Sigma w_i.M_0(p_i)$, *onde* $n = \#P$ *e* w_i *é um inteiro positivo,* $\forall p_i \in P \quad e \quad \forall M_k \in A(R; M_0)$.

O teorema C.0.2, no apêndice C, mostra que uma rede é conservativa se, e somente se, existe um vetor característico W de inteiros positivos tal que $C \cdot W = 0$.

Como vimos, a rede da figura 3.14.a não é uma rede *estritamente conservativa*, contudo esta rede é *conservativa* para um vetor de pesos $W =$

$(1,3,1,2,1)$, pois a soma ponderada permanece constante para qualquer disparo de transição ($\Sigma w_i.M_k(p_i) = 4, \quad \forall M_k \in A(R;M_0)$). Observe-se que uma rede *estritamente conservativa* é uma particularização das redes *conservativas*, onde o peso associado a cada lugar é 1.

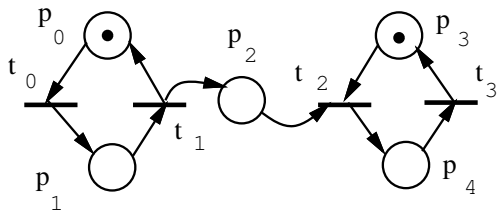

Figura 3.15: Rede não Conservativa

A rede da figura 3.15 é uma rede não-conservativa, pois não há um vetor de pesos com valores inteiros positivos que satisfaça a propriedade de conservação. Através do vetor $W = (1,1,1,1,0)$ obtemos um somatório constante de marcas, para isso, no entanto, associamos ao lugar p_5 um peso 0, o que vai de encontro à definição de rede conservativa. Dizemos contudo que esta rede tem componentes conservativos (dado que este vetor é uma solução) ou ainda que esta rede é *parcialmente conservativa.*

Definição 3.2.3 - Rede Parcialmente Conservativa*: uma rede marcada $N = (R;M_0)$ é dita parcialmente conservativa com relação a um vetor de pesos $W = (w_1, w_2, ..., w_n)$, para um $W \neq 0$, onde $n = \#P$ e w_i é um inteiro não-negativo, se $\Sigma w_i.M_k(p_i) = \Sigma w_i.M_0(p_i), \forall p_i \in P \quad e \quad \forall M_k \in A(R;M_0)$.*

O teorema C.0.3, no apêndice C, mostra que uma rede é parcialmente conservativa se, e somente se, existe um vetor característico W de inteiros não-negativos tal que $C \cdot W = 0$ e $W \neq 0$.

Na rede que apresentamos na figura 3.15, verificamos a presença de componentes conservativos (*parcialmente conservativa*). Note-se que nos lugares p_0, p_1 e p_3, p_4 o número de marcas pemanece constante.

O conservacionismo de uma rede indica a necessidade de recursos. Por exemplo, se utilizarmos uma rede de Petri para moldar o comportamento de um programa, se a rede for conservativa, sabemos que o programa necessita de espaço constante para a sua execução.

3.2.3 Repetitividade

Uma rede marcada é classificada como *repetitiva* se para uma marcação e uma seqüência de transições disparáveis, para esta marcação, todas as transições da rede são disparadas ilimitadamente.

Definição 3.2.4 - Rede Repetitiva: *seja $N = (R; M_0)$ uma rede marcada e s uma seqüência de transições.* N *é dita repetitiva se existe uma seqüência s tal que $M_0[s > M_i$ e toda $t_i \in T$ dispara um número infinito de vezes em* s.

O teorema C.0.4, no apêndice C, mostra que uma rede é repetitiva se, e somente se, existe um vetor característico $\overline{s}$ de inteiros positivos $C.\overline{s} \geq 0$ e $\overline{s} \neq 0$.

Suponha que existe um vetor $\overline{s}$ onde cada componente deste vetor é um inteiro positivo, tal que $M_i - M_0 = C.\overline{s} \geq 0$ isto significa que a seqüência s pode ser repetida indefinidamente.

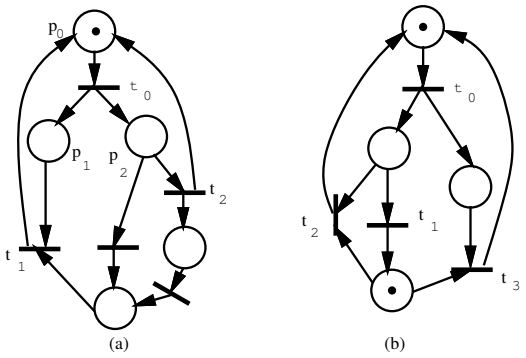

Figura 3.16: Repetitividade

A figura 3.16.a apresenta uma rede repetitiva. Observe-se que quando disparamos a transição t_0 geramos duas marcas, uma no lugar p_1 e uma no lugar p_2, o que possibilita o disparo das demais transições. Se dispararmos a transição t_2 e a transição t_1 o lugar p_0 aumentará sua marcação com relação a sua marcação inicial.

Definição 3.2.5 - Rede Parcialmente Repetitiva: *seja* $N = (R; M_0)$ *uma rede marcada e s uma seqüência de transições.* N *é dita parcialmente repetitiva se existe uma seqüência s tal que* $M_0[s > M_i$ *e algumas transições* t_i *disparam um número infinito de vezes em* s.

O teoremaC.0.5, no apêndice C, mostra que uma rede é parcialmente repetitiva se, e somente se, existe um vetor $\overline{s}$ não-nulo, cujos componentes são números naturais e $C.\overline{s} \geq 0$.

Na figura 3.16.b apresentamos uma rede parcialmente repetitiva. Nessa rede, as transições t_0, t_1 e t_3 podem ser disparadas indefinidamente, no entanto a transição t_2 só pode ser disparada uma vez.

3.2.4 Consistência

Uma rede de Petri tem a propriedade de consistência se disparando uma seqüência de transições a partir de uma marcação inicial M_0 retorna-se a M_0, porém todas as transições da rede são disparadas pelo menos uma vez.

Definição 3.2.6 - Consistência: *seja* $N = (R; M_0)$ *uma rede marcada e s uma seqüência de transições. N é dita consistente se* $M_0[s > M_0$ *e toda transição* t_i *dispara ao menos uma vezes em s.*

O teoremaC.0.6, no apêndice C, mostra que uma rede é consistente se, e somente se, existe um vetor $\overline{s}$ não-nulo de inteiros positivos tal que $C.\overline{s} = 0$.

Suponhamos que existe um vetor característico $\overline{s}$ de inteiros positivos, tal que ao disparar-se uma seqüência de transições retorna-se à marcação inicial $M_0 = M_0 + C.\overline{s} = 0$, isso significa que a seqüência pode ser repetida.

Definição 3.2.7 - Consistência Parcial: *seja* $N = (R; M_0)$ *uma rede marcada e s uma seqüência de transições. N é dita parcialmente consistente se* $M_0[s > M_0$ *e alguma* $t_i \in T$ *dispara ao menos uma vez em s.*

O teorema C.0.7, no apêndice C, mostra que uma rede é parcialmente consistente se, e somente se, existe um vetor $\overline{s}$ não-nulo de naturais tal que $C.\overline{s} = 0$.

3.3 Análise das Redes de Petri

Na seção, anterior apresentamos as propriedade estruturais relacionadas às redes de Petri. Dedicamos esta seção ao estudo dos métodos de análise [19] [4] [9] das redes de Petri. Esses métodos serão utilizados para a verificação das propriedades apresentadas anteriormente. Os métodos de análise das redes de Petri podem ser classificados em três grupos: análise baseada na árvore de cobertura, os métodos baseados na equação fundamental das redes de Petri e as técnicas de redução.

3.3.1 Árvore de Cobertura

O método de análise denominado *Árvore de cobertura* baseia-se na construção de uma árvore que possibilite a representação de todas as possíveis marcações de uma rede.

Para uma dada rede de Petri, com uma marcação inicial, é possível obtermos diversas marcações para um grande número de transições potencialmente habilitadas [15]. Essas marcações podem ser representadas por uma árvore, onde os nós são as marcações e os arcos as transições disparadas. Algumas propriedades, tais como limitação (*boundedness*), segurança (*safeness*), transição morta (*dead*) e alcançabilidade de marcações [32], podem ser analisadas através da árvore de cobertura.

A árvore de cobertura é um gráfico utilizado para representar finitamente um número infinito de marcações. Para possibilitar a representação finita das marcações, através da árvore, utilizamos o símbolo especial ω, apresentado na seção 1.1.

Definição 3.3.1 - Árvore de Cobertura: *seja uma rede de Petri marcada* $N = (P, T, I, O, K, M_0)$. *Define-se árvore de cobertura pelo par* $AC = (S, A)$, *onde* S *representa as marcações e* A *os arcos rotulados por* $t_j \in T$.

A árvore de cobertura pode ser construída seguindo-se o algoritmo abaixo:

1. Rotule a marcação inicial M_0 como a raiz e descrimine-a como *nova*.

2. Enquanto houver marcações *nova*s, faça:

- Selecione uma *nova* marcação M.
  - Se M é idêntica a uma marcação no caminho desde a raiz até M, então rotule-a como *antiga* e selecione uma nova marcação.
- Se nenhuma transição está habilitada para M, rotule-a esta marcação como *final*.
- Enquanto houver transição habilitada para M, faça o seguinte para cada transição habilitada:
  - Obtenha a nova marcação M' que resulta do disparo de uma transição t habilitada para M.
  - Se no caminho da raiz para a marcação M existe uma marcação M'' tal que para todo lugar $p \in P \quad M'(p) \geq M''(p)$ e $M' \neq M''$ (M'' é acessível de M') então substitui-se $M'(p)$ por ω para cada p tal que $M'(p) > M''(p)$.
  - Introduza M' como um nó e crie um arco rotulado com t de M para M' e rotule a marcação M' como *nova*.

Para uma rede de Petri limitada, a árvore de cobertura é denominada *árvore de alcançabilidade*, dado que esta contém todas as possíveis marcações da rede. A figura 3.17.b mostra a árvore de cobertura para a rede *Ri*.

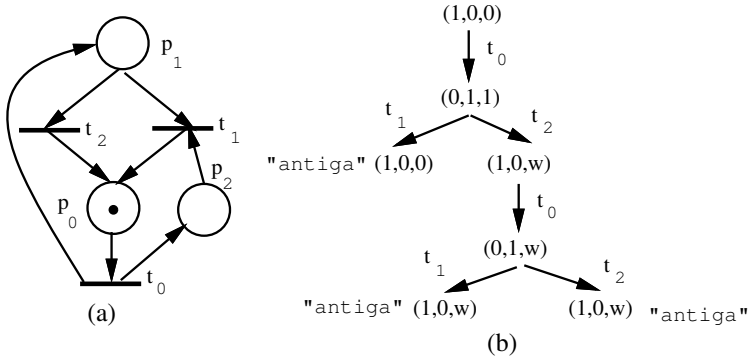

Figura 3.17: Árvore de Cobertura

Uma outra possível representação finita das marcações acessíveis pode ser obtida através do grafo de cobertura [2].

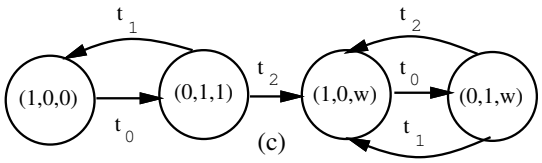

Figura 3.18: Grafo de Cobertura

Definição 3.3.2 - Grafo de Cobertura: *seja uma rede de Petri marcada* $N = (P, T, I, O, K, M_0)$, *define-se grafo de cobertura pelo par* $GC = (S, A)$, *onde* S *representa o conjunto de todos nós na árvore de cobertura e* A *os arcos rotulados por* $t_j \in T$ *representando todas os possíveis disparos de transição tal que* $M_i[t_k > M_j$, *onde* $M_i, M_j \in S$.

Se a rede é limitada o grafo de cobertura é denominado de *grafo de alcançabilidade*. Na figura 3.18 apresentamos o grafo de cobertura da rede da figura 3.17.a.

Podemos observar que essa rede não é uma rede *bounded*, dado a presença do símbolo ω na árvore de cobertura, no entanto podemos observar que os lugares p_0 e p_1 são lugares seguros(*safe*), dado que esses não acumulam mais do que uma marca.

Na figura 3.19.b temos a árvore de cobertura da rede da figura 3.19.a. Analisando-se a árvore constatamos que a rede é segura(*safe*) e não há transições mortas(*dead*). Observe-se também que esta rede é reversível ao estado *home*.

Em redes limitadas(*bounded*) a árvore de cobertura pode ser denominada de *árvore de alcançabilidade*, dado que esta contém todas as possíveis marcações acessíveis da rede, o mesmo ocorrendo com o grafo de cobertura (ver figura 3.20).

Embora, através desse método de análise, possamos analisar as propriedades de limitação(*boundedness*), segurança(*safeness*), conservação e problemas de cobertura de marcações(*coverability*), não possibilita a resolução de *liveness* e alcançabilidade(*reachability*) e reversibilidade em redes não-limitadas. Este problema está relacionado à introdução do símbolo ω que é uma informação da qual não sabemos o seu real valor. Como podemos observar através das redes da figura 3.21, essas redes têm a mesma árvore de

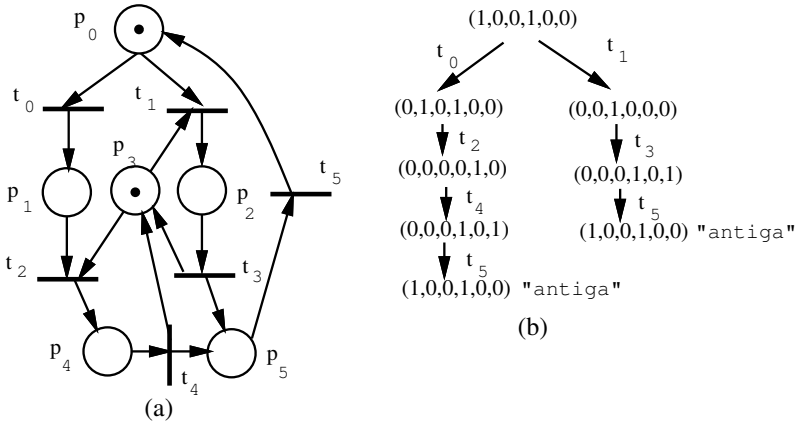

Figura 3.19: Árvore de Cobertura - Verificação de Propriedades

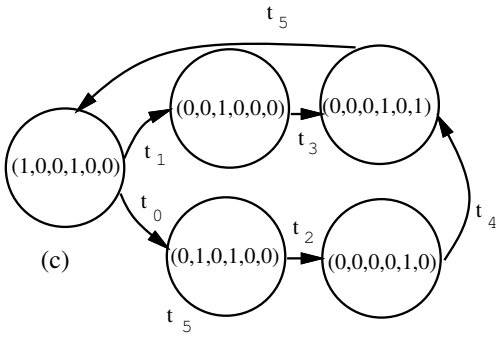

Figura 3.20: Grafo de Cobertura - Verificação de Propriedades

cobertura, no entanto o lugar p_2 da rede N_2 acumula um número de marcas que é um inteiro não-negativo múltiplo de 2, enquanto o mesmo lugar da rede N_1 acumula qualquer valor inteiro não-negativo. A árvore de cobertura não permite detectarmos essa característica, dado que o ω "mascara" esta informação.

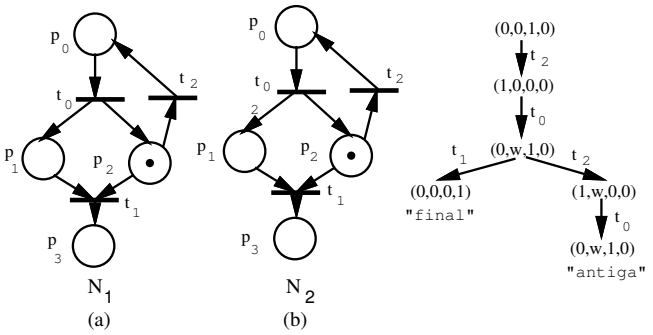

Figura 3.21: Problemas com a Alcançabilidade

Na figura 3.22 temos duas redes que têm mesma árvores de cobertura, porém a rede da figura 3.22.a é *live* e a da figura 3.22.b não, pois, se a seqüência $t_1t_2t_3$ for disparada, o sistema fica em *deadlock*.

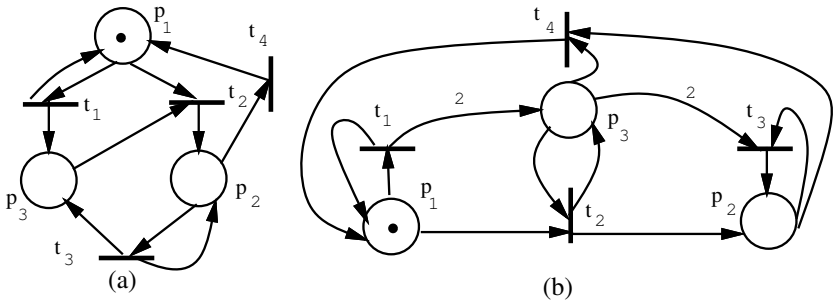

Figura 3.22: Problemas com *Liveness*

A árvore de cobertura não possibilita verificarmos propriedades de *li-*

veness e de *alcançabilidade* em redes genéricas (com qualquer estrutura e marcação), contudo possibilita a verificação dessas propriedades em redes mais restritas [4, 19].

3.3.2 Análise Sobre a *EFRP*

A equação fundamental das redes de Petri(EFRP) ou equação de estados possibilita a verificação de diversas propriedades de modelos [8]. Nesta seção, utiliza-se esta equação para analisar a acessibilidade das marcações, bem como o número de vezes que cada transição tem que ser disparada para atingir-se essa marcação. Além do processo de análise, apresentado nesta seção, através da equação fundamental das redes de Petri, são derivados outros métodos para a análise de propriedades estruturais, tais como análise dos invariantes de transição e de lugar, que serão apresentados nas seções seguintes.

A equação fundamental das redes de Petri provê meios para verificar a *acessibilidade* às marcações. Nesta seção, assumiremos que as redes estudadas são redes puras ou mesmo transformadas em redes puras através da inserção de lugares duais visando formar pares *dummy*, dado que a matriz de incidência não representa a estrutura de redes impuras.

Como visto na seção 1.8, a equação fundamental possibilita a representação de aspectos comportamentais da rede, pois descreve a inserção e remoção de marcas nos lugares, conforme o disparo das transições.

A Equação Fundamental das Redes de Petri é

$$M'(p) = M_0(p) + C.\overline{s}^t, \quad \forall p \in P$$

onde $\overline{s}$ é o vetor característico cujos componentes s_i são naturais e representam o número de vezes que cada transição t_i foi disparada para obter-se a marcação $M'(p)$ a partir de $M_0(p)$.

Essa técnica de análise tem sido utilizada para a verificação de problemas de alcaçabilidade, ou seja, podemos verificar se uma determinada marcação M_k é acessível a partir de M_0.

Na figura 3.23 apresentamos uma rede em que faremos a verificação da acessibilidade da marcação $M' = (0, 0, 0, 1, 1, 0, 1)$ a partir da marcação inicial apresentada.

A estrutura da rede através da matriz de incidência **C** é apresentada como:

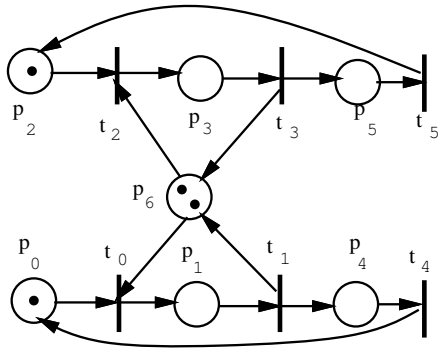

Figura 3.23: Alcançabilidade de uma Marcação

$$\begin{vmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{vmatrix} - \begin{vmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 2 \end{vmatrix} = \begin{vmatrix} -1 & 0 & 0 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ -1 & 1 & -1 & 1 & 0 & 0 \end{vmatrix} \times \begin{vmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{vmatrix}$$

A solução para o sistema matricial da figura 3.23 é

$$\overline{s} = (s_0, s_0, 1 + s_3, s_3, s_0 - 1, s_3).$$

Se considerarmos $s_0 = 1$ e $s_3 = 0$, teremos $\overline{s} = (1, 1, 1, 0, 0, 0)$ como solução do problema. Esta solução significa que com um disparo de t_0, t_1 e t_2 obteremos a marcação M', partindo-se da marcação M_0, ou seja, esta marcação é alcançável a partir de M_0. Observe-se, contudo, que a marcação M' pode ser alcançável em outras relações de disparo de transições.

Esta técnica de análise tem sido muito utilizada no processo de validação, embora não seja possível obter a seqüência de disparo das transições. Ainda é possível a obtenção de resultados que não representam o comportamento do sistema. Por exemplo, a rede da figura 3.24. A solução do sistema matricial

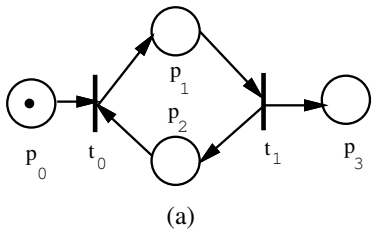

Figura 3.24: Não-Alcançabilidade

$$\begin{vmatrix} 0 \\ 0 \\ 0 \\ 1 \end{vmatrix} - \begin{vmatrix} 1 \\ 0 \\ 0 \\ 0 \end{vmatrix} = \begin{vmatrix} -1 & 0 \\ 1 & -1 \\ -1 & 1 \\ 0 & 1 \end{vmatrix} \times \begin{vmatrix} s_0 \\ s_1 \end{vmatrix}$$

é $\overline{s} = (1, 1)$. Isto significa que com um disparo de t_0 e t_1 obtemos a marcação final ($M' = (0, 0, 0, 1)$). Contudo isto não é verdade, dado que as transições t_0 e t_1 não estão habilitadas para a marcação inicial apresentada.

3.3.3 Invariantes de Transição

Nesta seção, apresentamos um método de análise que verifica a existência e fornece *componentes repetitivos estacionários* [20] nos modelos.

Se em uma rede é possível, a partir de uma marcação M', dispararmos cada transição n_i vezes e retornarmos a marcação inicial M', ou seja, a marcação obtida após o disparo da seqüência de transições (σ) é a marcação de partida ($M' = M_0 + C.\overline{s}$), dizemos que esta rede tem componentes repetitivos estacionários, onde estes componentes correspondem a comportamentos cíclicos da rede. Os componentes repetitivos estacionários são obtidos solucionando-se o sistema $C.\overline{s}$. Dizemos que o vetor característico encontrado $\overline{s} \geq 0$ é o invariante de transição I_t (*T-invariante*), ou seja, um invariante de transição é um vetor de dimensão igual ao número de transições na rede, onde cada componente deste vetor corresponde ao número de vezes que cada transição deve ser disparada. Cada invariante fornece um conjunto de transições que, quando disparadas, não alteram a marcação da rede.

Definição 3.3.3 - Invariante de Transição: *seja um vetor $I_t \geq 0$ de inteiros não negativos. I_t é chamado invariante de transição de uma rede de Petri se e somente se $C.I_t = 0$.*

Se I_{t1} e I_{t2} são invariantes de uma rede R, então $I_{t1} + I_{t2}$ e $k.I_{t1}$ $(k \in N)$ também são invariantes de R [5].

A seguir, apresentamos o cálculo dos invariantes de transição da rede da figura 3.23. Ao solucionarmos o sistema de equações $C.I_t^T = 0$ obtemos como resultado $s_0 = s_1 = s_4$ e $s_2 = s_3 = s_5$, onde I_t^T é o vetor I_t transposto. Desta forma, para $s_0 = 1$ e $s_2 = 0$ temos o invariante $I_{t1} = (1, 1, 0, 0, 1, 0)$. Se fizermos $s_0 = 0$ e $s_2 = 1$ temos um outro invariante $I_{t2} = (0, 0, 1, 1, 0, 1)$. A soma desses invariantes também nos fornece um outro invariante de transição, ou seja, $I_t = I_{t1} + I_{t2} = (1, 1, 1, 1, 1, 1)$. Observe-se que para um dada marcação que habilita essas transições, o disparo dessa transições conforme o invariante garante o retorno a marcação atual, ou seja, o disparo desta transições não altera a marcação original da rede.

