Uma Ferramenta Interativa Baseada em Redes de Petri para Modelagem, Simulação e Análise de Sistemas Complexos

Itamar de Souza Lima

Dissertação de Mestrado submetida à Coordenação dos Cursos de Pós-Gradução em Engenharia Elétrica da Universidade Federal da Paraíba - Campus II como parte dos requisitos necessários para obtenção do grau de Mestre em Engenharia Elétrica.

Área de Concentração: Processamento da Informação

Angelo Perkusich, D.Sc.

Orientador

Jorge César Abrantes de Figueredo, D.Sc.

Orientador

Campina Grande, Paraíba, Brasil

Uma Ferramenta Interativa Baseada em Redes de Petri para Modelagem, Simulação e Análise de Sistemas Complexos

Itamar de Souza Lima

Dissertação de Mestrado apresentada em Dezembro de 1997

Angelo Perkusich, D.Sc.

Orientador

Jorge César Abrantes de Figueredo, D.Sc.

Orientador

Antonio Marcus Nogueira Lima, Doutor.

Componente da Banca

Maria Izabel Cavalcanti Cabral, D.Sc.

Componente da Banca

Campina Grande, Paraíba, Brasil, Dezembro de 1997

L732f Lima, Itamar de Souza
Uma ferramenta interativa baseada em redes de petri para modelagem, simulacao e analise de sistemas complexos / Itamar de Souza Lima.- Campina Grande, 1997.
103 f.

Dissertacao (Mestrado em Engenharia Eletrica) - Universidade Federal da Praiba, Centro de Ciencias e Tecnologia.

1. Redes de Petri. 2. Redes de Petri - Sistemas Complexos. 3. Redes de Petri - Ferramenta Interativa. 4. Dissertacao I. Perkusich, Angelo , Prof. D.Sc. II. Figueiredo, Jorge Cesar Abrantes de. , Prof. D.Sc. III. Universidade Federal de Campina Grande - Campina Grande (PB) IV. Título

CDU 261.355(043)

UMA FERRAMENTA INTERATIVA BASEADA EM REDES DE PETRI PARA MODELAGEM, SIMULAÇÃO E ANÁLISE DE SISTEMAS COMPLEXOS

ITAMAR DE SOUZA LIMA

Dissertação Aprovada em 29.12.1997

PROF. ANGELO PERKUSICH, D.Sc., UFPB
Orientador

PROF. JORGE CESAR ABRANTES DE FIGUEIREDO, D.Sc., UFPB
Orientador

PROF. ANTONIO MARCUS NOGUEIRA LIMA, Dr., UFPB
Componente da Banca

MARIA IZABEL CAVALCANTI CABRAL, D.Sc., UFPB
Componente da Banca

CAMPINA GRANDE - PB
Dezembro - 1997

Dedicatória

Dedico este trabalho à Yolanda de Souza Lima, exemplo raro de mãe, de justiça, de bondade, de inocência, de amor ao próximo e de dedicação aos seus filhos, parentes e trabalho.

Agradecimentos

Agradeco aos meus orientadores, professores Angelo e Jorge pela paciência, amizade e pela incansável orientação e colaboração, indispensáveis na realização deste trabalho. Ao CNPq, pelo suporte financeiro.

Resumo

O projeto de sistemas complexos requer um processo intensivo de modelagem, simulação e análise. As tarefas, as temporizações, as restrições de concorrência e de conflito são, geralmente, analisadas à partir de valores próximos (ou médios) que assumem as variáveis do sistema, para determinados valores de entrada e determinada resposta ou valores esperados de saída. Neste sentido, a simulação pode contribuir na determinação de parâmetros dos sistemas complexos e na obtenção de referências para novas aplicações. Pode, ainda, reprovar, cancelar e paralisar sistemas cujos projetos ou procedimentos estejam sendo tratados como corretos.

As redes de Petri têm sido largamente utilizadas como ferramentas gráficas e matemáticas no auxílio ao projeto de sistemas complexos. Estes sistemas são caracterizados pelo comportamento discreto, sequencial, assíncrono e não determinístico que, aliado aos processos concorrentes e conflitantes de suas atividades, os fazem de difícil (senão impossível) análise, quando aplicadas técnicas tradicionais de controle.

A presente dissertação apresenta uma breve introdução das redes de Petri e algumas de suas extensões e a ferramenta por mim desenvolvida denominada ManNet. Esta ferramenta é baseada em linguagem orientada a objeto e é compatível com computadores pessoais. Dentre suas diversas facilidades, permite a interação com o usuário em tempo real e objetiva auxiliar no processo de modelagem, simulação e análise de modelos que usam redes de Petri e extensões. Como exemplos de aplicação, apresentamos uma célula de manufatura e um conhecido protocolo de comunicação de dados.

Abstract

Complex Systems design require intensive modeling, analysis and simulation. In some cases, a complex system can be seem as a multi-task and independent sub-systems. Each task, its timing, concurrency and conflit dependencies may be analysed properly with respect of their input and output values. Therefore, simulation would help the designer to stablish complex parameters of sub-systems as alternative references for new applications.

In some cases supposed complete system may be modified or extinguished, after the analisys of its model. In general, complex system has huge dimensions or multi-level algorithms that means in complex sub-systems integration and high cost of design; as example, we can cite Flexible Manufacturing Systems (FMS) and data communication protocols.

Petri nets have proven to be very reliable in practice for the design of such systems named Discret Event Systems, DES. These complex systems are characterized by its assinchronous and sequential and stochastic behavior, by high level of concurrancy and conflit of its tasks and mutual exclusive resources. These properties are of difficult descrition and analysis (or impossible) when used tradictional Theory of Control.

In this dissertion we present the basic concepts of Petri nets and some temporal extensions and our interative Petri net tool named ManNet. We also exemplify the application of this tool by modeling and analysing two examples, a manufacturing cell and a data communication protocol.

Índice

1 Introdução **1**

1.1 Motivação 1

1.2 Conceitos Básicos e Trabalhos Relacionados 3

1.3 Objetivo 4

1.4 Estrutura da Dissertação 5

2 Conceitos Básicos **6**

2.1 Introdução 6

2.2 Redes Lugar/Transição (*Place/Transition, P/T*) 7

2.3 Extensões Temporais de Redes de Petri 11

2.3.1 Extensões Temporais Determinísticas 12

2.3.2 Redes de Petri Estocásticas 17

2.3.3 Redes de Petri Estocásticas Generalizadas 19

2.3.4 Redes de Petri com Temporização Nebulosa 20

2.4 Métodos de Análise 21

2.4.1 Propriedades Estáticas 21

2.4.2 Propriedades Dinâmicas 22

2.4.3 Invariantes de Lugar e Transição 22

2.4.4 Redução 24

2.4.5 Árvore Alcançabilidade/Cobertura 25

2.5 Modelagem de Sistemas Utilizando Sistemas de Rede 27

3 A Ferramenta ManNet **28**

3.1 Introdução 28

3.2 A Ferramenta Petrilyser 29

3.2.1 O Editor de Redes de Petri 30

3.2.2 A Análise da Estrutura da Rede de Petri 32

3.2.3 A Simulação 33

3.2.4 A Rede de Petri Exemplo de Molloy 34

3.2.5 Utilizando a Ferramenta Petrilyser 44

3.2.6 Principais rotinas 47

3.3 A Ferramenta ManNet 52

3.3.1 O Editor de Redes de Petri 55

3.3.2 As Propriedades Estruturais dos Modelos em Redes de Petri . . 64

3.3.3 O Simulador 67

3.3.4 Elementos do Simulador 68

3.4 Medidas de Desempenho 78

3.5 Interface com o Usuário 78

3.6 Ajuda On Line 79

4 Exemplos **82**

4.1 Introdução 82

4.2 Célula de Manufatura 82

4.2.1 Propriedades Estruturais e Temporais 83

4.3 Protocolo de Comunicação 87

4.4 Medidas de Desempenho 89

5 Conclusões **91**

Lista de Figuras

2.1 Representação de uma rede P/T, (a) grafo, (b) matriz de incidência . . 9

2.2 (a) Conflito, (b) Concorrência, (c) Junção, (d) Divisão, (e) Seqüência, (f) Sincronização 10

2.3 Rede de Petri temporal 14

2.4 Transformação do atraso de lugar em atraso de transição 15

2.5 Exemplo de uma rede TB 16

2.6 Uma rede P/T 26

2.7 (a) Árvore de cobertura e (b) Grafo de cobertura para a rede da exemplo 26

3.1 Iniciando a Ferramenta Petrilyzer 35

3.2 Menu Arquivos 36

3.3 Lendo Arquivo de Lugares 36

3.4 Editando Arquivo de Lugares 37

3.5 Apresentando Arquivo de Lugares 37

3.6 A Rede de Petri Exemplo de Molloy 38

3.7 Grafo de Alcançabilidade da Rede de Molloy 38

3.8 Árvore de Cobertura da Rede de Molloy 39

3.9 Simulação da Rede de Molloy 40

3.10 Exemplo de Célula de Manufatura 41

3.11 Modelo da Célula de Manufatura em rede de Petri 42

3.12 Janela de Simulação da ferramenta Petrilyzer - versão DOS/Win3.xx . 43

3.13 Janela de Simulação da Ferramenta Petrilyzer - versão Windows 44

3.14 Componentes da Ferramenta ManNet 53
3.15 Janela Principal da Ferramenta ManNet 55
3.16 Sub-Menu arquivo 56
3.17 Janela de localização de arquivos 57
3.18 Mensagem de aviso de erro 57
3.19 Pasta de edição de Lugares 58
3.20 Edição de Lugar que modela um Recurso Variável 59
3.21 Edição de transições 61
3.22 Edição de arcos 62
3.23 Edição das propriedades das redes de Petri 63
3.24 Edição das marcações iniciais 64
3.25 Submenu estrutura 65
3.26 Marcação final 66
3.27 Rede de Petri limitada e com bloqueio 67
3.28 Simulação e análise 68
3.29 Edição de lugares especiais 70
3.30 Modelo de uma tarefa 71
3.31 Estados das tarefas 73
3.32 Estados das tarefas 73
3.33 Edição do RTC/Relógios 77
3.34 Submenu de Ajuda 80
3.35 Ajuda *On Line* 81

4.1 Célula de Manufatura 83
4.2 Modelo em Redes de Petri da Célula de Manufatura 84
4.3 Janela com os resultados do Simulador 87
4.4 Um Algoritmo Determinístico para o protocolo de Passagem de Ficha . 88

Capítulo 1

Introdução

1.1 Motivação

O desenvolvimento, a análise e a integração de sistemas complexos tais como células de manufatura e protocolos de comunicação, entre outros, requer exaustivo processo de modelagem e análise, devido às grandes dimensões físicas (no caso das células de manufatura) e à complexidade de seus algoritmos e nível de abstração (protocolos de comunicação), entre outros fatores. O processo da modelagem objetiva definir meios para representar as principais características ou propriedades destes sistemas com aceitável precisão, possibilitando também a avaliação destas propriedades. Através da simulação e análise do modelo de um sistema complexo, este poderá ser (ou não) modificado, redefinido, ou mesmo extinto, sem a necessidade de sua parcial ou total implementação.

A modelagem utilizando redes de Petri [Mur89] consiste na utilização de uma estrutura constituída de símbolos que representam elementos do sistema e as relações entre eles. Quando definido o estado inicial a esta estrutura, é possível a montagem de diagramas de estados ou grafos, cuja análise pode auxiliar na identificação das propriedades do sistema, tais como *concorrência, conflito e bloqueio.* As redes de Petri são uma ferramenta que tem sido largamente desenvolvida e utilizada no auxílio ao projeto de sistemas complexos por seu modelo original não possibilitar a representação

de propriedades temporais e estocásticas dos sistemas reais, entre outras. As diversas técnicas e ferramentas desenvolvidas para este fim têm evoluído com o surgimento de novas aplicações. As extensões temporais às redes de Petri buscam atender a restrições temporais dos sistemas discretos.

Diversas propostas de ferramentas auxiliares foram apresentadas para vários ambientes computacionais. As primeiras ferramentas compatíveis para computadores pessoais nasceram, em sua maioria, em centros de pesquisa e universidades. Elas evoluíram por alguns anos, certamente, consumindo milhares de homem/hora no desenvolvimento, e tornaram-se obsoletas rapidamente devido à vertiginosa evolução do *hardware*, ao frequente surgimento de novas extensões às redes de Petri, a evolução da engenharia de *software* e seus produtos (sistemas operacionais, compiladores, linguagens, etc.). Mesmo as ferramentas desenvolvidas recentemente, requerem constante atualização e, embora façam uso de vastas bibliotecas de programas, linguagem orientada a objeto e apresentarem diversos recursos gráficos, são aplicáveis somente ao estudo lógico dos modelos de sistemas a eventos discretos (*Discret Event Dynamic Systems, DES's* [ZD93]). Algumas são ferramentas dedicadas (manufatura, por exemplo), porém a maioria não permite a simulação das propriedades temporais e estocásticas e são sistemas *fechados*, ou seja, que não permitem a intervenção do usuário. Entre as ferramentas disponíveis pela INTERNET, citamos a *PMaker* (ou *Petri Maker*) desenvolvida no *Developpement de l'Atelier PETRI Maker*, na França (http://www.istia.univ-angers.fr/pmaker/v3.1d) que está em sua sétima, senão oitava versão. Outras soluções possuem custo elevado, ou utilizam outras ferramentas às redes de Petri ou ambientes computacionais de maior porte, a exemplo da ULTRASAN, uma ferramenta desenvolvida na Universidade de Illinois, USA).

A ferramenta ManNet é um *software* de arquitetura *aberta* em fase final de desenvolvimento e consiste da implementação do trabalho de dissertação do Curso de Mestrado em Engenharia Elétrica do Centro de Ciências e Tecnologia da Universidade Federal da Paraíba, para atuar como uma ferramenta auxiliar no processo de modela-

gem e simulação de sistemas complexos. A ManNet permite a geração de grafos que auxiliam na análise do comportamento do modelo baseado em redes de Petri, a análise de desempenho à partir da modelagem e simulação de suas propriedades temporais e a interação em tempo real com o usuário. Permite, ainda, definir diversos procedimentos de *sorteio* e temporização e de modelos de entrada e saída de sistemas complexos.

1.2 Conceitos Básicos e Trabalhos Relacionados

Uma rede de Petri é um grafo direcionado bipartido, mais um estado inicial denominado marcação inicial [Mur89]. O grafo direcionado consiste de dois tipos de nós, denominados *lugares* e *transições*. Os nós em um a rede de Petri são relacionados (conectados) por arcos rotulados com pesos (inteiros positivos). Um arco não pode relacionar componentes do mesmo tipo. Graficamente, lugares são representados por círculos e transições por retângulos ou barras. Um lugar p é entrada para uma transição t se existe um arco direcionado conectando o lugar à transição, neste caso o lugar é um *lugar de entrada*. Um lugar p é saída para uma transição, se existe um arco direcionado conectando a transição ao lugar, neste caso o lugar é um *lugar de saída*. O grafo direcionado define a estrutura de um sistema representado por uma rede de Petri.

A definição informalmente introduzida para redes de Petri é também denominada grafo de suporte ou estrutura da rede. Uma marcação atribui a cada lugar p um número k, inteiro não negativo, de elementos denominados fichas. Fichas são representadas por pontos pretos. Quando um número k de fichas é atribuído ao lugar, diz-se que o lugar está marcado com k fichas. Uma marcação é um vetor com o mesmo número de lugares que a estrutura da rede. O comportamento do sistema modelado pela estrutura da rede pode ser caracterizado pelo movimento de fichas pelos lugares, quanto a rede é *executada*. Este movimento de fichas caracteriza o comportamento dinâmico do sistema em termos de estados e suas mudanças. Para mover fichas, transições disparam se *habilitadas*. Uma transição deve estar *habilitada* na marcação corrente para poder disparar. Uma transição é dita habilitada se todos os lugares de entrada são marcados,

por pelo menos, o mesmo número de fichas definido pelo pêso associado aos arcos conectando estes lugares à transição. Uma transição habilitada pode disparar. Quando uma transição dispara, o número de fichas associados aos pesos dos arcos de entrada são removidas dos lugares de entrada e depositadas nos lugares de saída de acordo com os pesos associados aos arcos saindo da transição, conectando os lugares de saída. Este movimento de fichas pela rede é também conhecido como *jogo de fichas.*

A análise de redes de Petri é principalmente baseada no grafo de alcançabilidade ou em técnicas algébricas lineares. Um *grafo de alcançabilidade* representa o conjunto de estados alcançáveis, e pode ser usado para verificar uma variedade de propriedades, tal como, se a rede é livre de bloqueio (*deadlock*). Técnicas algébricas lineares são utilizadas para calcular *invariantes.* A idéia é representar a rede por uma matriz de incidência e marcações por vetores de controle. Esta representação pode, então, ser utilizada para caracterizar a dinâmica do sistema utilizando-se a equação de controlabilidade utilizada em teoria de sistemas de controle. Portanto, pode-se derivar equações algébricas lineares cujas soluções características são denominadas invariantes. Dois tipos de invariantes podem, então, ser identificados: *invariantes de lugar* e *invariantes de transição.* Para uma introdução detalhada a estes e outros aspectos relacionados a redes de Petri o leitor pode referir-se à [Mur89, Pet81]. No Capítulo 2 é apresentada uma revisão dos conceitos básicos relacionados a redes de Petri bem como conceitos relacionados com suas extensões termporais determinísticas [Ram74] e estocásticas [AM89].

1.3 Objetivo

Finalmente, o objetivo do presente trabalho está na especificação, no desenvolvimento e na documentação de um *software* que possa ser utilizado como ferramenta interativa e auxiliar no processo de modelagem, simulação e análise de sistemas complexos, modelados em redes de Petri temporais e em técnicas de modelagem e de escalonamento temporal. Com a ferramenta ManNet é possível realizar a análise da estrutura do modelo em rede de Petri e a análise de comportamento dinâmico através da geração de

grafos e da simulação.

1.4 Estrutura da Dissertação

O restante da dissertação está organizada da seguinte forma:

- no Capítulo 2 são apresentados alguns conceitos básicos de redes de Petri e extensões temporais, e da técnica de modelagem em que se baseia a ferramenta ManNet;
- no Capítulo 3 é apresentada a ferramenta ManNet, suas propriedades e aplicabilidade;
- no Capítulo 4 são apresentados modelos e resultados da análise (obtidos mediante simulação) de dois exemplos: o primeiro, que explora uma célula de manufatura, e o segundo, um conhecido protocolo de comunicação; e
- no Capítulo 5 são apresentadas as conclusões e perspectivas.

Ainda, em anexo, apresenta-se a descrição do suporte de ajuda da ferramenta ManNet.

Capítulo 2

Conceitos Básicos

2.1 Introdução

Neste capítulo são introduzidos os conceitos básicos necessários a compreensão desta dissertação.

Bancos de dados, protocolos de acesso a canal, tempos de resposta, são elementos de um complexo sistema de comunicação. Da mesma forma, peças, paletes, robôs, máquinas, veículos, esteiras, são elementos de Sistemas Flexíveis de Manufatura (*Flexible Manufacture System, FMS*). Ambos são Sistemas a Eventos Discretos (*Discrete Event Systems, DED's*), cujo projeto e operação requerem processo intensivo de modelagem, simulação e análise. Estes sistemas, de grande importância no universo dos sistemas complexos pela alta taxa de crescimento, complexidade e diversidade de aplicações, são caracterizados por seu comportamento assíncrono, sequencial, concorrente, conflitante e não determinístico de tarefas e pela exclusão mútua dos recursos utilizados.

Diversas técnicas de modelagem já foram definidas para as extensões temporais de redes de Petri, visando atender às propriedades de tais sistemas complexos, porém não existe uma técnica formal baseada em redes de Petri que descreva todas estas de forma conjunta ou que permita sua modelagem. Com o objetivo de gerenciar a complexidade do grande número de estados e diferentes tipos de propriedades, a verificar, um

sistema complexo pode ser abordado como um conjunto de sub-sistemas multi-tarefas modulares e independentes, onde cada sub-sistema representa um conjunto compacto de atividades e de estados ou uma partição do sistema como um todo. Possue, ainda, número definido de atividades, recursos e propriedades (comportamento estocástico, dependências estruturais, etc.). Propriedades estas que devem ser analisadas segundo as possíveis entradas e saídas. Em alguns casos, os resultados destas análises podem facilitar futuras e inesperadas mudanças no sistema.

2.2 Redes Lugar/Transição (*Place/Transition, P/T*)

As Redes *Lugar/Transição (P/T)* estão entre os modelos de rede de Petri mais aplicados. Em redes *P/T*, os lugares podem ser marcados por uma ou mais fichas não estruturadas, os quais, na maioria dos casos representam contadores e possuem capacidades [Mur89] definidas[1]. Nestas redes, os arcos possuem um peso[Rei87] associado[2].

Definição 2.1 *Redes Lugar/Transição*

A tupla $\mathcal{N} = (P, T; F, K, W, M_0)$ é denominada uma rede Lugar/Transição se e somente se:

1. $(P, T; F)$ é uma rede onde P é o conjunto de lugares e T é o conjunto de transições.
2. $K : P \rightarrow \mathbb{N}^+ \cup \{\infty\}$ é a função de capacidade.
3. $W : F \rightarrow \mathbb{N}^+$ é a função de peso.
4. $M_0 \rightarrow \mathbb{N}$ é uma função de marcação inicial satisfazendo $\forall p \in P : M_0(p) \leq K(P)$.

Uma rede *P/T* tal que $\forall p \in P : K(p) = \infty$ e $\forall f \in F : W(f) = 1$ pode ser denotada simplesmente por $\mathcal{N} = (P, T; F, M_0)$ e e denominada um *rede de Petri ordinária.*

[1] A capacidade de um lugar indica o número máximo de fichas que este pode receber

[2] o peso de um arco indica o número inteiro de fichas que podem "fluir" através dele a cada ocorrência ou disparo das transições envolvidas.

Definição 2.2 *Regra de transição*

Considerando-se que $\mathcal{N} = (P, T; F, K, W, M_0)$ é uma rede P/T.

1. A função $M : P \rightarrow \mathbb{N}$ é dita uma marcação de $\mathcal{N}$ se e somente se $\forall p \in P : M_0(p) \leq K(p)$.

2. Um transição $t \in T$ está habilitada em M se e somente se $\forall p \in P : W(p,t) \leq M(p) \leq K(p) - W(t,p)$.

3. Se $t \in T$ é uma transição habilitada na marcação M, então t pode ocorrer, resultando em uma nova marcação M' dada pela equação: $M'(p) = M(p) - W(p,t) + W(t,p)$, $\forall p \in P$.

4. A ocorrência ou disparo de t altera a marcação M em uma nova marcação M', e é denotada por $M[t\rangle M'$.

5. $[M_0\rangle$ é a classe de alcançabilidade (para a frente), e é definida como o menor conjunto de marcações de $\mathcal{N}$ tal que: $M_0 \in [M_0\rangle$, e se $M_1 \in [M_0$ e $M_1[t\rangle M_2$ para alguma $t \in T$, então $M_2 \in [M_0\rangle$.

Antes de introduzir outros conceitos relacionados com sistemas de rede, discutir-se-á a regra de transição ou disparo como apresentada na Definição 2.2. Esta regra de disparo determina o comportamento da rede P/T em termos de estados do sistema (marcações) e suas mudanças.

Uma marcação é representada por um vetor coluna transposto, $[M(p_1), M(p_2), \cdots, M(p_n)]$, onde $M(p_i)$ é a marcação do lugar p_i. Esta evolução de estados permite simular o comportamento dinâmico do sistema modelado por uma rede P/T. Quando uma transição está habilitada, por exemplo, a transição a na Figura 2.1, ela pode disparar. Quando a transição dispara, fichas são removidas dos lugares de entrada e são depositadas nos lugares de saída. No exemplo da Figura 2.1, duas fichas são removidas do lugar A, uma

vez que o peso do arco conectando-o à transição a é dois, e uma ficha é removida do lugar B (o peso um associado ao arco conectando o lugar B a transição a está implícito). Após as fichas serem removidas dos lugares de entrada, uma ficha é depositada no lugar de saída C. Em outras palavras, o sistema evolui do estado $[2, 1, 0, 0]$, para o estado $[0, 0, 1, 0]$.

Definição 2.3 *Matriz de incidência para uma rede P/T*

Para uma rede P/T com n transições e m lugares, a matriz de incidência $\mathcal{I}$ é uma matriz de inteiros $n \times m$ e uma entrada típica é dada por:

$$a_{ij} = a_{ij}^{+} - a_{ij}^{-} \tag{2.1}$$

onde $a_{ij}^{+} = w(i, j)$, com $w \in W$, é o peso do arco da transição i para o lugar de saída j, e a_{ij}^{-} é o peso do arco para a transição i do seu lugar de entrada j.

Uma rede P/T pode ser representada por um grafo orientado, e algebricamente por uma matriz de incidência. Na Figura 2.1 é apresentado um exemplo de uma rede P/T e sua matriz de incidência.

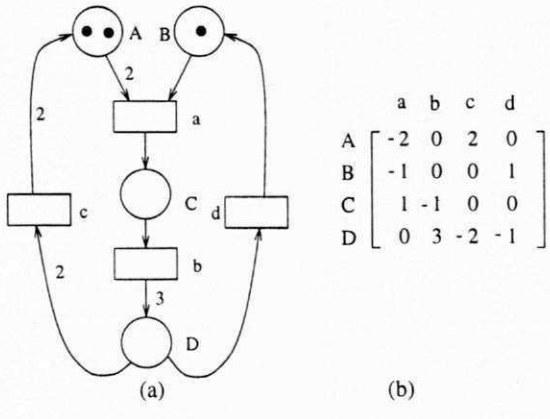


Figura 2.1: Representação de uma rede P/T, (a) grafo, (b) matriz de incidência

As redes de Petri têm sido extensivamente usadas em diversas áreas. Na computação destacam-se na avaliação de desempenho, na comunicação de protocolos, na modelagem e análise de sistemas distribuídos, entre outras. As redes de Petri apresentam

formalismo satisfatório para modelar e analisar muitos sistemas complexos, especialmente aqueles que exibem uma das seguintes características: conflito, concorrência, junção, divisão, seqüência e sincronização. Essas características podem ser facilmente representadas através das redes de Petri como mostrado na Figura 2.2.