Através do cálculo dos invariantes de transição é possível verificar a existência dos componentes repetitivos estacionários, bem como verificar a consistência parcial ou completa de uma rede. A rede analisada nesta seção é uma rede completamente consistente, pois existe um invariante (de inteiros não-negativos) que prova que ao disparar-se todas as transições dessa rede uma vez, segundo alguma seqüência, retornamos à marcação inicial.

3.3.4 Invariantes de Lugar

Nesta seção, apresentamos um método de análise baseado na equação fundamental das redes de Petri que nos fornece os *componentes conservativos* do sistema [21], sem a necessidade da observação exaustiva da árvore de cobertura.

A obtenção dos invariantes de lugar de uma rede provê o que é denominado por componentes conservativos do modelo. Esses componentes estão associados ao conjunto de lugares em que a soma ponderada de marcas, desses lugares, permanece constante, quando do disparo de uma seqüência de transições. Como vimos na seção 3.2.2, uma rede é conservativa se a soma ponderada de marcas dessa rede não é alterada para qualquer seqüência de disparo possível.

Se em uma rede há componentes conservativos, existe uma soma pondera-

da das marcas dos lugares associados de tal forma que esta soma é constante. A ponderação é fornecida por um vetor de pesos $I_p = (w_1, w_2, ..., w_n)$, onde w_i é o peso relativo a cada lugar da rede. Tratamos de soma ponderada e não simplesmente de uma soma das marcas dos lugares associados, porque o disparo de uma transição pode gerar mais ou menos marcas do que o número previamente existente, sem contudo comprometer a conservação.

$$w_1.M(p_1) + w_2.M(p_2) + ... + w_n.M(p_n) = S$$

onde S é um valor constante. Desta forma podemos escrever:

$$\Sigma w_i.M(p_i) = S, \quad onde \quad w_i \geq 0 \quad e \quad i = 1, 2, ..., n$$

Se em uma rede é possível, a partir de uma marcação M_0, alcançarmos uma marcação M' de tal forma que a soma ponderada de marcas é constante, temos que:

$$I_p^T.M_0 = I_p^T.M'$$

Da equação fundamental das redes de Petri temos:

$$M' = M_0 + C.\overline{s}^t$$

à esquerda por I_p^T obtemos:

$$I_p^T.M' = I_p^T.M_0 + I_p^T.C.\overline{s}$$

sendo a rede conservativa temos

$$I_p^T.M_0 = I_p^T.M_0 + I_p^T.C.\overline{s}$$

logo

$$I_p^T.C.\overline{s} = 0.$$

Como sabemos que $\overline{s} \neq 0$, portanto

$$I_p^T.C = 0.$$

O vetor I_p^T, um vetor que tem como dimensão o número de lugares, de invariante de lugar ou *P-invariante*.

Definição 3.3.4 - Invariante de Lugar: *seja um vetor I_p^T de inteiros não-negativos. I_p^T é chamado invariante de lugar de uma rede de Petri se e somente se $I_p^T.C = 0$.*

A seguir, apresentamos o cálculo dos invariantes de lugar da rede da figura 3.23. Ao solucionarmos o sistema de equações $I_p^T.C = 0$ obtemos como resultado $I_p = (w_0, 2.w_0, w_2, w_0 + w_2, w_0, w_2, w_0)$. Se fizermos $w_0 = 1$ e $w_2 = 1$ teremos $I_p = (1, 2, 1, 2, 1, 1, 1)$. Este resultado nos mostra que a rede analisada é conservativa.

3.3.5 Invariantes por Justaposição e por Concatenação

Nesta seção, apresentamos um método para a obtenção de invariantes de uma rede formada de sub-redes. Os invariantes da rede global podem ser obtidos diretamente dos invariantes desta sub-redes, sem a necessidade do cálculo de matrizes de grandes dimensões [20, 21].

A técnica de composição de matrizes é realizada através da fusão de transições ou lugares das sub-redes. A técnica utilizada para a obtenção dos invariantes possibilita tanto a conservação dos invariantes das sub-redes, quanto a obtenção de novos invariantes no modelo global.

Utilizaremos aqui a denominação *elemento* de uma rede associada aos lugares ou transições da rede. A técnica de fusão de elementos de sub-redes possibilita uma forma sistemática para a construção de redes de grandes dimensões.

Na figura 3.25 apresentamos a fusão de elementos de sub-redes de forma a obtermos redes de maiores dimensões. Sejam duas redes $R_1 = (P_1, T_1, I_1, O_1, K)$ e $R_2 = (P_2, T_2, I_2, O_2, K)$, cujas matrizes de incidência são $C_1 = O_1 - I_1$ e $C_2 = O_2 - I_2$, respectivamente. Seja R a rede obtida pela a fusão dos últimos k lugares de R_1 e os primeiros k lugares de R_2. A matriz de incidência da rede R é apresentada a seguir:

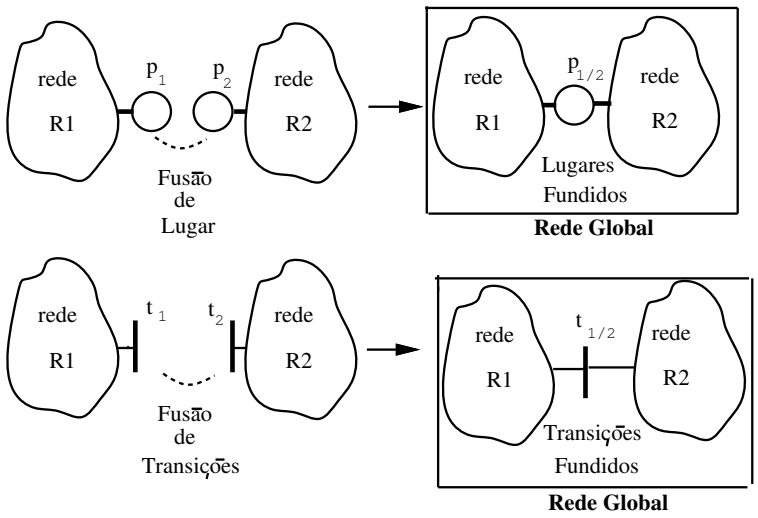

Figura 3.25: Fusão de Elementos

$$
\left|\begin{array}{cccccc|c}
c_1^{1,1} & ... & c_1^{n,1} & 0 & ... & 0 & \\
. & . & . & . & . & . & p_1^1 \\
. & . & . & . & . & . & . \\
. & . & . & . & . & . & . \\
c_1^{m-k} & ... & c_1^{n,m-k} & 0 & ... & 0 & . \\
c_1^{1,m-k+1} & ... & c_1^{n,m-k+1} & c_2^{1,1} & .... & c_2^{h,1} & p_1^{m-k+1}, p_2^1 \\
. & . & . & . & . & & . \\
. & . & . & . & . & & . \\
. & . & . & . & . & & . \\
c_1^{1,m} & ... & c_1^{n,m} & c_2^{1,k} & .... & c_2^{h,k} & p_1^m, p_2^k \\
0 & ... & 0 & c_2^{1,k+1} & ... & c_2^{h,k+1} & \\
. & . & . & . & . & . & . \\
. & . & . & . & . & . & . \\
. & . & . & . & . & . & . \\
0 & ... & 0 & c_2^{1,q} & ... & c_2^{h,q} & p_2^q
\end{array}\right|
$$

Sendo $I_t^T = (s_1^1, ..., s_1^n, s_2^1, ..., s_2^h)$ e solucionando o sistema de equações ge-

rado de $C.I_t^T = 0$, obtemos os invariantes de transição da rede, que podemos analisar conforme mostrado a seguir:

$$c_1^{1,i}.s_1^1 + ... + c_1^{n,i}.s_1^n = 0$$

para $i = 1, ..., m - k$,

$$c_1^{1,i}.s_1^1 + ... + c_1^{n,i}.s_1^n + c_2^{1,i}.s_2^1 + ... + c_2^{h,i}.s_2^h = 0$$

para $i = m - k + 1, ..., m$,

e

$$c_2^{1,i}.s_2^1 + ... + c_2^{h,i}.s_2^h = 0$$

para $i = k + 1, ..., q$.

Observamos, desta forma, que os invariantes da rede R_1 são soluções de

$$c_1^{1,i}.s_1^1 + ... + c_1^{n,i}.s_1^n = 0$$

para $i = 1, ..., m - k$ e

$$c_1^{1,i}.s_1^1 + ... + c_1^{n,i}.s_1^n + c_2^{1,i}.s_2^1 + ... + c_2^{h,i}.s_2^h = 0$$

para $i = m-k+1, ..., m$ desde que $s_2^1, ...s_2^h$ sejam zero. De forma semelhante, também observamos que os invariantes de R_2 são soluções de

$$c_1^{1,i}.s_1^1 + ... + c_1^{n,i}.s_1^n + c_2^{1,i}.s_2^1 + ... + c_2^{h,i}.s_2^h = 0$$

para $i = k + 1, ..., q$, desde que $s1_1, ..., s1_n$ sejam zero. Isto significa que os invariantes de transição de R podem ser obtidos pela concatenação dos invariantes de transição de R_1 e de R_2.

Observando ainda o problema de fusão de k lugares de sub-redes, consideramos a resolução do sistema de equações $Ip.C = 0$, para a obtenção dos invariantes de lugar da rede R, construída pela composição das sub-redes R_1 e R_2. Sendo $I_p^T = (w_1^1, ..., w_1^{m-k}, w_2^1, ..., w_2^q)$, onde $w_1^{m-k+i} = w_2^i$ para $i = 1, ..., k$, temos como solução deste sistema de equações:

$$c_1^{i,1}.w_1^1 + ... + c_1^{i,m}.w_1^m = 0$$

para $i = 1, ..., n$ e

$$c_2^{i,1}.w_2^1 + ... + c_2^{i,q}.w_2^q = 0$$

para $i = m - k + 1, ..., q$.

Verificamos que $c_1^{i,1}.w_1^1 + ... + c_1^{i,m}.w_1^m = 0$ para $i = 1, ..., n$ tem o número de equações igual ao número de lugares de R_1, assim como $c_2^{i,1}.w_2^1 + ... + c_2^{i,q}.w_2^q = 0$ para $i = m - k + 1, ..., q$ tem o número de equações igual ao número de lugares de R_2 e que os k lugares de cada rede fundidos, influenciam em todas as equações do sistema. Desta forma, só serão invariantes da rede global, os invariantes das sub-redes, desde que os componentes correspondentes aos elementos fundidos desses invariantes de cada sub-rede sejam idênticos, ou seja, ($w_1^{m-k+i} = w_2^{m-k+i}$ para $i = 1, ..., k$). Assim, temos:

- se $w_1^{m-k+i} = 0$ e/ou $w_2^{m-k+i} = 0$ para $i = 1, ..., k$ todos os invariantes de R_1 e/ou de R_2 serão invariantes de R;
- se $w1_{m-k+i} = w2_{m-k+i} \neq 0$ para $i = 1, ..., k$ todos os invariantes de R_1 e/ou de R_2 serão invariantes de R;
- se $w1_{m-k+i} \neq w2_{m-k+i} \neq 0$ para $i = 1, ..., k$, então é necessário encontrarmos um invariante de R_1 e de R_2 tal que a solução para a incógnitas fundidas sejam idênticas, para construirmos o invariante de R por justaposição destes invariantes. Se não for encontrada tal solução, os invariantes das sub-redes não serão invariantes da rede global.

Resultados semelhantes são obtidos se analisarmos este problema através da fusão de transições das sub-redes, pois lugares e transições podem ser vistos como elementos duais.

Algumas propriedades que sintetizam o processo de obtenção dos invariantes de uma rede R, obtida pela composição de sub-redes, são apresentados a seguir:

- Propriedade 1: Todos os invariantes de elemento dual (lugar ou transição) de R_1 e de R_2 são invariantes de elemento dual de R. Estes invariantes são obtidos por concatenação. É possível o aparecimento de novos invariantes.

- Propriedade 2: Todos os invariantes de elemento de R_1 e de R_2, onde os valores das incógnitas associadas aos elementos fundidos são zero, serão invariantes de elemento de R. Estes invariantes são obtidos pela justaposição dos invariantes das sub-redes R_1 e R_2.

- Propriedade 3: Se existe um invariante de R_1 e um invariante de R_2 tal que os pesos associados aos elementos fundidos sejam iguais, então a justaposição desses invariantes é invariante de R. É possível o desaparecimento de invariantes de R_1 e R_2 em R.

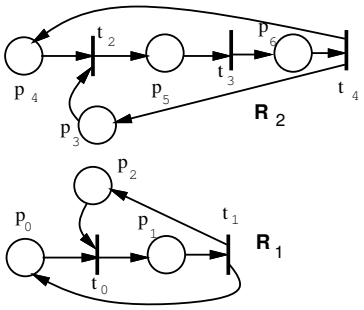

Figura 3.26: Sub-redes R_1 e R_2

Na figura 3.26 temos as rede R_1, a rede R_2 e suas respectivas matrizes de incidências são apresentadas a seguir:

$$\begin{array}{c} t_0 \quad t_1 \\ C_1 = \left| \begin{array}{cc|l} -1 & 1 & p_0 \\ 1 & -1 & p_1 \\ -1 & 1 & p_2 \end{array} \right. \end{array}$$

Os invariantes de transição podem ser obtidos solucionando-se os sistemas de equações $C_1.I_t^{1'} = 0$ e $C_2.I_t^{2'} = 0$, representados a seguir:

$$C_1 = \left| \begin{array}{cc} -1 & 1 \\ 1 & -1 \\ -1 & 1 \end{array} \right| \times \left| \begin{array}{c} x_0 \\ x_1 \end{array} \right| = \left| \begin{array}{c} 0 \\ 0 \\ 0 \end{array} \right|$$

$$C_2 = \begin{vmatrix} -1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{vmatrix} \times \begin{vmatrix} y_0 \\ y_1 \\ y_2 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ 0 \\ 0 \end{vmatrix}$$

ou sejam,

$$\begin{cases} -x_0 + x_1 &= 0 \\ x_0 - x_1 &= 0 \\ -x_0 + x_1 &= 0 \end{cases}$$

$$\begin{cases} -y_0 + y_2 &= 0 \\ y_0 - y_2 &= 0 \\ y_0 - y_1 &= 0 \\ y_1 - y_2 &= 0 \end{cases}$$

então $x_0 = x_1$. Se $x_0 = 1$ isto implica $x_1 = 1$, portanto $I_t^{1'} = (1,1)$ é uma possível solução de $C_1.I_t^{1'} = 0$. De forma semelhante, $y_0 = y_1 = y_2$. Fazendo-se $y_0 = 1$, tem-se que $y_1 = y_2 = 1$. Portanto $I_t^{2'} = (1,1,1)$ será uma possível solução de $C_2.I_t^{2'} = 0$.

A rede da figura 3.27 pode ser obtida pela composição das sub-redes R_1 e R_2 representadas na figura 3.26, através fusão dos lugares p_2 e p_3.

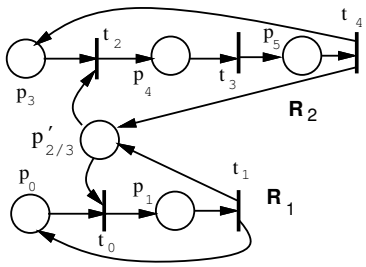

Figura 3.27: Rede R

$$t_0 \quad t_1 \quad t_2 \quad t_3 \quad t_4$$

$$I_t^{'} = \begin{pmatrix} -1, & 1, & 0, & 0, & 0 \end{pmatrix}$$

$$I_t^{'} = \begin{pmatrix} I_t^{1'} & , & 0, & 0, & 0 \end{pmatrix}$$

$$I_t^{''} = \begin{pmatrix} 0, & 0, & 1, & 1, & 1 \end{pmatrix}$$

$$I_t^{''} = \begin{pmatrix} 0, & 0, & I_t^{2'} \end{pmatrix}$$

Os invariantes de transição de R (elemento dual, pois o elemento fundido foi lugar) pode ser obtido pela concatenação dos invariantes de transição de R_1 e R_2, conforme afirmado no início desta seção, como por exemplo: $I_t^{'} = (1,1,0,0,0)$ e $I_t^{''} = (0,0,1,1,1)$.

Os invariantes de lugar de R (figura 3.27) podem ser obtidos pela justaposição dos invariantes de R_1 e R_2 (figura 3.27). Os invariantes de lugar de R_1 e de R_2 são obtidos solucionando-se os sistemas de equações $C_1.I_p^1 = 0$ e $C_2.I_p^2 = 0$, respectivamente. Uma das possíveis soluções é: $I_p^{1'} = (1,2,1)$ como invariante de R_1 e $I_p^{2'} = (1,1,2,2)$ como invariante de R_2. Os invariantes de elemento lugar(o elemento fundido foi um lugar) da rede R podem ser obtidos pela justaposição dos invariantes das redes que a compõem, ou seja, pelos invariantes de lugar de R_1 e de R_2.

$$\begin{array}{rcl} & p_0 \quad p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 & \\ I_p^{1'} = & \begin{pmatrix} 1, & 2, & 1 \end{pmatrix} & \\ I_p^{2'} = & & \begin{pmatrix} 1, & 1, & 2, & 2 \end{pmatrix} \\ I_p^{'} = J \begin{pmatrix} I_p^{1'}, & I_p^{2'} \end{pmatrix} = & \begin{pmatrix} 1, & 2, & 1, & 1, & 2, & 2 \end{pmatrix} & \end{array}$$

Observe-se que o peso associado ao elemento fundido (p_2), nos invariantes obtidos é 1 em $I_p^{1'}$ e em $I_p^{2'}$. Desta forma, conforme a propriedade *P3* a justaposição desses invariantes fornece um invariante de R.

Um outro invariante de lugar de R_2 é $I_p^{2''} = (2,1,3,3)$. Para efetuarmos a justaposição dos invariantes $I_p^{1'} = (1,2,1)$ e $I_p^{2''} = (2,1,3,3)$ das redes R_1 e R_2, respectivamente, é necessário multiplicar estes invariantes por números inteiros não-negativos, de forma que os valores associados aos elementos fundidos sejam iguais dois a dois. Representamos os invariantes de lugar de R_1 por dois conjuntos de pesos *w1* e *wk1 Ip1(w1, wk1)*, onde *wk1* corresponde

aos pesos associados aos elementos fundidos e *w1* aos demais elementos. De forma semelhante, representamos os invariantes de R_2 por $I_p^2(wk2, w2)$. Se $w1 \neq w2$ então é necessário encontrar $a, b \in N$ de forma que $a.w1 = b.w2$. Para o exemplo deste parágrafo, podemos adotar $a = 2$ e $b = 1$, pois $2.wk1 = 1.wk2$, ou seja $I_p^{1''} = 2.I_p^{1'} = (2, 4, 2)$ e $1.I_p^{2''} = (2, 1, 3, 3)$. Observe que em ambos os invariantes ($I_p^{1''}$ e $I_p^{2''}$) os valores associados ao elemento fundido são iguais; isto torna possível a justaposição destes invariantes.

$$
\begin{array}{rcl}
 & & p_0 \quad p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \\
 & \left(\begin{array}{ccc} 1, & 2, & 1 \end{array} \right) \times 2 = I_p^{1''} = & \left(\begin{array}{ccc} 2, & 4, & 2 \end{array} \right) \\
I_p^{2''} = & & \left(\begin{array}{cccc} 2, & 1, & 3, & 3 \end{array} \right) \\
I_p^{'} = & J \left(\begin{array}{cc} I_p^{1''}, & I_p^{2'} \end{array} \right) = & \left(\begin{array}{cccccc} 2, & 4, & 2, & 1, & 3, & 3 \end{array} \right)
\end{array}
$$

Caso não seja possível encontrar valores para a e b que possibilitem esta condição, não é plausível a justaposição desses invariantes.

Estes resultados podem ser comprovados. Para isto, podemos calcular os invariantes da rede R solucionado-se $C.Ip = 0$, onde C é a estrutura da rede R.

3.3.6 Regras de Redução para Análise

Uma outra técnica muito utilizada na análise de propriedades nas redes de Petri são as regras de redução. A análise de propriedades em redes de grandes dimensões não é um problema trivial, portanto métodos que possibilitem a transformação de modelos preservando as propriedades dos modelos originais têm sido estudados [19, 41, 1]. Apresentaremos aqui algumas regras de redução que facilitam a análise de grandes sistemas.

A técnica de validação por redução baseia-se na transformação de uma rede de grandes dimensões em uma outra rede mais abstrata, de forma que propriedades como *liveness*, limitação(*boundedness*) e segurança(*safeness*) sejam preservadas no modelo resultante após a transformação. Isto é, uma redução é uma relação binária ($f : N \rightarrow N'$) que transforma uma rede marcada $N = (R, M_0)$ em uma outra rede marcada, como por exemplo $N' = (R', M_0')$, preservando as propriedades de *liveness*, *boundedness* e *safeness*, se a rede original possui essas propriedades. A transformação reversa,

ou seja, a transformação de modelos abstratos em modelos refinados, de forma hierárquica, pode ser utilizada no processo de síntese. Estas regras são denominadas regras de síntese.

A seguir apresentamos algumas destas regras de redução, que possibilitam a transformação de redes pela fusão de lugares e transições e eliminação de *loops*.

1. Fusão de Lugares Série:

   Seja uma rede $N = (R, M_0)$, onde existe uma transição t_i com entrada e saída únicas ($I(t_i) = [p_j]$ e $O(t_i) = [p_k]$). Podemos transformar $N = (R, M_0)$ em $N' = (R', M_0')$ pela fusão de p_j e p_k e eliminação de t_i. Para isto, criamos um novo lugar $p_{j/k}$, que representa a fusão destes lugares, onde $I(p_{j/k}) = I(p_j)$ e $O(p_{j/k}) = O(p_k)$ (ver figura 3.28).

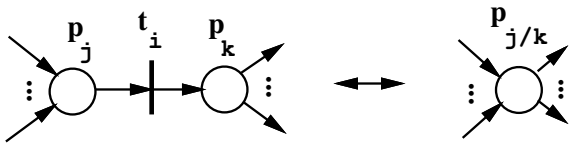

Figura 3.28: Reduções - Fusão de Lugares Série

2. Fusão de Transições Série:

   Seja uma rede $N = (R, M_0)$, onde existe um lugar p_i com entrada e saída únicas ($I(p_i) = [t_j]$ e $O(p_i) = [t_k]$). Podemos transformar $N = (R, M_0)$ em $N' = (R', M_0')$ pela fusão de t_j e t_k e eliminação de p_i. Para isto, criamos uma nova transição $t_{j/k}$, que representa a fusão destas transições, onde $I(t_{j/k}) = I(t_j)$ e $O(t_{j/k}) = O(t_k)$ (ver figura 3.29).

3. Fusão de Lugares Paralelos:

   Seja uma rede $N = (R, M_0)$, onde existem dois lugares p_i *e* p_h, tal que $I(p_i) = I(p_h) = [t_j]$ e $O(p_i) = O(p_h) = [t_k]$. Podemos transformar $N = (R, M_0)$ em $N' = (R', M_0')$ pela fusão dos lugares p_i e p_h. Para isto criamos um novo lugar $p_{j/k}$, que representa a fusão destes lugares, onde $I(p_{i/h}) = I(p_i) = I(p_h))$ e $O(p_{i/h}) = O(p_i) = O(p_h)$ (ver figura 3.30).

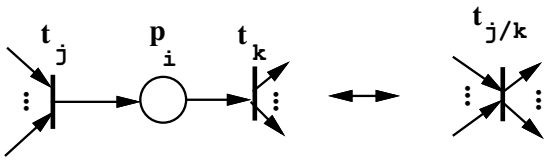

Figura 3.29: Reduções - Fusão de Transições Série

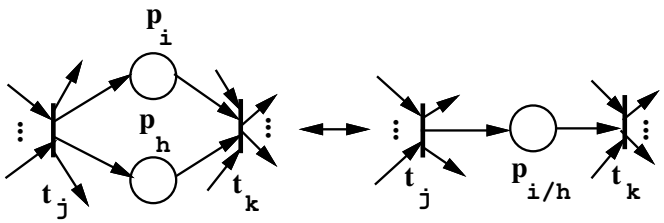

Figura 3.30: Reduções - Fusão de Lugares Paralelos

4. Fusão de Transições Paralelas:

   Seja uma rede $N = (R, M_0)$, onde existem duas transições t_i e t_h, tal que $I(t_i) = I(t_h) = [p_j]$ e $O(t_i) = O(t_h) = [p_k]$. Podemos transformar $N = (R, M_0)$ em $N' = (R', M_0')$ pela fusão das transições t_i e t_h. Para isto criamos uma nova transição $t_{j/k}$, que representa a fusão destas transições, onde $I(t_{i/h}) = I(t_i) = I(t_h)$ e $O(t_{i/h}) = O(t_i) = O(t_h)$ (ver figura 3.31).

5. Eliminação de Self-loop:

   Seja uma rede $N = (R, M_0)$ e e_i uma transição ou um lugar de N. Caso $e_i \in P$, então $M(e_i) \geq \#O(e_i)$. Se $I(e_i) = O(e_i)$ então podemos eliminar e_i de N (ver figura 3.32).

As regras de redução podem ser utilizadas para a simplificação dos modelos de forma estruturada, onde propriedades como *liveness* e limitação (*boundedness*) são mantidas nos modelos mais abstratos, o que facilita o uso dos métodos de análise nesses modelo.

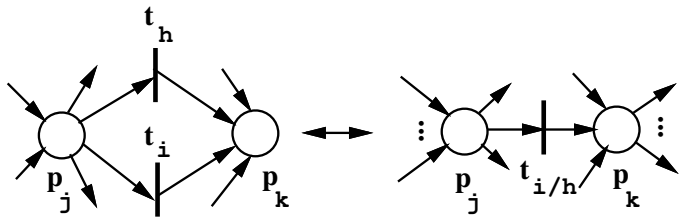

Figura 3.31: Reduções - Fusão de Transições Paralelas

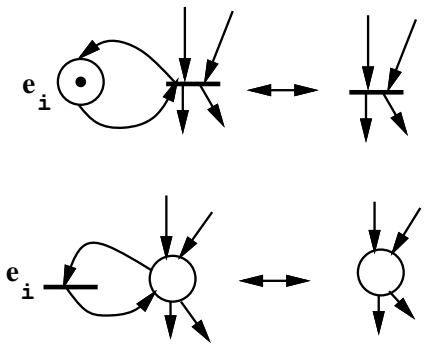

Figura 3.32: Reduções - Eliminação de Self-loop

3.4 Complexidade e Decidibilidade

Nesta seção, apresentamos um sumário sobre alguns problemas de decidibilidade e complexidade na verificação de propriedades e equivalência de modelos em redes de Petri [59, 4, 61, 13].

Quando procuramos uma solução para um problema, é necessário considerar a possibilidade que não haja elementos para dizer se o problema admite ou não solução, ou seja, tal problema é indecidível. Se a solução existe, é necessário considerar quanto tempo ou recursos são necessários, ou seja, a complexidade do problema, para a obtenção dessa solução.

Karp e Miller [63] apresentaram *o Sistema de Adição de Vetores* que possibilitava a analise de propriedades em programas paralelos. Nesse trabalho foram apresentados modelos para programas paralelos, assim como um procedimento para a verificação de propriedades nesses modelos denominado Árvore de Alcançabilidade. Esse procedimento permitia decidir sobre limitação(*boundedness*) e cobertura de marcações(*coverability*). No final da década de sessenta, observou-se que o Sistema de Adição de Vetores e as redes de Petri eram matematicamente equivalentes, o que incetivou fortemente a pesquisa sobre decidibilidade em redes de Petri.

Um conceito fundamental no estudo da decidibilidade é a *redutibilidade*. Solucionar um problema por redução é reduzir o problema estudado em um problema o qual já se conhece a solução, ou seja, se o *Problema-1* é redutível ao *Problema-2* e o *Problema-2* é decidível, o algoritmo utilizado para solucionar o *Problema-2* soluciona o *Problema-1*, isto é, o *Problema-1* também é decidível.

O outro aspecto a ser considerado está relacionado à eficiência das técnicas de verificação de propriedades e de equivalência, ou seja, o custo(tempo e espaço) para solucionar-se esses problemas.

A maioria das propriedades de interesse para a verificação em redes de Petri são decidíveis, contudo a sua verificação, normalmente, é de alta complexidade. A verificação da decidibilidade dessas propriedades é feita por redução aos problemas de alcançabilidade e limitação, os quais são decidíveis.

Karp e Miller [63] mostraram que uma rede de Petri é limitada se o grafo de marcações acessíveis ($A(R, M_0)$) é finito e que esse problema é decidível. Uma rede de Petri é dita ilimitada se, e somente se, existe uma seqüência de transições s tal que, $M[s > M + M'$, onde M' é um vetor não-nulo de dimensão $\#P$. A seqüência s é denominada"gerador de marcas", pois gera

marcações superiores à marcação original. O algoritmo desenvolvido no trabalho de Karp e Miller para a verificação de "geradores de marcas" mostrou-se ineficiente. Rackoff [62] apresentou um algoritmo que requeria no máximo $2^{c.n.\log n}$, onde n é o tamanho e em c uma constante. Rossier e Yen refinaram o resultado de Rackoff utilizando três paramentros, $2^{c.k.\log k}.(l+\log n)$, onde k é o número de lugares, n o número de transições e l o número máximo de transições de entrada e saída. Lipton [64] provou que era necessário ao menos $2^{C.\sqrt{n}}$ para decidir sobre a limitação.

O problema de alcançabilidade consiste em saber se dada uma rede de Petri marcada $N = (R, M_0)$ e uma marcação M', se M' pode ser acessível a partir de M_0. Muitos problemas são redutíveis ao acima citado, sendo assim esse foi um ponto fundamental no estudo da teoria das redes de Petri. Sacerdote e Tenney [65] afirmaram que a alcançabilidade era decidível, contudo não apresentaram uma prova completa que corroborasse essa afirmação. Em 1981, Mayr [66] apresentou uma prova dessa conjectura e esta foi simplificada por Kosaraju [67], contudo a prova permanecia muito complexa. O trabalho de Lambert [68] apresenta prova mais simples e auto-contida.

Diversos problemas são redutíveis ao problema de alcançabilidade, tais como: Alcançabilidade de *home-state* [69] e Alcançabilidade de submarcação [67], onde se deseja verificar se a marcação de um subconjunto de lugares atingiu uma determinada marcação, a existência de marcação zero, ou seja, se nenhum lugar possui marcas, em uma dada marcação acessível, a existência de marcações que não armazenam marcas em lugares específicos. O resultado apresentado por [70] mostra que a decidibilidade do problema de alcançabilidade é obtida em espaço exponencial.

Hack [71] mostrou que o problema de *liveness* é redutível ao de alcançabilidade, o mesmo apresentando-se em [72] com relação às redes sem transições mortas(*deadlock-freedom*).

Em [73, 66, 74] apresentam-se os resultados que confirmam a decidibilidade do problema de persistência.

Contrapondo-se aos resultados positivos obtidos para a verificação dessas propriedades, os resultados sobre equivalência dos conjuntos de marcações acessíveis e de linguagens entre redes têm sido negativos [61].

Duas redes têm equivalência de marcações se possuem um mesmo conjunto de lugares e um mesmo conjunto de marcações acessíveis [61]. O estudo que comprova a indecidibilidade da equivalência de marcações [71] baseia-se na computação-fraca de funções utilizando-se redes (*Weak-Computation*).

Mostrar a indecidibilidade do problema de equivalência de conjuntos de marcações acessíveis baseia-se na redução desse problema ao décimo problema de Hilbert(indecidível). O décimo problema de Hilbert consiste em: dado um polinômio $f(x_1, ..., x_r)$, com r variáveis com coeficientes inteiros, existe um vetor de inteiros $(x_1, ..., x_r)$ que seja raiz desse polinômio(equação Diofantina) ? A prova desse problema pode ser encontrada em [71] e consiste em mostrar que as redes de Petri podem efetuar a computação fraca de funções e desta forma do polinômio diofantino, reduzir a indecidibilidade do décimo problema de Hilbert ao problema de grafo de inclusão e então reduzir o problema de inclusão ao problema de equivalência de marcações. A solução desse problema está demonstrada em [71, 4, 59]. Vale salientar, contudo, que em classes especiais de redes esse problema é decidível. Por exemplo, no caso de redes limitadas, onde o conjunto de marcações acessíveis é finito, temos que o problema de equivalência de marcações é decidível.

Capítulo 4

Extensões às Redes de Petri

Neste capítulo apresentamos brevemente algumas extensões às redes de Petri [5, 6, 7, 75], que elevam o poder computacional dessas ao da máquina de Turing e possibilitam uma modelagem mais compacta e estruturada dos sistemas em geral. Definimos as redes com arco inibidor, as redes coloridas, as redes hierárquicas e redes temporizadas determinísticas.

4.1 Rede de Petri com Arco Inibidor

Um dos problemas das redes de Petri até então apresentadas é a impossibilidade do teste a zero de uma variável (lugar). Para tal, até o momento, era necessário a criação de um lugar dual e através dele efetuava-se o teste. Esta extensão inclui um arco diferenciado dos demais, o arco inibidor [4] [1] [2], que inibe o disparo de uma transição, caso o lugar, que é pré-condição à transição, esteja marcado.

Uma das principais limitações das redes de Petri é a impossibilidade da realização do *teste de zero* marcas em lugares de capacidade ilimitadas. Na figura 4.1, temos uma rede limitada (*bounded*) formada por três transições e seis lugares. Nessa rede, o lugar p_1 é limitado a k marcas, pois um lugar complementar (p_2) impossibilita o armazenamento de um número maior de marcas nesse lugar. O número de marcas dos lugares permanece constante para qualquer marcação acessível da rede. Observe-se que a transição t_3 só poderá ser disparada quando o lugar p_2 tiver k marcas ($M(p_2) = k$).

Embora as redes de Petri possibilitem o teste de zero marcas em redes

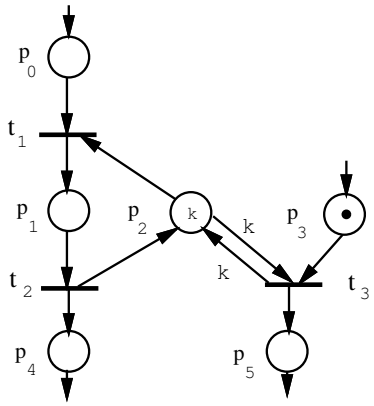

Figura 4.1: Teste de Zero Marcas em Lugares Limitados

limitadas, para redes ilimitadas (*unbounded*) esse teste não é possível. A extensão mais simples que proporciona o teste de zero marcas em redes ilimitadas é a rede com arco inibidor. Graficamente o arco inibidor é um arco de um lugar para uma transição (pré-condição) terminando com um pequeno círculo.

A regra de disparo das transições que têm como entrada arcos inibidores é alterada. Essas transições só estarão aptas a serem disparadas caso os lugares de entrada, relacionados às transições pelos arcos inibidores, não contiverem marcas.

A figura 4.2 apresenta uma rede com arco inibidor, possibilitando o teste de zero marcas no lugar p_0, ou seja, a transição t_2 só estará apta a ser disparada quando não houver marcas em p_0.

Observe-se que não é necessário sabermos a capacidade do lugar "testado" (neste caso, p_0)), pois a semântica de disparo de transições que possuam arco inibidor só possibilita o disparo se todos os lugares que estão associados à transição por arcos inibidores não possuírem marcas.

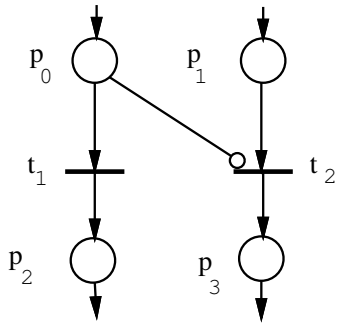

Figura 4.2: Teste de Zero Marcas

4.2 Rede de Petri Colorida

Embora as redes de Petri sejam uma técnica de descrição formal poderosa que possibilita a modelagem de sistemas concorrentes de forma natural, a sua aplicação para sistemas reais tem se mostrado inadequada, pois os modelos obtidos são de grandes dimensões e não hierárquicos dificultando a sua compreensão.

Um grande número de extensões às redes de Petri, denominadas redes de Petri de alto nível, têm sido propostas [5, 76]; entre estas, as redes de Petri coloridas [75, 2, 35, 37]. Nas redes ordinárias existe apenas um tipo de marca, o que não permite a diferenciação de recursos em um lugar, sendo necessários lugares distintos para expressarmos recursos similares. As redes ordinárias não possibilitam ainda níveis de refinamento do modelo, ou seja, não há hierarquização.

O principal objetivo das redes de Petri coloridas é a redução do tamanho do modelo, permitindo que marcas individualizadas (*cores*) representem diferentes processos ou recursos em uma mesma sub-rede.

Inicialmente, as marcas das redes coloridas eram representadas por cores ou mesmo por padrões que possibilitam a distinção das marcas [2]. Em trabalhos mais recentes [75] temos as marcas representadas por estruturas de dados complexas, não relacionando cores às marcas, a não ser pelo fato de que as marcas são distingüíveis.

Na figura 4.3 apresentamos uma rede colorida segundo a abordagem a-

presentada em [2]. Nessa classe de rede os arcos são rotulados com uma ou mais das possíveis cores ou ainda com variáveis. Na figura 4.3 os arcos são rotulados com cores (a, b e c). Para que uma transição desta rede esteja habilitada, é necessário que os lugares que são entrada desta transição tenham marcas do tipo (cor) associado ao arco que interliga estes lugares à transição. A transição t_0 não se encontra habilitada, pois o lugar p_0 não possui uma marca da cor a, contudo a transição t_1 está habilitada, pois o lugar p_1 possui uma marca da cor a e o lugar p_2 possui marcas da cores a,b, o que satisfaz as condições rotuladas nos arcos que interligam estes lugares à transição t_1. O disparo desta transição retira as marcas das cores associadas aos arcos, dos lugares de entrada e adiciona aos lugares de saída marcas de cor igual à cor associada ao arco que interliga a transição aos lugares de saída. Neste caso, uma marca de cor c é depositada no lugar p_4.

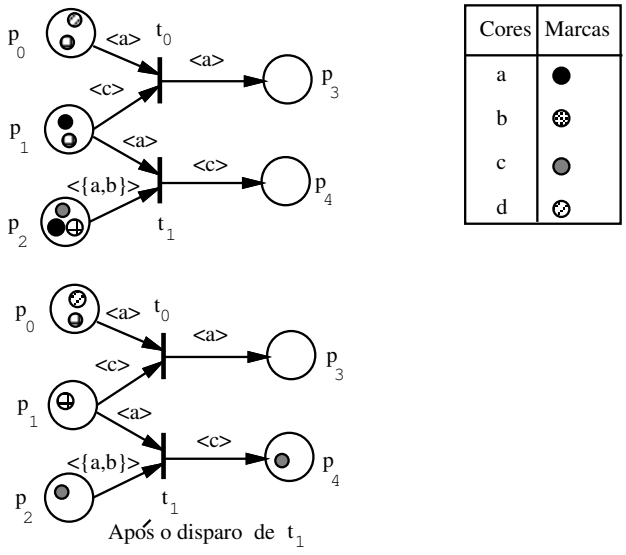

Figura 4.3: Redes Coloridas - Estudos Iniciais

Nas redes que apresentamos, as marcas são identificadas por cores, contudo as marcas podem ser estruturas mais complexas. Nas redes coloridas apresentadas em [75], as marcas são definidas como tipos de dados e é possível efetuar operações complexas sobre estes dados. Passamos a descrever esse tratamento para as redes de Petri coloridas.

Para apresentarmos as redes de Petri coloridas, utilizamos um exemplo em que temos uma rede que modela a manufatura de dois tipos de produtos distintos através de duas linhas de manufatura, onde recursos (máquinas) são compartilhados nos processos de fabricação. Inicialmente, apresentamos uma rede ordinária (*place-transition*) como o modelo (ver figura 4.4) do sistema e, posteriormente, tratamos a rede colorida correspondente. Faremos uso desse exemplo para descrever informalmente as redes de Petri coloridas. Em seguida, apresentamos uma definição formal para essa classe de redes.

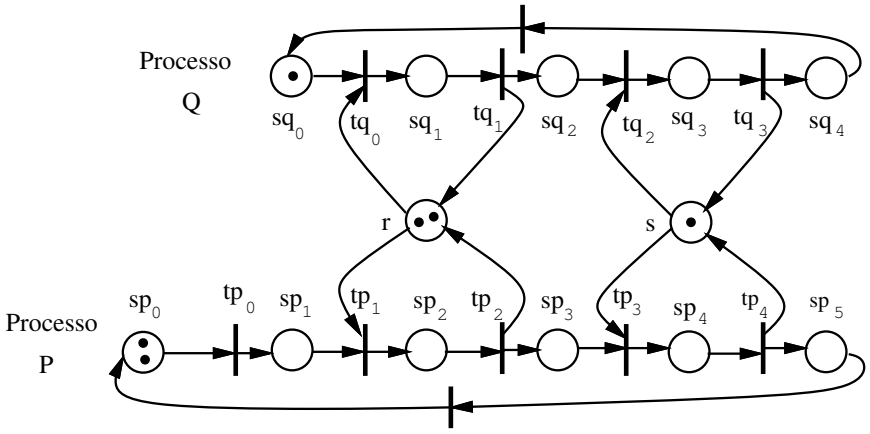

Figura 4.4: Modelo da Linha de Manufatura Usando Redes Ordinárias

Neste exemplo, o sistema é representado por seis estados locais para os processo denominados $p(sp_0, sp_1, sp_2, sp_3, sp_4, sp_5)$ e cinco estados locais para os processos $q(sq_0, sq_1, sq_2, sq_3, sq_4)$; temos uma instância do processo q e duas do processo p, representados por uma e duas marcas nos lugares sq_0 e sp_0, respectivamente. Os recursos compartilhados são representados pelas marcas dos lugares r e s. Observe-se que se tivéssemos um terceiro processo semelhante compartilhando os recurso r e s, teríamos que descrever todos os seus estados, ações e relações entre o novo processo os atuais e os recursos diponíveis. Portanto, torna-se claro que, para um número não muito grande de processos, haveria um variação considerável nas dimensões dos modelos.

As redes de Petri coloridas são compostas por três diferentes partes:

- estrutura,

- declarações e
- inscrições.

A estrutura é um grafo dirigido com dois tipos de vértices (lugares e transições). Os lugares são representados graficamente por círculos ou por elipses e as transições por retângulos. Cada lugar pode armazenar diversas marcas e cada uma dessas marcas pode representar valores associados a tipos de dados complexos. As declarações compreendem a especificação dos conjuntos de cores e declarações de variáveis. Cada lugar tem um conjunto de cores associado, ou seja, um tipo associado. Dessa forma as marcas que são armazenadas nesse lugar devem ter a mesma cor. Nessa classe de redes, cada lugar, transição e arcos têm inscrições associadas. Os lugares têm três tipos de inscrições: nomes, conjunto de cores e expressão de inicialização (marcação inicial). As transições possuem dois tipos de inscrições: nomes e expressões guardas, e os arcos apenas um tipo de inscrição dado pela expressão. Para facilitar a distinção dessas inscrições, os nomes são escritos usando-se letras planas, as cores em itálico, as expressões de inicialização são sublinhadas e as expressões guardas são colocadas entre colchetes. Os nomes associados aos lugares e transições não têm nenhum significado formal, contudo facilitam a identificação. A expressão de inicialização de um lugar é avaliada para *bags* do conjunto de cores. As expressões guardas associadas às transições são expressões *booleanas* que devem ser atendidas para que seja possível o disparo das transições. As expressões associadas aos arcos podem conter variáveis, constantes, funções e operações.

Na figura 4.5 apresentamos uma rede colorida que descreve o processo descrito na figura 4.4. Na rede *place/transition* que modela o processo, os subestados de cada processo são explicitados. Esses estados foram representados através de lugares comuns na rede colorida correspondente, contudo as marcas são distinguíveis.

A rede é formada por sete lugares e seis transições. O conjunto de cores que são armazenados nos lugares s_0, s_1, s_2, s_3, s_4 e s_5 são da cor $Proc*I$, onde $Proc = p|q$ e $I = INT$. As cores associadas ao lugar *recurso* correspondem aos tipos de recursos disponíveis, no caso $rec = r|s$. Nos lugares não-marcados, por conveniência, omitimos as expressões de inicialização. Os lugares s_0, s_1 e *recurso* possuem as seguintes funções de inicialização, respectivamente: $2'(p,0)$, $1'(q,0)$ e $2'r + 1's$. Por exemplo, o lugar s_0 possue duas marcas, onde a cor do processo é p e o zero corresponde ao número

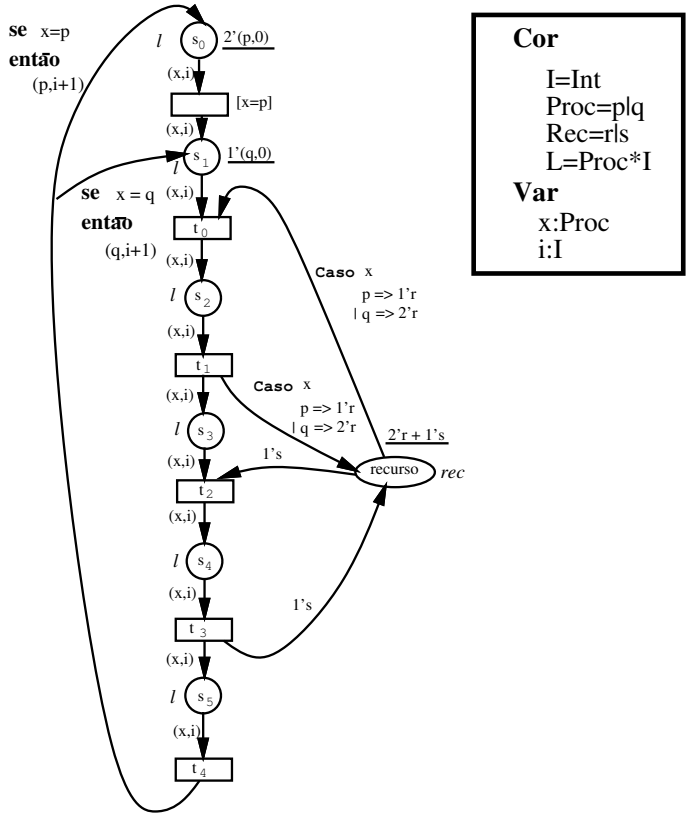

Figura 4.5: Modelo da Linha de Manufatura Usando Redes Coloridas

de iterações realizadas. As marcas inicialmente disponíveis no lugar $recurso$ são duas marcas da cor r e uma da cor s.

Inicialmente, as transições t_0 e t_1 estão habilitadas. A transição t_1 pode ser disparada, pois há uma marca no lugar s_1 e marcas no lugar $recurso$. Sendo atribuída a cor q à variável x, a expressão do arco que interliga o lugar $recurso$ e a transição t_1 é avaliada para q. Dessa forma, o valor associado ao arco é $2'r$. O disparo dessa transição remove uma marca da cor $(q, 0)$ do lugar s_1 e duas marcas da cor $2'r$ do lugar $recurso$.

A seguir, apresentamos um outro exemplo em que verificamos o uso e os benefícios da modelagem com as redes de Petri coloridas em função do seu maior poder de concisão quando comparado com as redes lugar/transição (*place/transition*). Na figura 4.6 apresentamos uma rede que é o modelo do problema do jantar do filósofos descrito no capítulo 2. Nessa rede temos três filósofos que podem esta *comendo, pensando* ou *com-fome*. Os filósofos compartilham os garfos representados pelos lugares $garfo_1, garfo_2$ e $garfo_3$.