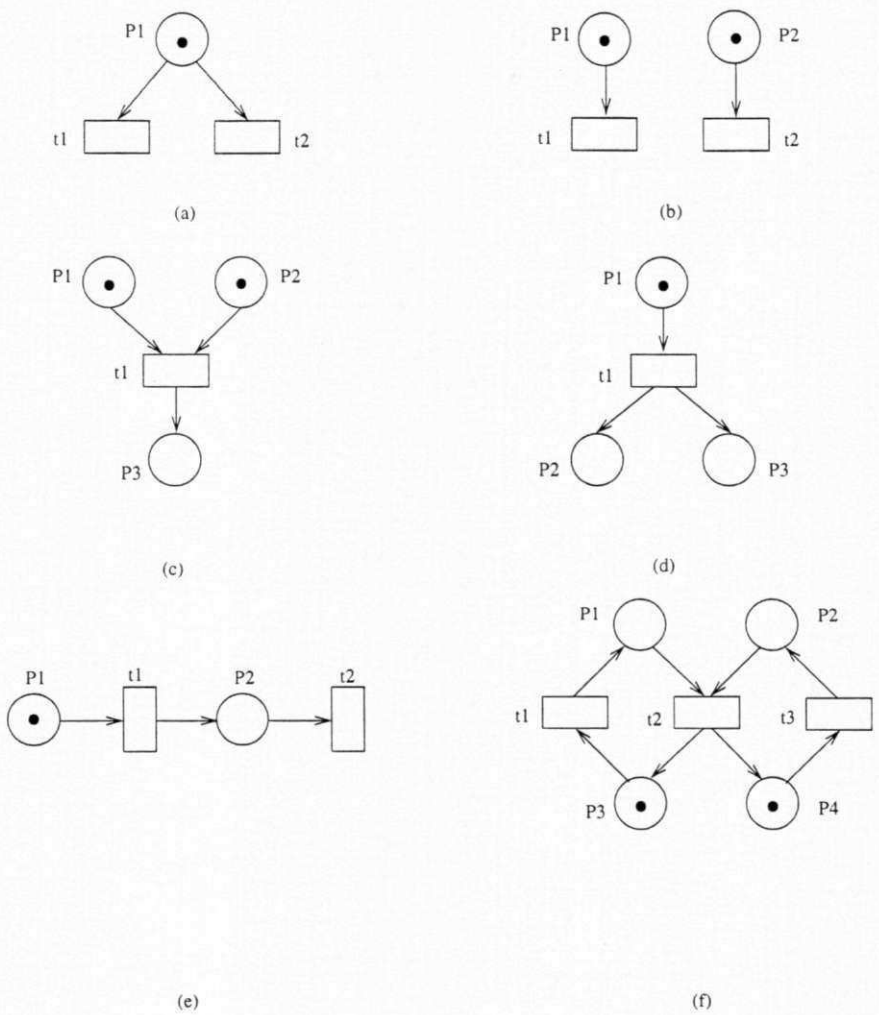


Figura 2.2: (a) Conflito, (b) Concorrência, (c) Junção, (d) Divisão, (e) Seqüência, (f) Sincronização

Na Figura 2.2(a), uma situação de conflito é modelada. As transições *t1* e *t2* são ambas sensibilizadas mas, apenas uma delas pode disparar. Se *t1* dispara, a transição *t2* é desabilitada e vice-versa. A Figura 2.2(b) modela a concorrência, onde as transições *t1* e *t2* são ambas sensibilizadas e representam atividades que podem ser executadas con-

correntemente. As Figuras 2.2(c) e 2.2(d) representam, respectivamente, as situações de junção e divisão. Na Figura 2.2(e), a transição *t2* dispara depois do disparo de *t1*, representando atividades que são executadas sequencialmente. As redes de Petri são também adequadas para representar sincronização como mostrado na Figura 2.2(f). Mesmo considerando que as transições *t1* e *t3* possam ser executadas concorrentemente, o disparo da transição *t2* depende das fichas nos lugares *P1* e *P2*.

A partir da definição original de redes de Petri, muitas extensões lógicas foram propostas e que são largamente aplicadas. Exemplos dessas extensões lógicas são arcos múltiplos e arcos inibidores [Mur89]. Os dois principais tipos de extensões de redes de Petri são discutidos na seqüência.

2.3 Extensões Temporais de Redes de Petri

Nas redes de Petri clássicas, a caracterização das propriedades temporais de um sistema não é possível. Em outras palavras, com as redes de Petri clássicas é possível representar apenas as propriedades qualitativas (não relacionadas ao tempo) de um sistema. No sentido de tornar possível a representação de propriedades quantitativas (relacionadas ao tempo) dos sistemas, algumas extensões de redes de Petri foram propostas. Diferentes técnicas foram usadas, as quais diferem basicamente em dois aspectos:

1) **localização:** As restrições de tempo podem ser associadas aos lugares ou transições.

2) **tipo:** A natureza das especificações das restrições de tempo (atrasos fixos, intervalos, atrasos estocásticos, etc).

Na seqüência, são apresentadas algumas das extensões de redes de Petri para a caracterização de restrições de tempo. Estas extensões utilizam uma abordagem determinística [GMMP89, MF76, Ram74, Sif80] ou estocástica [AMBC84, HS86, Mol82b]. Uma abordagem diferente foi proposta por Suzuki [SL89]. Em seu trabalho, em contraste às abordagens determinísticas e estocásticas, as restrições temporais são representadas por operações em lógica temporal. Suzuki afirma que as extensões temporais

tradicionais apresentam bons resultados analíticos na área de avaliação de desempenho mas falham na representação de idéias sobre as relações causais e temporais entre eventos. Logo, as *redes de Petri com lógica temporal (Temporal Petri Nets)* foram propostas para mostrar claramente as dependências causais e temporais entre eventos bem como representar, de forma elegante, propriedades fundamentais dos sistemas como eventualidade e justiça. As *Temporal Petri Nets* são definidas como as redes de Petri clássicas juntamente com uma linguagem para descrever as restrições temporais.

2.3.1 Extensões Temporais Determinísticas

As *Redes de Petri Temporizadas (Timed Petri Net - TdPN)* [Ram74] são uma extensão de redes de Petri nas quais uma duração ou um tempo de disparo é associado a cada transição da rede. Nas *TdPNs*, as transições são sensibilizadas da mesma forma que as transições nas redes de Petri clássicas. Quando sensibilizadas, as transições disparam instantaneamente mas, as fichas só são depositadas nos lugares de saída após decorrido t unidades de tempo após o disparo da transição, em que t é o tempo de disparo associado com a transição.

Definição 2.4 *Rede de Petri Temporizada*

Uma *Rede de Petri Temporizada* é uma 6-tupla $(P,T;F,\tau,M_0)$ em que $(P,T;F)$ e M_0 são elementos conhecidos e τ é uma função de tempo $\tau : T \rightarrow \{1, 2, ...\}$, mapeando cada transição na rede nos números naturais.

As *TdPNs* foram usadas para fazer análise de desempenho de sistemas. Ramchandani [Ram74] estudou o comportamento dos estados das *redes de Petri temporizadas* e desenvolveu métodos para calcular a taxa de *throughput* para certas classes dessa rede. Zuberek [Zub80, Zub91] estendeu o trabalho de Ramchandani e construiu um grafo dirigido rotulado finito representando o comportamento de uma *rede de Petri temporizada*. Devido à similaridade desses grafos com a cadeia de Markov com estados finitos,

as técnicas Markovianas podem ser usadas para efetuar análise. Ho [RH80] também usou as *redes de Petri temporizadas* para fazer avaliação de desempenho de sistemas.

No modelo de *Rede de Petri Temporal (Time Petri Net - TPN)*, um intervalo $[t_{min}, t_{max}]$ é associado com cada transição da rede[MF76] em que, t_{min} representa o tempo mínimo que deve ocorrer a partir do instante em que as condições de sensibilização de uma transição são satisfeitas até o tempo em que a transição pode disparar. t_{max} representa o tempo máximo que a transição pode permanecer sensibilizada. Após t_{max}, a transição deve disparar.

Definição 2.5 *Rede de Petri Temporal*

Uma *rede de Petri temporal* é uma 6-tupla $(P,T;F,E,M_0)$ em que P, T; F e M_0 são definidos como nas *redes de Petri temporizadas*. E é um intervalo de tempo $E : T \rightarrow [t_{min}, t_{max}]$, em que t_{min} e $t_{max} \in I\!N$ e $t_{max} \geq t_{min}$.

O modelo *TPN*, proposto por Merlin, engloba o modelo *TdPN*, pois é possível representar um tempo de disparo t através do intervalo $[t, t]$. As *Redes de Petri Temporais* foram usadas, principalmente, na modelagem de protocolos de comunicação [MB83, Mer79]. A Figura 2.3 mostra um protocolo comunicando dois processos modelados por uma *rede de Petri temporal*. O exemplo mostra como as *redes de Petri temporais* podem ser usadas para recuperar mensagens em um protocolo de comunicação. Inicialmente, uma ficha no lugar *P1* e uma uma ficha no lugar *P3* indicam respectivamente que o processo A está pronto para enviar uma mensagem e o processo B está pronto para receber uma mensagem. Quando o processo A envia uma mensagem, uma ficha é depositada nos lugares *P2* e *P4*. O significado de uma ficha no lugar *P4* é manter a informação a ser usada no caso da perda de uma mensagem. A recuperação da mensagem é acionada pelo disparo da transição *t3* que é determinado pelos intervalos de sensibilização a ela associados. Nesse exemplo, o limite inferior do intervalo de sensibilização, a, é maior do que o tempo estimado para o processo A

receber o reconhecimento do processo B, representado pelo disparo da transição *t5*. Os demais intervalos de sensibilização não foram representados na figura.

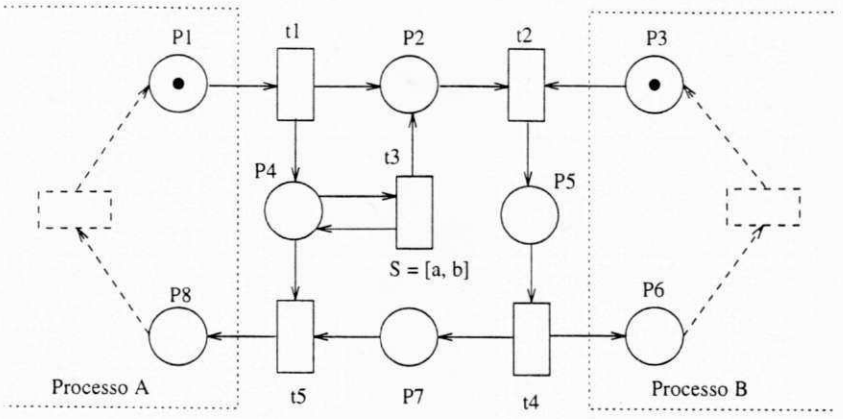


Figura 2.3: Rede de Petri temporal

Berthomieu [BD91] apresentou um método de análise para as *redes de Petri temporais*. O método gera o grafo de alcançabilidade de uma *TPN* pela parametrização do tempo de disparo para cada transição que dispara em uma marcação. Cada nodo nesse grafo representa classes de estado ao invés de estados, que são descritos por um conjunto de domínios de tempo de disparo que são computados pela solução de um sistema de inequações lineares. Uma abordagem similar foi usada por Srinivasan [SJ92] mas em um contexto diferente. A análise funcional e de desempenho de uma *TPN* foi investigada por Majmudar [MJ92].

As *Redes de Lugar-Transição Temporizadas (Timed Place-Transition Nets - TdPTNs)* diferem das *TdPNs* devido à localização da caracterização das restrições de tempo. Nas *TdPTNs*[Sif80], as restrições de tempo são representadas nos lugares da rede ao contrário das extensões anteriores. Neste caso, as fichas ao chegarem em um determinado lugar, permanecem indisponíveis por um período t, em que t é o tempo associado ao lugar, até se tornarem válidas para uma possível sensibilização de uma transição.

Definição 2.6 *Rede de lugar-transição temporizada*

Uma *rede de lugar-transição temporizada* é uma 6-tupla (P, T, I, O, τ, M_0) em que P, T, I, O e M_0 representam respectivamente um conjunto de lugares, um conjunto de transições, funções de entrada, funções de saída e marcação inicial. τ é uma função de tempo $\tau : T \rightarrow \{1, 2, ...\}$, mapeando cada lugar na rede nos números naturais.

Sifakis define algumas regras de transformação para obter o modelo de *rede de Petri temporizada* a partir do modelo de *rede de lugar-transição* foram apresentadas e concluiu-se que os modelos são equivalentes[Sif80]. A Figura 2.4 mostra como transformar um atraso de lugar (*TdPTN*) em um atraso de transição (*TdPN*). O lugar é decomposto em dois lugares e uma transição, e o tempo (atraso) que foi associado com o lugar agora é associado com a transição.

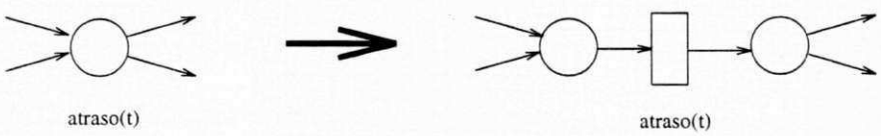


Figura 2.4: Transformação do atraso de lugar em atraso de transição

Stotts utilizou *TdPTN* para representar procedimentos concorrentes no modelo de um sistema de software[SP85]. Stotts definiu, ainda, um grafo de alcançabilidade modificado para o modelo *TdPTN* para suportar uma variedade de estudos de desempenho de procedimentos em tempo real[SP].

As três extensões apresentadas acima são consideradas como as extensões determinísticas básicas para o modelo de redes de Petri. Em alguns casos, como por exemplo para modelar sistemas complexos, é necessário aplicar mecanismos para reduzir sua complexidade. Portanto, as redes de Petri de alto nível podem ser usadas para reduzir a complexidade dos sistemas.

Algumas redes de Petri de alto nível foram estendidas para a caracterização do tempo. As fichas não são mais anônimas e carregam algumas restrições temporais. Na seqüência, duas extensões diferentes que aplicam redes de alto nível são apresentadas.

Ghezzi propôs as *Redes Básicas de Tempo (Time Basic Nets - TB nets)* que consistem, basicamente, na associação de um valor de tempo (*timestamp*) a cada ficha,

representando o tempo de disparo da transição que a gerou[GMMP89]. Além desse *timestamp* associado à ficha, associam-se ainda ações às transições, representando como os *timestamps* das fichas dos lugares sensibilizados determinam o valor de *timestamp* que é associado a cada ficha depositada nos lugares de saída. Na Figura 2.5, uma *TB net* é mostrada. O lugar *P2* representa a disponibilidade de um ítem perecível que está armazenado em uma loja. O lugar *P1* representa a compra de um ítem perecível e o lugar *P3* indica a disponibilidade de outros itens necessários para se fazer geléia. A transição *t1* modela a venda do ítem, a transição *t2* modela a ação de jogar um ítem podre no lixo, e a transição *t3* modela a feitura da geléia. Uma vez que o ítem em questão é um ítem perecível, a execução de uma ação que é representada pelo disparo de uma transição depende não apenas da presença das fichas nos lugares de entrada mas também de certas condições de tempo. Essas condições temporais são representadas por intervalos de tempo que podem depender dos valores que são carregados pelas fichas. Na Figura 2.5, os intervalos *i1*, *i2* e *i3* estão associados às transições *t1*, *t2* e *t3*, respectivamente. Por exemplo, o intervalo *t1* associado com a ação da venda de um ítem indica que este pode ser vendido se não está verde nem está estragado.

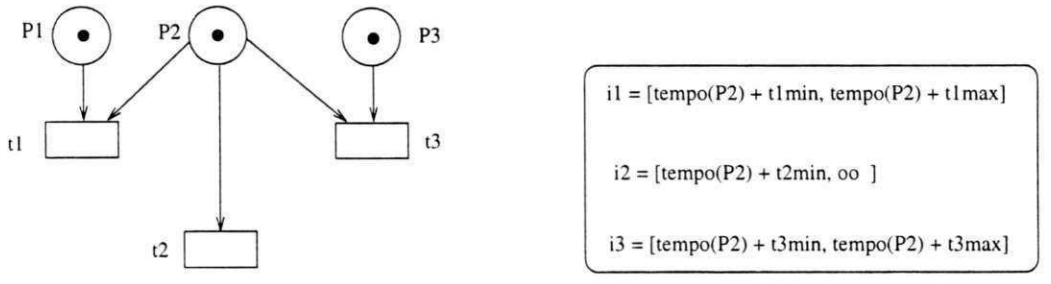


Figura 2.5: Exemplo de uma rede TB

Duas interpretações diferentes são definidas para esse modelo: a primeira, dita modelo forte, é similar ao modelo de Merlin e a segunda, dita modelo fraco, difere do modelo forte pois, nesse caso, a transição pode disparar depois de um dado tempo, mas não é forçada a isto.

2.3.2 Redes de Petri Estocásticas

Processos estocásticos são modelos matemáticos úteis para a descrição de fenômenos de natureza probabilística como uma função de um parâmetro que comumente tem o significado no tempo[AM89]. Entre a classe dos processos estocásticos, podem ser citados os processos Markovianos [Kle75]. Quando o espaço de estados de um processo Markoviano é enumerável, o processo é conhecido como cadeia de Markov. Os processos estocásticos e as cadeias de Markov são a base das extensões temporais estocásticas de redes de Petri.

O modelo de *rede de Petri estocástica (SPN)* foi proposto inicialmente por Molloy [Mol81] e Natkin [Nat80] no início dos anos 80. As *SPN*s utilizam uma abordagem estocástica ao invés da abordagem determinística utilizada nos modelos descritos anteriormente. Nas *redes de Petri estocásticas*, uma taxa de disparo distribuída exponencialmente é assinalada para cada transição [Mol82b]. Os modelos *SPN*s são isomórficos aos processos homogêneos de Markov então, combinando a simplicidade e facilidade de representação das redes de Petri com as bem conhecidas técnicas de análise de Markov, as *SPN*s são adequadas para a estimação de desempenho. A análise de uma *SPN* propicia a computação de índices de desempenho agregado. Entre eles, os mais comuns são:

1. a probabilidade de um evento que é definida através da marcação dos lugares,
2. o número médio de fichas em um lugar, e
3. a freqüência de disparo de uma transição.

Se uma SPN é um processo Markoviano, então é um processo *sem memória* e isto significa dizer que a probabilidade P de que um estado (ou marcação M) seja alcançado é função exclusiva do estado presente, conforme a expressão [Pap65]:

$$P\{M_j \leq M_j | M_{j-1} = M_{j-1}, \cdots, M_0 = M\} = P\{M_j <= M_j | M_{j-1} = M_{j-1}\} \quad (2.2)$$

Definição 2.7 *Rede de Petri Estocática (SPN)*

Uma *SPN* é uma sextupla dada por $SPN = (P, T; F, M_0, \Lambda)$,
onde:

$P, T; F, M_0$ são elementos das redes de Petri já conhecidos

Λ é o conjunto das taxas de disparo exponenciais negativas λ_i associadas às transições t_i.

Estas taxas podem depender da marcação sendo, então, escritas na forma: $\lambda_i(M_j)$.

O tempo médio de disparo é dado por $[\lambda_i(M_j)]^{-1}$. Em situações de conflito, o valor mínimo entre duas taxas λ_{i1} e λ_{i2} exponencialmente distribuídas é, também, uma variável aleatória exponencialmente distribuída com taxa $\lambda_{i+1} = \lambda_{i1} + \lambda_{i2}$. Então, o tempo médio de permanência em uma marcação M_k é uma variável aleatória exponencialmente distribuída, cuja média é dada por: $[\sum_{i:t_i \in E(M_j)} \lambda_i(M_j)]^{-1}$ onde E(M_j) é o conjunto de todas as transições habilitadas em M_j.

A probabilidade de uma transição disparar t_k é igual a probabilidade de M_{j+1} que é dada por

$$P\{t_k|M_j) = \frac{\lambda_k(M_j)}{\sum_{i:t_i \in E(M_j)} \lambda_i(M_j).} \tag{2.3}$$

com: $t_k \in E(M_j)$ e $M_{j+1}[t_k > M_j$.

Em [Pap65, AMBC84] podem ser encontradas estas, entre outras relações que definem os processos Markovianos e que são aplicáveis às redes de Petri que possuem este tipo de comportamento.

Regra de Disparos das SPN's

Existem duas regras básicas de disparo:

- a primeira estabelece que a cada marcação, disparará a transição habilitada (regra padrão) cujo atraso houver expirado primeiramente; e

- a segunda se baseia no fato de que sempre que uma transição se torna habilitada por uma marcação (última) ela dispara um temporizador associado com sua taxa exponencial, que poderá ser parado pelo disparo de uma transição conflitante e que deverá voltar a contar quando a mesma for novamente habilitada; esta será disparada somente quando seu temporizador chegar a 0 (zero).

Diferentes variações para o modelo *SPN* foram propostas, as quais utilizam uma função de densidade de probabilidade mais geral, tal como: redes de Petri estocásticas generalizadas (*GSPN*) [AMBC84], redes de Petri estocásticas estendidas (*Extended Stochastic Petri Nets - ESPN*) [DTGN84] e redes de Petri estocásticas regenerativas(*Regenerative Stochastic Petri Nets*) [HS86]. A extensão de *SPN* mais conhecida e usada é a *GSPN* [AM89, AMBC84] proposta para diminuir a complexidade de análise do modelo *SPN*. O modelo *GSPN* tem dois tipos de transições: as *transições temporizadas*, as quais são associadas atrasos aleatórios como nas *SPN*s e as *transições imediatas*, as quais têm prioridade sobre as outras transições e disparam instantaneamente. Além do mais, as *SPN*s foram usadas com redes de Petri de alto nível para reduzir a complexidade gráfica do modelo *SPN*. As definições dos modelos *SPN de alto nível* e *SPN colorida* foram introduzidas respectivamente em [ML87] e [Zen85].

2.3.3 Redes de Petri Estocásticas Generalizadas

Para solucionar problemas surgidos na modelagem com redes de Petri estocásticas, tais como a complexidade e a velocidade de ciclo da rede, foram propostas as Redes de Petri Estocásticas Generalizadas (*Generalized Stochastic Petri Nets, GSPN's*), que empregam dois tipos de transições: as transições de atraso, igualmente às redes estocásticas e transições imediatas; e acrescenta [ZD93, Mur89, Sil95, CLBJM92]:

- um conjunto de arcos inibidores;

- um conjunto de prioridades associados a transições imediatas; e
- conjunto de pesos para a computação da probabilidade de cada transição, se é do tipo imediata.

2.3.4 Redes de Petri com Temporização Nebulosa

As redes de Petri com temporização nebulosa (*Fuzzy Timed Petri Nets, FTPN*) [dF94] são uma extensão temporal de redes de Petri que se propõe a auxiliar na modelagem de sistemas a tempo real, quanto na avaliação de desempenho de sistemas complexos, combinando as propriedades das redes temporais determinísticas e das estocásticas.

O conceito *nebulosa* está associado a incerteza dos eventos. Por exemplo: a) o Natal é, uma data festiva que ocorre *precisamente* aos 25 de Dezembro de cada ano em nosso calendário; b) se uma outra festa vai ocorrer em Dezembro, esta é uma data *imprecisa*, pois a possibilidade da mesma ocorrer em cada dia deste mês (intervalo) é a mesma, ou 1/31; contudo, se há maiores chances (definidas por funções de densidade de probabilidade) desta ocorrer em um intervalo que contém a data correta, então, esta festa possui uma data *nebulosa*.

No modelo FTPN, as fichas carregam uma função nebulosa de tempo que determina a probabilidade de sua existência em determinado instante a partir de um determinado momento. Em outras palavras, a ficha tem uma *vida útil* associada a cada lugar. Transcorrido este período, esta ficha não mais contribuirá para o disparo de transições.

Dois intervalos nebulosos, ainda, estão associados à cada transição: E, ou intervalo de sensibilização e D, intervalo de disparo. Uma transição permanece sensibilizada por um período E antes de disparar e começa a disparar por um período D. Após o disparo, as fichas dos lugares de entrada são removidas e uma ficha que carrega o valor da função nebulosa de tempo é depositada em cada lugar de saída da transiçã o. Em [dF94] podem ser encontradas estas definições, entre outras.

Para maiores detalhes sobre a teoria e aplicação de FTPN, bem como a definição formal, o leitor interessado pode consultar [dF94, dFPC94, PdFC94, dFPM93, PdFM93,

dFP94, dFPC95, dFP95b, dFP95c, PdF95, dFP95a, dFP96, PPC96, PdF97, dFP97].

2.4 Métodos de Análise

Nesta seção são introduzidas de modo informal técnicas e métodos de análise para sistemas de redes de Petri. A maneira mais direta de análise é a simulação, a qual em muitos aspectos é bastante similar ao teste e execução de programas[Jen92]. Simulação é extremamente útil para o entendimento e depuração de um sistema. Este aspecto é particularmente relevante durante a fase de concepção e validação prematura de um grande sistema complexo. Entretanto, por meios de simulação é impossível obter-se uma completa prova ou verificação das propriedades dinâmicas de um sistema, devido à complexidade espacial e temporal. Portanto, é muito importante vislumbrar métodos formais de análise (i.e., métodos que são baseados em técnicas de prova matemática). Duas classes de propriedades podem ser verificadas ou analisadas para sistemas de redes, propriedades estáticas e propriedades dinâmicas.

2.4.1 Propriedades Estáticas

Propriedades estáticas ou estruturais podem ser derivadas da definição da rede em questão – sem considerar as seqüências de disparo ou ocorrência das transições. Propriedades estáticas são principalmente importantes para caracterizar redes com alguma tipo de propriedade especial.

Para redes de Petri P/T, é possível definir diversas propriedades estáticas que possam ser verificadas. Murata [Mur89] apresenta uma excelente introdução a estas propriedade estáticas. Entre as principais citam-se: vivacidade estrutural (*structral liveness*), controlabilidade, repetitividade, e consistência.

2.4.2 Propriedades Dinâmicas

Propriedades dinâmicas ou comportamentais caracterizam o comportamento de redes individuais, por exemplo, se é possível ou não alcançar uma marcação na qual nenhuma transição estaria habilitada. A verificação de propriedades dinâmicas pode ser extremamente difícil quando nenhum método formal é disponível, isto pois o número de possíveis combinações de casos a serem simulados pode ser proibitivo. Portanto, é muito importante definirem-se métodos normais para a análise dos vários tipos de sistemas de redes.

Exemplos de propriedades comportamentais são: alcançabilidade, limitabilidade, vivacidade, reversibilidade, estados originais (*home states*), cobertura (*coverability*), persistência, distância sincrônica (*synchronic distance*), e justiça. Todas estas propriedades dinâmicas são discutidas em [Mur89]. Métodos de análise para sistemas de rede podem ser classificados nos seguintes três grupos:

1. invariantes de lugar e transição
2. técnicas de redução e decomposição, e
3. método da árvore de alcaçabilidade (cobertura).

No que segue-se discute-se informalmente as principais técnicas de análise para redes de Petri.

2.4.3 Invariantes de Lugar e Transição

O método do invariante é conhecido por pelo menos duas vantagens: primeiramente, a análise pode ser executada em sub-redes locais ignorando-se como o sistema global se comporta; segundo, este método é aplicável para grande número de tipos redes de Petri.

A ídeia básica é analisar o comportamento dinâmico de um sistema através de equações lineares. Entretanto, como enfatizado em [Mur89], a solução destas equações

é um tanto quanto limitada. Isto deve-se a característica não determinística do comportamento de modelos de sistemas de redes, e devido à restrição de que as soluções devem pertencer ao conjunto dos inteiros não negativos, no caso de redes de baixo-nível. A seguir são introduzidos somente os aspectos conceituais da análise de invariantes. Para detalhes matemáticos refirir-se a [Mur89]. No caso da análise de invariantes definem-se equações de estado para o sistema. Uma marcação M_k é escrita como um vetor coluna m. A $j^{ésima}$ entrada de M_k denota o número de fichas no lugar j imediatamente após a $k^{ésima}$ seqüência de disparo. A $k^{ésima}$ seqüência de disparo pode ser vista como um vetor de controle u_k. O vetor de controle u_k é um vetor coluna $n \times 1$ com $n-1$ zeros e um a entrada não nula, um 1 na $j^{ésima}$ posição indica que a transição j dispara no $k^{ésimo}$ disparo. Uma vez que a $i^{ésima}$ linha na matriz de incidência C denota a mudança de uma marcação como resultado do disparo da transição i, pode-se escrever a seguinte equação de estado:

$$M_k = M_{k-1} + C^T u_k, \quad k = 1, 2, \cdots. \tag{2.4}$$

Como detalhado em [Mur89], duas equações podem ser derivadas da Equação 2.4. Uma é denominada de *P-invariantes* para invariantes de lugar, e a segunda é denominada *T-invariantes* para invariantes de transição.

Definição 2.8 *Vetor de disparo*

Um vetor de disparo x é um vetor coluna $n \times 1$ de inteiros não negativos.

onde a $i^{ésima}$ entrada de x denota o número de vezes que a transição i deve disparar para transformar uma marcação M_0 para M_d.

Definição 2.9 *P-invariantes*

Dado que C é a matriz de incidência para um sistema de redes, e dado que x é um vetor de disparo, um P-invariante é uma solução inteira para o sistema de equações homogêneas:

$$C^T x = 0 \tag{2.5}$$

Definição 2.10 *T-invariantes*

Dado que C é a matriz de incidência para um sistema de redes, e dado que x é um vetor de disparo, um T-invariante é uma solução inteira para o sistema de equações homogêneas:

$$Cx = 0 \tag{2.6}$$

Informalmente um P-invariante corresponde a uma seqüência de disparo que não altera a soma das fichas nos lugares, e um T-invariante corresponde a uma seqüência de disparo que não altera a marcação da rede. A análise através de invariantes é um método bastante poderoso tanto para executar análise estrutural, como comportamental [MV91, Mur89].

2.4.4 Redução

Para simplificar a análise de grandes sistemas de redes, é freqüentemente necessário reduzir o modelo para um mais simples. Deve-se notar que esta redução deve garantir a preservação de propriedades. Existem diversas diferentes técnicas para transformar ou reduzir um sistema de redes.

A idéia básica por trás destas transformações está em escolher um ou mais tipos de propriedades a investigar (p.e. vivacidade ou limitabilidade). Então, define-se um conjunto de regras para redução, que quando aplicadas podem simplificar o sistema de rede – sem alterar as propriedades que estão sendo investigadas. Usualmente, as regras

são locais, no sentido de que cada uma delas permite que uma sub-rede seja substituída por uma outra mais simples. Está além do escopo desta dissertação discutir todas as regras de redução disponíveis para as redes de Petri mais utilizadas, o leitor pode referir-se a [LF85, Mur89].

2.4.5 Árvore Alcançabilidade/Cobertura

Para a descrição da árvore de alcançabilidade/cobertura toma-se a definição de Murata [Mur89]. Dada um sistema de rede $\mathcal{N}$, a partir da marcação inicial M_0, pode-se obter tantas marcações quantas forem as transições habilitadas. A partir de cada nova marcação pode-se então alcançar novas marcações. Este processo resulta em uma árvore de marcações. Para esta árvore, nós representam marcações geradas a partir de M_0 (a raiz da árvore) e seus sucessores, e cada arco da árvore representa o disparo de uma transição, o qual transforma uma marcação em outra. Entretanto, a representação em árvore crescerá indefinidamente no caso da rede não ser limitada. De modo a manter a árvore finita, introduz-se um símbolo especial ω, o qual pode ser considerado como *infinito*. Para redes de baixo-nível este símbolo apresenta a propriedade de que para cada inteiro não negativo n, $\omega > n$, $\omega \pm n$ e $\omega \geq \omega$. A árvore de cobertura para um sistema de rede $\mathcal{N}$ e uma marcação inicial M_0, pode ser construída aplicando-se o algoritmo apresentado em [Mur89].

No caso de um sistema limitado, a árvore de cobertura é denominada árvore de alcançabilidade, uma vez que esta contém todas as possíveis marcações alcançáveis. Neste caso todos os problemas de análise podem ser resolvidos pela análise da árvore de alcançabilidade. A desvantagem deste método reside no fato de ser um método exaustivo. Entretanto, de modo geral, os problemas de alcançabilidade e vivacidade não podem ser resolvidos somente com a árvore de cobertura, isto devido a introdução do símbolo ω.

Para um sistema de rede, o grafo de cobertura é definido pelo grafo direto rotulado $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Onde o conjunto de nós $\mathcal{V}$, é o conjunto de todos os nós rotulados distintos

na árvore de cobertura, e o conjunto de arcos $\mathcal{E}$, é o conjunto de arcos rotulados com uma única transição t_k, representando todos os possíveis disparos únicos de transições, de forma que $M_i[t_k\rangle M_j$, onde M_i e M_j estão em $\mathcal{V}$. Por exemplo, para a rede mostrada na Figura 2.6 a árvore de cobertura e o grafo de cobertura são mostrados na Figura 2.7 (a) e (b), respectivamente.

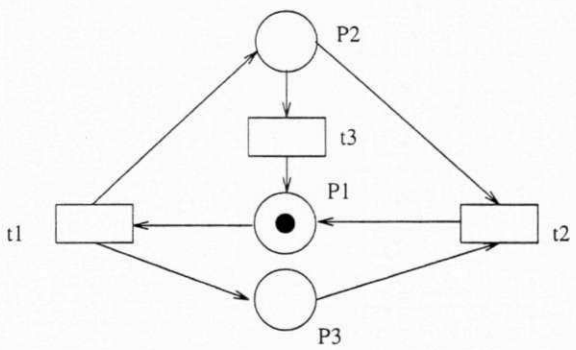


Figura 2.6: Uma rede P/T

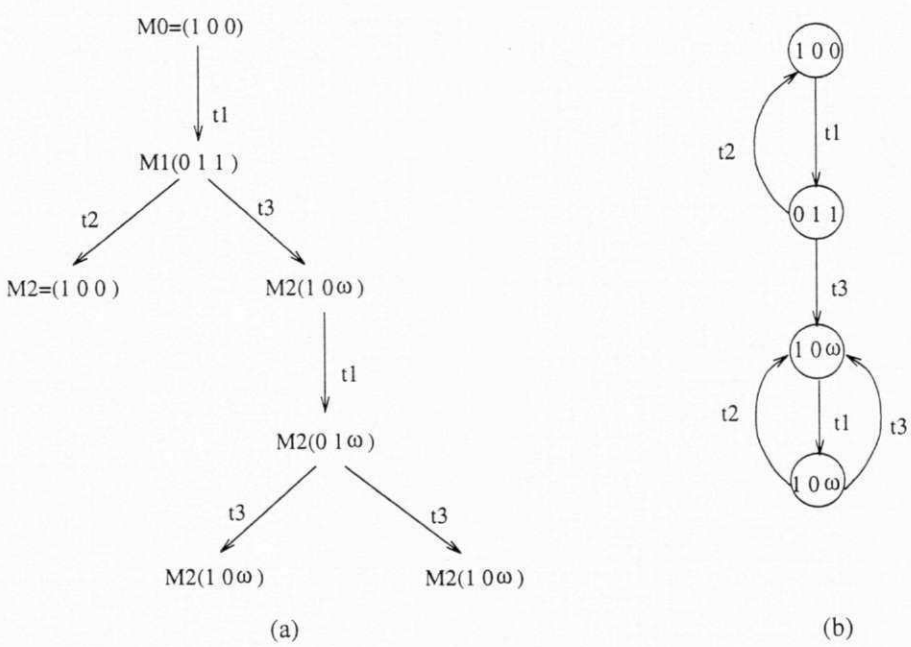


Figura 2.7: (a) Árvore de cobertura e (b) Grafo de cobertura para a rede da exemplo

2.5 Modelagem de Sistemas Utilizando Sistemas de Rede

No caso de sistemas complexos algumas metodologias e técnicas estão disponíveis para prover a possibilidade de construir o modelo do sistema de forma modular e hierárquica. Valette [Val79] introduz uma metodologia baseada em refinamentos sucessivos para ser aplicada na construção de redes de baixo-nível. Ele introduziu o conceito de *blocos bem formados*. A idéia básica é definir redes bem formadas com o comportamento desejado, p.e. vivacidade (*liveness*) ou limitabilidade (*boundness*), e então usá-las para construir uma rede mais complexa. Se a rede é construída usando este tipo de blocos, as propriedades desejadas para o sistema podem ser preservadas, e em conseqüência a rede complexa resultante não precisa ser analisada. Murata [Mur89] generalizou a metodologia introduzida por Vallete de modo a incluir também abstração. Infelizmente, esta abordagem pode somente ser aplicada para redes ou blocos com um par de transições para entrada e saída. Valette [ABD+84, VCD85] introduziu outra abordagem que possibilitava a fusão de lugares e transições, de modo que algumas propriedes possam ser preservadas. Para detalhes o leitor pode referir-se a [LF85, PBP+91, PBdFP91].

Outras construções modulares [SM91] baseiam-se na idéia geral de construir uma rede modelando um sistema de forma modular, e deduzir propriedades do sistema somente pela análise de seus componentes menores. A razão para introduzir este tipo de abordagem é que de um modo geral, a composição de sub-redes genéricas não possibilita a preservação de determinadas propriedades (especialmente vivacidade) ao nível da rede global. Diferentemente das abordagens previamente introduzidas, esta não introduz restrições às redes a serem compostas. Por exemplo, a restrição de haver somente um par de transições de entrada e saída, é suficiente para estruturalmente restringir um meio (uma terceira rede), a qual é usada para compor duas outras redes.

Capítulo 3

A Ferramenta ManNet

3.1 Introdução

A ferramenta ManNet é a continuidade do desenvolvimento de um *software* que teve início em uma disciplinas dos cursos de mestrado em Engenharia Elétrica e em Informática da Universidade Federal da Paraíba, UFPb, Campus II. O projeto inicial foi denominado *PetriLyser*, e consistiu no desenvolvimento de um programa em linguagem C que auxiliasse no estudo e análise das redes de Petri.

A ferramenta em sua primeira versão possuia estrutura de procedimentos e era voltada para o ambiente *MS DOS*, embora pudesse ser executada sobre os *Windows 3.1, 3.11 e 95.* Era constituída por:

- um editor de redes de Petri com interface com o usuário baseada nos estilos de *diálogo* e árvores de sub-menus;
- rotinas que geram a matriz de incidência e a *árvore e o grafo de cobertura* [Mur89] de uma rede de Petri, e;
- rotinas que realizam a remoção de *arcos inibidores* e a transformação de uma rede *de capacidade finita* em uma rede *de capacidade infinita*, se for o caso.

Porteriormente foram incorporadas:

- técnicas de programação que permitem o gerenciamento dos ciclos de máquina da *CPU* do *PC*, baseada na técnica de escalonamento *round robin não preemptiva*; e

- tempos de duração e estados às tarefas, baseados nas técnicas de Zhou e DiCesare [ZD93]).

A utilização de tais técnicas (incluindo tempos de duração às tarefas) permitiu a simulação de redes de Petri e a interação com o usuário em tempo real. Suas aplicações foram inicialmente voltadas para as *células de manufatura.* À partir deste estágio a ferramenta ganhou a atual denominação: ferramenta ManNet[LdFP96]. O protótipo abriu, então, novas perspectivas, tais como a aplicação com a maioria dos sistemas complexos, o estudo temporal das redes de Petri e a análise de desempenho, através da simulação e o desenvolvimento de nova interface com o usuário, facilitada pelas bibliotecas e pelo compilador Borland C++ Buider.

A ferramenta ManNet é, portanto, um programa em desenvolvimento onde se busca incorporar a ferramenta Petrilyser recursos atualizados de edição de redes de Petri, de interação com o usuário e de análise de desempenho.

3.2 A Ferramenta Petrilyser

A Petrilyser é uma ferramenta que foi inicialmente desenvolvida em linguagem C para o ambiente DOS e que logo evoluiu para os ambientes *Windows 3.xx e 95* para simular sistemas modelados em redes de Petri que realizam as operações de um supervisor ou controlador em estações de trabalho e sistemas responsáveis pela execução de tarefas em Sistemas Flexíveis de Manufatura *FMS's* modelados em redes de Petri Temporais. A ManNet consiste, então, em uma proposta de ferramenta com os mesmos propósitos, voltada para ambientes *Windows 95 e NT*, porém com maiores recursos de simulação, para atender a diversos modelos de sistemas complexos.

A ferramenta Petrilyser utiliza técnicas de modelagem e de escalonamento as quais são descritas com mais detalhes a seguir. Constituem esta ferramenta:

- um editor de redes de Petri;
- rotinas que auxiliam na análise da estrutura e do comportamento dinâmico das redes de Petri;
- as funções de um escalonador com politica round robin, não preemptivo;
- as funções de um "jogador" que sorteia a ativação das tarefa e a duração, quando necessárias; e
- as funções de um executivo em tempo real.

3.2.1 O Editor de Redes de Petri

Baseado na edição de texto e menus de opções, o editor permite editar, ler e salvar em disco arquivos (de uma rede de Petri).

Os Arquivos de uma Rede de Petri

Os arquivos das redes de Petri são três: o de lugares, o de transições e o de arcos, cujas extensões propostas são *.lug, *.tra e *.arc, respectivamente.

Estes elementos são descritos por estruturas *struct*. Cada conjunto de estruturas consiste em um banco de dados que dá suporte ao arquivo correspondente. Cada arquivo é finalizado por um bloco *(struct)* cujo primeiro caracter é *$ (cifrão)*, depois seguido do caracter de controle EOF.

Estruturas e Versões

A seguir são apresentadas as estruturas dos lugares, transições, arcos e dos descritores das tarefas, e informações sobre os campos das mesmas, as quais encontram-se, também, presentes nos arquivos fonte ou versões:

- MpnaD.c: versão da ferramenta Petrilyser para o ambiente DOS;
- MpnaW1.c: primeira versão para os ambientes *Windows 3.xx e 95*[1];
- MpnaW2.c: segunda versão para o ambiente *Windows 3.xx*[2].

```
typedef struct {
char nome[30], tipo;
unsigned int capac, linha, marca;} llugar;

typedef struct {
char nome[30]; } ttransi;

typedef struct {
char partida, tipo;
unsigned int origem, destino, peso;} aarcos;

typedef struct {
unsigned int tipo, transi, pai;
int mk[MMAX];} mmarca;

typedef struct {
char estado, antigo;
unsigned int dura, cont, tipo, tra_ini,
```

[1]Versão semelhante à apresentada para o ambiente DOS, não testada no ambiente *Windows NT*

[2]Nesta versão são permitidas alterações nos relógios base "RTC" e no que faz o escalonamento e a marcação da rede, ou "jogador". Esta versão também pode ser utilizada no ambiente *Windows 95*. Neste caso, durante o escalonamento/simulação, não é recomendado o término da ferramenta através do botão *CLOSE, ou X* devido à violação do gerenciamento do controle de interrupções pelo *Windows 95*, o que poderá inibir alguns procedimentos no retorno para o mesmo, tais como o uso do teclado, atualização de janelas, entre outros problemas até então detectados que, dependendo do instante do término da ferramenta podem "travar" parcial ou totalmente o microcomputador.

```
tra_fini, ordem, ordem_i, ordem_f;
void (*tar)(); } bct;
```

Se um arquivo gerado por esta ferramenta não é interpretado por um outro editor, a solucao prática de interfacear estas ferramentas está no desenvolvimento de um programa ou rotina que traduza este arquivo no outro (e vice-versa) de forma transparente para o usuario.

Quanto aos arquivos de lugares (*nome.lug*) e transições (*nome.tra*), o número de ordem (ou índice) é fornecido pelo programa. O gerenciamento do número de ordem tem por objetivo reduzir a dimensão e a esparcidade da matriz de incidência, facilitar a etapa de eliminação de lugares de capacidade finita e de arcos inibidores e evitar que os nomes dos lugares e transições se limitem a Pnn e Tnn. Seu ponto negativo está na necessidade do usuário tomar nota durante a edição e usá-lo quando necessário [3]. Tal tarefa é menos árdua quando os lugares e transições são editados na ordem crescente e contínua (p1, p2, ..., t1, t2, ..., e assim por adiante).

Quanto aos arquivos de arcos, o número de ordem (ou índice) de origem: é o número de ordem do lugar ou da transição de origem, enquanto que o número de destino é o número de ordem da transição ou do lugar de destino; partida pode ser "l" (quando o arco tem origem em um lugar) ou "t" (se tem origem em uma transição); tipo pode ser "i" (se é um arco do tipo inibidor) ou "n" (se normal), caso contrário[4].

3.2.2 A Análise da Estrutura da Rede de Petri

Para auxiliar na análise da estrutura das redes de Petri a ferramenta Petrilyser permite gerar e apresentar a matriz de incidência, os vetores marcação inicial e final[5], e a árvore

[3] A ser solucionado, com o reconhecimento de cadeia de caracteres em futuras versões.

[4] Na etapa de conversão para arcos normais e lugares de capacidade infinita há a criação de arcos e lugares extras que são identificados pela ferramenta como do tipo "x". Esta operação resulta na transformação gerando uma nova rede de Petri, porém com os mesmos grafos e estados da rede originária. Esta nova rede poderá ser armazenada, se desejado, com novo nome.

[5] Marcação ou estado da árvore de alcançabilidade

O Executivo

O executivo "marca" a rede de Petri e procede a execução das tarefas em progresso a cada *TIC* do relógio de tempo real. É sua função, também, gerenciar a "suspensão" ou interrupção de uma tarefa realizada pelo usuário (pelo teclado).

As redes TPN e SPN

Na opção "temporal" a duração de cada tarefa é dada pela constante descrita no seu bloco de controle ou *bct*. Em outras palavras, é dada por este valor vezes a constante do relógio "RTC", vezes o período da interrupção do mesmo.

Na opção "estocástica" a duração é dada por um valor sorteado uniformemente que está entre 0 (zero) e a duração máxima prevista no descritor, vezes o divisor da frequência do "RTC", vezes o período do mesmo.

As Tarefas

Uma tarefa é modelada por um lugar e sua "execução" é simulada pela chamada de uma rotina a cada *TIC*, na qual é decrementado um contador associado (descrito pelo *bct*). Sua ativação consiste na carga do contador com a frequência no *bct* presente e seu término após transcorrido o número de *TIC's* devidos. Se um TIC interromper durante a execução de uma tarefa, esta será indicada e terá sua execução retomada ao término desta interrupção.

3.2.4 A Rede de Petri Exemplo de Molloy

As Figuras 3.1, 3.2, 3.3, 3.4 e 3.5 apresentam a interface com usuário da ferramenta Petrilyser, a qual é baseada no diálogo e na árvore de menus quando na edição da rede de Petri apresentada pela Figura 3.6 e por Molloy[Mol82a]. Esta rede de Petri possui pequena estrutura, porém reune várias propriedades e será utilizada como referência neste trabalho, a título de comparação/validação dos resultados fornecidos

pela Petrilyser e que, aqui será chamada de *Exemplo de Molloy.*

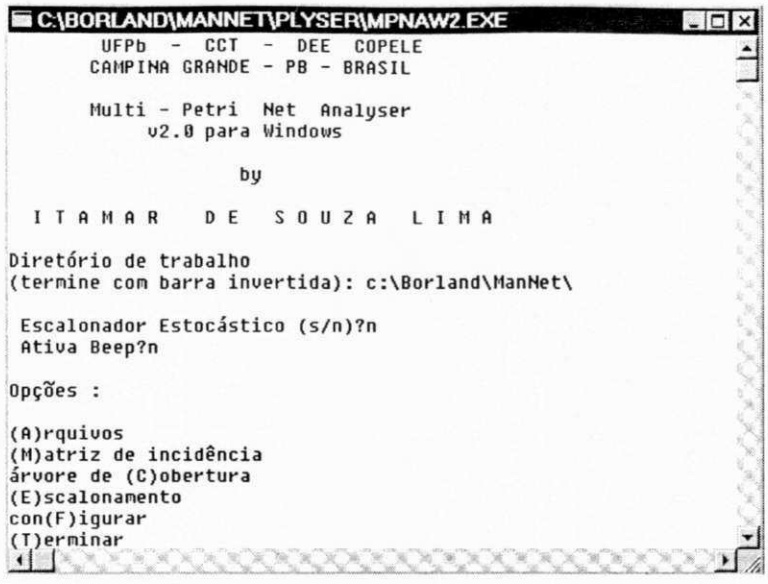


Figura 3.1: Iniciando a Ferramenta Petrilyzer

O Grafo de Alcançabilidade e a Árvore de Cobertura

Na Figura 3.7 apresentamos passos na montagem do grafo de alcançabilidade da rede de Molloy, que é possível com a obtenção de disparos consecutivos de transições habilitadas, considerando a marcação inicialmente estabelecida como (1,0,0,0,0). Nesta destacamos:

- a marcação encontrada com o disparo de t1 ou (0,1,1,0,0);
- a marcação encontrada com o disparo de t2 ou (0,0,1,1,0);
- um erro de operação, quando digitado "5" quando esperado "s" ou "n";
- um erro de montagem, quando comandado o disparo da transição t5, não estando esta habilitada; e
- a marcação encontrada com o disparo de t3 ou (0,0,0,1,1).