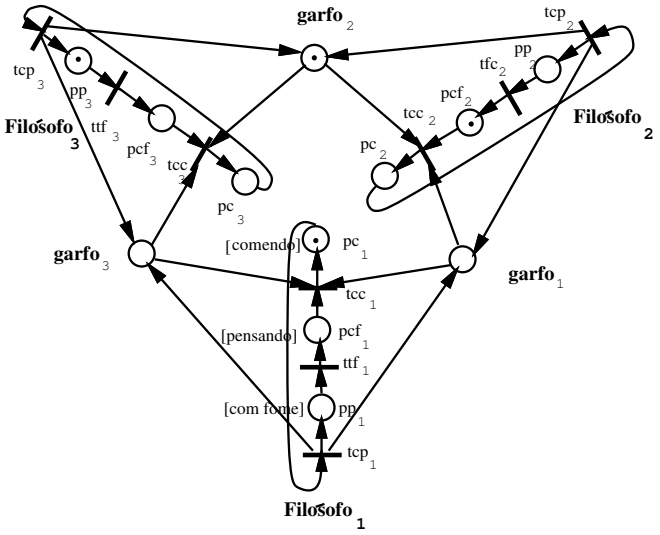

Figura 4.6: Jantar dos Filósofos - Modelo com Redes *Place/Transition*

Na figura 4.7 apresentamos uma rede colorida que representa o modelo acima, contudo nesse modelo é ressaltado o número de iterações de cada processo. Os lugares *H, E* e *Th* representam os estados dos processo e o

lugar *Fork* os recursos, ou seja, os garfos dos filósofos. O disparo da transição t_0 pode ser efetuado segundo as três atribuições possíveis às variáveis x e i, ou seja, a expressão de inicialição do lugar H quando avaliada resulta em três marcas de cores distintas, representando os filósofos e o número de iterações. Caso atribua-se à variável x a cor $p(t_0, x = p)$ a expressão do arco que interconecta o lugar $Fork$ ao lugar t_0 é avaliada para $1'f_1 + 1'f_2$. Os disparo de t_0 remove a marca $1'(p, 0)$ do lugar H e as marca $1'f_1 + 1'f_2$ do lugar $Fork$ e deposita a marca $1'(p, 0)$ no lugar E. Caso a atribuição seja outra, por exemplo $x = q$, a avaliação das expresões dos arcos seria outra e ao dispararmos a transição obteríamos uma outra marcação. Denomina-se o par (transição, atribuição) por *elemento de ocorrência.*

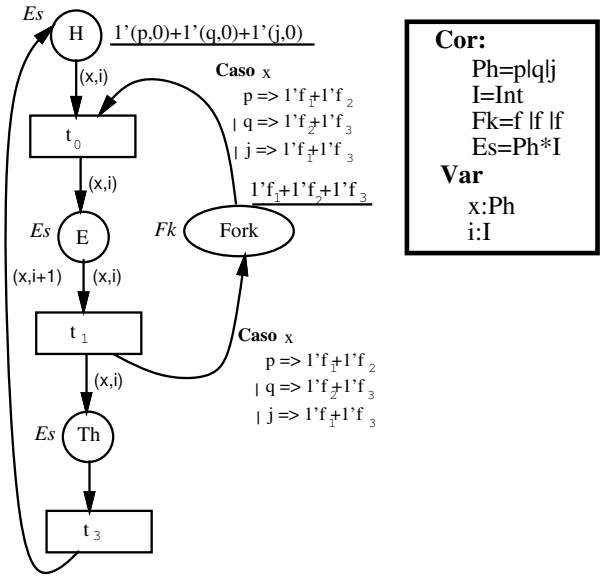

Figura 4.7: Jantar dos Filósofos - Modelo com Redes Coloridas

A seguir, apresentamos a definição formalizada das redes de Petri coloridas e algumas consideração sobre os aspectos dinâmicos dessa classe de rede.

Definição 4.2.1 - Redes de Petri Coloridas: *define-se as redes de Petri coloridas como* $RdPC = (R, \sum, C, G, E, In)$, *onde* $R = (P, T, A)$, P *é um*

conjunto de lugares, T *é um conjunto de transições,* A *é um conjunto de arcos. Os elementos de* A *são arcos que conectam transições a lugares ou lugares a transições* ($A \subseteq (PxT) \bigcup (TxP)$). $\sum$ *é o conjunto finito não-vazio de tipo, denominados cores.* C *é a função de coloração* $C : P \rightarrow Cor$, G *é a função de guarda* $G : T \rightarrow exp$, *onde* exp *é uma expressão tal que* $\forall t_i \in T \mid Tipo(G(t)) = bool$ *e o* $Tipo(Var(G(t))) \in \sum$, E *é uma função associada aos arcos* $E : A \rightarrow exp$, *onde* $\forall a \in A \mid Tipo(E(a)) = C$ *e o* $Tipo(Var(E(a))) \in \sum$ *e* In *é a expressão de inicialização, onde* $\forall p_i \in P \mid Tipo(In(p)) = C(p)$ *e* $Var(In(p)) = \emptyset$.

Consideramos $Var(t)$ como o conjunto de variáveis associadas à transição t. Essas variáveis podem ser associadas às transições de duas maneiras: nas expressões guardas($G(t), \ t \in T$) e nas expressões do arcos($E(a), \ a \in A$) que interconectam lugares e transições.

Definição 4.2.2 - Tipo da Ligação*: para uma transição* $t \in T$ *com* $Var(t) = \{v_1, ..., v_n\}$ *define-se o tipo da ligação por* $BT(t) = Tipo(v_1) \times ... \times Tipo(v_n)$.

Definição 4.2.3 - Conjunto de Ligações*: define-se o conjunto de ligações de uma transição por* $B(t) = (c_1, ...c_n), \ \in BT(t)$ *ou mesmo por* $B(t) = G(t) < v_1 = c_1, ..., v_n = c_n >$.

Definição 4.2.4 - Marcação*: a distribuição de marcas é uma função* $M : P \rightarrow \sum$ *e uma marcação de uma rede colorida é uma distribuição de marcas.*

A marcação inicial das redes coloridas é obtida através da avaliação da expressão de inicialização para cada lugar da rede, ou seja, $M_0(p) = IN(p) <> \forall p \in P$.

A distribuição de ligações é uma função $Y : T \rightarrow B$. Denomina-se elemento de Y o par (t, b), onde b é uma ligação, tal que $b \in Y(t)$. Denomina-se por *passo* (*step*) uma distribuição de ligações não-vazia.

Um passo ($Y = (t, b)$) está habilitado quando o número de marcas de cores correspondente à ligação do passo é maior ou igual à avaliação da expressão do arco que interliga os lugares de entrada à transição t.

Definição 4.2.5 - Passo Habilitado*: um passo está habilitado se, e somente se,* $\sum_{(t,b) \in Y} E(p, t) < b > \leq M(p), \ \forall p \in P$.

Um passo habilitado pode ocorrer e a ocorrência provoca uma alteração da marcação da rede, pela remoção de marcas dos lugares de entrada e adição de marcas aos lugares de saída.

Definição 4.2.6 - Ocorrência de um Passo: *seja M_0 uma marcação e Y um passo habilitado para essa marcação. A ocorrência do passo Y altera a marcação da rede para uma marcação M_1. A alteração da marcação da rede é definida pela equação:* $M_1 = (M_0 - \sum_{(t,b)\in Y} E(p,t) < b >) + \sum_{(t,b)\in Y} E(t,p) < b >$, *onde $E(p,t)$ e $E(t,p)$ correspondem às expressões dos arcos de entrada e saída da transição(em t) do passo, respectivamente.*

Diz-se que uma uma marcação M'' é diretamente acessível de M' se a partir de M' e pela ocorrência do passo Y alcança-se M'', ou seja, $M'[Y > M''$.

4.3 Redes Hierárquicas

Nesta seção, apresentamos as redes de Petri hierárquicas. A modelagem de sistemas de grandes dimensões só é viavel através do uso de mecanismos que possibilitem a estruturação. O conceito de hierarquia em redes de Petri permite o refinamento de lugares e transições.

Do ponto de vista teórico, a hierarquia é apenas uma conveniência gráfica que não adiciona nenhum poder computacional, contudo para a modelagem segura de sistemas de grandes dimensões é necessário o uso de ferramentas que utilizem esses mecanismos.

O uso dos conceitos de hierarquia na modelagem de sistemas possibilita a inspeção considerando-se níveis de detalhamento, visualização de partes dos sistema e re-uso de partes do modelo [79, 78, 77].

O refinamento de lugares e transições tem sido utilizado no processo de modelagem hierárquico, contudo o refinamento de elementos vizinhos apresenta dificuldades no controle da consistência do modelo. O refinamento de um elemento pode ser representado como uma sub-rede(página). Algumas abordagens que tratam hierarquia em redes de Petri possibilitam o refinamento de lugares, de transições, lugares e transições e ainda as que não possibilitam o refinamento de elemento adjacentes.

As redes hierárquicas aqui apresentadas propocionam uma descrição estruturada dos sistemas, possibilitando sua especificação em níveis de abs-

tração que pode ser refinada quando necessário. Nessa classe de rede, os lugares e as transições do níveis superiores podem ser refinados, ou seja, podem ser sub-redes da rede global. Estas sub-redes fornecem uma descrição mais detalhada do lugar ou da transição do nível superior. A *interface* entre um lugar ou transição de mais alto nível e a sub-rede correspondente é efetuada através de transições ou lugares denominados *port-transitions* e *port-places*, respectivamente.

O primeiro morfismo utilizado nesta abordagem para a obtenção de hierarquia em redes de Petri é a substituição de transições. Cada transição de alto nível designa uma página (denominada subpágina). Uma transição de alto nível é vista como uma caixa preta e a subpágina correspondente representa o detalhamento dessa transição. A transição de alto nível e a página em que está inserida são, para a subpágina, denominadas de superpágina.

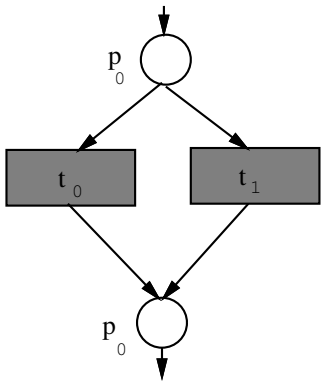

Figura 4.8: Superpágina

Na figura 4.8 apresentamos uma rede de Petri na qual temos duas transições de alto nível. A figura 4.9 apresenta uma página que representa o refinamento da transição t_0.

Para representar claramente a semântica da substituição de transições, apresentamos alguns passos [77], que possibilitam a representação não- hierárquica equivalente da rede:

- Remove-se a transição de alto nível.
- Insere-se a subpágina, que representa o refinamento da transição.

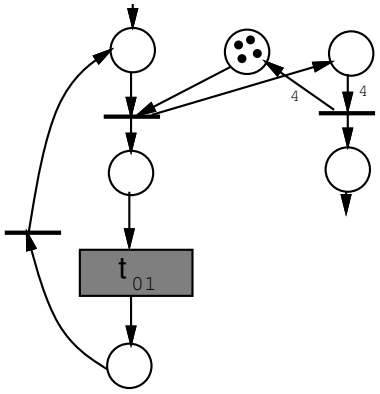

Figura 4.9: Subpágina

- Fundem-se os *port-places* da rede de mais alto nível e da subpágina.

De forma semelhante à substituição de transições, os lugares também podem ser substituídos por redes que representem o seu refinamento(páginas).

Para ilustrar o uso das redes de Petri hierárquicas, apresentamos um modelo que representa um sistema de aquisição e transferência de informações (ver figura 4.10). Esse modelo é dividido em páginas que representam as atividades de aquisição, transmissão e recepção de dados. Cada uma dessas tarefas está representada na rede da figura 4.10 pelas páginas *Ler Dados*, *Trans* e it Recep, respectivamente. O canal através do qual fluem as informação é representado pelo lugar *canal*. Observe que este lugar é interconectado às páginas it Trans e *Recep* por arcos em ambos os sentidos. Como vemos nas subpáginas, esse lugar será refinado, dando origem a dois lugares.

Na figura 4.11 apresentamos as subpáginas que representam a transmissão, recepção e o detalhamento do canal. A página que descreve o processo de transmissão(*Trans*) tem como *port-places* os lugares r_inf, r_proc, *mens* e *reconh*, repectivamente. Esse modelo possibilita a retransmissão da informação caso essa não seja adequadamente recebida.

A página *Recep* tem como *port-places* os lugares *mens* e *reconh*. Esse processo lê a informação enviada pelo transmissor, envia o reconhecimento da recepção da informação e volta a aguardar por uma nova informação a ser recebida. O lugar *canal* é representado pelos lugares *mens* e *reconh*.

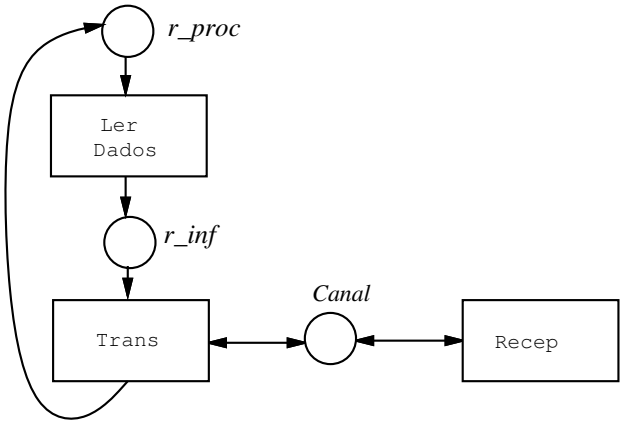

Figura 4.10: Superpágina - Aquisição e Transferência de Informações

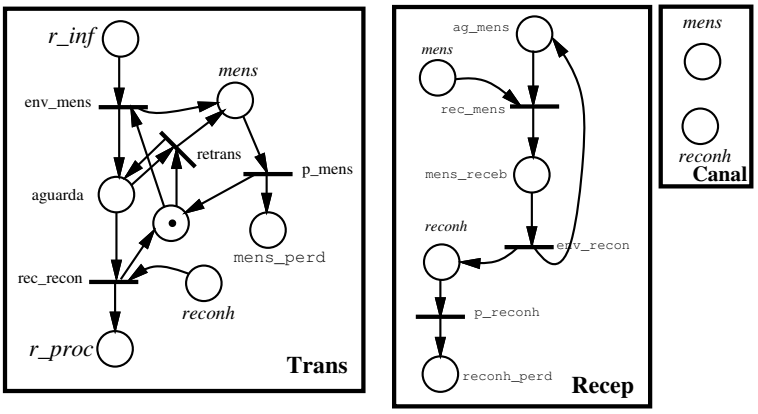

Figura 4.11: Subpáginas - Aquisição e Transferência de Informações

4.4 Rede de Petri Temporizada Determinística

Redes de Petri é um formalismo que possibilita a expressão do comportamento dinâmico de sistemas que tenham atividades concorrentes, assíncronas e não-determinísticas. O conceito de tempo não foi explicitado na definição original das redes de Petri. No entanto, na especificação de sistemas de tempo real, avaliação de desempenho e problemas de escalonamento (*scheduling*) é necessária a introdução de retardos de tempo. Nesta seção apresenta-se a rede de Petri temporizada determinística [2] [36].

Uma outra maneira de se tratar requisitos temporais através das redes de Petri é através do ponto de vista probabilístico do disparo das transições. Diversos autores propuseram independentemente a associação de retardos de disparo exponencialmente distribuídos às transições. Essas redes são denominadas redes de Petri estocásticas, pois o comportamento dessas redes pode ser descrito por um processo estocástico.

Entre as diversas abordagens de redes de Petri que tratam requisitos temporais, apresentamos algumas propostas com relação à forma em que os tempos são especificados:

- O tempo pode estar associado aos lugares, onde as marcas armazenadas nos lugares de saída, após o disparo de uma transição, só estarão disponíveis para disparar uma nova transição após um determinado tempo que é associado ao lugar.

- O tempo pode estar associado às marcas, onde as marcas possuem uma informação que indica quando a marca estará disponível para disparar a transição.

- O tempo é associado à transição.

Nesta seção, tratamos das redes de Petri temporizadas determinísticas com tempos associados às transições. Nesta classe de rede, após a sensibilização de uma transição o disparo desta deve ocorrer após o tempo associado.

Definição 4.4.1 - Rede Temporizada: *uma rede de Petri temporizada é uma dupla* $Rt = (RM, D)$, *onde* RM *é uma rede marcada e* D *é um vetor* $D : T \rightarrow N$ *com* $\#T$ *que associa a cada transição* t_i *de* RM *um tempo* d_i.

Essa classe de redes de Petri temporizadas, pode ser classificada como de disparo atômico ou de disparo de três fases. Nas redes onde o disparo das transições é atômico, as marcas permanecem nos lugares de entrada das transições até o tempo associado à transição ter sido"consumido" e então as marcas são removidas dos lugares de entrada e imediatamente armazenadas nos lugares de saída (ver figura 4.12).

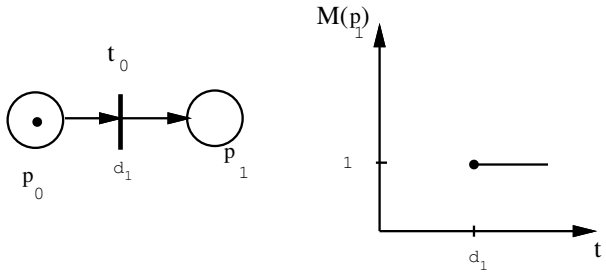

Figura 4.12: Transição com Disparo Atômico

Nas redes temporizadas com disparo em três fases, ao se habilitar uma transição, as marcas são imediatamente consumidas dos lugares de entrada, contudo são apenas depositadas nos lugares de saída após o tempo associado à transição disparada. Apresentamos aqui as redes de Petri temporizadas com disparo atômico, pois essas possibilitam explorar a teoria desenvolvida para as redes de Petri não-temporizadas.

Nas redes temporizadas com disparo atômico, uma transição ao ser habilitada, um contador com o tempo associado à transição é iniciado. O contador é decrementado a uma taxa constante. Quando o contador atinge o valor zero, a transição dispara.

Observe-se, contudo, o problema que se tem quando duas transições estão habilitadas para uma dada marcação M'. Na figura 4.13 temos uma rede que representa essa situação.

Nessa rede, a transição que será disparada será aquela que possuir o menor tempo associado. Um fato interessante é ressaltado com o disparo de uma dessas transições, por exemplo, suponha $d_1 < d_2$, o que proporciona o disparo da transição t_1. Nesse caso, qual será a situação do contador associado à transição t_2, após o disparo de t_1 ? Algumas políticas têm sido utilizadas para tratar esse problema, entre essas:

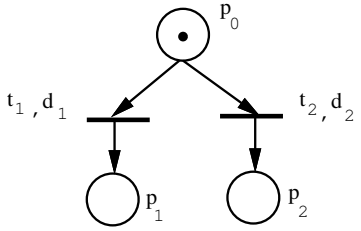

Figura 4.13: Marcação Habilitando Duas Transições

- Reamostragem: de acordo com essa política, a cada disparo de transição todos os contadores de todas as transições serão reiniciados, ou seja nenhuma informação anterior é registrada.

- Com Memória: nessa política, quando ocorre um disparo de uma transição, os contadores associados a cada transição que se torna desabilitada são reiniciados. Os contadores das transições que permanecem habilitadas mantêm os seus valores.

Para ilustrar o uso da redes de Petri temporizadas determinísticas, apresentamos a modelagem de um sistema de contagem síncrono (ver figura refrtc1) por redes de Petri temporizadas. Este sistema percorre quatro estados (S_0, S_1, S_2 e S_3), mudando de estado a cada pulso negativo do sistema de relógio. Dependendo da entrada (X) a contagem pode ser crescente ($S_i \rightarrow S_{i+1}$) ou decrescente ($S_{i+1} \rightarrow S_i$).

Dividimos o modelo desse sistema em dois submodelos: o do sistema de contagem propriamente dito e do sistema de relógio. O sistema de relógio (ver figura 4.15) fornece uma forma de onda com 700 ms em nível alto e 700 ms em nível baixo. Representamos o nível baixo por uma marca no lugar clk_L e o nível alto por uma marca no lugar clh_H. As mudança de nível são representadas pelas transições t_{LH} e t_{HL}. Obeserve-se que essas transições têm associado a elas o tempo de 700 ms.

Na figura 4.15 apresentamos a alteração da marcação dos lugares clk_L e clk_H e função do tempo.

O sistema de contagem (ver figura 4.16) possui quatro estados, representados pelos lugares S_0, S_1, S_2 e S_3, dois lugares que representam o alfabeto

Figura 4.14: Sistema de Contagem

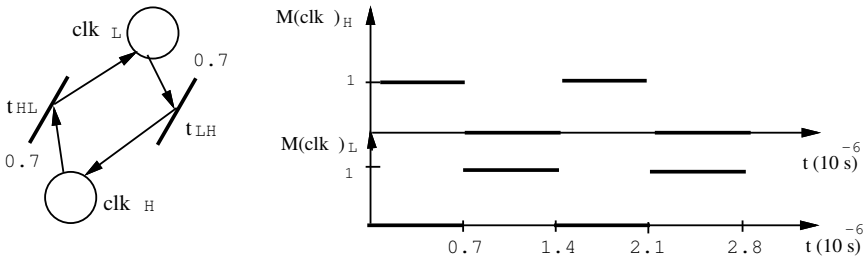

Figura 4.15: Relógio

de entrada($X = 0$ e $X = 1$) e as transições que representam as mudanças de estado, segundo o estado anterior e a entrada. Essas transições têm tempos de 500 *ms* associado.

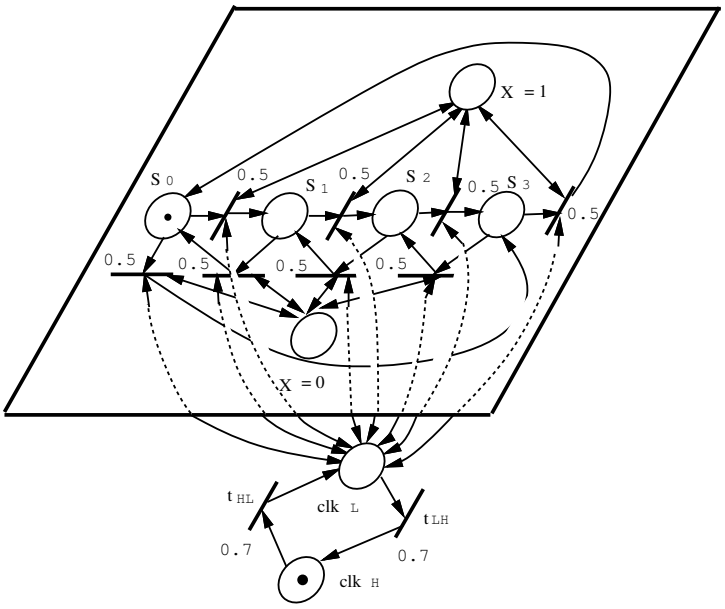

Figura 4.16: Modelo

Capítulo 5

Aplicações

Neste capítulo, apresentamos dois exemplos da utilização das redes de Petri na modelagem de sistemas reais. Inicialmente descrevemos um método de transcrição de programas concorrentes em *OCCAM* para redes de Petri. *OCCAM* é uma linguagem de programação, implementada a partir de *CSP*, projetada para expressar algoritmos concorrentes, possibilitando a descrição de aplicações através de coleções de processos concorrentes. Cada processo de uma aplicação descreve o comportamento de um aspecto particular da implementção.

Os programas em *OCCAM* são construídos através da combinação de processos *primitivos*. Neste capítulo, apresentamos os modelos básicos em redes de Petri que representam os processos primitivos e uma forma de obtenção de modelos mais complexos através das redes básicas e estruturas que possibilitam a combinação destes modelos.

Como segundo exemplo, desenvolvemos um método de especificação de sistemas de controle industrial [3, 22, 25, 27, 28, 80] através das redes de Petri. O processo de modelagem dos Sistemas de Controle Seqüencial parte da descrição comportamental de cada componente, através de regras operacionais. A partir das regras operacionais, são construídas as redes de Petri elementares que modelam cada componente do sistema. A rede de Petri global é obtida pela técnica de fusão de "lugares"comuns das redes elementares. Este processo garante a preservação das propriedades das redes elementares no modelo global. Apresentamos, ainda, um método de diagnóstico de falhas em sistemas de controle seqüencial baseado nas redes de Petri obtidas segundo o método de modelagem descrito em [22].

5.1 Tradução *OCCAM*-Redes de Petri

OCCAM [84] é uma linguagem de programação, implementada a partir de *CSP* [56], projetada para expressar algoritmos concorrentes, possibilitando a descrição de aplicações através de coleções de processos. Cada processo de uma aplicação descreve o comportamento de um aspecto particular da implementação. Iniciamos esta seção apresentando os processos primitivos, os combinadores de processos da linguagem *OCCAM* em estilo BNF e um método de tradução de programas em linguagem *OCCAM* para as redes de Petri. A seguir apresentamos a descrição dos processos primitivos, assim como dos combinadores:

$$\begin{aligned} P \quad ::= \quad & SKIP \mid STOP \mid x := e \mid ch?x \mid ch!e \\ & \mid IF(c_1p_1, ..., c_np_n) \mid ALT(g_1p_1, ..., g_np_n) \\ & \mid SEQ(p_1, ..., p_n) \mid PAR(p_1, ..., p_n) \end{aligned}$$

- O processo primitivo *SKIP* comporta-se como se nenhuma ação houvesse ocorrido e termina com sucesso.
- *STOP* é o processo que expressa o *deadlock*.
- $x := e$ representa a atribuição da valor da expressão e à variável x.
- $ch?x$ e $ch!e$ representam os processos de leitura e escrita nos canais de comunicação, respectivamente. O processo de leitura, quando executado, lê a informação disponível no canal, atribui essa informação à variável x e envia ao processo transmissor o reconhecimento da recepção da informação. O processo transmissor envia através do canal o valor da expressão e e aguarda o reconhecimento do processo receptor.
- *IF* é o combinador condicional que tem como argumento um lista de termos na forma c_ip_i, onde c_i é uma expressão lógica e p_i um processo.
- *ALT* é um combinador que tem como argumento uma lista de alternativas na forma g_ip_i, onde cada g_i é uma expressão guarda e p_i um processo. A primeira expressão guarda que se torna verdadeira ativa o processo correspondente. Estando duas expressões guardas habilitadas em um dado momento, uma das alternativas é escolhida não-deterministicamente. Quando nenhuma das expressões guardas é verdadeira, o processo comporta-se como o *STOP*. A expressão guarda pode ser uma expressão lógica ou mesmo o estado de um canal.

- $SEQ(p_1, ..., p_n)$ representa a composição seqüencial dos processos p_1, ..., p_n.
- $PAR(p_1, ..., p_n)$ representa a composição paralela dos processos p_1, ..., p_n.

Na seção seguinte, apresentamos um método de tradução de programas escritos em *OCCAM* para redes de Petri.

5.1.1 Modelagem dos Processos Primitivos

Os programas em *OCCAM* são construídos através da combinação de processos. O processo mais simples em *OCCAM* é uma ação. Uma ação pode ser uma atribuição, uma entrada ou mesmo uma saída.

5.1.1.1 Atribuição

Um exemplo é a atribuição a uma variável x o valor de uma expressão *exp*. Abaixo apresentamos a sintaxe da atribuição.

$$variável := exp$$

Consideremos o seguinte exemplo: $x := y + 1$. Para modelarmos este processo, usamos uma extensão das redes de Petri que associa as redes de Petri aos grafos de fluxos de dados. Esta extensão foi analisada no capítulo anterior.

Nesta rede os arcos do grafo de fluxo de dados (figura 5.1.b) são rotulados com p_1, de tal forma que só é possível realizar a operação *op*, particularmente a adição, quando houver uma marca em p_1 (figura 5.1.a).

5.1.1.2 Comunicação

A comunicação é uma das partes pricipais da linguagem *OCCAM*. Os processos comunicam-se através de canais através dos quais são enviados e recebidos valores, ou seja, dois tipos de ações podem ocorrer: uma ação de entrada *input* ou uma ação de saída *output*.

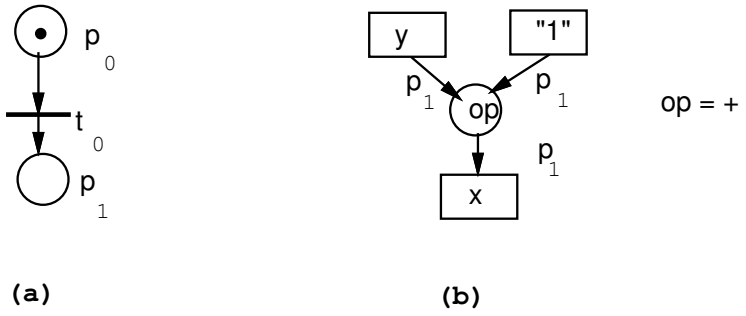

Figura 5.1: Atribuição

Input

Através de uma ação *input* o processo recebe um valor por um canal e este valor é atribuído à uma variável. Na figura 5.2 apresentamos a rede que representa o canal de comunicação de entrada $canal_1?x$. Neste exemplo, o canal denominado $canal_1$ aguarda uma informação; quando esta é recebida, então atribui-se este valor à variável x.

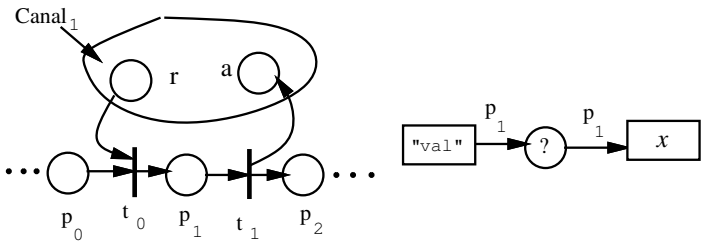

Figura 5.2: *Input*

Quando uma marca chegar ao lugar p_0, o processo aguarda por um valor *val* de entrada. Quando este valor for transmitido uma marca no lugar r indicará esta condição, possibilitando o disparo da transição t_0 (recepção da informação). Ao disparar-se esta transição, as marcas dos lugares p_0 e r são retiradas e colocada uma marca no lugar p_1. Observe-se que no grafo de fluxo de dados, os arcos são rotulados com p_1, o que possibilita a atribuição

do valor de entrada *val* à variável x. Uma marca no lugar p_1 possibilita o disparo de t_1, que retira uma marca deste lugar depositando uma marca nos lugares p_2 e a. A marca no lugar a informa ao processo transmissor a recepção da informação transmitida, e a marca no lugar p_2 possibilita a continuidade do processo.

Output

Através da ação *output* é possível transmitir-se um valor de uma expressão por um canal especificado. Na figura 5.3 apresentamos a rede que representa o canal de comunicação de saída $canal_2!y$. Neste exemplo o valor da variável y é transmitida por *canal* $_2$ e aguarda até que esta informação tenha sido recebida pela entrada correspondente.

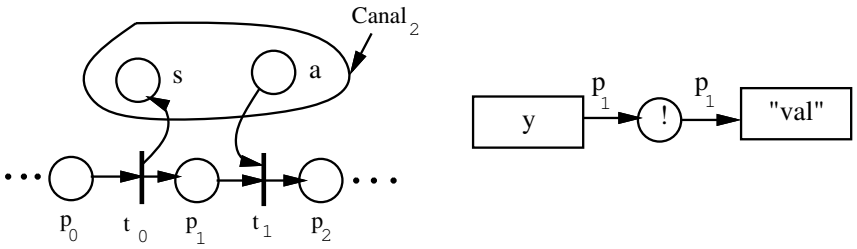

Figura 5.3: *Output*

Uma marca no lugar p_0 habilita o disparo da transição t_0. O disparo desta transição (transmissão) retira uma marca do lugar p_0 e deposita marcas nos lugares s e p_1. A marca no lugar s informa ao outro processo comunicante (receptor) que uma informação foi transmitida. A transição t_1 pode ser disparada quando o receptor enviar o reconhecimento da recepção da mensagem através de uma marca no lugar a, dado que o disparo de t_0 depositou uma marca no lugar p_1. O disparo de t_1 retira uma marca dos lugares p_1 e a e deposita uma marca no lugar p_2, que habilita a execução da próxima ação.

5.1.1.3 *Skip* e *Stop*

Nesta seção apresentamos a modelagem dos dois últimos processos primitivos de *OCCAM*. O processo *skip* quando iniciado não executa nenhuma

ação observável e então termina. A figura 5.4.a apresenta o modelo que corresponde ao processo *skip*. Representamos a ação executada pelo processo (nenhuma) como uma transição não-observável, ou seja, executa uma ação não relevante. Esta sub-rede poderia ser suprimida. A figura 5.4.b apresenta um exemplo simples, onde o processo através do canal p recebe uma informação, atribui esta informação à variável c, executa o processo *skip* e então transimite a informação da variável c através do canal s. A rede que modela este programa está representada na figura 5.4.c.

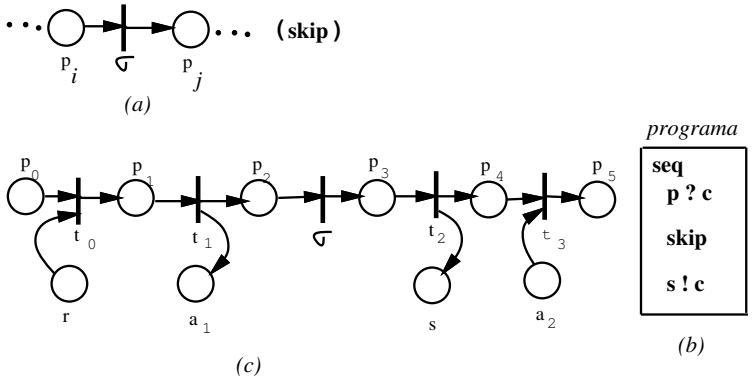

Figura 5.4: *Skip*

A transição σ poderia ser suprimida do modelo, dado que não expressa qualquer ação que venha alterar o comportamento do sistema. Desta forma os lugares p_2 e p_3 poderiam ser fundidos em um só lugar.

Quando o proceso *stop* é iniciado, nenhuma ação é executada e o processo nunca termina. A figura 5.4.a apresenta uma rede que representa este processo. Observe-se que o lugar p_i não é entrada de qualquer transição, portanto o disparo da transição σ não possibilita o disparo de uma transição subsequente. A figura 5.4.b apresenta um pequeno exemplo em que usamos o processo *stop*. Neste exemplo após a leitura do canal x executa-se o processo *stop*. A execução de stop nunca termina o que impossibilita a escrita através do canal y. A figura 5.5.c apresenta a rede que representa este programa. O lugar p_j não tem qualquer transição como saída, e o lugar p_2 não tem qualquer transição como entrada. Desta forma a transição t_3 nuca será disparada. A transição σ poderia ser suprimida do modelo, dado que não expressa qualquer ação que venha alterar o comportamento do sistema. No

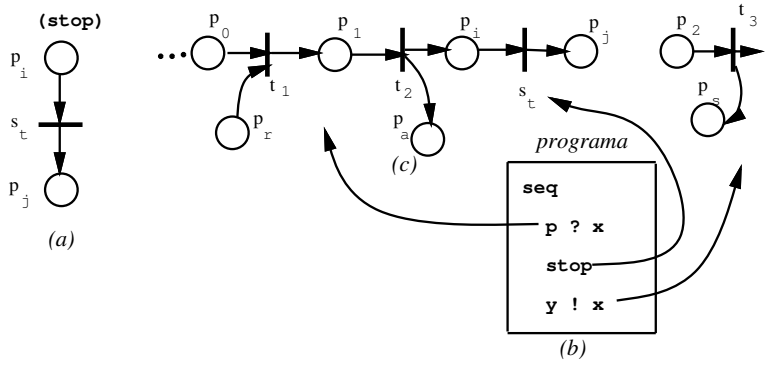

Figura 5.5: *Stop*

entanto, o lugar p_2 deve continuar sem qualquer transição como entrada.

5.1.2 Modelagem dos Combinadores

As aplicações desenvolvidas em *OCCAM* são construídas através da combinação de processos primitivos. Nesta seção apresentaremos as redes que representam estes combinadores de processos, bem como alguns pequenos exemplos que demostram a sua aplicação.

5.1.2.1 Seqüência

O primeiro combinador é o *seq*. Este construtor combina processos onde um só será executado após a execução do outro, como pode ser visto na figura 5.6.a. O programa apresentado é elaborado com a utilização de dois processos primitivos. O combinador *seq* determina que a execução desses processos ocorrerá de forma seqüencial, ou seja, a escrita sobre o canal s só será executada após a leitura sobre o canal k. A figura 5.6.b apresenta uma rede de Petri hierárquica, onde os blocos k e s são redes que modelam a comunicação através dos canais entrada (k) e saída (s) (ver seção 5.1.1.2).

É natural que os elementos de conexão da rede k e da rede s sejam lugares. Nesta rede (figura 5.6.b) observamos que o disparo de *seq* desencadeará o processo s.

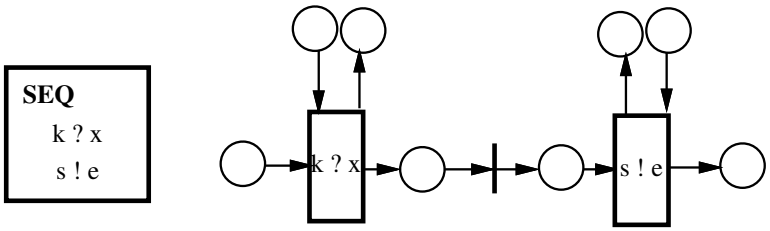

Figura 5.6: *Sequence*

OCCAM possibilita a replicação possibilitando que um número de processos similares sejam executados em seqüência. A modelagem desta contrução pode ser realizada de forma semelhante à apresentada na figura anterior.

5.1.2.2 *Conditional*

O combinador condicional *if* combina um conjunto de processos guardados por expressões *booleanas*. Estas expressões são avaliadas em seqüência; quando uma das expressões é avaliada como verdadeira, o processo associado a esta condição é executado. No entanto, se nenhuma expressão é avaliada como verdadeira, o processo comporta-se como o processo primitivo *stop*. Podemos usar, contudo, a expressão constante **TRUE**. Esta constante deve ser colocada como última opção, pois é sempre verdade, ou seja, se nenhuma outra expressão for avaliada como verdadeira, nesta a avaliação será. Neste contexto **TRUE** deve ser como *caso contrário*.

Como verificamos na seção 2.1.4, as redes de Petri possibilitam a modelagem da escolha não-determinística. No entanto observamos também na seção 1.10 que podemos associar, as transições da rede, condições externas (expressões *booleanas*, informações provenientes de sensores etc). Assim sendo utilizaremos estas características para a modelagem deste combinador.

A figura 5.7.a apresenta um programa que usa o combinador condicional. Neste exemplo, cada expressão booleana guarda os processo de comunicação, onde, se $x < y$ é verdadeira, é executada uma leitura através do canal k; se $x = y$ é verdadeira, uma escrita é executada através do canal s e, caso contrário (**TRUE**), efetua-se uma atribuição à variável x.

Quando houver uma marca no lugar p_0 e a expressão $x < Y$ for avaliada

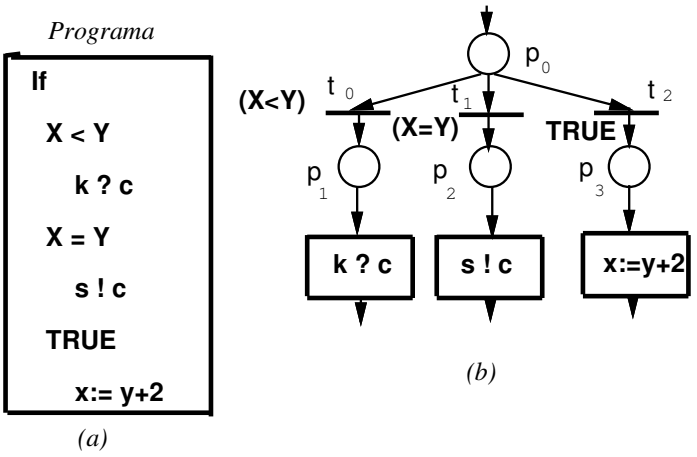

Figura 5.7: *Conditional*

como *verdade*, é possível disparar-se a transição t_0. O disparo desta transição retira a marca do lugar p_0 e deposita uma marca no lugar p_1, impossibilitando o disparo de t_1 e t_2. A marca no lugar p_1 possibilita a execução da rede k *?* c. Caso a avaliação da expressão $x < y$ não seja verdadeira e a expressão $x = y$ seja, a transição t_1 poderá ser disparada, o que impossibilitará o disparo de t_0 e t_2. O disparo de t_1 permitirá a execução da rede s *!* c. E caso as duas expressões anteriores não sejam avaliadas como verdadeira a terceira transição (t_2) poderá ser disparada, dado que a condição associada a esta transição é avaliada sempre como verdade, portanto, possibilitando a execução da rede $x := y + 2$.

5.1.2.3 *Loop*

Nesta seção apresentaremos a modelagem da repetição de processos, enquanto expressões *booleanas* são avaliadas como verdadeiras. Apresentamos, como exemplo, dois processos p e r que são repetitivamente executados, enquanto uma expressão *booleana* for vedadeira. Na figura 5.8 apresentamos esse exemplo e a rede de Petri que o representa.

A modelagem desse combinador é baseada na rede básica que descreve o conflito, sendo as transições rotuladas por expressões *booleanas* que serão

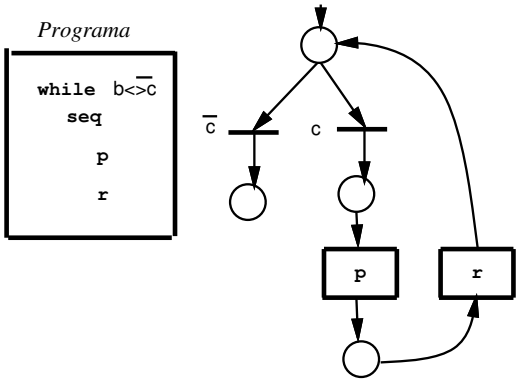

Figura 5.8: *Loop*

avaliadas, o que possibilitará ou não o disparo dessas transições. Observe-se no exemplo da figura 5.8 que o *corpo* da expressão de repetição é construído através de dois processos que são executados de forma sequüencial, ou seja, o processo p é executado antes do processo r. Enquanto a expressão b for falsa, esses processos serão executados repetitivamente.

5.1.2.4 Paralelismo

O combinador (combinador) *Parallel* é um dos mais poderosos da linguagem *OCCAM*, pois possibilita a combinação de processos que são executados concorrentemente. Os processos combinados através deste combinador começam a ser executados simultaneamente e este combinador só terminará quando todos os processo terminarem.

Baseamos a modelagem deste combinador nas redes básicas apresentadas nas seções 2.1.2, 2.1.3 (distribuição e junção, respectivamente). No exemplo da figura 5.9.a observamos um programa que tem como componentes três processos *E, K* e *S*, onde o processo *K* comunica-se com o processo *E* através do canal *t.in* e o processo *S* comunica-se com o processo *E* através do canal *t.out*. Na figura 5.9.b, uma marca no lugar p_0 habilita o disparo de t_0 o que possibilita a execução das sub-redes (páginas) que modelam os processos *E, K* e *S*, ou seja, o disparo da transição t_0 retira uma marca do lugar p_0 e coloca uma marca nos lugares p_1, p_2 e p_2 (*port-places*). Essas novas condições

estabelecidas permitem a execução de outras tarefas paralelamente, ou seja, p_1 é pré-condição para a execução da rede K, p_2 pré-condição para a execução da rede E e p_3 pré-condição para a execução de S.

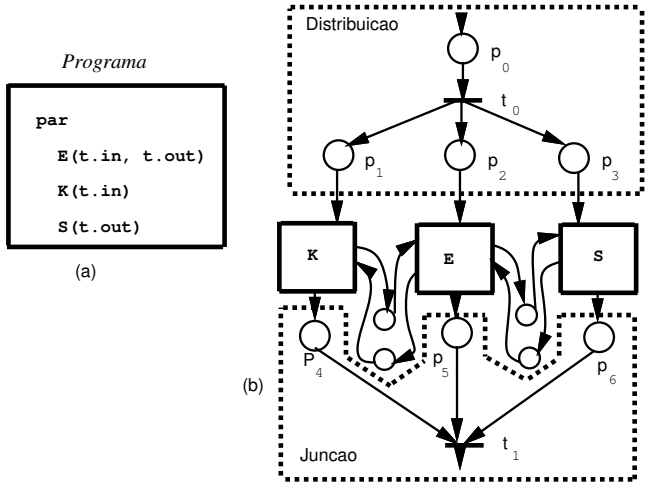

Figura 5.9: *Parallel*

Quando todas as redes forem executadas (K, E e S), serão colocadas marcas nos lugares p_4, p_5 e p_6 (*port-places*). Esta marcação possibilita o disparo da transição t_1, que representa o término da execução dos processos combinados (sincronização).

5.1.2.5 *Alternation*

Este combinador possibilita-nos agregar processos guardados por entradas e/ou expressões *booleanas*. Suponhamos que dois processos são combinados através do *Alternation*. A execução desses processos está diretamente associada à avaliação da expressão ou do canal que *guarda* estes processos, ou seja, o processo só será executado se a expressão *booleana* for avaliada como verdadeira ou se o canal que *guarda* esses processos tiverem alguma informação. No entanto, se os dois processo estiverem em condições de serem executados (canal pronto e/ou expressão *booleana* avaliada como verdadeira), apenas um destes será executado. Está escolha é feita não-determinísticamente. Caso os canais ou expressões *booleanas* não habilitem

a execução destes processos, o combinador aguarda até que algum desses processos torne-se habilitado. A figura 5.10.a exemplifica a utilização deste combinador. Neste exemplo, temos um processo primitivo de escrita através de um canal denominado *trans*.

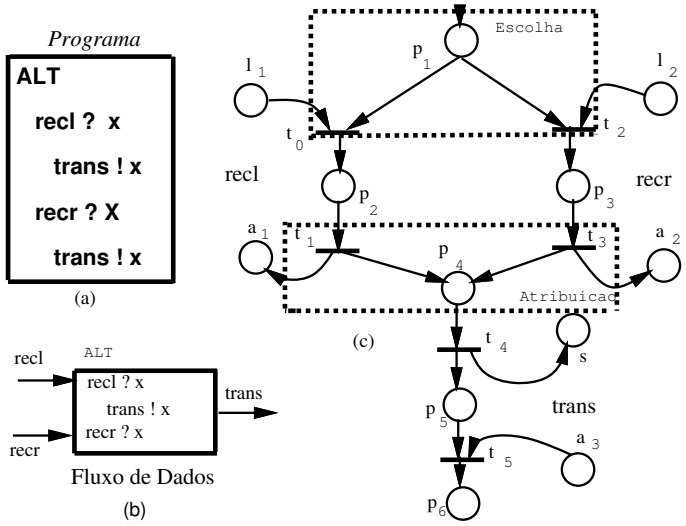

Figura 5.10: *Alternation*

O modelo desse combinador pode ser construído usando-se os modelos dos processos primitivos de escrita e leitura sobre canais, apresentados no início da seção, e as redes elementares que modelam a escolha e a atribuíção (ver seção 2.3.2).

No exemplo da figura 5.10.a apresentamos um *merging* de dois canais de entrada para um canal de saída, ou seja se o canal *recl* estiver pronto e o canal *recr* não estiver, *recl* será selecionado. Se o canal *recl* não estiver pronto e *recr* estiver, *recr* é selecionado. Caso os dois canais estejam prontos, um destes canais é selecionado de forma não-determinística; se nenhum destes canais estiverem prontos, aguarda-se até que algum desses torne-se habilitado. A figura 5.10.b apresenta o diagrama de fluxo de dados desse processo. Apresentamos a rede de Petri que modela esse o fluxo de controle na figura 5.10.c. Observe-se que esse modelo é construído por sub-redes: redes que modelam a recepção por canal de comunicação, transmissão, e a exclusão mútua. A representação de um canal pronto, na rede de Petri, é

feita através da marca nos lugares l_1 e l_2. Observe-se que o disparo de uma destas transições (t_0 ou t_2) impossibilita o disparo da outra, dado que com o disparo consome-se a marca do lugar p_1. A semântica das redes de Petri, garante a escolha não-determinística do disparo das transições t_0 ou t_2, caso haja marcas, em um mesmo instante, nos lugares l_1 e l_2. Com o disparo de uma da transições (t_0 ou t_2), deposita-se uma marca no lugar p_2 ou p_3, respectivamente. Uma marca no lugar p_2 habilita a transição t_1, bem como o lugar p_3 marcado habilita o disparo da transição t_3. O disparo de t_1 ou t_3 deposita marcas nos lugares a_1 ou a_2 que representam o reconhecimento pelos processo de recepção da informação, e no lugar p_4. Uma marca em p_4 possibilita a escrita no canal *trans* (disparo de t_4). O disparo de t_5 ocorre quando o processo que recebeu a informação transimitida, representada pelo disparo de t_5 e sinalizada através da marca no lugar s, reconhece a recepção dessa informação através do depósito de uma marca no lugar $a3$. Ao disparar-se t_5, coloca-se uma marca no lugar p_6, que representa a condição para a continuidade do processo.

Para mostrar a modelagem do combinador ALT, no qual os processos são guardados por expressão, apresentamos o exemplo da figura 5.11.a. Nesse exemplo, temos um programa que regula o fluxo de atividades entre os usuários e um sistema servidor, composto por diversos processadores. Nesse exemplo, os usuários solicitam ao sistema regulador, através do canal *dus*, a execução, pelos servidores, de tarefas. O sistema regulador, desde que haja servidores livres (processadores) ($i \geq 1$), decrementa uma variável que indica o número de processadores livres, e envia a solicitação para o sistema servidor através do canal *pse*. Quando os servidores finalizam suas tarefas, sinalizam para o regulador a finalização da tarefa (canal *dse*) e então o sistema regulador envia para os usuário, através do canal *pus*, esta informação, bem como incrementa o número de processadores livres. No programa apresentado, os *guardas* são os canais de entrada *dus*, *dse*, bem como a expressão $i \geq 1$.