Na Figura 3.8 apresentamos a árvore de cobertura da Rede de Molloy, onde:

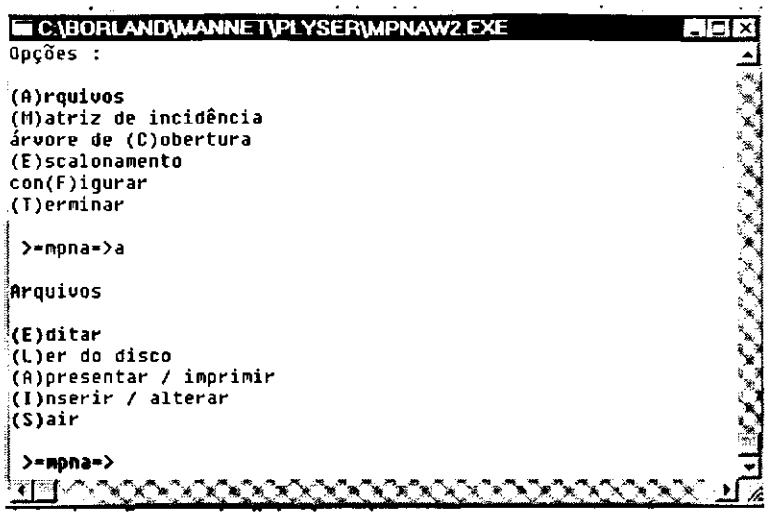


Figura 3.2: Menu Arquivos

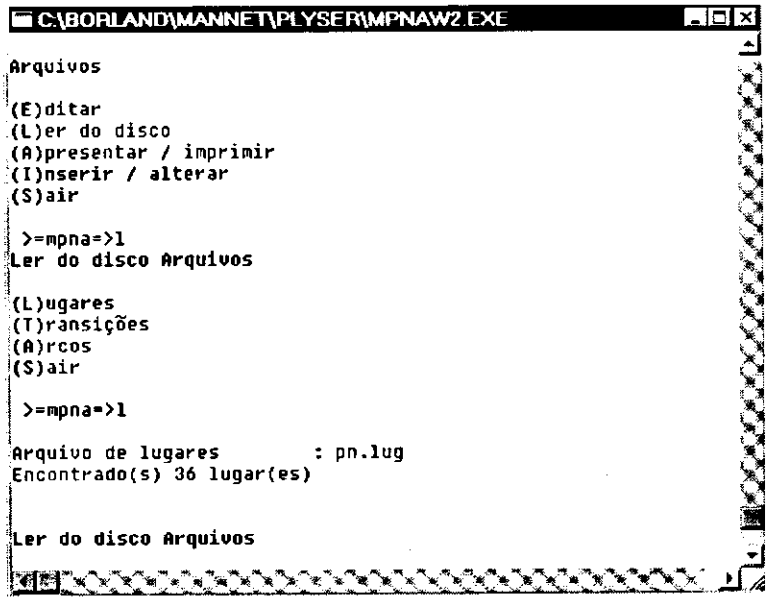


Figura 3.3: Lendo Arquivo de Lugares

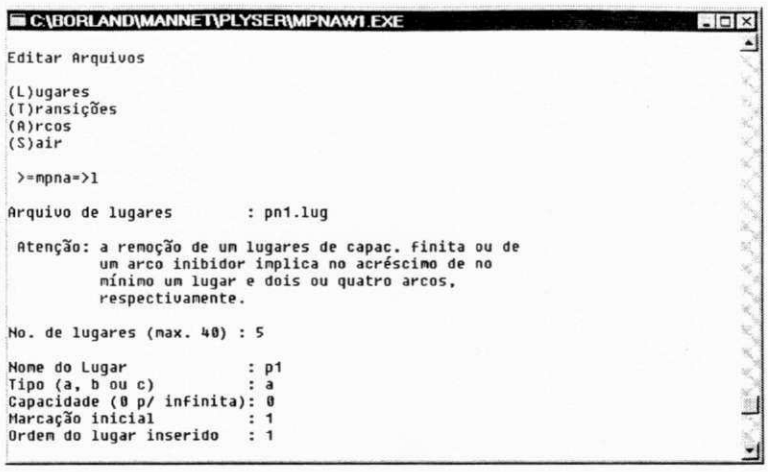


Figura 3.4: Editando Arquivo de Lugares

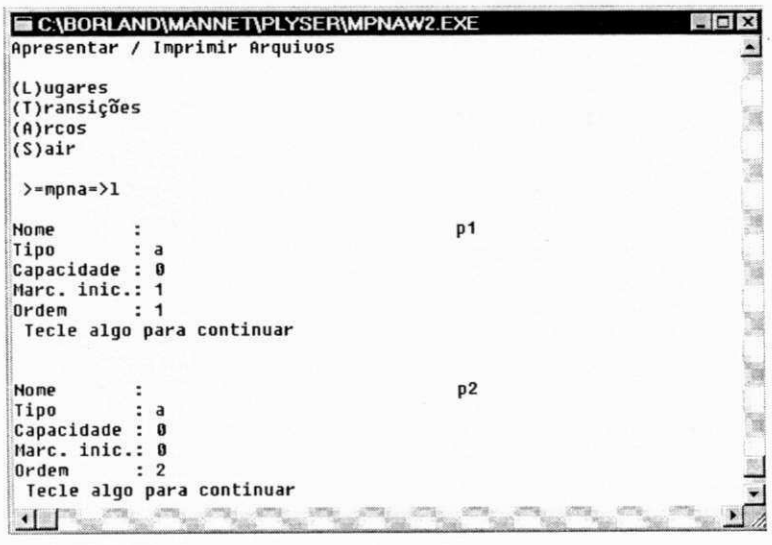


Figura 3.5: Apresentando Arquivo de Lugares

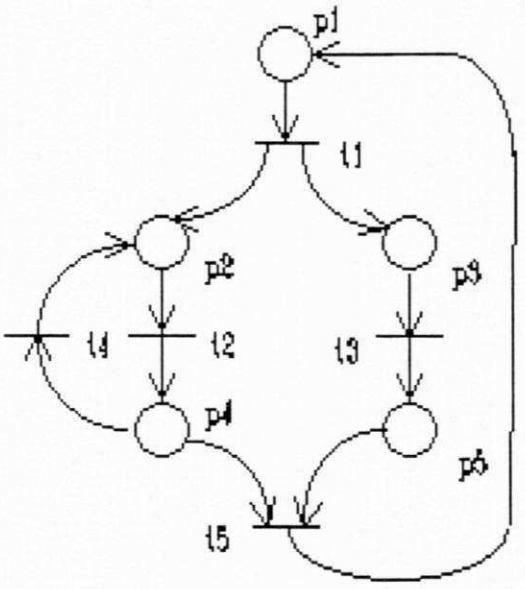


Figura 3.6: A Rede de Petri Exemplo de Molloy

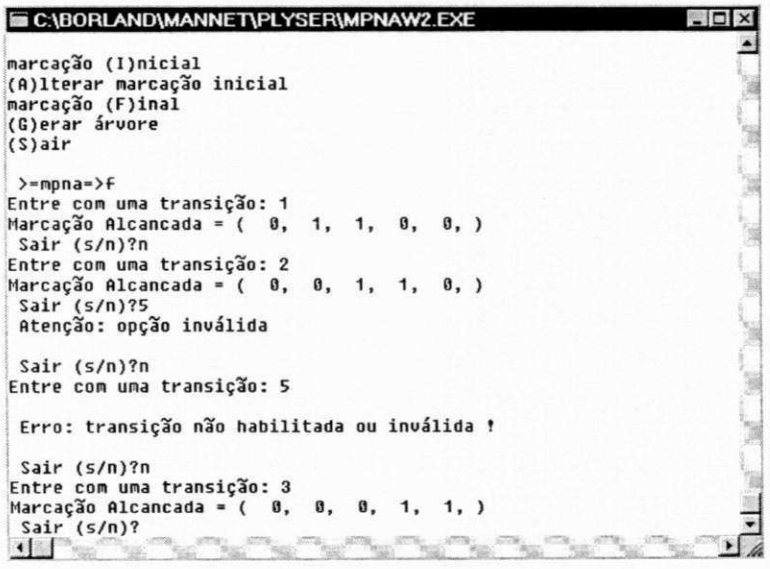


Figura 3.7: Grafo de Alcançabilidade da Rede de Molloy

- Ord indica a ordem ou índice de criação e armazenamento da marcação;
- o Tipo é o tipo propriamente dito, da marcação conforme definido por Murata[Mur89];
- Pai é a ordem da marcação antecessora;
- Trans é a ordem da transição cujo disparo a levou a este estado; e
- Marcação, o vetor marcação propriamente dito.

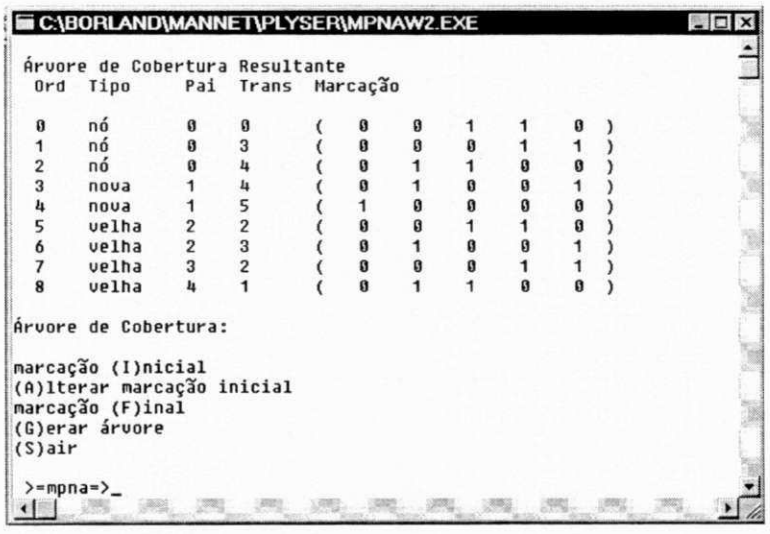


Figura 3.8: Árvore de Cobertura da Rede de Molloy

A Simulação

A simulação da rede de Molloy é apresentada pela Figura 3.9. Esta janela é dividida em três quadros: o das tarefas, o dos recursos e o dos comandos. No caso da Rede de Molloy, todos os lugares modelam tarefas, por ser o único elemento das redes de Petri, previsto pela Petrilyser, que possui propriedades temporais. A inexistência de recursos, ou elementos independentes do tempo, impede a descrição do quadro dos recursos. Contudo, é permitido apresentar algumas de suas propriedades, tais como:

- os lugares p1, p2, p3, p4 e p5 quando em atividade, ou "Rodando", possuem tempos de duração de até 40, 50, 25, 50 e 15 ut's, ou unidades de tempo, respectivamente;
- no instante da captura da janela, durante a simulação, apenas o lugar p1 está marcado, ou seja, é a única tarefa em atividade;
- a título de informação da estrutura da rede, o quadro dos lugares apresenta os nomes dados às atividades, a ordem dos lugares anteriores e posteriores de cada um dos lugares, indicados por Li e Lf, respectivamente; e
- a ordem das transições que as inicializam e finalizam, indicados por Ti e Tf, respectivamente.

Em uma rede de Petri maior, tal como a apresentada na Figura 3.11 que modela a célula de manufatura da Figura 3.10, é possível destacarmos que na Figura 3.12:

- no quadro das tarefas estão presentes as informações referentes a estas;
- no quadro dos recursos, a marcação corrente destes; e
- no quadro dos comandos, os acessíveis durante a simulação.

Figura 3.9: Simulação da Rede de Molloy

Mais detalhadamente, no primeiro quadro:

- Ct ou contador para baixo, que apresenta o tempo que falta para o término da tarefa se esta se encontra em operação ou "Rodando";

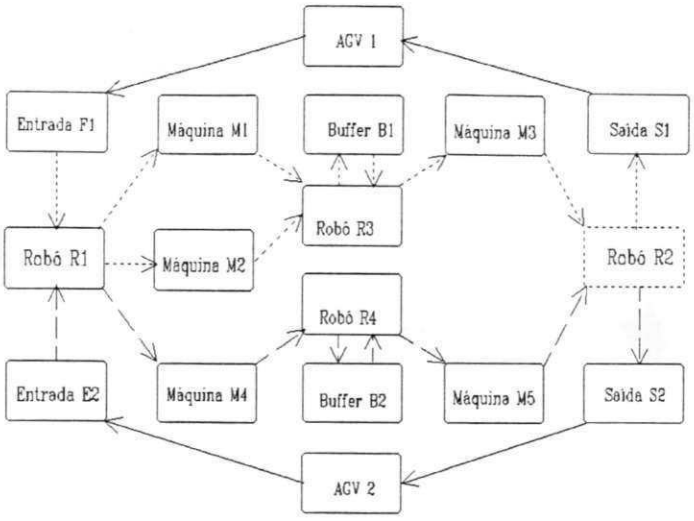


Figura 3.10: Exemplo de Célula de Manufatura

- Du, que consiste no tempo (número de ciclos do RTC) máximo de duração da tarefa;
- Estado pode ser um em quatro tipos[7] "Pronto", "Rodando", "Esperando" e "Dormindo";
- Atividade ou nome dado ao lugar associado à tarefa;
- Mk ou marcação corrente do lugar associado;
- S ou coluna onde o símbolo < indica a atividade sorteada a cada instante;
- Li ou coluna onde se informa a tarefa anterior a corrente;
- Or ou coluna onde se indica a ordem interna da tarefa;
- Lf ou coluna onde se informa a tarefa posterior a corrente;
- Ti ou coluna onde se indica a transição que inicializa a tarefa; e

[7]Na versão atual, ou ManNet, estes estados correspondem ao *Pronta, Ativa, Espera e Inativa*, respectivamente, os quais são descritos a seguir.

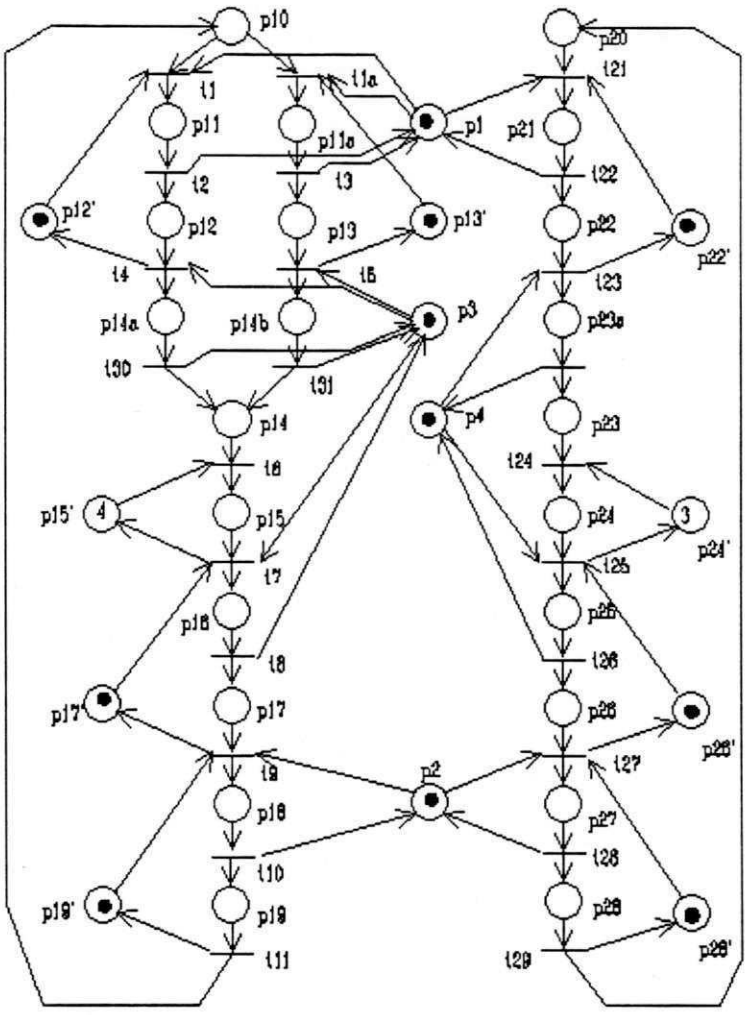


Figura 3.11: Modelo da Célula de Manufatura em rede de Petri

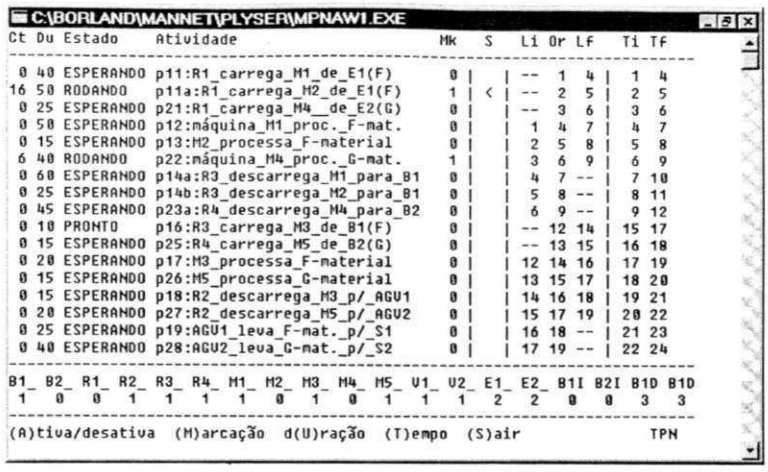


Figura 3.12: Janela de Simulação da ferramenta Petrilyzer - versão DOS/Win3.xx

- Tf ou coluna onde se indica a transição que finaliza a tarefa.

Nas colunas Li e Lf quando houver os símbolos $--$ e $**$, indicam que (anterior ou posteriormente à tarefa) "não existe" e "existe mais de uma" tarefa, respectivamente.

No segundo quadro encontram-se as iniciais dos lugares que modelam recursos fixos (máquinas, veículos, robôs, entre outros) e variáveis (p.e. materiais, paletes e ferramentas) e, abaixo, suas marcações. Um recurso fixo marcado (valor = 1) implica na sua disponibilidade.

No terceiro quadro encontra-se um menu de comandos e uma sigla que informa o comportamento temporal da rede (SPN = Rede de Petri Estocástica ou TPN = Rede de Petri Temporal). Os comandos trocam diálogos com o usuário no segundo quadro:

- Ativa/desativa: permite ativar e desativar uma tarefa ("Dormindo", ou "removê-la" da rede) – Na ativação o estado anterior da tarefa é restaurado;

- Marcação: permite alterar a marcação da rede (realizado em nova janela);

- Duração: permite alterar a duração das tarefas;

- Tempo: permite alternar o comportamento da rede de temporal para estocástico e vice-versa, que é indicado pelas siglas SPN e TPN;

- RTC: permite alterar a frequência do TIC; e
- Jogador: permite alterar a frequência ou taxa de escalonamento.

3.2.5 Utilizando a Ferramenta Petrilyser

O uso da Petrilyser pode ocorrer com a execução de um dos programas: o MpnaD.exe sobre o DOS ou sobre os Windows 3.xx, 95 e NT, o MpnaW1.exe ou o MpnaW2.exe, ambos exclusivamente sobre o Windows 3.xx, 95 ou NT). As Figuras 3.12 e 3.13 apresentam as janelas da simulação quando utilizados os programas MpnaW.exe e MpnaW2.exe, respectivamente.

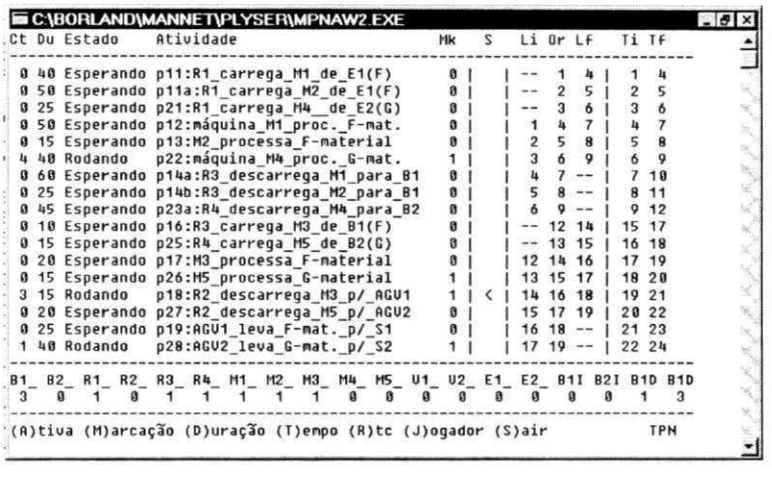


Figura 3.13: Janela de Simulação da Ferramenta Petrilyzer - versão Windows

Arquivos Utilizados

Os arquivos de lugares, transições e arcos da Rede de Molloy e da Célula de Manufatura são pn1.lug, pn1.tra e pn1.arc e pn.lug, pn.tra e pn.arc, respectivamente.

Célula de Manufatura

Na célula de manufatura são lugares que modelam tarefas ou atividades:

- p11: robô R1 carregando máquina M1 com F_material de E1

- p11a: robô R1 carregando máquina M2 com F_material de E1
- p21: robô R1 carregando máquina M4 com G_material de E2
- p12: máquina M1 processando F_material
- p13: máquina M2 processando F_material
- p22: máquina M4 processando G_material
- p14a: descarregando F_material de M1 para buffer B1
- p14b: descarregando F_material de M2 para buffer B1
- p23a: descarregando G_material de M4 para buffer B2
- p16: robô R3 carregando M3 com F_material de B1
- p25: robô R4 carregando M5 com G_material de B2
- p17: máquina M3 processando F_material
- p26: máquina M5 processando G_material
- p18: robô R2 descarregando F_material da máquina M3 para o veículo AGV1
- p27: robô R2 descarregando G_material da máquina M5 para o veículo AGV2
- p19: Veículo AGV1 levando F_material p/ a saída S1 e paletes vazios p/ a entrada E1
- p28: Veículo AGV2 levando G_material p/ a saída S2 e paletes vazios p/ a entrada E2

São lugares que modelam recursos fixos:

- p1: robô R1 disponível
- p2: robô R2 disponível

- p3: robô R3 disponível
- p4: robô R4 disponível
- p12': máquina M1 disponível
- p13': máquina M2 disponível
- p17': máquina M3 disponível
- p22': máquina M4 disponível
- p26': máquina M5 disponível
- p19': Veículo AGV1 disponível
- p28': Veículo AGV2 disponível

São lugares que modelam recursos variáveis:

- p15: buffer B1 armazenando F_material
- p24: buffer B2 armazenando G_material
- p10: peças tipo F disponíveis na entrada E1
- p20: peças tipo G disponíveis na entrada E2
- p23: peças tipo G disponíveis para armazenagem no buffer B2
- p15': slots vazios no buffer B1 disponíveis
- p24': slots vazios no buffer B2 disponíveis

3.2.6 Principais rotinas

A seguir é apresentada uma breve descrição das rotinas que compreendem a ferramenta Petrilyser[8].

- void beep(void): liga/desliga autofalante do PC;
- void warning(unsigned int i): apresenta advertência de índice i;
- void prompt(void): apresenta o *prompt*;
- void espera(void): espera uma tecla ser pressionada;
- unsigned int buffer(unsigned int i): verifica o estado do buffer de índice i (se o mesmo estiver vazio, envia mensagem para o usuário);
- unsigned int get_resp(unsigned int i): retorna resposta do usuário (0 ou 1 para FALSO ou VERDADEIRO) para a pergunta realizada de índice i;
- void liga(char *file): liga o diretório de trabalho ao arquivo corrente apontado por file;
- char menu (unsigned int i, unsigned int j, unsigned int k): apresenta o menu de índice j com cabeçalho dado por i e espera (k=1) ou não a digitação da tecla <enter> após dado o comando;
- void limpa_estado(void): define todos os *buffers* de dados como vazios;
- void limpa_tela(void): limpa o vídeo;
- void edit_um_lug(unsigned int i): edita lugar de índice i;
- unsigned int abre_lug(unsigned int i): acessa lugar de índice i;
- void fecha_lug(void): encerra edição de lugar;

[8] A maioria destas rotinas são extremamente simples, o que dispensa o detalhamento de seus algoritmos.