Representamos esse programa pela rede de Petri apresentada na figura 5.11.b. Como no exemplo anterior, representamos este combinador através da rede elementar de escolha e da atribuição, no entanto observe que a transição t_2 é rotulada com $i \geq 1$, ou seja, só é possível dipará-la quando houver uma marca nos lugares p_3, l_2 e essa expressão for avaliada como verdadeira.

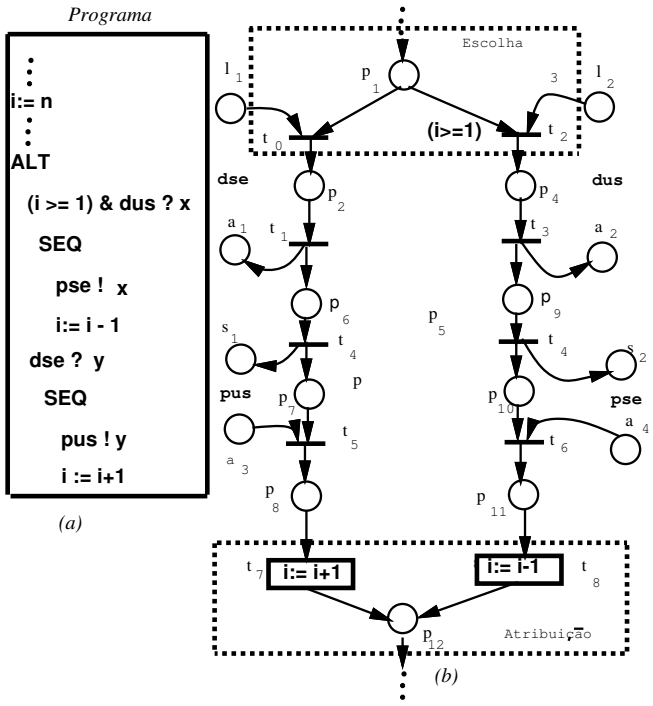

Figura 5.11: *Alternation* - Regulador de Fluxo

5.1.2.6 Exemplo

Nesta seção apresentamos um exemplo do uso do método de tradução apresentado. Consideremos a função de convolução descrita a seguir:

$$y_i = \sum_{j=1}^{n} x_{i-j} \times \omega_j \times \alpha_i, 1 \leq i \leq 2n-1$$

Onde ω_j é $\omega_{j+1} = b \times x_1 \times \omega_j$ e $\alpha_i = c_i + d_i$

$$c_i = \begin{cases} x_i & \text{se } x_i \geq 0 \\ \frac{x_i}{2} & \text{se } x_i < 0 \end{cases}$$

$$d_i = \begin{cases} x_{i+1} & \text{se } x_{i+1} \geq 0 \\ \frac{x_{i+1}}{2} & \text{se } x_{i+1} < 0 \end{cases}$$

O programa em *OCCAM* que descreve esta função pode ser implementada por um conjunto de processos concorrentes conforme mostrado abaixo [83, 81, 82].

```
CHAN OF INT P1.P4, P2.P4, P4.P3 :
CHAN OF [5] INT P3.P4 :
PAR
  INT c:
  SEQ i=0 FOR 2                                    (P1)
      IF(x[i]>=0 c:=x[i], x[i]<0 c:=x[i]/2)
      p1.p4 ! c

  INT d:
  SEQ i=0 FOR 2                                    (P2)
    IF(x[i]>=0 d:=x[i+1], x[i]<0 d:=x[i+1]/2)
    p1.p4 ! c

  INT c:
  SEQ i=0 FOR 2                                    (P3)
    p4.p3 ? w
    PAR j=0 FOR 4
      e[j]:=x[5*(i/(j+((j+1)/(i+1))))+(j-i)]*w
```

```
  p3.p4 ! e

INT c,d:
[5] INT e :
SEQ i=0 FOR 2                                    (P4)
  p4.p3 ! w
  PAR
    p1.p4 ? c
    p2.p4 ? d
    p3.p4 ? e
  PAR
    w:=k*e[j]
    PAR j = 0 FOR 4
      y[j] := y[j] + e[j] * (c + d)
```

A representação do programa em redes de Petri é discutida a seguir. O programa consiste no operador PAR que engloba quatro processos comunicantes. Desta forma, temos quatro blocos interconectados por arcos, que representam redes mais detalhadas do processo(ver figura 5.12). A tradução de $OCCAM$ para a rede de Petri equivalente é feita passo-a-passo usando a tradução dos combinadores apresentada na seção 5.1.

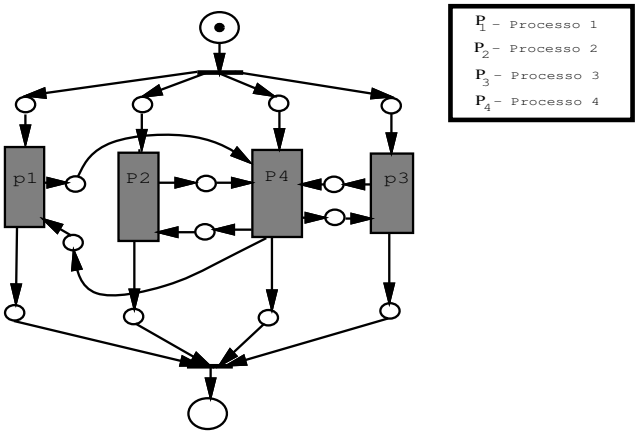

Figura 5.12: Rede dos Processos Comunicantes

O processo P_1 é formado por um operador de seqüenciamento, uma esco-

lha e transmissão de mensagem pelo canal $p_1.p_4$. Na figura 5.13 apresentamos o modelo que representa este processo.

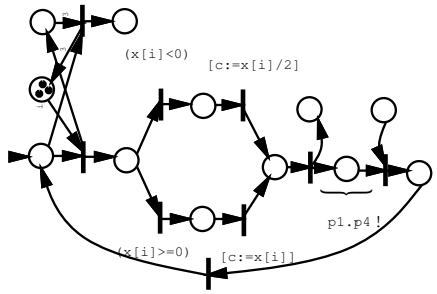

Figura 5.13: Processo P_1

Devido à similaridade estrutural entre os processos P_1 e P_2, as redes que representam estes processos têm estruturas semelhantes, diferenciando-se apenas pela funcionalidade das operações.

O processo P_3 é formado a partir dos construtores SEQ e PAR, processos primitivos de escrita e leitura e operações aritméticas. A rede de Petri que representa o fluxo de controle é apresentada na figura 5.14.

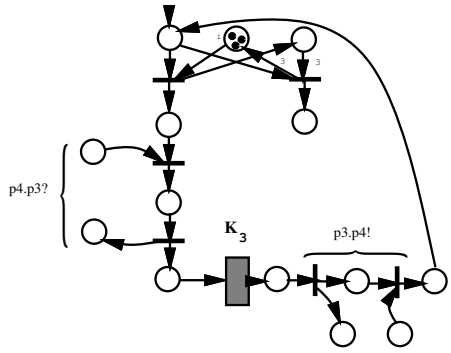

Figura 5.14: Processo P_3

A rede que representa a descrição detalhada da página K_3 é apresentada na figura 5.15

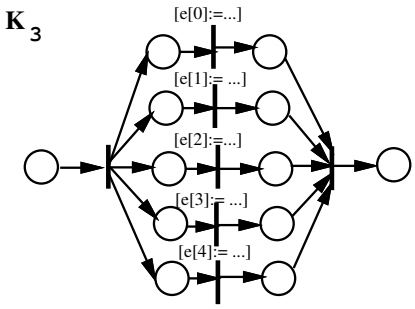

Figura 5.15: Página K_3

O processo P_4 é composto por processos primitivos de recepção, transmissão e operações aritméticas, assim como os construtores SEQ e PAR.

Na figura 5.16 temos a rede que representa este processo. A figura 5.17 apresenta o refinamento do bloco K_5.

5.2 Diagnóstico de Falhas em Sistemas de Controle Industrial

Nesta seção apresentamos um método de diagnóstico de falhas em sistemas de controle seqüencial utilizando-se redes de Petri [22, 27, 28, 80]. O sistema de diagnóstico faz uso de modelos especificados segundo um processo de modelagem descrito também nesta seção.

O processo de modelagem dos sistemas de controle seqüencial parte da descrição comportamental de cada componente, através de regras operacionais. A partir das regras operacionais, são construídas as redes de Petri elementares que modelam cada componente do sistema. A rede de Petri global é obtida pela técnica de fusão de "lugares" comuns das redes elementares. Esse processo garante a preservação das propriedades das redes elementares no modelo global.

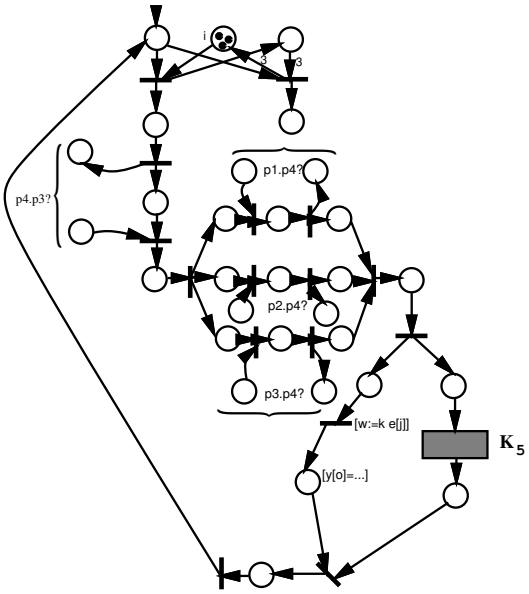

Figura 5.16: Processo P_4

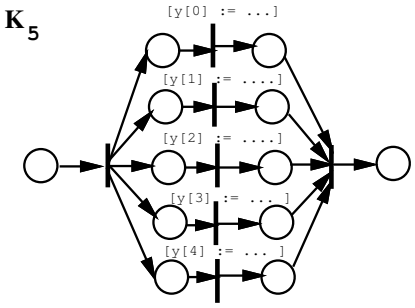

Figura 5.17: Página K_5

5.2.1 As Especificações Funcionais

Nesta seção, apresentamos uma visão geral sobre Sistemas de Controle Seqüencial (SCS). Apresentamos, também, uma técnica para especificar ou transcrever diagramas eletro/eletrônicos de controle seqüencial, assíncronos e paralelos, através de regras operacionais e a tradução destas para as redes de Petri.

Na figura 5.18 ilustramos a representação conceitual dos Sistemas de Controle Seqüencial.

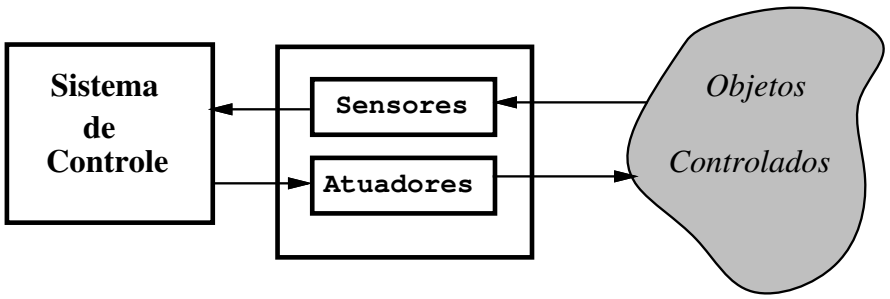

Figura 5.18: Sistema de Controle Seqüencial

O Controlador recebe sinais dos sensores, que informam os estados dos objetos controlados e envia sinais para os atuadores, os quais modificam os estados desses objetos. Este ciclo se repete durante toda a operação do sistema.

5.2.2 As Funções de Estado e Excitação

Em muitas aplicações industriais, o comando desses sistemas é formado por dispositivos, como por exemplo: relés, válvulas solenóides, termostatos, embreagens, portas lógicas, flip-flops etc. Em geral, esses dispositivos são formados por duas partes as quais denominamos de estruturas básicas:

- objetos acionados e
- acionadores.

Representamos o conjunto dos objetos acionados de um dispositivo por S_j e os elementos deste conjunto por sj. Cada dispositivo terá seu comportamento descrito por duas funções de estado, $g_j(x)$ e $h_j(x)$. A função $g_j(x)$ indica o estado dos objetos acionados, quando o dispositivo está desativado. E a função $h_j(x)$ indica o estado dos objetos acionados, quando o dispositivo está ativado. Portanto, chamamos estado g os estados dos objetos $s_j \in S_j$ descritos por $g_j(x)$ e estado h os estados representados por $h_j(x)$.

Os acionadores (A_j) dos dispositivos são estruturas capazes de modificar o estado de todos objetos acionados. Dois tipos de eventos podem atuar sobre os acionadores:

- Evento L_j : ativação de (A_j)
- Evento D_j : desativação de (A_j),

Esses eventos são mutuamente exclusivos. Para todo acionador (A_j) será associada uma função $f_j(x)$, que denominamos função de excitação, de forma que:

- Se a condição $f_j(x)$ for verdadeira, poderá ocorrer L_j;
- Se a condição $\overline{fj(x)}$ for verdadeira, poderá ocorrer D_j.

Observamos na figura 5.18 que os sensores e atuadores são entradas e saídas do controlador, respectivamente. Portanto, sensores são acionadores de entrada que percebem os eventos externos, e os atuadores são acionadores de saída.

Designamos variável de entrada aquelas variáveis que estão relacionadas aos sensores e variáveis de saída, aquelas associadas aos atuadores. Identificamos essas variáveis nas funções de estado e de excitação do seguinte modo:

- *Variáveis de entrada* são aquelas que figuram apenas nas funções de excitação e nunca aparecem nas funções de estado.
- *Variáveis de saída* são aquelas que figuram apenas nas funções de estado e não aparecem nas funções de excitação.

5.2.3 As Regras Operacionais

O controle do sistema é constituído pelo arranjo de várias estruturas elementares E_j interligadas, de modo que as funções de controle sejam executadas corretamente. As estruturas elementares são formadas pela combinação de estruturas básicas, ou seja, acionadores e objetos acionados. Descrevemos a relação entre as componentes de um estrutura elementar através da expressão abaixo:

$$Ej : identA_j < S_j >, \tag{5.1}$$

onde "ident" e um nome atribuído a Ej.

As estruturas elementares devem obedecer a regras operacionais (R_j) que formulam o comportamento destas estruturas. Essas regras garantem que, a qualquer instante, todos objetos s_j associados a qualquer acionador A_j só podem estar ou no estado-g ou no estado-h.

O comportamento das estruturas elementares, ou seja, os dispositivos que compõem o sistema de controle, é expresso através das regras operacionais:

(R_j):

$$L_j t_k : \quad se \quad < f_j(x) > . < g_j(x) > \quad então \quad < h_j(x) > \tag{5.2}$$

$$D_j t_k : \quad se \quad \overline{< f_j(x) >} . < h_j(x) > \quad então \quad < g_j(x) > \tag{5.3}$$

Onde L_j representa o evento de ativar o disposistivo j; $f_j(x)$ é a função de excitação que possibilita esta ativação; $g_j(x)$ é o estado desativado; $h_j(x)$ é o estado ativado; D_j representa o evento de desativar o dispositivo e t_i representa o tempo associado a execução do evento, ou seja, define o momento em que o evento ocorre. O tempo será especificado sempre que for relevante em sua descrição funcional.

5.2.4 Obtenção das Redes de Petri

Nesta seção apresentamos a técnica de modelagem por Redes de Petri, para a obtenção do modelo global do sistema. Esta técnica é do tipo *bottom*

up, ou seja, a partir da composição de modelos elementares (sub-redes) que representam componentes do sistema, obtemos o modelo global.

O procedimento de modelagem por Redes de Petri proposto, é baseado na fusão de lugares. Nesse procedimento seguimos as seguintes etapas :

- **A1** - construção e validação dos modelos elementares respeitando-se as regras operacionais.
- **A2** - construção do modelo global pela composição dos modelos elementares assegurando as regras operacionais das redes elementares.

obtemos os modelos elementares a partir das regras operacionais, segundo os passos descritos a seguir:

- **A1.2** - determinação das transições: eventos do tipo L ou D nas regras correspondem às transições. Associa-se a essas transições o tempo especificado na regra operacional.
- **A1.2** - determinação dos lugares de entrada: São as variáveis de estado das regras operacionais que representam as entradas dos disposistivos.
- **A1.3** - determinação dos lugares e dos lugares de saída: cada variável de estado origina um lugar com exceção das variáveis de entrada. Os lugares que correspondem às variáveis de saída são denominados lugares de saída.

5.2.5 Um Exemplo

Para ilustrarmos a metodologia proposta, apresentamos aqui um sistema que faz parte de uma linha de produção de bebidas, como mostrado na figura 5.19.

A esteira desloca recipientes vazios até o feixe de luz ser interrompido. A incidência de luz habilita o sensor óptico. Quando o sensor óptico é desativado, ele e o sensor laser, que é ajustado em função do nível do líquido no depósito, ao verificar o recipiente vazio, causam a abertura da válvula de enchimento e a desabilitação do movimento da esteira. Quando o recipiente estiver cheio, o sensor laser é ativado, provocando o fechamento da válvula de enchimento e inicia o movimento da esteira até o recipiente deixar o feixe de luz. Quando o recipiente deixa o feixe de luz, o sensor óptico é ativado,

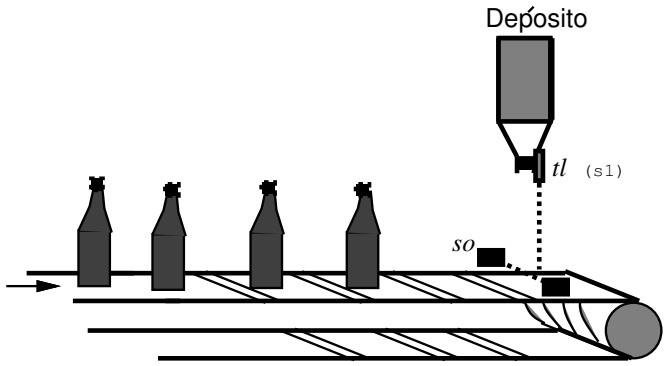

Figura 5.19: Linha de Produção de Bebidas

e o sensor laser desativado. O sensor óptico ativado provoca o movimento da esteira para o preenchimento do recipiente seguinte. Assume-se que o motor é ligado à esteira através de uma embreagem. Quando a embreagem é acoplada, a esteira entra em movimento. Caso contrário, permanece parada. Na figura 5.20, mostramos o diagrama elétrico do comando desse sistema e sua relação com sensores e atuadores.

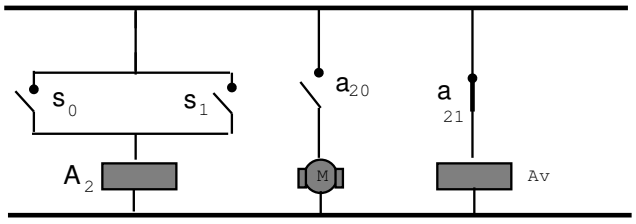

Figura 5.20: Diagrama Elétrico

5.2.6 Método para Obtenção das Regras Operacionais

As regras operacionais dos componentes do sistema são obtidas segundo os passos apresentados a seguir:

1. Identificação das estruturas elementares;
2. Definição das variáveis de estado das estruturas;
3. Construção das funções lógicas;
4. Construção das regras operacionais.

Passo 1: definição das estruturas elementares.

Observando-se a figura 5.20 , verificamos que o sistema é composto por cinco dispositivos, os quais definem cinco estruturas elementares :

-*E0*:sensor óptico A0(s00); A0 é o acionador e s00 o objeto acionado.

-*E1*:sensor laser A1(s01)

-*E2*:contator A2(s02,s12)

-*E3*:atuador do motor A3(s03)

-*E4*:atuador da válvula A4(s04)

Por exemplo, a estrutura E2 é formada por dois componentes básicos: o acionador A2 e dois objetos acionados que são os contatos s02 e s12.

Passo 2: atribuição das variáveis de estado.

Neste passo atribuímos variáveis de estado para cada estrutura do sistema:

E0: sensor óptico $A0(s00)$: atribuímos $\overline{X0}$ à não incidência de luz em $A0$ e a $X0$ à incidência de luz. Atribuímos $x00$ ao contato $s00$ no estado aberto e a $\overline{x00}$ ao contato $s00$ no estado fechado.

E1: sensor laser $A1(s01)$: atribuímos $\overline{X1}$ à distância entre o sensor e ao líquido que corresponde ao recipiente não cheio e a $X1$ à distância que corresponde ao recipiente cheio. Atribuímos $\overline{x01}$ ao contato $s01$ no estado aberto e a $x01$ ao contato $s01$ no estado fechado.

E2: relé $A2(s02, s12)$: atribuímos $\overline{x02}$ ao contato $s02$ no estado aberto e a $x02$ ao contato $s02$ no estado fechado. Atribuímos $\overline{x12}$ ao contato $s12$ no estado aberto e a $x12$ ao contato $s12$ no estado fechado.

E3: atuador do motor $A3(s03)$: atribuímos $\overline{x03}$ ao eixo do motor $s03$ no estado parado e a $x03$ ao eixo do motor $s03$ no estado girando.

E4: atuador da válvula $A4(s04)$: atribuímos $\overline{x04}$ à válvula $s04$ no estado aberto e a $x04$ à válvula $s04$ no estado fechado.

Passo 3:determinação das funções lógicas.

Determinamos as funções lógicas para cada estrutura observando a topologia do circuito (figura 5.20) e o estado das variáveis. Neste exemplo, temos:

E0: Sensor Óptico

- se há incidência de luz ($X0$) sobre sensor óptico então $A0$ é ativado, logo $f(x) = X0$.
- se não há incidência de luz($\overline{X0}$) então $A0$ é desativado, logo $f(x) = \overline{X0}$.
- estado-g: $g(x) = \overline{x00}$; corresponde ao contato $s00$ aberto.
- estado-h: $h(x) = x00$; corresponde ao contato $s00$ fechado.

E1: Sensor Laser

- se $X1$ então $A1$ é ativado, logo $f(x) = X1$
- se $\overline{X1}$ então $A1$ é desativado, logo $f(x) = \overline{X1}$
- estado-g: $g(x) = \overline{x01}$
- estado-h: $h(x) = x01$

E2: Relé

- se $x00 + x01$ então $A2$ é ativado, logo $f(x) = x00 + x01$
- se $\overline{x00}.\overline{x01}$ então $A2$ é desativado, logo $\overline{f(x)} = \overline{x00}.\overline{x01}$
- estado-g: $g(x) = \overline{x02}.x12$
- estado-h: $h(x) = x02.\overline{x12}$

E3: Atuador do Motor

- se $x02$ então $A3$ é ativado, logo $f(x) = x02$
- se $\overline{x02}$ então $A3$ é desativado, logo $\overline{f(x)} = \overline{x02}$
- estado-g: $g(x) = \overline{x03}$
- estado-h: $h(x) = x03$

E4: Atuador da Válvula

- se x12 então $A4$ é ativado, logo $f(x) = x12$
- se $\overline{x12}$ então $A4$ é desativado, logo $\overline{f(x)} = \overline{x12}$
- estado-g: $g(x) = x04$
- estado-h: $h(x) = \overline{x04}$

Passo 4: obtenção das regras operacionais.

Obtemos as regras operacionais aplicando as funções lógicas na forma expressa pelas sentenças 5.2 e 5.3. Para o exemplo, temos:

(R0): Sensor Óptico

L0: se$< X0 > \overline{< x00 >}$então $< x00 >$
D0: se $\overline{< X0 >} < x00 >$ então $\overline{< x00 >}$

(R1): Sensor Lazer

L1: se $< X1 > \overline{< x01 >}$ então $< x01 >$
D1: se $\overline{< X1 >} < x01 >$ então $\overline{< x01 >}$

(R2): Relé

L2': se $< x00 > \overline{< x02}.x12 >$ então $< x02.\overline{x12}$
L2": se $< x01 > \overline{< x02}.x12 >$ então $< x02.\overline{x12 >}$
D2: se $< \overline{x00}.\ \overline{x01} > < x02.\overline{x12 >}$ então $< \overline{x02}.x12 >$

(R3): Atuador do Motor

L3: se $< x02 > \overline{< x03 >}$ então $< x03 >$
D3: se $< \overline{x02} >< x03 >$ então $\overline{< x03 >}$

(R4): Atuador da Válvula

L4: se $< x12 >< x04 >$ então $\overline{< x04 >}$
D4: se $< \overline{x12} > \overline{< x04 >}$ então $< x04 >$

Observamos que as variáveis de estado $X0$ e $X1$ figuram apenas nas expressões $L0$ e $L1$ das regras $R0$ e $R1$, respectivamente. Portanto, são variáveis de entrada. Por outro lado, as variáveis $\overline{x03}$ e $\overline{x04}$ figuram apenas

nas funções de estado (ver expressões $L3, D3, L4$ e $D4$ das regras $R3$ e $R4$), logo são variáveis de saída. Dessa forma estabelecemos os vetores Vi e Vo, como respectivamente o vetor de entrada e saída, conforme descrevemos a seguir.