- void edit_lug(void): edita lugar;
- unsigned int car_lug(void): lê do disco arquivos de lugares;
- void apr_lug(void): apresenta o conteúdo de arquivos de lugares;
- unsigned int alt_lug(void): altera lugar;
- void edit_uma_tra(unsigned int i): edita transição de índice i;
- unsigned int abre_tra(unsigned int i): acessa transição de índice i;
- void fecha_tra(void): encerra edição de transição;
- void edit_tra(void): edita transição;
- unsigned int car_tra(void): lê do disco arquivos de transições;
- void apr_tra(void): apresenta o conteúdo de arquivos de transições;
- unsigned int alt_tra(void): altera transição;
- void edit_um_arc(unsigned int i): edita arco de índice i;
- unsigned int abre_arc(unsigned int i): acessa arco de índice i;
- void fecha_arc(void): encerra edição de arco;
- void edit_arc(void): edita arco;
- unsigned int car_arc(void): lê do disco arquivos de arcos;
- void apr_arc(void): apresenta o conteúdo de arquivos de arcos;
- void editar(void): apresenta menu de edição de arquivos;
- void ler_disco(void): apresenta menu de leitura de arquivos;
- void apresentar (void): apresenta menu de visualização de arquivos;

- void alterar(void): apresenta menu de alteração de arquivos;
- void arquivos(void): apresenta menu geral para arquivos;
- void rem_finita(unsigned int nlugar): remove lugar de capacidade finita de índice nlugar, chamada por cap_fin, e possui como algoritmo o seguinte:
  1. cria um segundo lugar com o mesmo nome seguido de ' (linha)
  2. define as capacidades do lugar nlinha e o novo lugar como sendo 0 (zero, ou infinita)
  3. para o novo lugar estabelece tipo "x" (ou criado no processo de remoção)
  4. cria arcos que chegam em todas as transições e saem do novo nlinha, com os mesmos pesos, porém com sentido contrário aos que saem da transição e chegam no lugar nlinha, de forma que para todo arco que chega ao lugar nlinha há um arco com o mesmo peso que sai do novo lugar e vai em direção à transição de entrada do lugar nlinha.
  5. repete o passo anterior criando arcos que saem da transição de saída e chegam ao novo lugar.
- void rem_arco(unsigned int inarco): remove arco inibidor de índice inarco, chamada por arc_inib, e possui o seguinte algoritmo:
  1. cria lugar com mesmo nome do lugar de destino seguido de ' (linha);
  2. define as capacidades do novo lugar como sendo 0 (zero, ou infinita);
  3. para o arco inibidor estabelece tipo "x" (ou ex arco inibidor);
  4. cria arcos (p',t) e (t,p').
- unsigned int arc_inib (void): localiza e remove arco inibidor
- unsigned int cap_fin(void): localiza e remove lugar de capacidade finita;

- unsigned int rede(unsigned int i): gera rede de capacidade infinita, chama as rotinas arc_inib e cap_fin;
- void matriz(unsigned int k): gera matriz de incidência se os arquivos tiverem sido lidos de disco ou editados (*buffers* não vazios);
- unsigned int mar_ini(unsigned int d, unsigned int p): gera o vetor marcação inicial apresentando-o entre parênteses, se d é diferente de zero e com pausa, se p não nulo;
- unsigned int alt_ini(unsigned int k): altera marcação inicial;
- unsigned int monta_vetor(unsigned int k); monta vetor de disparos o qual é utilizado no disparo de uma transição (ou várias simultaneamente);
- unsigned int opera_mat(void): retorna a marcação resultante após aplicado o vetor de disparo na equação de controle da rede de Petri em m_output[], retorna FALSO se a transição (ou conjunto de transições) não estiver habilitada;
- unsigned int dispara(unsigned int l): verifica se uma transição é disparável;
- unsigned int mar_fin(unsigned int k, unsigned int m): apresenta marcação final (utiliza dispara e opera_mat);
- unsigned int gera_filha(unsigned int m): gera todas as marcações alcançáveis pelo disparo de todas as transições habilitadas uma vez a partir da marcação de índice m e classificando-a como NOVA, FINAL ou NÓ;
- unsigned int verif_velha(unsigned int k): verifica as NOVAS marcações, se existentes as denomina VELHAS;
- unsigned int ger_arv(void): gera árvore de cobertura;
- void tarn (void): rotina_tarefa que decrementa os contadores de eventos da tarefa n em execução;

- void inicialize (void): inicializa o RTC, definindo um novo endereço de execução;
- void finalize (void): restaura o endereço de execução *default* do RTC;
- void interrupt newint8(void): nova rotina do RTC;
- int p (int val): testa semáforo para o recurso de ordem val e impede o acesso se ocupado;
- int v(int val): libera o acesso ao recurso de ordem val;
- void proc (val): procedimento associado ao recurso de ordem val;
- void tela_b_fixa(void): apresenta parte do terceiro campo da tela de resultados da simulação ou iniciais dos recursos fixos e variáveis;
- void tela_fixa (void): apresenta o segundo campo da tela de resultados da simulação ou índices;
- void tela_b_var(void): apresenta parte do terceiro campo da tela de resultados da simulação ou as marcações dos recursos fixos e variáveis;
- void tela_var(void): apresenta a parte variável do primeiro campo da tela de resultados da simulação ou os valores dos contadores e tempo de duração das tarefas e seus estados;
- void inicia_bct(void): inicializa os buffers de controle das tarefas no início da simulação, classificando as tarefas segundo a disposição destas na rede, identificando assim as condições de conflito, sequência, junção, entre outras propriedades importantes no processo de marcar a rede e atribuir estados às tarefas;
- void tempo(void): muda TPN para SPN e vice versa;
- void dura(void): altera a duração da tarefa;
- void ativa(void); ativa ou desativa tarefa;

- void comandos(void): interpretador dos comandos quando na tela de simulação;
- void atual_rede(void): atualiza estados da rede;
- void escalo(void): *loop* realiza o escalonador do simulador;
- void arvore(void): apresenta menu de opções;
- void config(unsigned int i): inicializa ferramenta;
- void main (void): rotina ou *loop* principal.

3.3 A Ferramenta ManNet

A ferramenta ManNet consiste em uma nova proposta da ferramenta Petrilyser. Em outras palavras, trata-se de uma versão interativa com o usuário destinada a ambientes *Windows 95* e superiores. Esta ferramenta é constituída por um variado conjunto de funções ou rotinas que são funcionalmente divididas em módulos (apresentados pelas Figura 3.14), que implementam:

- um editor de Redes de Petri Estocásticas;
- um simulador-analisador de redes de Petri; e
- um conjunto de funções de ajuda ao usuário ou *help on line.*

Estes módulos estão definidos nos arquivos auxiliares e bibliotecas que são acessados pelo programa executável ManNet.exe. A ferramenta ManNet pode ser utilizada em ferramentas *Windows 3.1* e versões mais atualizadas (NT, inclusive) e exige componentes de *hardware*[9] mínimo:

- CPU 80486 a 100 MHz;

[9] A configuração poderá ser insuficiente em redes com elevado número de lugares que modelam atividades temporais e recursos de acesso mutuamente exclusivo. Nestes casos, recomenda-se uma CPU de maior desempenho, maiores memórias *cache* e RAM.

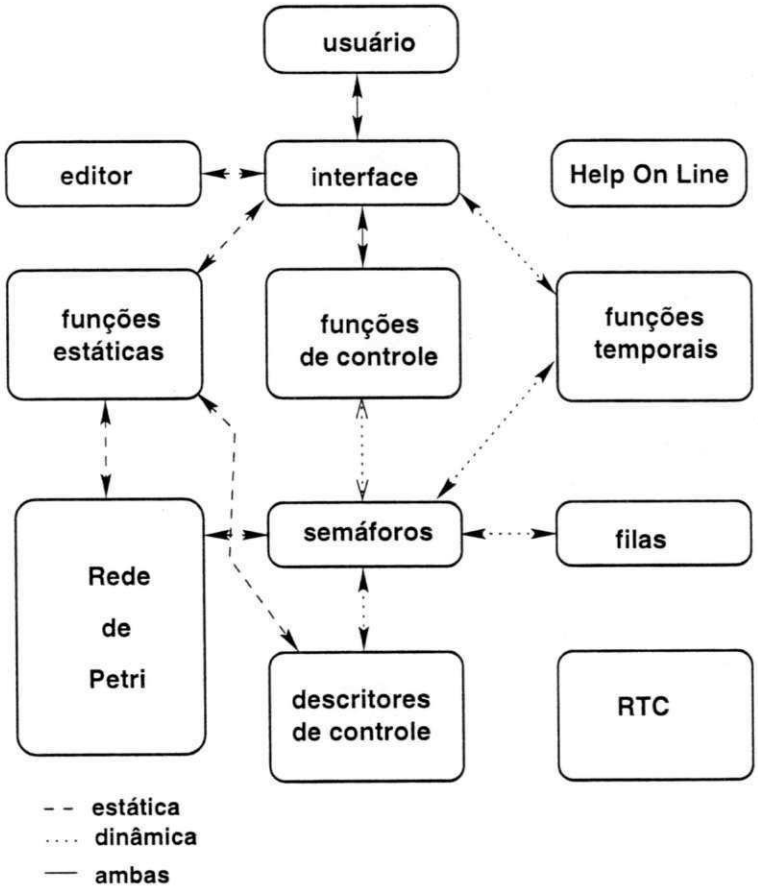


Figura 3.14: Componentes da Ferramenta ManNet

- 8 Mbytes de memória RAM;
- 4 Mbytes (para instalação) de disco rígido ;

Executando a Ferramenta ManNet

A execução da ferramenta ManNet é realizada de forma semelhante aos programas executáveis para os ambientes *Windows*:

- através do Gerenciador de Arquivos (presente em todos os ambientes *Windows*), com um duplo *click* com o botão esquerdo do *mouse* (*DoubleLeftClick*);
- através dos comandos *Iniciar* | *Executar* do Windows 95 ou *Arquivo* | *Executar* nas versões Windows 3.1 e 3.11;
- utilizando atalhos permitidos pelo *Windows* em uso, que facilitem o acesso ao programa executável ManNet.exe.

Uma vez iniciada, a ferramenta ManNet apresenta sua primeira janela, conforme apresenta a Figura 3.15, que contém:

- uma paleta que contém o cabeçalho: "ManNet - Ferramenta de Simulação" e três ícones ou atalhos para os comandos de minimizar totalmente a janela, para maximizar e, finalmente, para encerrar a ferramenta ManNet;
- o menu principal de operações que dá acesso aos sub-menus *Arquivo*, *Marcações*, *Estrutura*, *Simulação* e *?(sub-menu de ajuda)*;
- uma barra de ícones com ajuda sensível ao posicionamento do cursor, que podem ser utilizados como atalhos para janelas ou comandos que permitem editar um nova rede de Petri, ler/abrir arquivos existentes em disco, salvar e imprimir arquivos, editar propriedades temporais, iniciar/finalizar simulação, apresentar dados estatísticos, tabelas e, finalmente, encerrar o uso da ferramenta ManNet; e

- uma barra de estados, na base da janela, onde são apresentadas outras informações sobre comandos apontados pelo cursor.

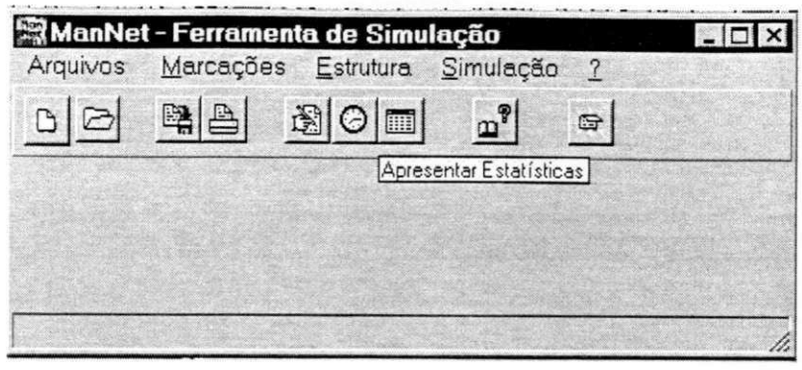


Figura 3.15: Janela Principal da Ferramenta ManNet

3.3.1 O Editor de Redes de Petri

Nesta seção detalha-se as funcionalidades da ferramenta ManNet. Inicialmente a edição de uma rede de Petri pode ser dada a partir da leitura de uma rede já editada e presente em disco ou a partir da criação de novos arquivos.

Os Arquivos de uma Rede de Petri

Os arquivos de uma rede de Petri são quatro: o arquivo de modelos, o de lugares, o de transições e o de arcos, onde os três últimos armazenam, exclusivamente, informações relacionadas com a estrutura da rede de Petri (lugares, transições e arcos, respectivamente) e não podem ser manipulados separadamente. São, então arquivos gerenciados pelo arquivo de modelos. Possuem as extensões .lug, .tra, e .arc, respectivamente (não obrigatoriamente). No arquivo de modelos constam os nomes dos demais arquivos, as últimas marcações iniciais utilizadas, as propriedades temporais dos elementos das redes de Petri e os parâmetros de configuração do simulador, que são apresentadas a seguir. Em resumo, o gerenciamento dos arquivos das redes de Petri é realizado à partir do arquivo de modelos.

Abrindo um Arquivo

Para abrir um arquivo de modelos deve-se ativar o comando *Arquivo —Abrir* ou pressionar simultanemente as teclas CTRL e A, resultando na apresentação da janela mostrada pela Figura 3.16. Em seguida, a ferramenta apresenta a janela presente na Figura 3.17, mediante a qual poder-se-á localizar e selecionar (editando o nome ou com o botão esquerdo do *mouse*) o diretório e o arquivo desejado ou *Cancelar* a operação, além de outras operações.

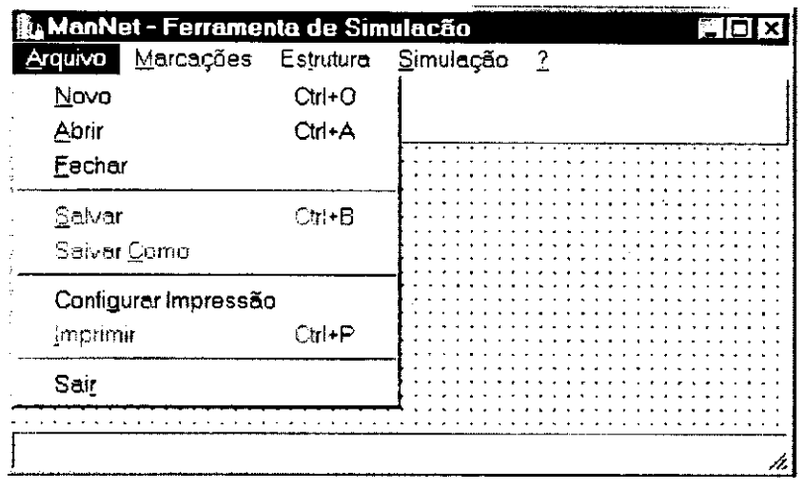


Figura 3.16: Sub-Menu arquivo

Determinado o arquivo de modelo a ser lido, a ferramenta executa a "leitura" deste e dos demais arquivos por este gerenciados, apresentando em uma janela com um fichário de pastas que conterá todas as informações lidas do disco. A Figura 3.19 apresenta a pasta de edição de lugares. Esta pasta, a exemplo das demais, é composta por grupos de comandos os quais se tornam visíveis a medida em que o usuário define as propriedades dos lugares. Quando na edição de um lugar que modela um recurso variável, por exemplo, a pasta citada apresentará os grupos necessários como mostra a figura 3.20.

As pastas são janelas embutidas de edição. Nestas, caixas de combinação (*combo-box*, de listagem (*listbox*), de textos e botões de configuração que estão relacionadas com listas descritoras de lugares, transições e arcos, e a estruturas de dados que, transpa-

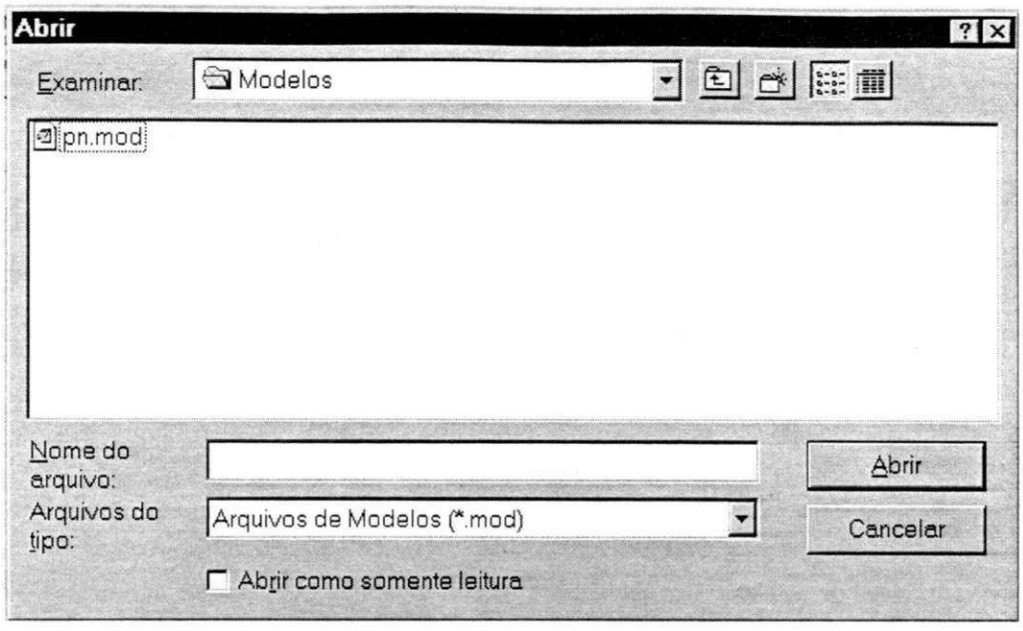


Figura 3.17: Janela de localização de arquivos

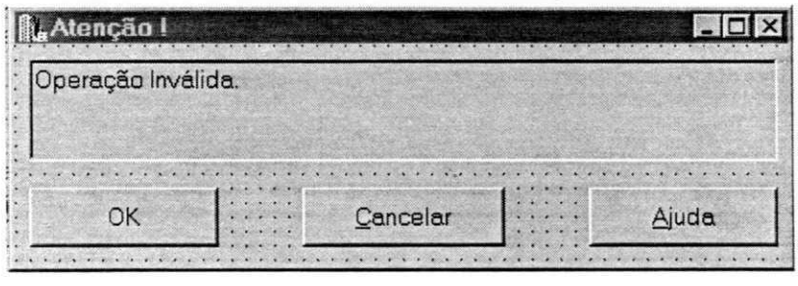


Figura 3.18: Mensagem de aviso de erro

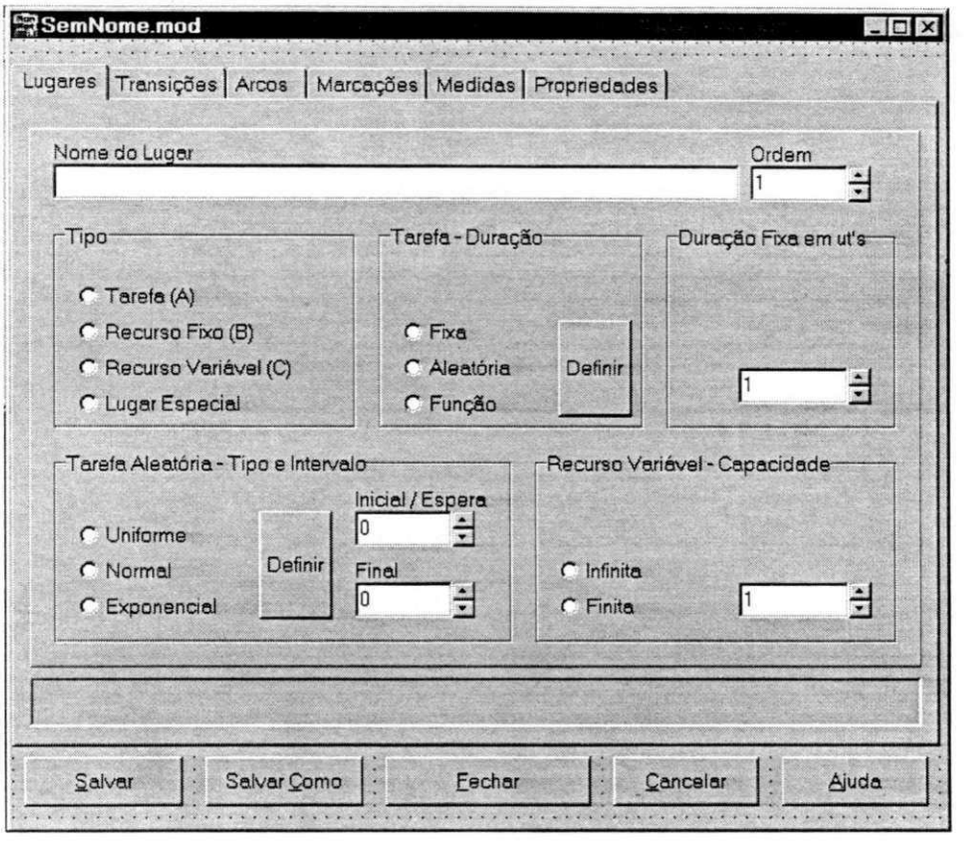


Figura 3.19: Pasta de edição de Lugares

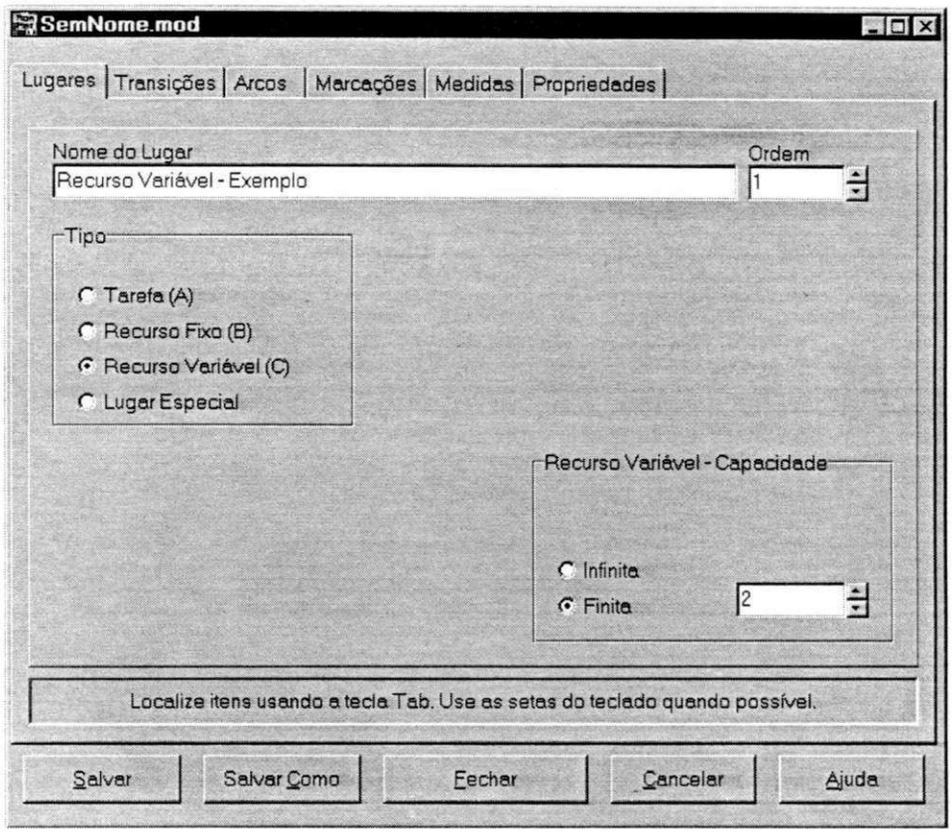


Figura 3.20: Edição de Lugar que modela um Recurso Variável

rentes para o usuário, organizam o armazenamento destes na memória do computador e na leitura e gravação em disco.

A janela apresenta, ainda, cinco botões em através dos quais pode-se salvar os arquivos em ediçao, mudando seu nome ou não, cancelar e finalizar a edição, e acessar o arquivo de ajuda.

Falhas de operação

Na ocorrência de uma falha, a operação será abandonada e uma mensagem de erro será emitida com esta informação, de forma semelhante à apresentada pela Figura 3.18[10].

Alterando Dados

Uma vez aberto um arquivo, ou mesmo durante a edição de um novo arquivo, o usuário deverá proceder do mesmo modo que edita um novo arquivo, ou seja, removendo, alterando, ou acrescentando novos elementos ou propriedades temporais.

Criando um novo Arquivo

Quando na criação de um novo arquivo (mediante o comando *Arquivo* | *Novo*, Figura 3.16), a ferramenta apresenta automaticamente a janela presente na Figura 3.19 se e somente se, esta janela não estiver aberta, caso contrário uma janela de mensagem é emitida, a verificar se as edições já realizadas devam ser abandonadas ou a operação cancelada.

Edição das Propriedades Estruturais das Redes de Petri

Consiste na edição das propriedades não relacionadas com a marcação *Inicial* e o comportamento temporal das redes de Petri.

[10]Em alguns casos, através da janela de mensagem é possível obter informações complementares, através do botão *Ajuda*.

A edição das propriedades estruturais dos lugares consiste na edição do nome do lugar, o qual é associado a um índice automaticamente. A Figura 3.19 apresenta esta janela, a qual permite ainda a edição de suas propriedades temporais e de simulação (descritas a seguir).

Da mesma forma, a Figura 3.21 apresenta a pasta que permite a edição do nome da transição e de suas propriedades temporais (descritas adiante).

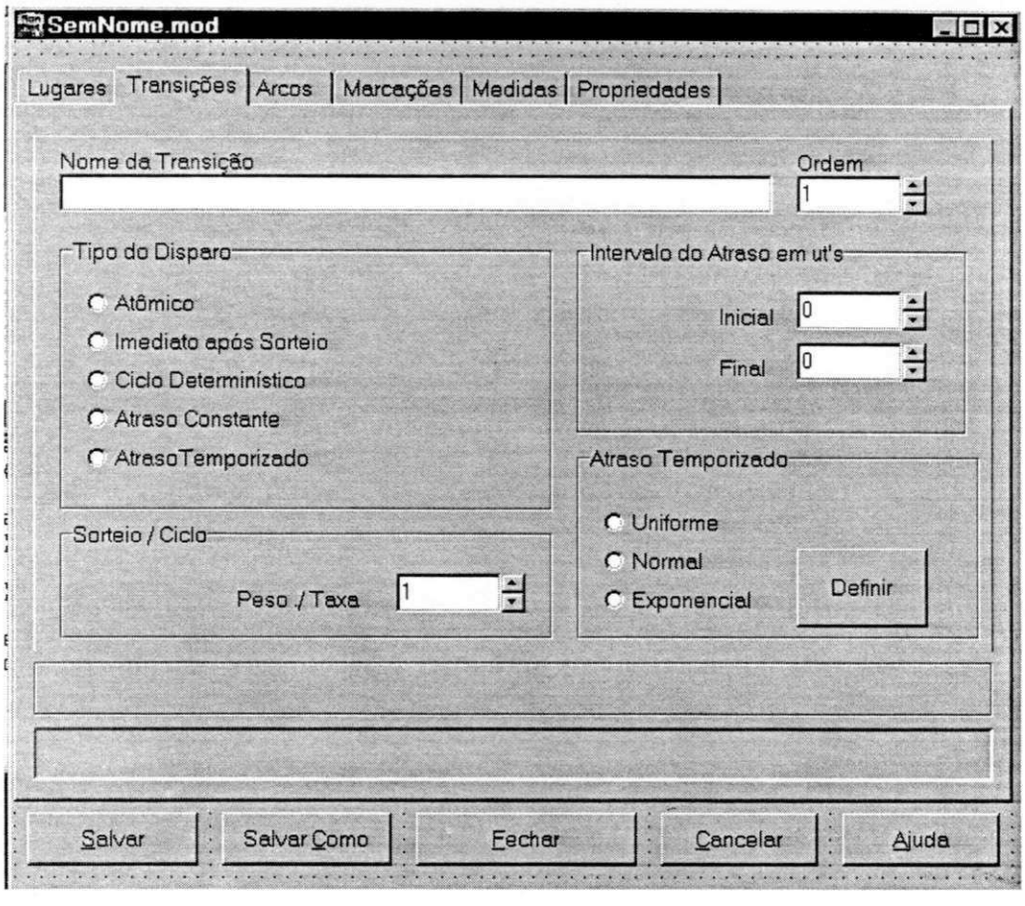


Figura 3.21: Edição de transições

A edição das propriedades estruturais dos arcos (Figura 3.22), consiste na edição do nome do arco (que pode ser "" (vazio) se não for de interêsse nomeá-lo), o seu tipo (padrão ou inibidor), se tem origem em um lugar ou em uma transição, as ordens destes elementos que interligam e seu peso. A proposta da interface da ferramenta ManNet, inicialmente, previa uma janela com duas caixas de combinação com lugares

e transições, com as quais poder-se-ia estabelecer os arcos com maior agilidade e clareza. Contudo, as diversas mudanças de ambientes de desenvolvimento, suas limitações e o tempo necessário para esta, entre outras implementações, impediram-nos de fazê-la. Contudo, por ser uma ferramenta aberta esta, senão outra solução, poderá ser adotada nas futuras versões.