-vetor de entrada $Vi = (X0, X1)$

-vetor de saída $Vo = (\overline{x03}, \overline{x04})$

Podemos apresentar a relação entre as entradas e saídas do sistema da figura 5.20 através da seguinte tabela:

X0	X1	$\overline{x03}$	$\overline{x04}$	motor	válvula
0	0	1	1	parado	aberta
0	1	0	0	movimento	fechada
1	0	0	0	movimento	fechada

Tabela 5.1: Relação entre Vi e Vo

Tendo obtido as regras operacionais, a próxima etapa é a obtenção dos modelos elementares a partir destas regras. Apresentamos este procedimento a seguir:

5.2.7 Modelagem e Validação

Conforme os passos **A1** e **A2**, que apresentamos no ínicio da seção 5.2.4, temos:

Passo A1 - Construção de Modelos Elementares

Considerando a regra operacional *(R0)* e os passos **A1.1**, **A1.2** e **A1.3**, temos:

A1.1 - determinação das transições: os eventos originam as transições.

A1.2 - determinação dos lugares de Entrada (eventos externos): os eventos externos são representados por lugares de entrada, por exemplo: a variável *X0* torna-se o lugar *X0*; incidência de luz sobre o sensor e a variável $\overline{X0}$ torna-se o lugar $\overline{X0}$; não incidência de luz sobre o sensor.

A1.3 - determinação dos demais lugares: as demais variáveis são representadas por lugares, por exemplo: a variável $\overline{x00}$ torna-se o lugar $\overline{x00}$.

Baseado nessas considerações, obtemos o modelo *N0* conforme mostra figura 5.21.a, onde a ficha no *p1* representa o contato associado ($\overline{s0}$) no estado aberto.

Empregando o mesmo procedimento, obtemos os demais modelos elementares. Mostramos estes modelos na figura 5.21.

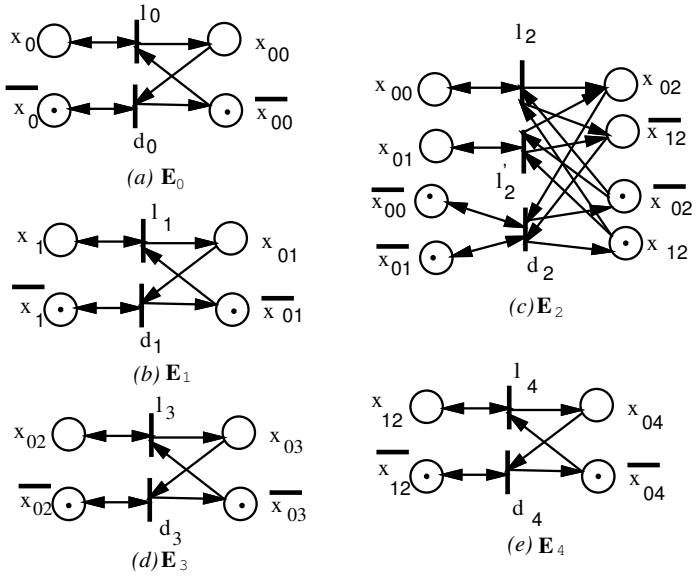

Figura 5.21: Modelos Elementares

Passo A2 - Construção do Modelo Global **N**

O modelo global do controle do sistema é formado pela composição dos modelos elementares, através da fusão de lugares de mesma denominação, conforme mostramos através da figura 5.22.

5.2.8 Elemento de Falha de Vetor Sintoma

- Elementos de Falha

Conforme mostrado na seção 5.2.2, para todo acionador A_j pode-se de-

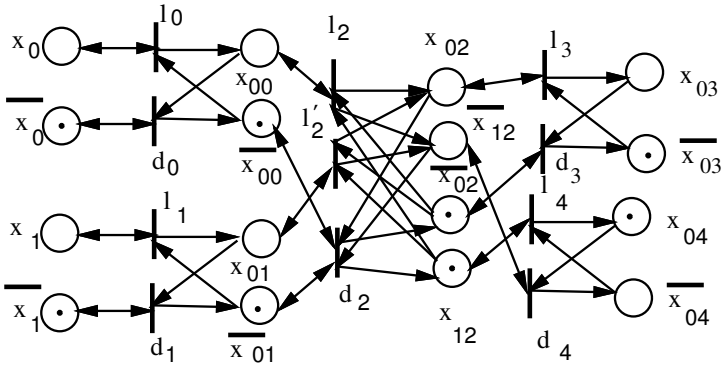

Figura 5.22: Modelo Global

terminar uma função $fj(x)$, denominada função de excitação, de forma que: se $fj(x) = 1$, poderá ocorrer o evento Lj e se $\overline{fj(x)} = 1$, poderá ocorrer o evento Dj,

Na verdade as funções $fj(x)$ estabelecem as condições em termos de variáveis de estado para a ocorrência dos eventos Lj ou Dj. Daqui por diante as variáveis de estado que figuram nas funções $f(x)$, com exceção das variávies de entrada (etiquetas) e das variáveis de saída, serão chamadas de variáveis de falhas. Portanto, os lugares correspondentes às variáveis de falha serão denominados lugares de falha.

Como demonstrado anteriormente, as redes modeladas a partir destas estruturas, que representam componentes de dois estados, como são os componentes estudados no escopo deste trabalho, têm para todo lugar de falha pf um lugar dual $\overline{pf}$ correspondente. Uma transição no modelo global que tem simultaneamente um mesmo lugar de falha como entrada e saída, será chamada transição de falha tf.

Considerando-se o modelo da figura 5.22, temos o seguinte conjunto que relaciona transições e lugares de falha:$\{(d2, \overline{x00}, \overline{x01}); (l2, x00); (l2', x01); (l3, x02); (d3, \overline{x02}); (l4, x12); (d4, \overline{x12})\}$. Apresentamos também um outro conjunto que relaciona os lugares com seus respectivos lugares duais:

$\{x00, \overline{x00}\}$; $\{x01, \overline{x01}\}$; $\{x02, \overline{x02}\}$; $\{x12, \overline{x02}\}$.

Como mencionado, o controle do sistema possui duas estruturas básicas:

objetos acionados e acionadores. Dessa forma durante a operação do sistema poderá ocorrer, a qualquer instante, falhas em objetos acionados e/ou acionadores. Portanto, o diagnosticador deve ser capaz de apresentar ao usuário dois tipos listas, ou seja: lista-A: relaciona acionadores defeituosos e a lista-S: relaciona objetos acionados defeituosos. Logo, a lista-A contém transições tf enquanto que a lista-S, os lugares pf.

- **O Vetor Sintoma**

Considerando-se a tabela 5.1, para o vetor de entrada $Vi = (0, 1)$, temos o vetor de saída $Vo = (1, 0)$. Entretanto o sistema pode apresentar falhas, como, por exemplo, o motor não girar. Portanto, o sintoma, em termos da saída, será $Vs = (0, 0)$ (vetor sintoma). Deste modo, denominamos vetor sintoma (Vs) todo vetor de saída obtido em um estado de falha.

5.2.8.1 Princípio do Diagnóstico

O diagnóstico de falhas é feito através da desabilitação das transições habilitadas para um sintoma (Vs) apresentado, ou seja, removendo-se as marcas dos lugares de falha que habilitam as transições, para os lugares duais correspondentes, até a obtenção de uma marcação estável. O diagnóstico é, portanto, a negação dos eventos e das condições observadas. A implementação de sistema de diagnóstico, baseia-se em um simulador de redesde Petri e uma função que faz o diagnóstico.

- função Simulador: simula a evolução dos estados do sistema de controle modelado.
- função Diagnosticador: o módulo de diagnóstico utiliza a estrutura matricial que representa rede do sistema e a marcação fornecida pelo simulador. O simulador utiliza como informações adicionais, os estados das entradas do sistema, ou seja, os eventos externos associados. Com estas informações o simulador parte da marcação inicial até a obtenção de uma marcação estável. A partir desta marcação, o sintoma deve ser fornecido. Baseado neste sintoma, o módulo de diagnóstico investiga quais eventos não deveriam ter ocorrido e que condições não deveriam ser verdadeiras e concorreram para a falha.

Baseado nesse princípio, apresentamos a seguir um procedimento que descreve as ações do sistema de diagnóstico de falhas.

0) Ler estrutura e estado inicial da máquina.

1) Evoluir o modelo a partir de seu estado inicial até o estado que seria alcançado, caso não tivesse ocorrido a falha.

2) Modificar a marcação de saída obtida no passo **1**) conforme vetor sintoma Vs,

3) Colocar todas as transições habilitadas na lista-A,

4) Desabilitar cada transição, armazenada na lista-A, removendo-se as fichas de seus lugares de falha para os respectivos lugares duais,

5) Colocar na lista-S os lugares de falhas decorrentes do passo **4**),

6) Enquanto houver transições habilitadas, repetir os passos **3**), **4**) e **5**),

7) Converter as informações da lista-A e lista-S em informações de alto nível para o usuário.

5.2.8.2 Aplicação do SDF

Para exemplificar a construção das lista-A e lista-S pelo diagnosticador, considera-se o exemplo da figura 3, onde, a princípio, tem-se a esteira parada e a valvula aberta. Quando o recipiente enche, a válvula é fechada e a esteira entra em movimento. No entanto, uma falha ocorre: a esteira não se movimenta. Neste caso, o vetor sintoma é $Vs = (0, 0)$.

Mostra-se a seguir como o SDF efetua o diagnóstico para este sintoma.

estado inicial *S0*:

$M(\overline{x0}) = 1$, $M(\overline{x1}) = 1$, $M(\overline{x00}) = 1$, $M(\overline{x01}) = 1$, $M(\overline{x02}) = 1$, $M(x12) = 1$, $M(\overline{x03}) = 1$, $M(x04) = 1$

Passo 1:

Evoluindo-se o modelo apartir do estado inicial (ver figura 5.23) até a obtenção de uma marcação estável (ver figur 5.24), temos uma ficha nos seguintes lugares:

estado *Se*:

$M(\overline{x0}) = 1$, $M(x1) = 1$, $M(\overline{x00}) = 1$, $M(x01) = 1$, $M(x02) = 1$, $M(\overline{x12}) = 1$, $M(x03) = 1$, $M(\overline{x04}) = 1$

Passo 2:

Modificando-se a marcação dos lugares de saída conforme vetor sintoma

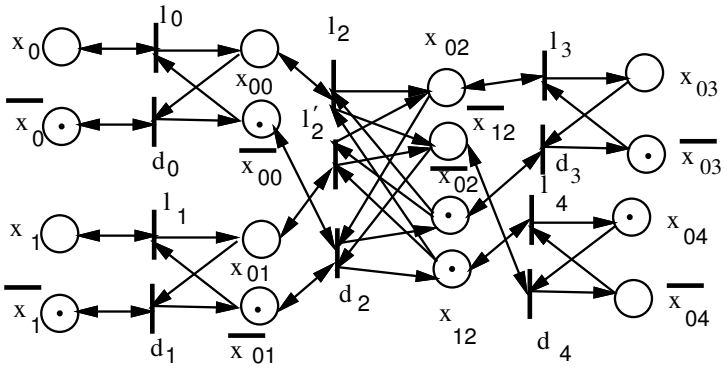

Figura 5.23: Estado $S0$

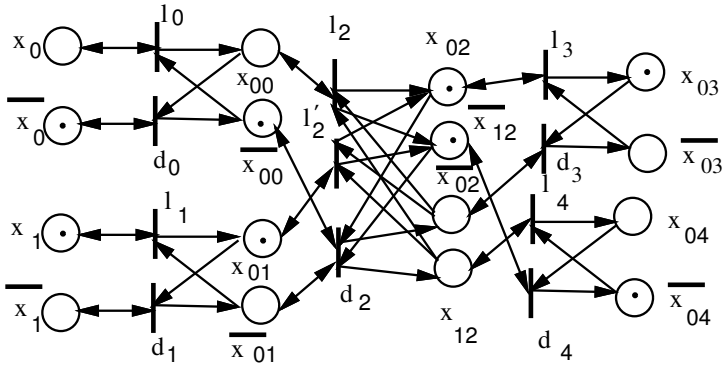

Figura 5.24: Estado Se

$Vs = (00)$, temos uma ficha nos seguintes lugares:

$M(\overline{x0}) = 1$, $M(x1) = 1$, $M(\overline{x00}) = 1$, $M(x01) = 1$, $M(x02) = 1$, $M(\overline{x12}) = 1$, $M(\overline{x03}) = 1$, $M(\overline{x04}) = 1$

Observe-se que se moveu uma marca do lugar $M(x03) = 1$ para o lugar $M(\overline{x03}) = 1$ (ver figur

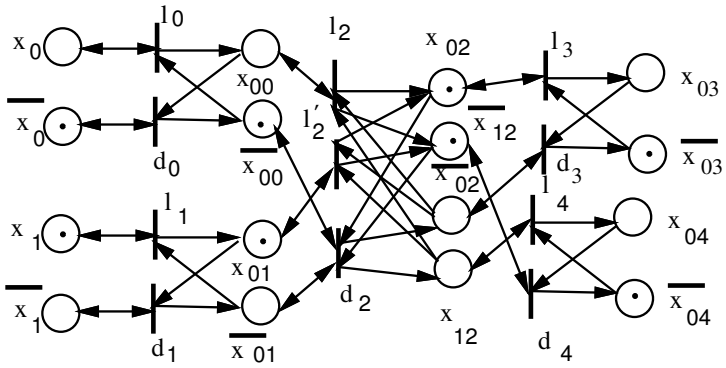

Figura 5.25: Sintoma

Passo 3:

Dentro da marcação obtida no passo 5, temos apenas a transição *l3* habilitada. Portanto, deve ser listada na lista-A, ou seja:

lista-A: *l3*

Passo 4:

Desabilita-se a transição do passo 6 movendo-se a marca do lugar de falha $x02$ para seu lugar dual $x12$. Isso conduz a seguinte marcação (ver figura 5.26):

$M(\overline{x0}) = 1$, $M(x1) = 1$, $M(\overline{x00}) = 1$, $M(x01) = 1$, $M(x12) = 1$, $M(\overline{x02}) = 1$, $M(\overline{x03}) = 1$, $M(\overline{x04}) = 1$

Passo 5:

Move-se o lugar $x02$ de falhas para a lista-S, ou seja:

lista-S: $x02$

Passo 6:

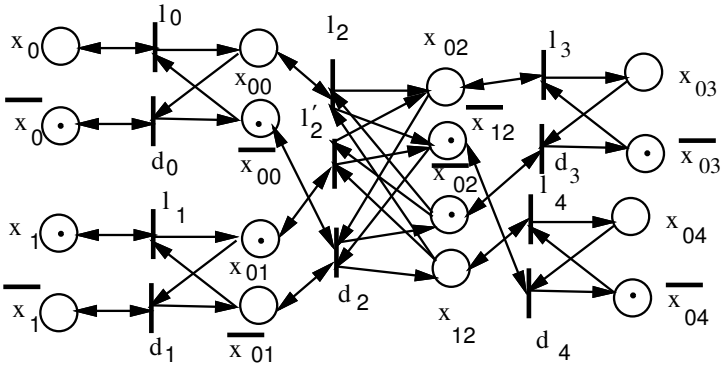

Figura 5.26: Desabilitação de *l3*

Verifica-se que, dentro da marcação do passo 5, não existem mais transições habilitadas. Portanto, para este sintoma o diagnóstico está determinado.

Passo 7:

Nesse exemplo o conteúdo das listas A e S pode ser relacionado com o diagrama elétrico da figura 5.20 através das seguintes mensagens:

lista-A: A3 [desativado]

lista-S: s02 [aberto]

Estas mensagens informam ao usuário que o Acionador $A3$ está desativado ou que o contato $s02$ não está fechando. De fato o diagnóstico está correto, visto que se $A3$ não é ativado (acionador do motor está danificado) ou $s02$ não fecha (que é o contato que aciona $A3$) a saída Vs corresponde ao sintoma apresentado.

5.3 Algumas Considerações

Neste capítulo apresentamos a modelagem de sistemas de software e hardware através das redes de Petri. Incialmente modelamos a linguagem de programa ção para sistemas concorrentes *OCCAM*. Modelamos os processo primitivos e seus combinadores de processos. Para ilustrar o método, apre-

sentamos uma série de pequenos exemplos em que descrevemos os processos primitivos, os combinadores da linguagem e os respectivos modelos em redes de Petri.

Apresentamos também uma metodologia para a especificação de sistemas de controle industrial, adotando o processo *botton-up* para a modelagem desses sistemas. Nessa metodologia, capturamos o comportamneto dos dispositivos que compoẽm o sistemas, através do que denominamos regras operacionais. Tendo capturado o comportamento dos dispositivos, o passo seguinte foi transcrever as regras operacionais para redes de Petri (modelos elementares). O modelo do sistema é obtido através da composição dos modelos elementares. O processo de combinação dessas redes garante a manutenção das propriedades dos modelos elementares no modelo global do sistema. Apresentamos ainda, um método de diagnóstico de falhas dos sistemas de controle industrial utilizando-se os modelos obtidos segundo o método de modelagem apresentado.

Podemos observar, portanto, a potencialidade das redes de Petri como técnica para a descrição e análise de propriedades de sistemas que tenham atividades concorrentes, observando também o seu poder de abstração.

Capítulo 6

Considerações Finais

Dedicamos este capítulo para alguma considerações finais e para apresentarmos algumas das aspectos que têm sido tema de pesquisas no contexto das redes de Petri, atualmente.

Inicialmente, apresentamos sumariamente os assuntos tratados em cada capítulo:

Na introdução, apresentamos um histórico das redes de Petri, procurando situar o desenvolvimento e áreas de aplicação, sendo, também, descrito os conceitos básicos, tais como: estrutura, marcação, grafo de marcações acessíveis etc. No capítulo 2, apresentamos a modelagem de problemas clássicos, procurando ressaltar, ao leitor, o poder de modelagem das redes de Petri. No capítulo 3, tratamos das propriedades estruturais, comportamentais, assim como dos métodos de análise para a verificação destas propriedades. No capítulo 4, apresentamos algumas extensões às redes de Petri e finalmente, no capítulo 5, apresentamos o uso das redes de Petri, na modelagem de programas escritos em liguagem de programação concorrente e na modelagem e diagnóstico de falhas em sistemas de controle industrial.

Atualmente, diversos esforços feitos, por diferentes grupos situados na Europa, Estados Unidos da América e Ásia, tanto na pesquisa de aspectos teóricos, quanto no estudo aplicado das redes de Petri [85]. Nos estudos que tratam sobre os aspectos teóricos das redes de Petri, podemos destacar pesquisas que estudam equivalência e bissimulação, álgebras e métodos de prova e o estudo de classes estruturadas de redes. Podemos, também, destacar algumas linhas de pesquisa que tratam sobre os aspectos aplicados das redes de Petri, entre estas as que fazem estudos correlatos às linguagens de

programação, estudos sobre redes de alto nível, especificação e estudos de casos.

O grupo de pesquisa da UFPE/UPE, tem desenvolvido trabalhos diversos que fazem uso de redes de Petri. Dentre alguns desses trabalhos, podemos destacar a aplicação das redes de Petri no processo de paralelização de código e o uso destas como formato intermediário entre linguagens, no diagnóstico de falhas em sistema de controle industrial, na modelagem de sistemas digitais, na captura de requisitos temporais e modelagem de arquitetura de computadores no contexto do hardware/software *codesign* e na especificação de protocolos de comunicação.

Bibliografia

[1] G.W. Brams. *Réseaux de Petri: Théorie et Pratique, tome 1.* Masson Editions, 1983.

[2] G.W. Brams. *Réseaux de Petri: Théorie et Pratique, tome 2.* Masson Editions, 1983.

[3] M. Courvoisier, R. Valette. *Systèmes de Commande en Temps Réel.* SCM Editions, 1980.

[4] J. L. Peterson. *Petri Nets an Introduction.* Pretence-Hall, Inc, 1981.

[5] W. Reisig. *Petri Nets: An Introduction.* Springer-Verlag, 1982.

[6] S. M. Chen, J. Sheng Ke, J. F. Chang. *Knowledge Representation Using Fuzzy Petri Nets.* IEEE Trans. on Knowledge and Data Engineering, 1982.

[7] Y. Deng, S. K. Chang. *A G-Net Model for Knowledge Representation and Reasoning.* Y. Deng, S. K. Chang, 1990.

[8] T. Murata. *State Equation, Contrallability, and Maximal of Petri Nets.* IEEE Trans. on Automatic Control, 1977.

[9] T. Murata. *Modelling and Analysis of Concurrent Systems, Handbook of Software Engineering.* Van Norstrand Reinhold Company Inc., 1984.

[10] F. Dicesare, I. Koh. *Modular Transformation Methods for Generaliz Nets and Their Aplications to Automated Manufafacturing Systems.* IEEE Trans. on Systems,Manuf. and Cybernetics, 1991.

[11] T. Murata, D. Zhang. *A Predicate-Transition Net Model for Parallel of Logic Programs* IEEE Trans. on Software Engineering, 1989.

[12] R. Vallet, M. Courvoisier, J. M. Bigou, P. Esteban. *A Programmable Logic Controller Based on High Level Specification Tool.* LAAS, Toulouse, France, 1983.

[13] P. R. F. Cunha, T. S. E. Mainbaum. *A Synchronization Calculus for Message Oriented Programming.* , 2nd. International Conference on Distributed Computing Systems, Paris, France, 1982.

[14] P. R. F. Cunha, J. F. B. Castro. *Especificando Protocolos através de uma Metodologia Baseada em Redes de Petri.* V SBRC, São Paulo, Abril, 1987.

[15] F. Dicesare, S. J. Chang, G. Goldbogen. *Failure Propagation Trees for Diagnosis in Manufacturing Systems.* IEEE Trans. on Systems,Manuf. and Cybernetics, 1991.

[16] Tomohiro Murata, N. Komoda, K. Matsumoto, K. Haruna. *A Petri Net-Based Controller for Flexible and Maintainable Sequence Control and it's Aplications in Factory Automation.* IEEE Trans. on Industrial Electronics, 1986.

[17] R. A. Nelson, L. M. Haibt, P. B Sheiridan. *Casting Petri Nets into Programs.* IEEE Trans. on Software Engineering, 1983.

[18] R. Vallet, M. Courvoisier, C. Desclaux. *Putting Petri Nets for Work for Controlling Flexible Maufacturing Systems.* IEEE International Symposium on Circuits and Systems, 1985.

[19] T. Murata. *Petri Nets: Properties, Analysis and Aplications.* Proceding of The IEEE, 1989.

[20] D. S. Barbalho. *Conception et Mise en Oeuvre de la Fonction Coordination pour une Commande Distribuée d'Atelier.* tese de doutorado Université Paul Sabatier, Toulouse, France 1985.

[21] T. C. Barros. *Uma Técnica de Modelagem por Redes de Petri Voltada a Automação da Manufatura.* dissertação de mestrado UFPE, Recife, 1990.

[22] P.R.M. Maciel. *Um Diagnosticador de Falhas Para Sistemas de Controle Sequencial Baseado em um Jogador de Redes de Petri.* dissertação de mestrado UFPE, Recife, 1993.

[23] D.S. Barbalho, M.R. Santos, T. C. Barros. , Uma Técnica Estruturada de Modelagem por Redes de Petri: *Anais do* **VIII** *Congresso Brasileiro de Automática.* Função de Coordenação em Sistemas de Transporte, 1989.

[24] D.S. Barbalho, M.R. Santos, T. C. Barros. *Uma Metodologia Estruturada e Sistemática de Suporte à Automação da Manufatura.* Anais do **I** ERAI, 1989.

[25] P.R.M. Maciel, M.R. Santos, T. C. Barros. *Uma Ferramenta de de Diagnóstico de Falhas Voltada para Sistemas de Controle Industrial.* Anais do **V** Simpósio de Computadores Tolerantes a Falha, São Jose dos Campos, SP, 1993.

[26] P.R.M. Maciel, J. B. Castro. *Desenvolvimento de Sistemas Distribuídos de uma Forma Precisa.* 2ª Infornordeste, 1993.

[27] P.R.M. Maciel, J. B. Castro. *Uso de Redes de Petri e TCL para o Diagnóstico de Sistemas de Controle Industrial.* Anais do **XXI** SEMISH, 1994.

[28] P.R.M. Maciel, M.R. Santos, T. C. Barros. *Diagnosticador de Falhas para Sistemas de Controle Industrial: uma Abordagem por Redes de Petri.* Anais do **VI** Congresso Latino Americano de Controle e Automática em Conjunto com **X** Congresso Brasileiro de Automática, 1994.

[29] S.M. Shatz, K. Mai, S. Tu. *Design and Implementation of Petri Net Based Toolkit for Ada Tasking Analysis.* IEEE Transaction on Parallel and Distributed Systems, 1990.

[30] P. R. F. Cunha, C. G. Fernandes, J. F. B. Castro. *DIMAC - Um Protocolo de Acesso por Passagem de Tokem.* IV SBRC, Recife, Março, 1986.

[31] A.M. Tyrrell, D.J. Holding. *Design of Reliable Software in Distributed Systems Using the Conversation Scheme.* IEEE Transaction on Software Engineering, 1986.

[32] H.C. Yen. *Unified Aproach for Deciding the Existence of Certain Petri Net Path.* Information and Computation, 1992.

[33] A. Datta, S. Ghosh. *Synthesis of a Class of Deadlock-Free Petri Nets.* Journal of The Association for Computing Machinery, 1986.

[34] . C. T. King, W. H. Chou, L. M. Ni. *Pipelined Data – Parallel Algorithms: Concept and Modeling / Design.* IEEE Transaction on Parallel and Distributed Systems, 1990.

[35] G. Balbo, G. Chiola, S.c. Bruell, P. Cheng. *An Example of Modeling and Evaluation of Concurrent Program Using Colored Stochastic Petri Nets: Lamports's Fast Mutual Exclusion Algorithm.* IEEE Transaction on Parallel and Distributed Systems , 1992.

[36] J.E. Coolahan, N. Roussopoulos. *Timing Requirements for Time-Driven Systems Using Augmented Petri Nets.* IEEE Transaction on Software Engineering, 1983.

[37] I. Gorton. *Parallel Program Design Using High-Level Petri Nets.* Concurrency: Practice and Experience, 1993.

[38] K. Bilínsk, M. Adamski, J.M. Saul, E.L. Dagless. *Parallel Controller Synthesis from a Petri Net Specification.* Proceedings EURODAC-94 , 1994.

[39] P. Eles, K. Kuchcinski, Z. Peng, M. Minea. *Synthesis Concurrent Process.* Proceedings EURODAC-94, 1994.

[40] G. Dohmen. *Petri Nets as Intermediate Representation Between VHDL and Symbolic Transition Systems.* Proceedings EURODAC-94 , 1994.

[41] J. Esparza. *Reduction and Synthesis of Live and Bounded Free Choice Petri Nets.* Information and Computation , 1994.

[42] J. Mesguer, U. Montanari. *Petri Nets Are Monoids.* Information and Computation , 1988.

[43] S. M. Rezaul Islam, H. Ammar. *Performability Analysis of Distributed Real-time Systems.* IEEE Transaction on Computers, 1991.

[44] G. Lefranc, M. Villalobos. *Control Celdas Flexibles de Produccion Mediante Redes de Petri.* Anais do VI Congresso Latino Americano de Controle e Automática em Conjunto com X Congresso Brasileiro de Automática, 1994.

[45] W. Colombo, J. Pellicer, M. Martín, K. Kuchen. *Simulador de Sistemas Flexibles de Manufactura Usando Redes de Petri Temporizadas.* Anais do VI Congresso Latino Americano de Controle e Automática em Conjunto com X Congresso Brasileiro de Automática, 1994.

[46] C. A. Heuser. *Modelagem Conceitual de Sistemas.* EBAI, 1988.

[47] S. Bardinelli, A. Fuggetta, C. Ghezzi, L. Lavazza. *SPADE: An Environment for Software Process Analysis, Design, and Enactance.* Software Process Modelling and Technology, editado por: A. Finkelstein, J. Kramer, B. Nuseibeh, Research Studies Press Ltd., 1994.

[48] D. Walker. *Introduction to a Calculus of Communicating Systems.* Dept. Computer Science, University of Edinburgh, Lecture notes, 1987.

[49] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[50] L. M. Barroca, J. A. McDermid. *Formal Methods: Use and Relevance for the Development of Safety-Critical Systems.* The Computer Journal, 1992.

[51] V. M. Milutinović. *Computer Architecture.* Editado por: V. M. Milutinović, North Holland Press, 1988.

[52] M. Hack. *Analysis of Production Schemata by Petri Nets.* Master's Thesis, Departament of Eletrical Engineering, Massachusetts Institute of Technology Massachusetts, Febuary, 1972.

[53] F. Commoner. *Deadlock in Petri Nets.* Applied Data Research Inc. Massachusetts, 1972.

[54] Edited by W. Brauer. *Net Theory and Applications.* Lecture Notes in Computer Science, Springer-Verlag, Hamburg, 1979.

[55] J.M. Spivey. *Z Notation - A Reference Manual.* Prentice Hall International, 1989.

[56] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice Hall International, 1985.

[57] *LOTOS - A Formal description Technique Based on the Temporal Ordering of Observational Behaviour* . Information Processing System - Open Systems Interconnection, 1987.

[58] R. Goldblatt. *Axiomatizing the Logic of Computer Programming.* LNCS, Springer-Verlag, 1982.

[59] M. Jantzen. *Complexity of Place/ Transition Nets.* Lecture Notes in Computer Science, Springer-Verlag, edited by W. Brauer, W. Reisig, G. Rozenberg Bad Honnef, 1986.

[60] H. J. Genrich, K. Lautenbach, P. S. Thiagarajan. *Elements of General Net Theory.* Lecture Notes in Computer Science, Springer-Verlag, Edited by W. Brauer Hamburg, 1979.

[61] J. Esparza, M. Nielsen. *Decidability Issues for Petri Nets.* Gesellschaft für Informatik, 1994.

[62] C. Rackoff. *The Covering and Boundedness Problem for Vector Addition Systems.* Theorical Computer Science, vol. 6, p. 223-231. 1978.

[63] R. Karp, R. Miller. *Parallel Program Schemata.* Journal of Computer and System Science, vol. 3, N. 4, p. 167-195. 1969.

[64] R. Lipton. *The Reachability Problem Requires Exponential Space.* Research Report 62, Departament of Computer Science, Yale University, 1976.

[65] G. S. Sacerdote, R. L. Tenney. *The Decidibility of the Reachability Problem for Vector Addition System.* 9th Annual Sympossium on Theory of Computing, p. 61-76, Boulder, 1977.

[66] E. W. Mayr. *Persistence of Vector Replacement System is Decidable.* Acta Informatica 15, p. 309-318. 1981.

[67] S. R. Kosaraju. *Decidibility of Reachability in Vector Addition Systems.* 14th Annual ACM Symposium on Theory of Computing, p. 267-281. San Francisco, USA, 1982.

[68] J. L. Lambert. *Vector Addition Systems and Semi-linearity.* SIAM Journal of Computing, 1994.

[69] D. Frutos, C. Johne. *Decidability of Home States in Place Transition Systems.* 14th Internal Report, Dpto. Informatica y Automatica, Univ. Complutense de Madrid, 1986.

[70] E. Cardoza, R. J. Lipton, A. R. Meyer. *Exponential Space Complete Problems for Petri Nets and Commutative Semigroups.* 8th Symposium on Theory of Computing, p. 50-54. 1976.

[71] M. H. T. Hack. *Decidability Questions for Petri Nets.* PhD Thesis, MIT, 1976.

[72] A. Cheng, J. Esparza, J. Palsberg. *Complexity Results for 1-safe Nets.* 13th Conference on Foundations of Software Technology and Theorical Computer Science, Bombay, 1993.

[73] J. Grabowsky. *The Decidability of Persistence for Vector Addition Systems.* Information Processing Letters 11, vol-1, p. 20-23, 1980.

[74] H. Müller. *On the Reachability Problem for Persistent Vector Replacement Systems.* Computing Supplements, vol-3, p. 89-104, 1981.

[75] K. Jensen. *Coloured Petri Nets: A High Level Language for System Design and Analysis.* Lecture Notes in Computer Science, vol-483, p. 342-416, Springer-Verlag, Edited by G. Rozenberg 1990.

[76] H. J. Genrich. *Predicate/Transition Nets.* Lecture Notes in Computer Science, part I, vol-254, p. 207-247, Springer-Verlag, Edited by G. Rozenberg 1987.

[77] K. Jensen, P. Huber, R. M. Shapiro. *Hierarchies in Coloured Petri Nets.* Lecture Notes in Computer Science, vol-483, p. 313-341, Springer-Verlag, Edited by G. Rozenberg 1990.

[78] G. Dittrich. *Modeling of Complex Systems Using Hierarchical Petri Nets.* Codesign - Computer-Aided Software/Hardware Engineering, p. 128-144, IEEE Press, Edited by J. Rozenblit, K. Buchenrieder 1995.

[79] R. Fehling. *A Concept of Hierarchical Petri Nets with Building Blocks.* Lecture Notes in Computer Science, vol-674, p. 148-168, Springer-Verlag, Edited by G. Rozenberg 1993.

[80] P. R. M. Maciel, T. M. P. Medeiros, L. C. Albuquerque, J. F. B. Castro. *Uso de Redes de Petri Temporizadas para o Diagnóstico de Falhas em Sistema Digitais.* VI Simpósio de Computadores Tolerantes a Falhas, p. 181-200, Canela, RS, 1995.

[81] P. R. M. Maciel, E. N. S. Barros. *Captura de Requisitos Temporais Usando Redes de Petri para o Particionamento de Hardware/Software.* IX Simpósio Brasileiro de Concepção de Circuitos Integrados, p. 383-396, Recife, PE, 1996.

[82] P. R. M. Maciel, E. N. S. Barros. *Capturing Time Constraints by Using Petri Nets in the Context of Hardware/Software Codesign.* a ser publicado no 7th IEEE International Workshop on Rapid System Prototyping, Porto Caras, Thessaloniki, Grécia, 1996.

[83] E. Barros, A. Sampaio. *Towards Provably Correct Hardware/Software Partitioning Using OCCAM.* Procedings of

the Third International Workshop on Hardware/Software Co-desig, 1994.

[84] G. Jones. *Programming in OCCAM*. C.A.R. Hoare Series Editor, Prentice-Hall International Series in Computer Science, 1987.

[85] E.Best. *Esprit Basic Research Action 3148 DEMON (design methods based on nets)*. Lecture Notes in Computer Science, p. 1-20, Springer-Verlag, Edited by G. Rozenberg 1992.

[86] J. A. M. de Queiroz, P. R. F. Cunha. *Sistemas Distribuídos: de Epecificação LOTOS a implementações*. IX Escola de Computação, 1994.

Apêndice A

Teoria Bag

Neste apêndice apresentamos uma breve introdução à teoria *bag*.

Definição A.0.1 - Bag: *definimos um bag* M *sobre um conjunto não-vazio* C, *por uma função* $M : C \rightarrow \mathbb{N}$, *onde* $m(c)$ *representa o número de ocorrência do elemento* c *em M.*

Utilizamos o símbolo [] para denotarmos os *bags* e {}, para os conjuntos.

Alternativamente, podemos representar o número de ocorrência de um elemento em um *bag*, pela representação explícita dessa desse elemento n vezes, ou seja, $M_1 = [2a, b, 3c] \quad = \quad [a, a, b, c, c, c]$.

Definição A.0.2 - Cardinalidade: *seja o bag* $M \in C_M$. *A cardinalidade de* M $(|M|)$ *é o total de ocorrência de todos os elementos no bag* M.

Definição A.0.3 - Igualdade: *sejam os bags* M_1 *e* $M_2 \in C_M$,*onde* C_M *é um conjunto.* $M_1 = M_2$ *se, e somente se, para todo elemento* $i \in C_M$, $m_1(i) = m_2(i)$,.

Definição A.0.4 - Subbag: *sejam os bags* M_1 *e* $M_2 \in C_M$. $M_1 \subseteq M_2$ *se, e somente se, para todo elemento* i, $m_1(i) \leq m_2(i)$

Definição A.0.5 - Soma: *sejam os bags* M_1 *e* $M_2 \in C_M$. $M_1 + M_2 = M_s$, *onde* $m_s(i) = m_1(i) + m_2(i)$, *para todo elemento* $i \in C_M$.

Definição A.0.6 - União: *sejam os bags* M_1 *e* $M_2 \in C_M$. $M_1 \bigcup M_2 = M_u$, *onde* $m_u(i) = max(m_1(i), m_2(i))$, *para todo elemento* $i \in C_M$.

Definição A.0.7 - Interseção: *sejam os bags* M_1 *e* $M_2 \in C_M$.

$M_1 \bigcap M_2 = M_{in}$, *onde* $m_{in}(i) = min(m_1(i), m_2(i))$, *para todo elemento* $i \in C_M$.

Definição A.0.8 - Diferença: *sejam os bags* M_1 *e* $M_2 \in C_M$. $M_2 - M_1 = M_s$, *onde* $m_s(i) = m_2(i) - m_{in}(i)$, *para todo elemento* $i \in C_M$. $m_{in}(i) \in M_{in}$, *onde* M_i *é o bag intersecção entre* M_1 *e* M_2.

Apêndice B

Álgebra Matricial

Neste apêndice apresentamos os conceitos básicos sobre matrizes, conceitos esses importantes na resolução de problemas em redes de Petri.

Definição B.0.9 - Igualdade: *sejam as matrizes* $A_{m\times n} = [a_{ij}]_{m\times n}$ *e* $B_{r\times s} = [b_{ij}]_{r\times s}$. $A_{m\times n} = [a_{ij}]_{m\times n}$ *e* $B_{r\times s} = [b_{ij}]_{r\times s}$ *são iguais (A=B), se têm o mesmo número de linhas (m=r), o mesmo número de colunas (n=s), e todo* $a_{ij} = b_{ij}$.

- **Tipos**

  Definição B.0.10 - Matriz Quadrada: *seja uma matriz* $A_{m\times n} = [a_{ij}]_{m\times n}$. $A_{m\times n} = [a_{ij}]_{m\times n}$ *é dita matriz quadrada se, e somente se,* $m = n$.

  Definição B.0.11 - Matriz Nula: *seja uma matriz* $A_{m\times n} = [a_{ij}]_{m\times n}$. $A_{m\times n} = [a_{ij}]_{m\times n}$ *é dita matriz nula se, e somente se,* $a_{ij} = 0$, *para todo em i e* j.

  Definição B.0.12 - Matriz Coluna: *seja uma matriz* $A_{m\times n} = [a_{ij}]_{m\times n}$. $A_{m\times n} = [a_{ij}]_{m\times n}$ *é dita matriz coluna se, e somente se,* $n = 1$.

  Definição B.0.13 - Matriz Linha: *seja uma matriz* $A_{m\times n} = [a_{ij}]_{m\times n}$. $A_{m\times n} = [a_{ij}]_{m\times n}$ *é dita matriz lina se, e somente se,* $m = 1$.

Definição B.0.14 - Matriz Diagonal: *seja uma matriz* $A_{m\times m} = [a_{ij}]_{m\times m}$ *(matriz quadrada).* $A_{m\times m} = [a_{ij}]_{m\times m}$ *é dita matriz diagonal se, e somente se,* $a_{ij} = 0$*, para todo* $i \neq j$ *e* $a_{ij} \neq 0$*, para todo* $i = j$*.*

Definição B.0.15 - Matriz Identidade: *seja* $A_{m\times m} = [a_{ij}]_{m\times m}$ *uma matriz diagonal.* $A_{m\times m} = [a_{ij}]_{m\times m}$ *é dita matriz identidade se, e somente se,* $a_{ii} = 1$ *e* $a_{ij} = 0$*, para todo* $i \neq j$*.*

Definição B.0.16 - Matriz Triangular Superior: *seja uma matriz* $A_{m\times m} = [a_{ij}]_{m\times m}$ *(matriz quadrada).* $A_{m\times m} = [a_{ij}]_{m\times m}$ *é dita matriz triangular superior se, e somente se,* $a_{ij} = 0$*, para todo* $i < j$*.*

Definição B.0.17 - Matriz Triangular Inferior: *seja uma matriz* $A_{m\times m} = [a_{ij}]_{m\times m}$ *(matriz quadrada).* $A_{m\times m} = [a_{ij}]_{m\times m}$ *é dita matriz triangular inferior se, e somente se,* $a_{ij} = 0$*, para todo* $i > j$*.*

Definição B.0.18 - Matriz Simétrica: *seja uma matriz* $A_{m\times m} = [a_{ij}]_{m\times m}$ *(matriz quadrada).* $A_{m\times m} = [a_{ij}]_{m\times m}$ *é dita matriz simétrica se, e somente se,* $a_{ij} = a_{ji}$*, para todo* i *e* j*.*

- **Operações**

Definição B.0.19 - Adição: *sejam as matrizes* $A_{m\times n} = [a_{ij}]_{m\times n}$ *e* $B_{m\times n} = [b_{ij}]_{m\times n}$ *de mesma ordem. Representamos a matriz soma por* $A + B$*, onde os elementos dessa matriz são somas dos elementos correpondentes de* A *e de* B*, ou seja,* $A + B = [a_{ij} + bij]_{m\times n}$*.*

Propriedades: Sejam as matrizes A, it B e C de mesma ordem:

1. $A + B = B + A$
2. $A + (B + C) = (A + B) + C$
3. $A + 0 = A$, onde 0 denota a matriz nula de mesma ordem de A

Definição B.0.20 - Multiplicação por Escalar: *sejam a matrizes* $A_{m\times n} = [a_{ij}]_{m\times n}$ *e um número* k*. Define-se o produto de escalar entre em* k *e* A *por:* $k \cdot A = [ka_{ij}]_{m\times n}$*.*

Propriedades: Sejam as matrizes A e it B de mesma ordem e os números k, k_1 e k_2.

1. $k(A+B) = kA + kB$
2. $(k_1 + k_2)A = k_1A + k_2A$
3. $0 \cdot A = \mathbf{0}$, isto é, a multiplicação do escalar 0 por uma matriz resulta na matriz nula.
4. $k_1(k_2A) = (k_1k_2)A$

Definição B.0.21 - Transposição: *seja a matriz* $A_{m\times n} = [a_{ij}]_{m\times n}$. *Define-se a matriz trasposta* $A^T_{n\times m} = [b_{ij}]_{n\times m}$, *onde* $b_{ij} = a_{ji}$.

Propriedades: Sejam as matrizes A e it B de mesma ordem, as A^T e B^T, as respectivas matrizes transposta e o escalar k.

1. Uma matriz A é simétrica se, e somente se, $A = A^T$
2. $(A+B)^T = A^T + B^T$
3. $(kA)^T = kA^T$

Definição B.0.22 - Multiplicação de Matrizes: *sejam as matrizes* $A_{m\times n} = [a_{ij}]_{m\times n}$ *e* $B_{n\times p} = [b_{rs}]_{n\times p}$. *Representamos a matriz produto por* $AB = [c_{xy}]_{m\times p}$, *onde* $c_{xy} = \sum_{k=1}^{n} a_{xk}b_{ky} = a_{x1}b_{1y} + ... + a_{xn}b_{ny}$.

Apêndice C

Teoremas

Teorema C.0.1 *Uma rede de Petri $R = (P, T, I, O, K)$ é estruturalmente limitada se, e somente se, existe um vetor de inteiros positivos W com dimensão $\#P$ tal que $W \cdot C \leq 0$.*

Prova:

Seja uma marcação $M \in A(R, M_0)$, de forma que multiplicando-se o vetor de inteiros positivos pela equação fundamental das redes de Petri, temos:

$$W.M = W.M_0 + W.C.\overline{s}$$

Sendo $W.C \leq 0$ e $\overline{s} \geq 0$, temos:

$$W.M \leq W.M_0$$

Então a marcação de cada lugar da rede é limitada a:

$$M(p) \leq (W.M_0)/W(p)$$

onde $W(p)$ é p-ésimo componente do vetor W.

Teorema C.0.2 *Uma rede marcada $N = (R; M_0)$ é dita conservativa se, e somente se, existe a um vetor de pesos $W = (w_1, w_2, ..., w_n)$, de inteiros positivos tal que, $W \cdot C = 0$, onde $n = \#P$*

Prova:

Seja a equação fundamental $M = M_0 + C.\overline{s}$ e vetor de inteiros positivos W, então:

$$W.M = W.M_0 + W.C.\overline{s}$$

Sendo $W \cdot C = 0$, temos: $W \cdot M = W \cdot M_0$

Teorema C.0.3 *Uma rede marcada $N = (R; M_0)$ é dita parcialmente conservativa se, e somente se, existe a um vetor de pesos $W = (w_1, w_2, ..., w_n)$, de naturais de forma que, $W \cdot C = 0$ e $W \neq 0$, onde $n = \#P$*

Prova:

Seja a equação fundamental $M = M_0 + C.\overline{s}$ e vetor inteiros não-negativos $W > 0$, então:

$$W.M = W.M_0 + W.C.\overline{s}$$

Sendo $W \cdot C = 0$, temos: $W \cdot M = W \cdot M_0$

Teorema C.0.4 *Uma rede é repetitiva se, e somente se, existe um vetor característico $\overline{s}$ de inteiros positivos, tal que $C.\overline{s} \geq 0$ e $\overline{s} \neq 0$*

Prova:

Seja a equação fundamental $M = M_0 + C.\overline{s}$ e um vetor característico $\overline{s_1}$ de inteiros positivos, que representa o número de disparo de cada transição em uma seqüência s_1, tal que $C \cdot \overline{s_1} \geq 0$. Então, existe uma marcação $M \geq M_0$. Sendo assim, a seqüência s_1 pode ser repetida indefinidamente. Então, todas as transição sempre podem ser disparadas.

Teorema C.0.5 *Uma rede é parcialmente repetitiva se, e somente se, existe um vetor não-nulo ($\overline{s} \neq 0$), cujos componentes são números naturais e $C.\overline{s} \geq 0$.*

Prova:

Seja a equação fundamental $M = M_0 + C.\overline{s}$ *e um vetor característico* $\overline{s_1}$ *de* $\mathbb{N}$*, não-nulo, que representa o número de disparo de cada transição em uma seqüência* s_1*, tal que* $C \cdot \overline{s_1} \geq 0$*. Então, existe uma marcação* $M \geq M_0$*. Sendo assim, a seqüência* s_1 *pode ser repetida indefinidamente. Logo, algumas das transição sempre podem ser disparadas.*

Teorema C.0.6 *Uma rede é consistente se, e somente se, existe um vetor* $\overline{s}$ *não-nulo* ($\overline{s} \neq 0$) *de inteiros positivos, tal que* $C.\overline{s} = 0$

Prova:

Seja a equação fundamental $M = M_0 + C.\overline{s}$ *e um vetor característico* $\overline{s_1} \geq 0$ *de inteiros positivos, que representa o número de disparo de cada transição em uma seqüência* s_1*, tal que* $C \cdot \overline{s_1} = 0$*. Então, ao disparar-se a seqüência* s_1 *retorna-se à marcação inicial, isso significa que a seqüência pode ser repetida indefinidamente. Desta forma, todas as transição sempre podem ser disparadas.*

Teorema C.0.7 *Uma rede é parcialmente consistente se e somente se existe um vetor* $\overline{s}$ *não-nulo* ($\overline{s} \neq 0$) *de naturais de forma que* $C.\overline{s} = 0$.

Prova:

Seja a equação fundamental $M = M_0 + C.\overline{s}$ *e um vetor característico* $\overline{s_1} \geq 0$ *de* $\mathbb{N}$*, que representa o número de disparo de cada transição em uma seqüência* s_1*, de forma que* $C \cdot \overline{s_1} = 0$*. Então, ao disparar-se a seqüência* s_1 *retorna-se à marcação inicial; isso significa que a seqüência pode ser repetida indefinidamente. Desta forma, algumas das transições sempre podem ser disparadas.*