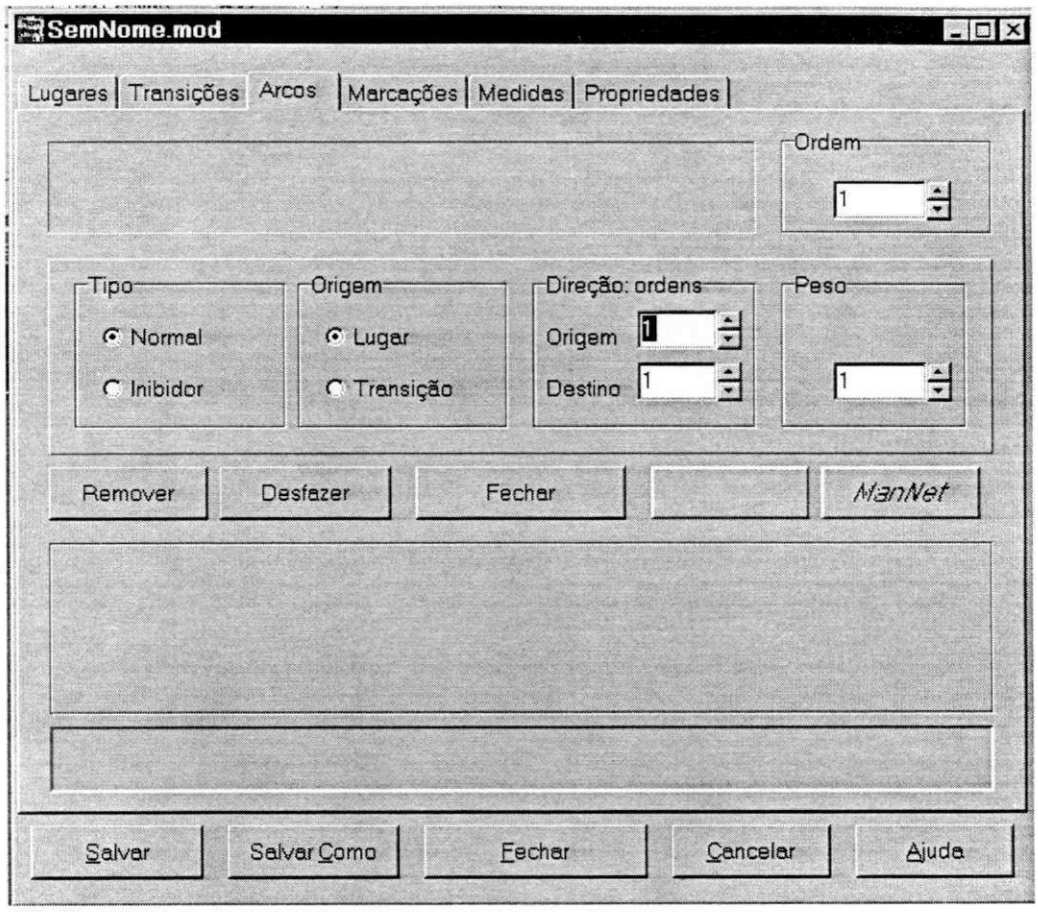


Figura 3.22: Edição de arcos

Definindo os Arquivos da Rede de Petri

As redes de Petri são definidas por quatro arquivos como já foram citados, um dos quais gerencia os demais. Seus nomes e extensões poderão ser alterados a qualquer

momento através da pasta mostrada pela Figura 3.23[11],que permite a edição de outras propriedades descritas adiante.

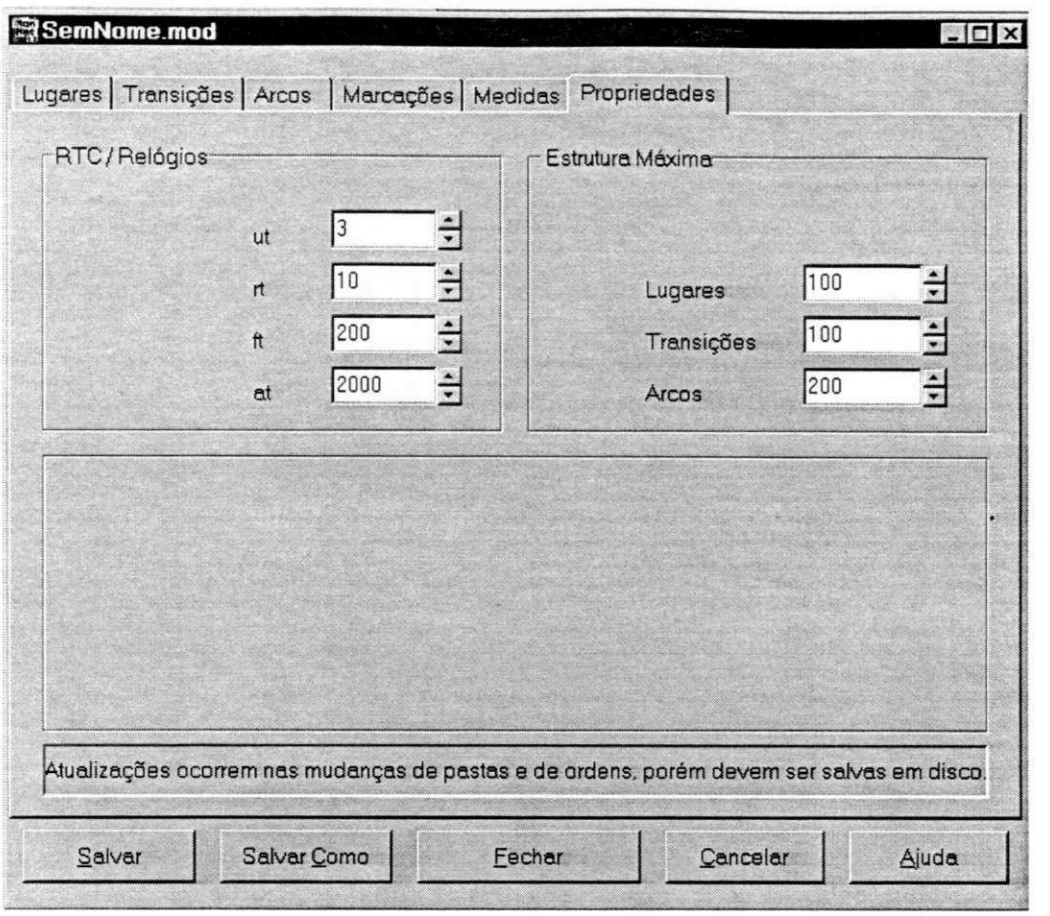


Figura 3.23: Edição das propriedades das redes de Petri

Editando Marcações

A ferramenta ManNet permite a edição de quatro marcações iniciais, uma denominada *inicial* e as demais de marcações de *Teste #1*, *Teste #2* e *Teste #3* que podem ser utilizadas quando desejada a comparação entre resultados obtidos a partir de duas a quatro marcações iniciais diferentes. Esta edição é realizada a partir pasta apresentada na Figura 3.24, que pode ser obtida a partir do comando *Marcacões Inicial e Testes*

[11]A abertur a do arquivo de modelo de uma rede de Petri abre também, automáticamente, os arquivos de lugares, transições e arcos.

—*Editar*, caso o usuário esteja com a janela principal ativa. Para acelerar a edição das marcações de teste (que geralmente apresentam pequenas diferenças em relação à marcação *Inicial*) deve-se iniciar a edição pela marcação Inicial. A ferramenta copiará automaticamente os valores editados para a marcação *Inicial* para as marcações de *Teste*, a cada marcação editada.

Esta cópia automática é interrompida sempre que se altera uma marcação de teste durante a edição da marcação *Inicial*.

As marcações *Inicial* e as de *Testes* são armazenadas no arquivo de modelos.

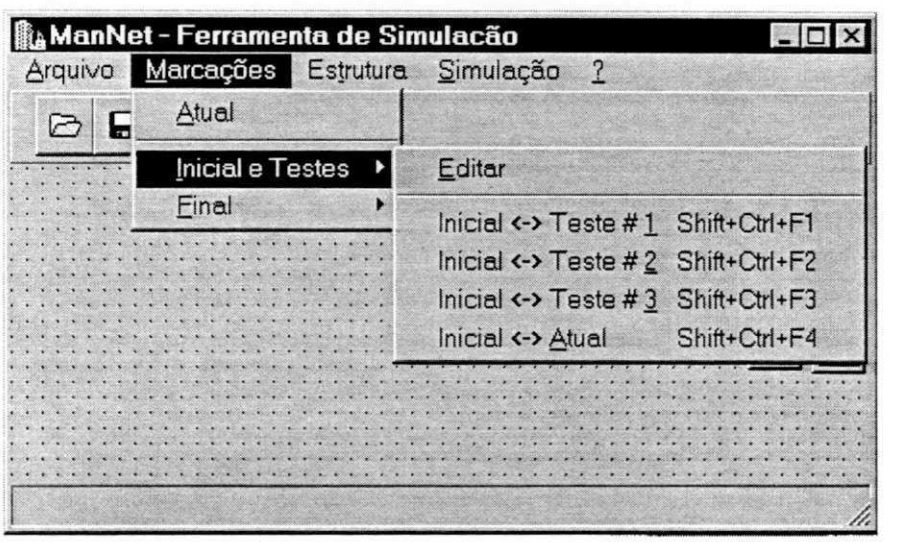


Figura 3.24: Edição das marcações iniciais

3.3.2 As Propriedades Estruturais dos Modelos em Redes de Petri

São propriedades que não consideram as propriedades temporais dos modelos em Redes de Petri, as quais são ativadas a partir do sub-menu *Estrutura* (conforme apresenta a janela na Figura 3.25), tais como:

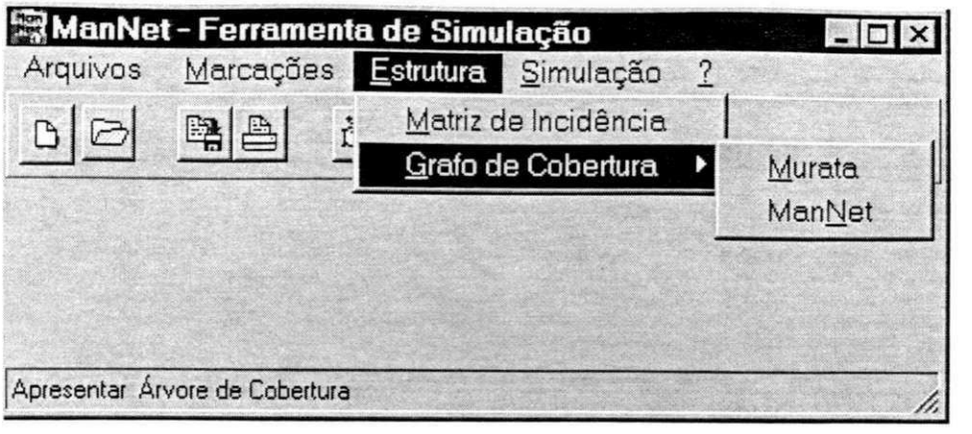


Figura 3.25: Submenu estrutura

- a montagem, armazenamento e impressão[12] da matriz de incidência[13].
- a obtenção da próxima marcação com o disparo de uma transição habilitada (quando selecionada pelo mouse ou de uma sequência possível de disparos, conforme apresenta a Figura 3.26[14] e;
- geração, armazenamento e impressão da árvore de cobertura, conforme apresentada pela Figura 3.25.

[12]Uma vez apresentada a janela de montagem da matriz de incidência ou da árvore de cobertura, os comandos de Salvar e Imprimir são disponíveis em sub-menu *Pop Up* ativado com o botão direito do *mouse* quando o cursor estiver sobre esta janela.

[13]Na existência de arcos inibidores e/ou lugares de capacidade finita, a matriz de incidência somente será gerada após a conversão em arcos normais e/ou lugares de capacidade infinita. Procedimentos estes realizados automaticamente quando na geração da matriz de incidência se e somente se a marcação *Inicial* já estiver definida, alterando-a e acrescentando novos lugares e arcos à rede de Petri inicialmente editada. Aos lugares criados, serão atribuídas às marcacões de *Teste* os mesmos valores para a correspondente marcação *Inicial*. Este processo é irreversível, porém, o usuário poderá *salvar* os arquivos na nova rede com novo nome.

[14]Neste caso, a rede de Petri assume (e mantém) a marcação resultante (*Atual*) se e somente se a transição ou sequência de transições for disparada com sucesso. O sub-menu *Marcações* | *Final* pode ser utilizado na obtenção da árvore de alcançabilidade das redes de Petri.

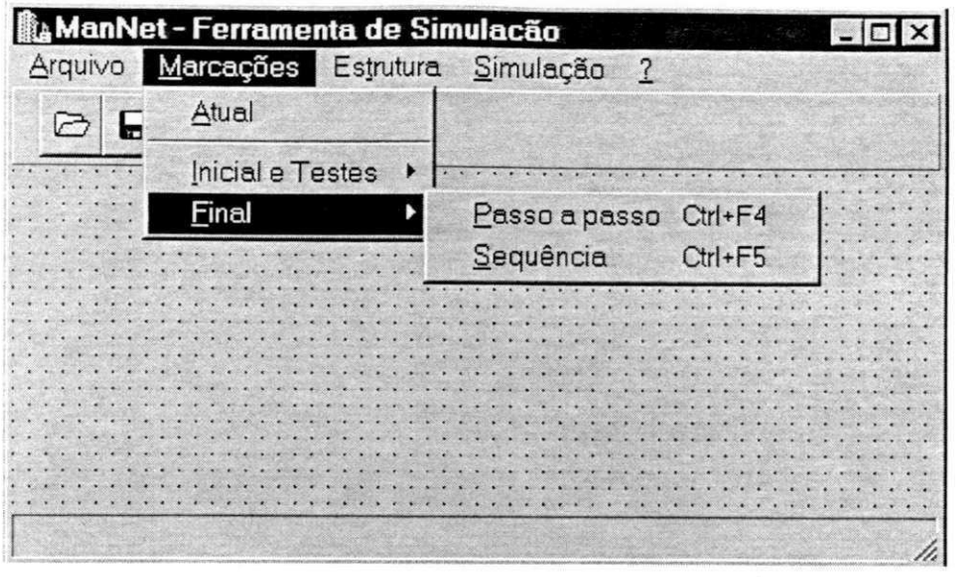


Figura 3.26: Marcação final

A Árvore e o Grafo de Cobertura

A ferramenta ManNet permite montar e apresentar:

- a árvore de cobertura proposta por Murata [Mur89], onde as marcações já existentes (ditas *velhas*) são geradas tantas vezes quantas apareçam na árvore;
- e o grafo de cobertura em uma versão proposta pela ferramenta ManNet, onde cada marcação é gerada apenas uma vez.

Nesta segunda alternativa, a cada marcação ou estado é associado um vetor com campos, onde são registradas as transições cujos disparos levaram a rede de Petri a este estado, a ordem da marcação atual, da(s) antecedente(s) e seu tipo. Esta solução é apropriada para redes, como a apresentada na Figura 3.27, cuja árvore de cobertura apresenta número de marcações muito grande, embora *limitada* e *não viva*.

Nestes casos há significativa redução da memória necessária para armazená-la, e maiores chances de que uma marcação esperada (de *bloqueio*, por exemplo) ocorra, antes do *estouro* da memória do sistema. Exige, portanto, um esforço computacional maior.

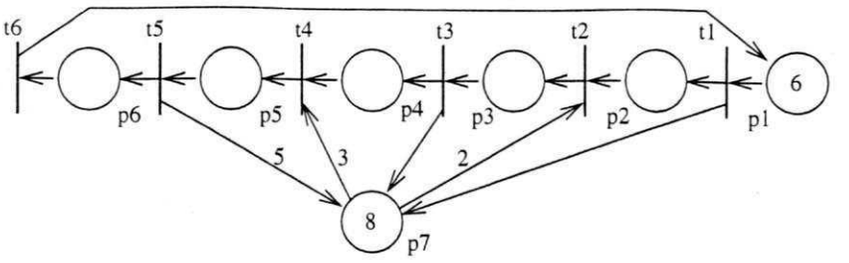


Figura 3.27: Rede de Petri limitada e com bloqueio

As marcações (ou estados) alcançáveis de uma rede de Petri são classificadas de acordo com as suas disposições na árvore de cobertura. Podem ser:

- inicial, ou marcação no início (ou topo) da árvore;
- nó, marcação que pode gerar mais do que uma outra marcação, porém só ocorre uma vez na árvore;
- nova, se ela gera uma única outra marcação e só ocorre uma única vez;
- velha, se ela já existe em algum ponto da árvore, ou seja, se ela pode ser gerada por diferentes disparos de transições a partir de marcações outras; e
- final ou de bloqueio, ou marcação que impossibilita novos disparos de transições, ou seja, que não gera outras marcações.

3.3.3 O Simulador

Compreende um conjunto de funções ou comandos que permite simular sequências de disparos em uma rede de Petri de acordo com as propriedades temporais estocásticas atribuídas às atividades modeladas pela rede e transições. A Figura 3.28 apresenta o sub-menu *Simulação* e seus comandos, com os quais pode-se configurar as propriedades do Simulador, iniciar a simulação e obter os dados estatísticos colhidos, segundo as medidas definidas de desempenho.

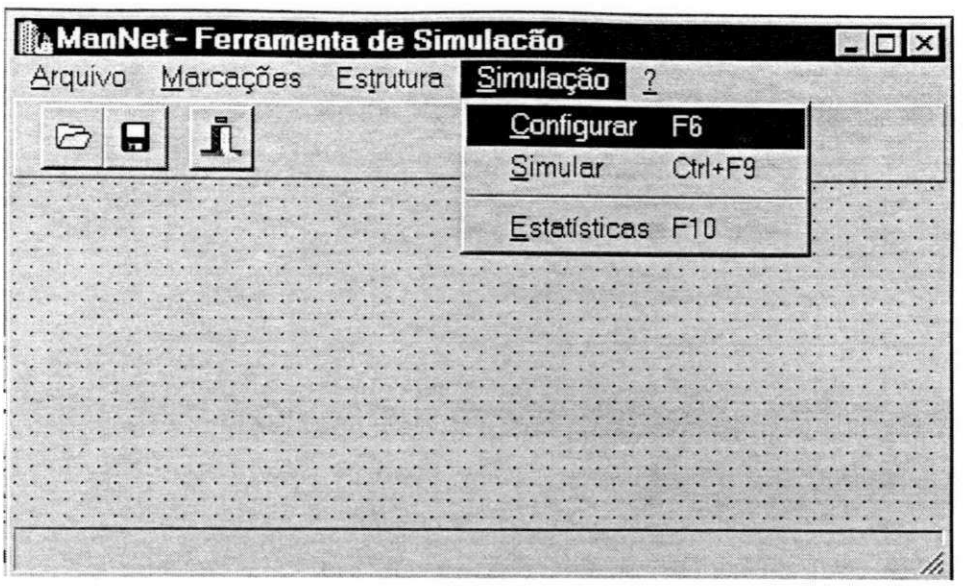


Figura 3.28: Simulação e análise

3.3.4 Elementos do Simulador

Constituem o Simulador:

- rotinas que implementam relógios gerados a partir da interrupção do RTC e contadores;
- as funções de um "escalonador" com política *Round Robin não preemptiva*;
- as funções geradoras de números aleatórios;
- um conjunto de procedimentos que implementam um "executivo" em tempo real;
- procedimentos que realizam o controle de tarefas e recursos mutuamente exclusivos;
- lugares "produtores" e "consumidores" de fichas, para a simulação das entradas e saídas do modelo em redes de Petri;
- funções de estatística, usadas para análise entre experimentos realizados; e
- a multiprogramação (escalonada) de funções de aplicação e de controle de tarefas e recursos.

Configurando o Simulador

A configuração do Simulador, compreende:

- a edição das propriedades temporais para transições e lugares, através das pastas apresentadas pelas Figuras 3.19, 3.21 e 3.22;
- a edição das taxas de produção e consumo para os lugares especiais, através da pasta de edição de lugares, conforme apresenta a Figura 3.29; e
- a edição das medidas de desempenho estabelecidas para os lugares que deseja-se avaliar (vide Medidas de Desempenho);
- a determinação das taxas dos temporizadores, através da pasta mostrada na Figura 3.23.

Escalonador Temporal

O escalonador é um componente da ferramenta que, sob o controle de um relógio interno baseado na interrupção do RTC, atualiza a marcação da rede (número de fichas dos lugares) a cada disparo de uma transição. Determina, ainda, que tarefas se encontram "prontas" para sua execução e verifica e atualiza outras informações exclusivas de cada tarefa, as quais estão descritas em um bloco dedicado de memória ou *Descritor de Tarefas.*

As Tarefas e os Recursos

Na ferramenta ManNet, para efeito de simplificação dos *algoritmos* as tarefas são modeladas sempre por lugares (conforme apresentado na Figura 3.30) e são iniciadas pelo disparo de transições, que podem ser:

- *atômicas* ou com tempo de duração nulo;

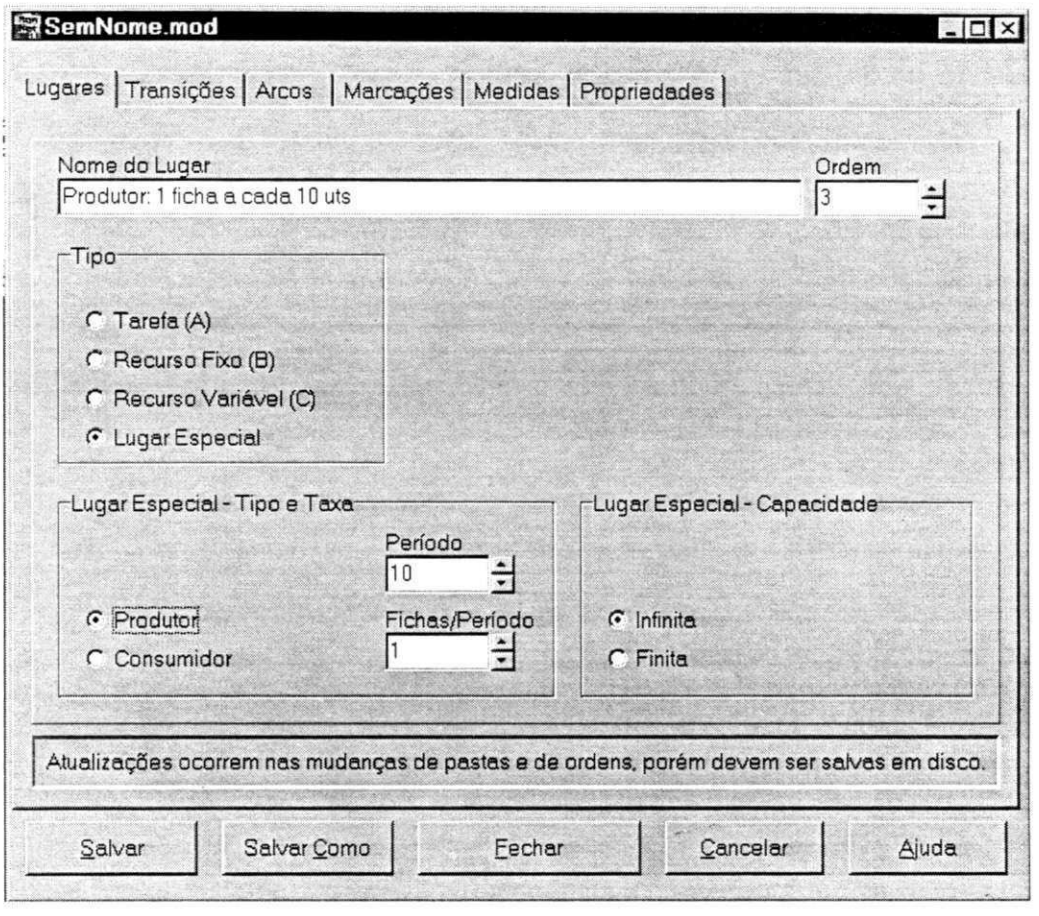


Figura 3.29: Edição de lugares especiais

- *cíclico-determinísticas*, quando são definidas taxas de ciclo de execução, para transições conflitantes;
- *sorteadas*, quando definidas taxas de disparo ou função de distribuição de probabilidade; ou
- *temporizadas*, quando são simuladas pela contagem de tempo de duração determinística ou estocástica.

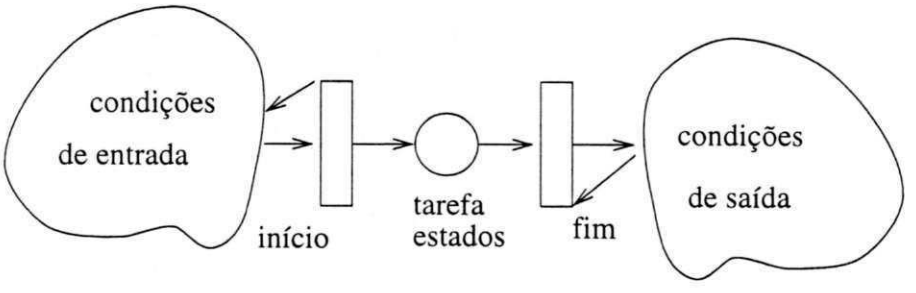


Figura 3.30: Modelo de uma tarefa

A simulação de uma tarefa consiste na execução de uma rotina em instantes estabelecidos pelo *RTC*, nos quais são decrementados contadores associados a esta rotina. As informações temporais e de controle de cada tarefa estão definidas em descritores individuais, que consistem em conjuntos de dados de acesso mutuamente exclusivo. Este acesso é gerenciado com o auxílio de filas (*pipeline)* e *semáforos.* São dados armazenados pelo descritor:

- a *ordem* ou índice da tarefa;
- seu *estado*;
- a *ordem* ou índice de todas as transições que a inicializam;
- a *ordem* ou índice de todas as transições que a finalizam;
- informações relativas aos procedimentos de temporização, tais como *modo*, contadores de eventos, limites, enfim, parâmetros que dependem da terefa e do seu comportamento; e

- a *fila da tarefa*, um banco de dados utilizado para armazenar temporariamente os dados variáveis da tarefa, quando "suspensa" (impedida de fazer uso de um recurso mutuamente exclusivo) e quando "interrompida" (pelo *RTC*).

Uma tarefa em execução pode ser desativada e ativada a qualquer instante, através de comando dado pelo *mouse*. Estas facilidades são úteis quando na análise de comportamento de sistemas quando são modeladas falhas nos equipamentos, rotas alternativas de produção, sincronização entre etapas concorrentes, entre outras aplicações.

Os Estados das Tarefas

A dinâmica da rede de Petri é gerenciada em função dos estados que as tarefas podem assumir. Os estados das tarefas podem ser seis, dois dos quais são internos e transparentes para o usuário:

1. *inativa(I)*: tarefa desativada pelo usuário;
2. *espera(E)*: tarefa em espera de atualização pelo escalonador;
3. *pronta(P)*: tarefa cujas condições de entrada e saída estão satisfeitas;
4. *ativa(A)*: tarefa em execução, não acessando um recurso de exclusão mútua;
5. *crítica (C,interno):* tarefa acessando um recurso de exclusão mútua; e
6. *suspensa (S, interno):* tarefa impedida de acesso a recurso de exclusão mútua, quando seu dados ficam salvos na fila (ou *lifo*) enquanto permanece neste estado.

Na Figura 3.31 são apresentados os estados das tarefas e as possíveis transições entre os mesmos.

Durante a simulação na janela (apresentada na Figura 3.32), ao lado dos lugares selecionados para visualização são apresentados indicadores circulares, onde são informados seus estados (*I, E, P, A, C ou S*), ou seja:

- I, quando marcado indica que a tarefa está inativa;

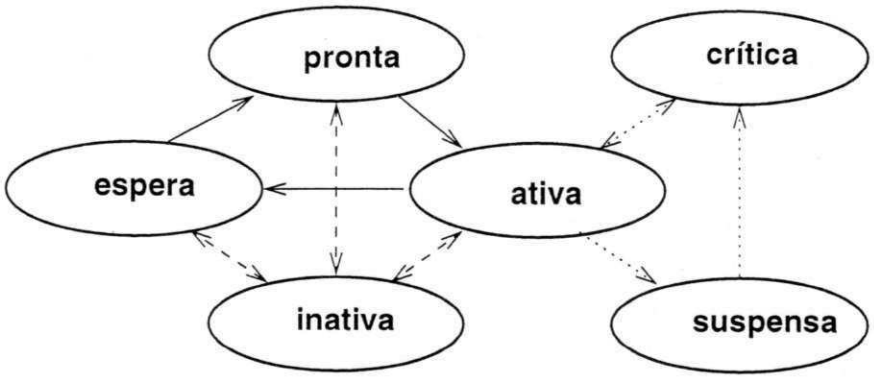


Figura 3.31: Estados das tarefas

- E, indica tarefa em estado de espera;
- P, indica tarefa pronta;
- A, indica tarefa ativa;
- C, indica tarefa acessando um recurso mutuamente exclusivo, ou em uma região crítica; e
- S, indica tarefa com atividade suspensa.

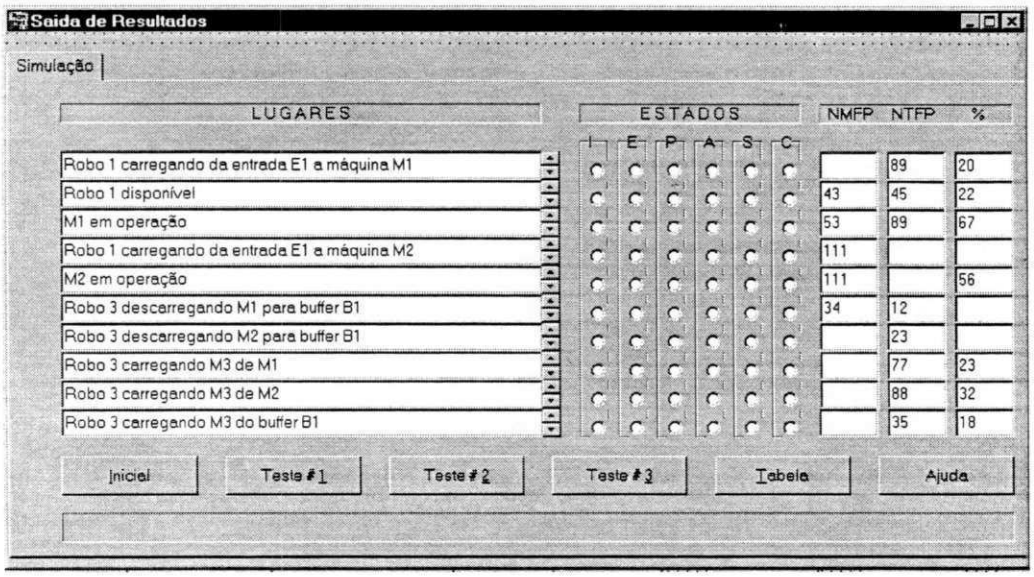


Figura 3.32: Estados das tarefas

Regras para as Mudanças de Estado

As tarefas mudam de estados à medida que são executadas na rede de Petri ou quando há uma intervenção realizada pelo usuário.

A seguir descreve-se quando ocorrem estas mudanças:

- *inativa* $\longleftrightarrow$ *pronta*: quando na inserção e remoção da tarefa da rede é realizada pelo usuário (neste caso, a ferramenta não altera parâmetros no descritor da tarefa em ambas as mudanças);
- *inativa* $\longleftrightarrow$*ativa*: quando na intervenção realizada pelo usuário (neste caso, a ferramenta salva parâmetros e realiza uma *pausa*);
- *inativa* $\longleftrightarrow$*espera*: idem a *inativa* $\longleftrightarrow$*ativa*;
- *espera* $\longrightarrow$*pronta* (realizada pelo escalonador): quando satisfeitas as condições necessárias para o início da tarefa (condições de entrada da transição cujo disparo a iniciará e a levará para o estado *ativa*);
- *pronta* $\longrightarrow$*ativa* (realizada pelo executivo): quando a transição de entrada é disparada (a ferramenta inicializa, então, seus parâmetros associados);
- *ativa* $\longrightarrow$*espera* (realizada pelo escalonador): quando ao fim de sua execução - a seguir, a ferramenta atualiza indicadores de estado e de parâmetros;
- *ativa* $\longleftrightarrow$*suspensa*: quando ocorre uma das condições:
  - quando a tarefa tem seu processamento "suspenso" pela interrupção do RTC[15](realizada pelo executivo); ou
  - quando a tarefa tem seu processamento "suspenso" por ocasião do acesso a um dado mutuamente exclusivo[16];

[15]Neste estado seus dados são salvos na *pilha* de seu descritor e sua atividade é suspensa e controlada pela fila (*fifo, "first in first out"*) até que retorne para o estado *ativa* e seja retomado o processamento.

[16]A tarefa permanecerá neste estado até que o dado esteja disponível - ver *suspensa* $\longleftrightarrow$*crítica*

- *suspensa* $\longrightarrow$*crítica*: ocorre quando um dado mutuamente exclusivo (cuja disponibilidade estava sendo aguardada) ficou disponível e está sendo acessado; e
- *inativa* $\longleftrightarrow$*crítica*: quando a tarefa utiliza e libera um recurso mutuamente exclusivo disponível.

Os Geradores de Números Aleatórios

A função destes geradores é decidir, através de procedimentos de seleção ou de "sorteio", quando nas situações de conflito entre tarefas na inicialização (tarefas *prontas* e na finalização (tarefas *ativas*), segundo as taxas de probabilidade de disparo associadas às transições envolvidas.

Esta seleção envolve a definição de *taxas de ciclo*[17] entre transições em conflito.

O Executivo em Tempo Real

A função do executivo é monitorar tarefas em operação (*ativas*) e atualizar seus estados segundo o comportamento temporal de cada e/ou os resultados fornecidos pelos geradores de números aleatórios. Em outras palavras, o executivo é o responsável pela *marcação* da rede de Petri e a execução das tarefas em progresso sob a cadência do serviço de interrupção de um relógio (ou temporizador baseado no *RTC*). Realiza, ainda, os comandos que são determinados pelo usuário (tais como ativar e desativar tarefas ou indisponibilizar recursos) durante a simulação.

A Intervenção do Usuário

Durante a simulação o usuário pode intervir sobre todas as atividades presentes na janela de Simulação (vide Figura 3.28). Esta janela é sensível ao botão direito do *mouse*, cujo acionamento faz surgir no vídeo um sub-menu Pop Up que contém os Comandos

[17] Uma taxa de ciclo é dada pelo número definido de disparos de uma transição quando em conflito, dividido pelo somatório de todos estes números definidos para as transições envolvidas.

de Ativar e o de Desativar. Permite, ainda, em tempo real alterar as propriedades temporais da tarefa selecionada, entre outras operações.

O Relógio de Tempo Real

Para a obtenção de relógios, a frequência do *RTC* é dividida por valores inteiros pré-definidos, utilizada para temporizar e simular o comportamento dinâmico do modelo. São quatro os relógios internos:

- unitário (ut), que fornece o menor tempo de duração possível de uma tarefa temporizada;
- de função (ft), período base para o escalonamento das funções selecionadas pelo usuário;
- de rede (rt), período base para o escalonamento das funções temporais da rede;
- de alarme (at) [DR85, Ric85], utilizado como alarme da ocorrência de uma das situações possíveis: quando um bloqueio é atingido ou quando a ut, rt ou o ft precisam ser ajustados, ou seja, quando há um número de tarefas e funções em processamento concorrente superior ao que a CPU pode suportar, sem que a análise temporal seja comprometida. Para tal verificação, um gráfico em barra está disponível para a observação do comportamento das filas de controle.

A Figura 3.33 apresenta a janela de edição das constantes base de tempo geradas a partir do RTC.

Contadores de Eventos

Os contadores de eventos são propriedades dos lugares que, quando configurados, podem fornecer o número de fichas por um período pré-estabelecido ou o tempo de permanência destas nos lugares. Tais medidas são úteis na avaliação de desempenho do modelo.

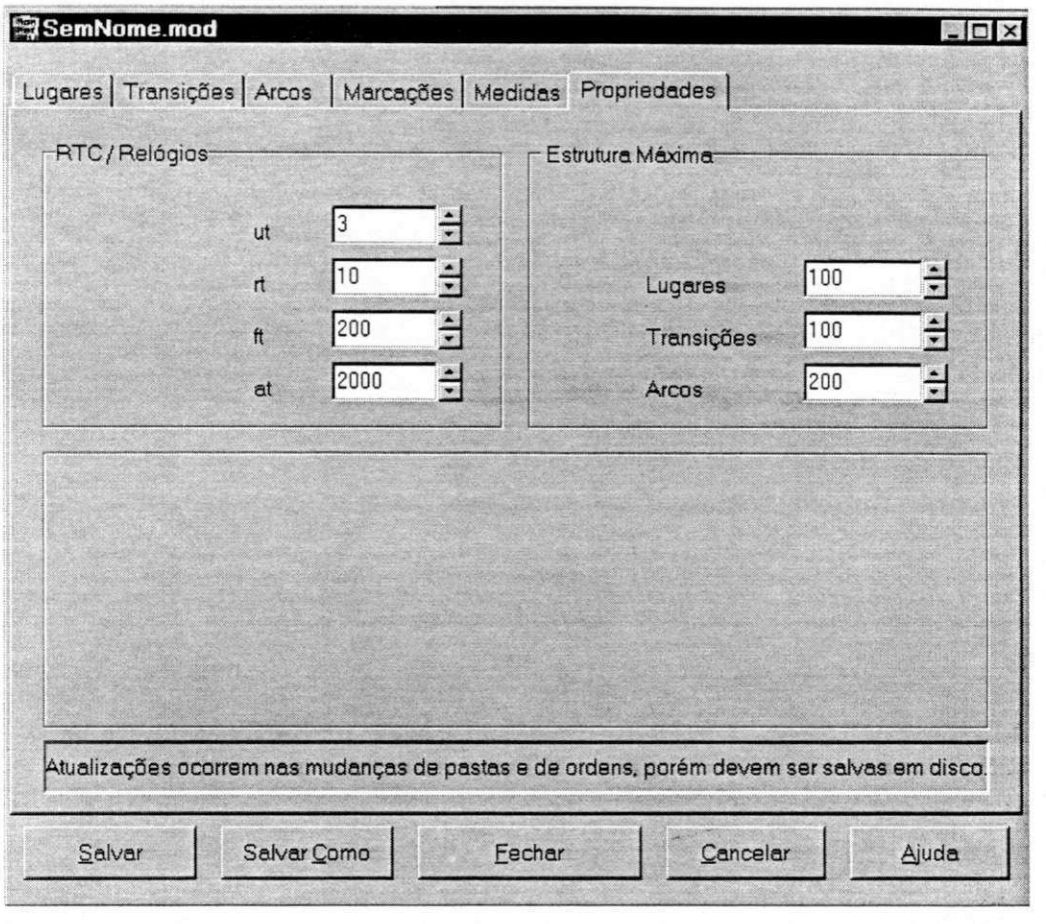


Figura 3.33: Edição do RTC/Relógios

3.4 Medidas de Desempenho

Como citado anteriormente, são medidas possíveis para efeito de avaliação de desempenho:

- número médio de fichas em um dado lugar;
- frequência de disparo de uma transição;
- atraso médio de disparo de uma transição;
- tempos médios de tarefa e ociosidade de recursos fixos (máquinas, robôs, etc.), entre outras.

Os resultados dos ensaios poderão ser salvos em arquivos do tipo texto e impresso, através da ativação de um sub-menu *Pop Up*, acionado com o uso do botão direito do *mouse*.

3.5 Interface com o Usuário

Nesta seção são apresentadas as características da interface empregada e os motivos pelos quais foi adotada. As ferramentas de interfaceamento com o usuário baseadas em manipulação do *mouse*, no acionamento de comandos através de ícones, são sem dúvida, as mais utilizadas na atualidade, pela riqueza de suas propriedades:

- as aplicações são bem apropriadas a consulta e ao processamento de dados;
- libera a memória do usuário para outras tarefas (não obrigando-o a memorizar comandos que venham a utilizar ocasionalmente) e permite fácil interação com novos usuários;
- utiliza dispositivos apontadores e de seleção (teclas de movimento, *mouse*, teclas de comando (*return*, *enter*, etc.) e teclas aceleradores (de função);

- há redução do esforço de digitação (logo reduzindo os erros);
- é possível a apresentação de todas as operações disponíveis, situando o usuário no contexto de seu trabalho;
- rápida realimentação/certificação ao usuário de que o comando escolhido foi "aceito" ou que está em processamento;
- permite o uso de dispositivos alternativos de entrada (indexação, letras chaves destacadas no comando, entre outros);
- porém, é restritiva em não permitir ao usuário mais experiente "saltar" etapas, mediante o uso do agrupamento de comandos (sequência que utiliza costumeiramente), exigindo tempo e esforço ainda significativos.

3.6 Ajuda On Line

Nesta seção será apresentada uma breve descrição dos recursos de ajuda ao usuário quanto ao uso da ferramenta ManNet.

A Ajuda *On Line* consiste de um programa de ajuda a ser acessado à partir do submenu *?* presente no menu principal, conforme apresenta a Figura 3.34, ativando-se o comando *Ajuda* ou acionando o ìcone *?* presente na barra de ìcones.

Acionada a *Ajuda*, uma janela sensível ao contexto é apresentada, composta de menu principal, através do qual é possível realizar diversas operações com arquivos, localizar palavras, palavras chaves, imprimir tópicos de ajuda e *navegar* avançando pelo utilitário, acionando palavras sensíveis ao contexto e percorrer o caminho realizado de volta.

Outras formas de se obter ajuda estão presentes em quase todas as janelas da ferramenta ManNet. Estão disponíveis, ainda, janelas de mensagens de recomendação, de erro e de confirmação de procedimento. Estas mensagens *pop up* são emitidas sob o controle do programa e quando solicitadas pelo usuário.

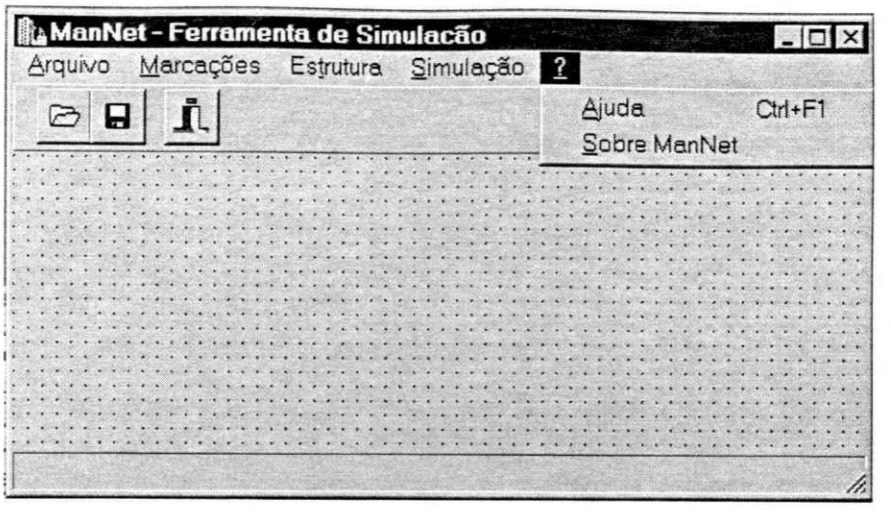


Figura 3.34: Submenu de Ajuda

No Anexo 1 encontra-se uma descrição mais detalhada do Ajuda *On Line*.

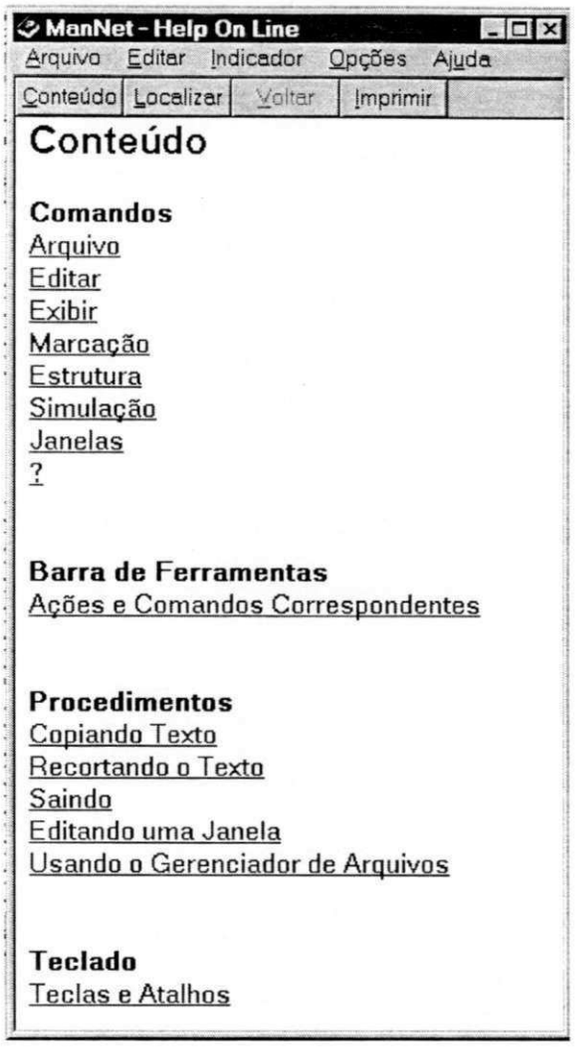


Figura 3.35: Ajuda *On Line*

Capítulo 4

Exemplos

4.1 Introdução

Neste capítulo são apresentados dois exemplos de aplicação da ferramenta ManNet. O primeiro exemplo apresenta a modelagem e análise de uma célula de manufatura. No segundo exemplo é discutido um protocolo de comunicação do tipo passagem de ficha.

4.2 Célula de Manufatura

Para exemplificar a utilização da ferramenta ManNet são apresentados nas Figuras 4.1 e 4.2 uma célula de manufatura e seu modelo usando redes de Petri, respectivamente. A célula de manufatura compreende duas linhas seriais de produção, F e G, que compartilham entre si robôs e uma via de retorno para seus veículos autoguiados. Apresenta, ainda, cinco máquinas operatrizes, cinco robôs e dois armazenadores temporários (*buffers*) de capacidades limitadas. Mediante a figura 4.1, pode ser observado que:

- entrada por E_1, uma peça do tipo F pode ser trabalhada pela máquina M_1 ou M_2, depois por M_4 diretamente (se disponível ou não); e, finalmente, transportada como produto até a saída S_1;

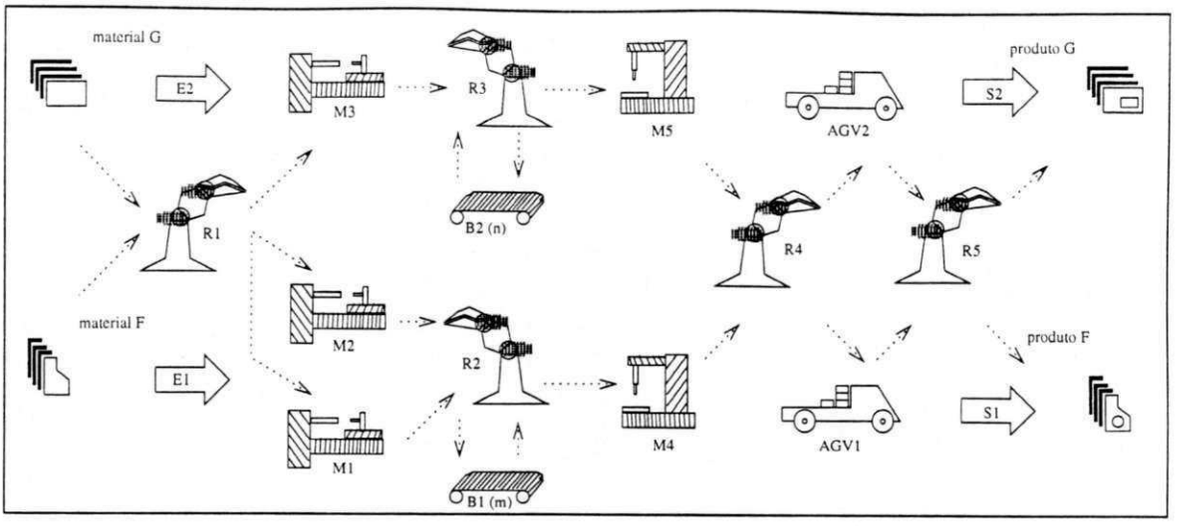


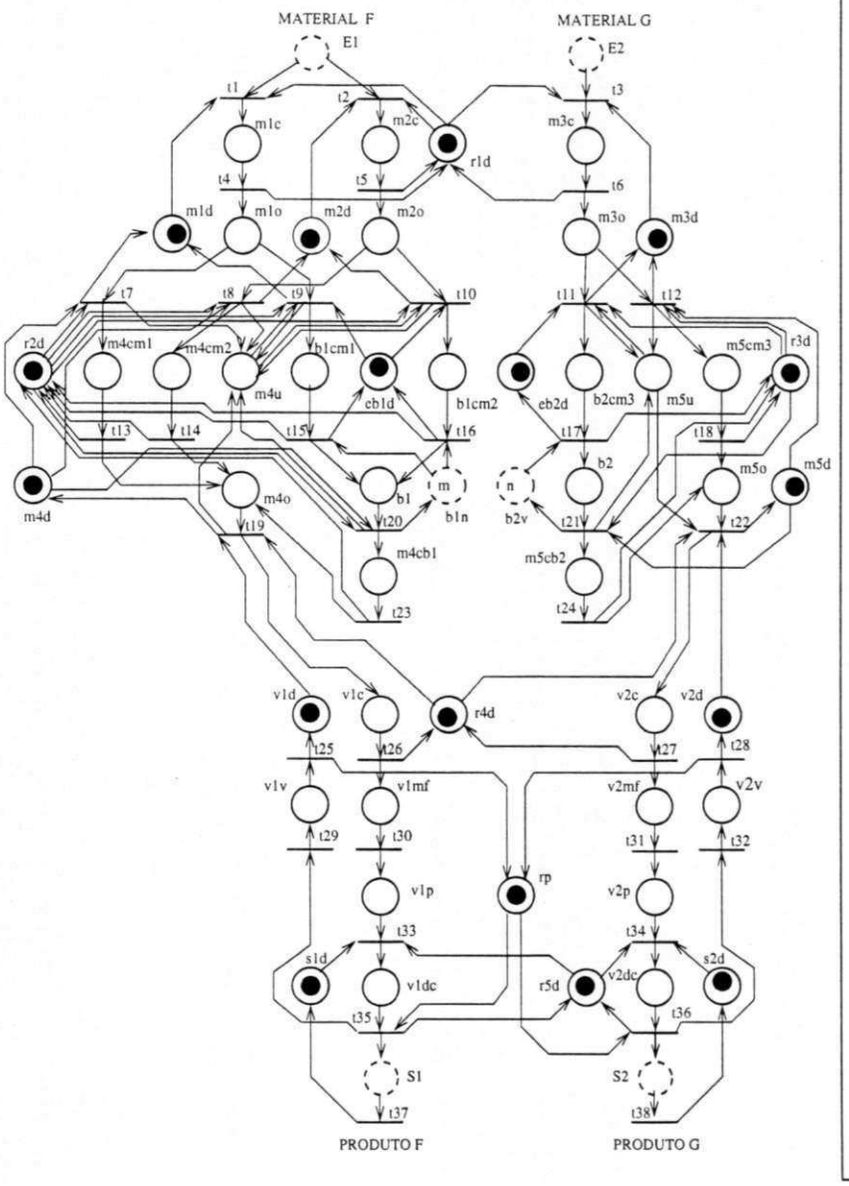
Figura 4.1: Célula de Manufatura

- entrada por E_2, uma peça do tipo G é trabalhada pela máquina M_3, depois pela M_5 e depois transportada até a saída S_2; e
- as máquinas M_4 e M_5 possuem dois elementos de armazenamento temporário de peças (*buffers*), de capacidades m e n, respectivamente, que são utilizados para receber peças vindas de outras máquinas quando não se encontram disponíveis ou livres e que, geralmente, são utilizados para ajustar velocidades (sincronismo) entre duas etapas de produção (evitando-se assim retenções e superdimensionamentos).

4.2.1 Propriedades Estruturais e Temporais

No presente exemplo buscou-se verificar as taxas de ocupação das máquinas M_1, M_2 e do robô R_1, entre outras informações, sem a dependência das propriedades da máquina M_4, da velocidade do AGV_1 e do consumo (ou saída) em S_1. A Figura 4.3 apresenta a janela principal do simulador após a simulação da produção de duzentos produtos na linha F, utilizando-se as duas linhas simultaneamente.

Para a simulação deste exemplo e do que se segue, foram utilizadas versões dedicadas



LUGAR:		TIPO:
E1:	entrada de material tipo F	C
E2:	entrada de material tipo G	C
m1c:	máquina M1 em carga	A
m2c:	máquina M2 em carga	A
r1d:	robô R1 disponível	B
m3c:	máquina M3 em carga	A
m1d:	máquina M1 disponível	B
m1o:	máquina M1 em operação	A
m2d:	máquina M2 disponível	B
m2o:	máquina M2 em operação	A
m3o:	máquina M3 em operação	A
m3d:	máquina M3 disponível	B
r2d:	robô R2 disponível	B
m4cm1:	máquina M4 em carga de M1	A
m4cm2:	máquina M4 em carga de M2	A
m4u:	máquina M4 em uso	A
b1cm1:	buffer B1 em carga de M1	A
eb1d:	entrada do buffer B1 disponível	B
b1cm2:	buffer B1 em carga de M2	A
eb2d:	entrada do buffer B2 disponível	B
b2cm3:	buffer B2 em carga de M3	A
m5u:	máquina M5 em uso	B
m5cm3:	máquina M5 em carga de M3	A
r3d:	robô R3 disponível	B
m4d:	máquina M4 disponível	B
m4o:	máquina M4 em operação	A
b1:	buffer B1	C
b1v:	locações do buffer B1 vazias	C
b2v:	locações do buffer B2 vazias	C
b2:	buffer b2	C
m5o:	máquina M5 em operação	A
m5d:	máquina M5 disponível	B
m4cb1:	máquina M4 em carga de buffer B1	A
m5cb2:	máquina M5 em carga de buffer B2	A
v1d:	veículo V1 disponível	B
v1c:	veículo V1 em carga	A
r4d:	robô R4 disponível	B
v2c:	veículo V2 em carga	A
v2d:	veículo V2 disponível	B
v1v:	veículo V1 voltando	B
v1mf:	veículo V1 em movimento p/ frente	A
v2mf:	veículo V2 em movimento p/ frente	A
v2v:	veículo V2 voltando	A
v1p:	veículo V1 parado	B
rp:	retorno permitido	B
v2p:	veículo V2 parado	B
s1d:	saída S1 disponível	B
v1dc:	veículo V1 em descarga	A
r5d:	robô R5 disponível	B
v2dc:	veículo V2 em descarga	A
s2d:	saída S2 disponível	B
S1:	saída de produto tipo F	C
S2:	saída de produto tipo G	C

Figura 4.2: Modelo em Redes de Petri da Célula de Manufatura

da ferramenta Petrilyser utilizando rotinas não orientadas ao objeto da ferramenta ManNet, por a primeira ser bastante limitada quanto a avaliação de desempenho e por na segunda ainda não terem sido vencidos os problemas de gerenciamento do RTC[1]. Foram, então definidas as seguintes propriedades:

- manufatura total na linha F prevista para duzentas peças;
- baixa produção na linha G, ou seja, duração de operação da máquina M_3 de cinquenta ut's;
- taxas de produção nas entradas E_1 e E_2 de 1 peça a cada 20 e 10 ut's, respectivamente, sem limitação de capacidade (dos lugares que as modelam);
- buffers B_1 e B_2 de capacidade elevada (vinte cada, para não interferirem na avaliação);
- tempo de duração das operações das máquinas M_1 e M_2 de trinta e vinte ut's máximos, respectivamente, com função de densidade de probabilidade uniforme entre os intervalos de uma ut e seus respectivos valores máximos[2]; e
- transições (todas) atômicas.

No lado direito, os indicadores informam que:

[1]No *Windows 95* não se pode garantir a interatividade em tempo real devido aos níveis de prioridade de interrupção cujo gerenciamento não é acessível ao usuário. Neste *Windows*, para solucionar este problema (que basicamente paralisou o desenvolvimento da ferramenta ManNet) são permitidas duas soluções:

- utilizar um processador auxiliar que trabalhe totalmente transparente ao Windows 95 e que processe o escalonamento e a simulação das execução das tarefas; ou
- reescrever parte dos programas fonte, reorganizando-os, de forma que os procedimentos diretamente relacionados com o escalonamento e a interatividade sejam compilados com o nível mais baixo possível da estrutura de dados e programas gerenciada pelo Windows 95 ou *VBX*.

[2]Neste caso, as máquinas sempre operam, ou melhor, nunca falham.

- oitenta e nove peças foram trabalhadas pela máquina M_1 e cento e onze, na máquina M_2; e
- por exclusão, estas máquinas estiveram trinta e três e quarenta e quatro porcentos do tempo total gastos fora de operação, respectivamente;
- enquanto o robô R_1 esteve vinte e dois porcento disponível.

Os dados de uma simulação somente fazem sentido quando o que se deseja avaliar está claro ou bem definido e são excluídas totalmente interferências de outras variáveis ou elementos cujas propriedades não podem interferir diretamente nos dados em análise. Como exemplo as propriedades definidas para os elementos existentes na linha F, a partir das máquinas M_1 e M_2.

Como foi observado, a taxa de produção deste conjunto de máquinas M_1 pode ser aumentada, considerando-se que o robô R_1 possui significativo percentual de disponibilidade sem prejuízos para a segunda linha de produção (que também é atendida por R_1).

Até quanto pode crescer esta produção, como também otimizar a linha G, são questões que podem ser resolvidas, possivelmente, com a *Teoria das Filas* à partir das taxas de *nascimento* e *morte* das peças, porém, torna-se mais prático fazê-lo de forma interativa, com o auxílio da ferramenta ManNet, pela facilidade de se alternar entre um modelo e outro e realizar rapidamente comparações entre os resultados obtidos.

Como apresenta o exemplo, várias questões comuns a ambientes de manufatura são de difícil solução algébrica [ZZ94, Taz95], porém possíveis de análise com a simulação interativa. Como exemplos podem ser citados:

- a necessidade da avaliação do desempenho em função da taxa de entrada de peças;
- a simulação de falhas e manutenção preventiva de máquinas;
- a determinação da taxa de contenção de máquinas (tais como M_1, M_2 e M_3) em função do tamanho dos *buffers* de estocagem ou sincronismo (B_1 e B_2, por exemplo);

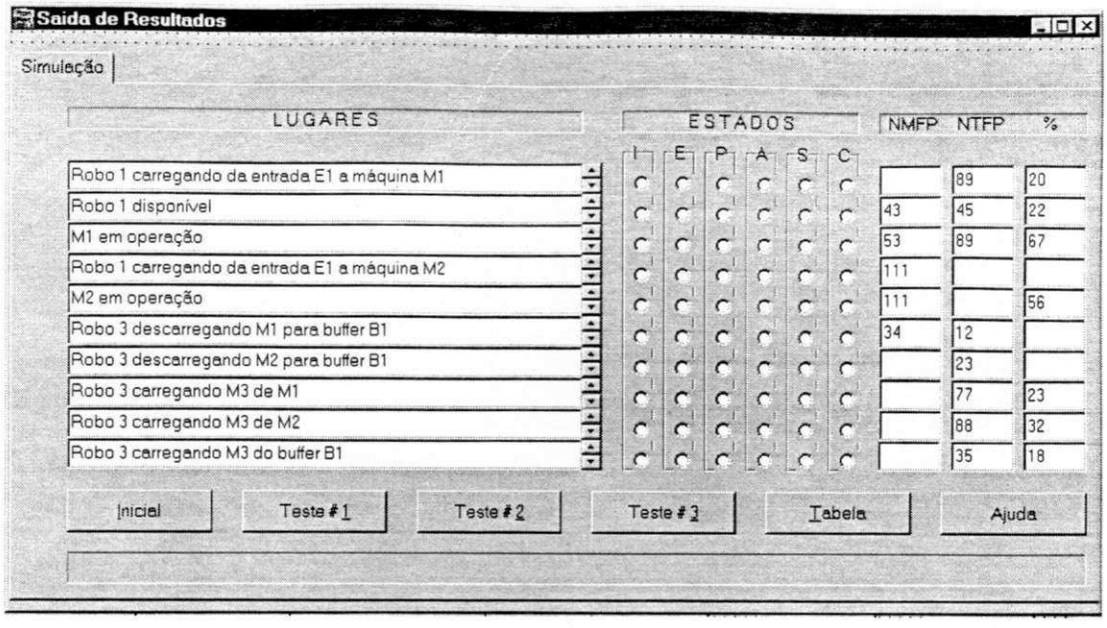


Figura 4.3: Janela com os resultados do Simulador

- a identificação dos tamanhos otimizados dos *buffers* para uma estimada produção, ou vice versa; e
- a determinação do número de veículos necessários em função dos limites de velocidade e de caminhos mutualmente exclusivos e vice versa, entre outras.

4.3 Protocolo de Comunicação

A segunda proposta de exemplo de aplicação da ferramenta ManNet trata-se da avaliação de um protocolo de passagem de permissão (*token-passing*). Estes protocolos estabelecem que um sub-sistema para fazer uso de um dado recurso (que pode ser um canal de comunicação ou mesmo um dado de acesso mutuamente exclusivo) deve estar de posse de uma "permissão" que, em alguns casos, é "administrada" por um dos sub-sistemas que compoem o sistema como um todo. Estes protocolos, também descrevem os procedimentos básicos de aquisição e fornecimento da "permissão", uma vez que todos os sub-sistemas possuem, *a priori*, os mesmos direitos de acesso ao recurso.

A Figura 4.4 apresenta o modelo de um algoritmo de passagem de ficha (*Token-*

Passing Mutex Algorithm)[Rei85], que pode ser confiável (determinístico) ou não. O conceito básico deste algoritmo é que dois sub-sistemas, dispostos lado a lado, acessam um recurso mutuamente exclusivo controlado por meio de uma "permissão", por ambos disputada. O lado que possuir a "permissão" possui, então, o direito de acesso ao recurso.

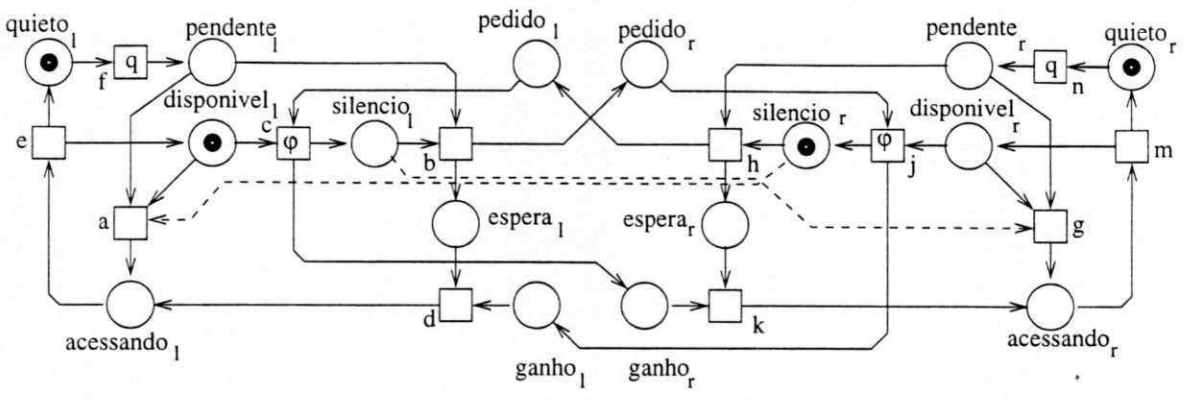


Figura 4.4: Um Algoritmo Determinístico para o protocolo de Passagem de Ficha

As seguintes considerações são possíveis, se não forem considerados os arcos tracejados:

- os lados são estruturalmente simétricos, mas os estados iniciais não o são;
- não é possível a associação a uma máquina de estados sequenciais;
- dois pares de transições ou ações (f, c) e (j, r) podem ocorrer concorrentemente;
- se o lado l possui a "permissão" e está disponível, as transições a e c podem estar em conflito;
- da mesma forma com j e g, em respeito ao lado r;
- então, algumas transições podem nunca disparar, ou seja, é possível que um lado *monopolize* a "permissão" por longos períodos de tempo, não permitidos em determinados níveis do protocolos de comunicação[3];

[3]Neste exemplo não é possível uma análise satisfatória do comportamento se não forem incluídos

- é possível aplicar probabilidades condicionais a atividades em conflito. Por exemplo, substituir automaticamente a probabilidade de solicitar a "passagem" a esquerda pela maior, se a última passagem feita foi a esquerda (α, tal que $\alpha + \beta = 1$);

Agora, considerando os arcos tracejados, temos um algoritmo mais justo no que se diz respeito a análise de comportamento e ao direito mútuo de acesso ao recurso. Neste caso, fica assegurado que o risco de monopólio da ficha inexiste e que o maior atraso que um lado pode ter em obter a "permissão" é pouco superior ao maior tempo de acesso ao recurso por vez. Assim sendo, são possíveis a atribuição de tempos de permanência aos estados, de resposta de reconhecimento de pedido e passagem da "permissão", estabelecer alarmes, taxas de ciclo, etc., e a observação do comportamento do algoritmo através da simulação interativa.

Finalmente o exercício de simulação e análise realizadas não permitiu chegar a conclusão satisfatória ou algum resultado próximo do que se esperava obter. É possível supormos que, pela definição formal das redes de Petri, uma transição habilitada *pode disparar ou não*, o que não é usual para o modelo. Em outras palavras, no primeiro modelo, sem taxas não nulas de propabilidade de acesso a ficha, um sistema poderá ser impedido de utilizar o canal de comunicação mesmo que o outro não deseje fazê-lo. Neste caso, o primeiro modelo pode ser dito impróprio para este método de análise.

4.4 Medidas de Desempenho

Apesar dos esforços realizados na tentativa de apresentar resultados diversos e comparações entre alternativas de configuração nos exemplos propostos (e seus modelos apresentados nas seções anteriores ... principalmente pela falta de recursos e pelo tempo que seria necessário para a adaptação da ferramenta para o ambiente Windows

dados que definam o comportamento estocástico das atividades ou transições concorrentes e em conflito.

95 e garantir a interatividade a tempo real). Pelos motivos já mensionados foi desenvolvida versões dedicadas da ferramenta Petrilyser para processar os exemplos propostos. Para a rede exemplo de Molloy (Figura 3.6) estabelecemos propriedades que permitam a comparação entre os resultados encontrados por Molloy[Mol82a] e os obtidos pela ferramenta Petrilyser/ManNet[4].

[4]Lembramos que a segunda é complementar à primeira

Capítulo 5

Conclusões

A proposta de simulação de Sistemas Complexos modelados em redes de Petri é uma solução versátil para tais sistemas a eventos discretos, nos quais a análise, o planejamento e a avaliação de desempenho são tarefas árduas, que exigem modelos matemáticos descritivos complexos, principalmente, por seus componentes possuirem comportamentos estocásticos diversos. Portanto, apoiada na evolução da *Engenharia de Software*, dos processadores e das extensões (e aplicações) das redes de Petri, as ferramentas de simulação estão, cada vez mais, trazendo agilidade e praticidade na solução dos problemas de Engenharia, Processamento e Comunicação de Dados e em outras áreas tecnológicas.

Este trabalho apresentou a ferramenta Petrilyser, uma ferramenta baseada em redes de Petri para auxiliar no processo de modelagem, simulação e análise de sistemas complexos; ainda, uma proposta de ferramenta, sucessora da Petrilyser denominada ManNet que, além dos recursos já apontados promoveria a simulação da rede de Petri com amplos recursos temporais e a interatividade em tempo real com o usuário. Foram apresentados, finalmente, exemplos de aplicação nas áreas da Engenharia de Manufatura e na de Comunicação de Dados.

Apesar de que as medidas propostas nos exemplos apresentados possam ser obtidas mediante o uso de métodos analíticos, estamos certos de que a simulação aliada

aos recursos de ativação, desativação, o ajuste de tempos e das taxas de duração de execução das tarefas, a remoção e o acréscimo de fichas, de forma interativa e a avaliação de desempenho em tempo real, resultam em tarefas que exige menos esforços e qualificacação técnica para aplicações e obtenção de resultados.

Considerando a amostragem e os procedimentos de contagem de eventos, não são esperados resultados milimetricamente precisos da ferramenta porque são possíveis perdas por tempo de processamento não recuperáveis durante o escalonamento, em períodos do RTC. Algumas fontes de erro podem ser evitadas, mediante a configuração otimizada dos relógios, uma habilidade que para grandes redes somente será alcançada com a prática do uso da ferramenta. No caso de pequenas redes, estes problemas não deverão ocorrer. Como a análise, geralmente, é realizada sobre médias após repetidas simulações, grande parte destas incertezas serão minimizadas, senão inexistentes. Pela média dos resultados, poderão ser encontrados os valores a serem considerados. Nosso trabalho permite propor novos estudos ou complementares, tais como:

- ferramenta para a simulação de multi-células agrupadas;
- simulação de um sistema de supervisão;
- ferramenta para simulação de métodos de escalonamento de diversos tipos de tarefas e sistemas, entre outras; e
- com a padronização das redes de Petri (que espera-se que ocorra ainda este ano), integrar a ferramenta ManNet com outras ferramentas.

Seria ainda de interesse introduzir a possibilidade da ferramenta tratar as rede de Petri com temporização nebulosa e desenvolver a interface gráfica. Contudo, muitas destas sugestões dependem da padronização comentada.

Bibliografia

[ABD+84] P. Alache, K. Benzakour, F. Dollé, P. Gillet, P. Rodrigues, and R. Valette. PSI: A petri net based simulator for flexible manufacturing systems. In G. Rozenberg, editor, *Advances in Petri Nets 1984*, volume 188 of *Lecture Notes in Computer Science*, pages 1–14. Springer Verlag, 1984.

[AM89] M. Ajmone Marsan. Stochastic petri nets: An elementary introduction. In *Advances in Petri Nets 1989*, volume 424 of *Lecture Notes in Computer Science*. Springer-Verlag, 1989.

[AMBC84] M. Ajmone Marsan, D. Balbo, and G. Conte. A class of generalised stochastic petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, 2(2):93–122, May 1984.

[BD91] B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time petri nets. *IEEE Transactions on Software Engineering*, 17(3):259–273, March 1991.

[CLBJM92] C. Chaouiya, S. Lefrebvre-Barbaroux, and A. Jean-MariePapoulis. *Scheduling Theory and Its Applications.* John Wiley and Sons Ltd., INRIA, Centre Sophia Antipolis, 1992.

[dF94] J. C. A. de Figueiredo. *Fuzzy Time Petri Net.* PhD thesis, Universidade Federal da Paraíba - Campus II, Campina Grande, Paraíba, 1994.

[dFP94] J.C.A. de Figueiredo and A. Perkusich. Análise temporal baseada em redes de petri para sistemas de software. In *Anais do XIX Seminário Integrado de Software e Hardware, SEMISH 94*, August 1994.

[dFP95a] J.C.A. de Figueiredo and A. Perkusich. Distributed control of track-vehicle system with fault-tolerant characteristics: a petri net based approach. In *Proc. of IEEE Int. Conf. on Systems Man and Cybernetics*, pages 377–382, Vancouver, Canada, October 1995.

[dFP95b] J.C.A de Figueiredo and A. Perkusich. Fault tolerance in real-time distributed systems using petri nets extension. *Journal of Computing and Information*, 1(2):924–946, November 1995.

[dFP95c] J.C.A de Figueiredo and A. Perkusich. Tratamento antecipado de falhas: Uma abordagem por redes de petri. In *Proceeding of 6th Simpósio Brasileiro de Tolerância a Falhas*, pages 125–142, Canela, RS, August 1995.

[dFP96] J.C.A. de Figueiredo and A. Perkusich. Faults and timing analysis in real-time distributed systems: A fuzzy time petri-net-based approach. *International Journal Fuzzy Sets and Systems*, 83(2):143–168, 1996.

[dFP97] J.C.A. de Figueiredo and A. Perkusich. Towards a modular timing analysis of real-time software system. In *Proc. of IEEE Int. Conf. on Systems Man and Cybernetics*, pages 4442–4447, October 1997.

[dFPC94] J.C.A. de Figueiredo, A. Perkusich, and S.K. Chang. Timing analisys of real-time software systems using fuzzy time petri nets. In *Proc. of The Sixth International Conference on Software Engineering and Knowledge Engineering*, pages 243–253, Riga, Latvia, June 1994.

[dFPC95] J.C.A de Figueiredo, A. Perkusich, and S.K. Chang. Anticipated faults in real-time distributed systems. In *Proc. of The Seventh International Con-*

ference on Software Engineering and Knowledge Engineering, SEKE'95, pages 411–418, Washington, USA, June 1995.

[dFPM93] J.C.A. de Figueiredo, A. Perkusich, and M.E. Morais. Tolerância a falhas em sistemas de software utilizando uma abordagem por redes de petri. In *Anais do V Simpósio de Computadores Tolerantes a Falhas*, October 1993.

[DR85] M. Didie and G. Richter. Time and clocks and task management. *Proceedings of International Workshop on Timed Petri Nets*, pages 1–10, 1985.

[DTGN84] J.B. Dugan, K.S. Trivedi, R.M. Geist, and V.F. Nicola. Extended stochastic petri nets: Applications and analysis. In *Performance '84*, December 1984.

[GMMP89] C. Ghezzi, D. Mandrioli, S. Morasca, and M. Pezze. A general way to put time in petri net. In *5th International Workshop on Software Specifications and Design*, pages 60–66, May 1989.

[HS86] P.J. Haas and G.S. Shedler. Regenerative stochastic petri nets. *Performance Evaluation*, 6(3):189–204, September 1986.

[Jen92] K. Jensen. *Coloured Petri Nets: Basic Concepts, Analysis, Methods and Practical Use.* EACTS – Monographs on Theoretical Computer Science. Springer-Verlag, 1992.

[Kle75] L. Kleinrock. *Queueing Systems Volume I: Theory.* John Wiley, 1975.

[LdFP96] I. de S. Lima, J.C.A. de Figueiredo, and A. Perkusich. An interactive petri net tool for modeling, analysis and simulation of complex systems. In *Proc. of IEEE Int. Conf. on Systems Man and Cybernetics*, pages 870–875, Beijing, China, October 1996.

[LF85] K.H. Lee and J. Favrel. Hierarchical reduction method for analysis and decomposition of petri nets. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-15(2):272–280, March 1985.

[MB83] M. Menasche and B. Berthomieu. Time petri nets for analyzing and verifying time dependent protocols. In *3rd International Workshop on Protocol Specification, Testing and Verification*, pages 161–172, June 1983.

[Mer79] P.M. Merlin. Specification and validation of protocols. *IEEE Transactions on Communications*, COM-27(11):1671–1680, November 1979.

[MF76] P.M. Merlin and D.J. Farber. Recoverability of communication protocols - implications of a theoretical study. *IEEE Transactions on Communication*, COM-24(9):1036–1043, September 1976.

[MJ92] K. Majmudar and M.A. Jafari. Functional and performance analysis of time petri nets. In *International Conference on Systems, Man, and Cybernetics*, volume 2, pages 980 – 985, October 1992.

[ML87] D. Marinescu and C. Lin. On stochastic high-level petri nets. In *International Workshop on Petri Nets and Performance Models*, August 1987.

[Mol81] M.K. Molloy. *On the Integration of Dealy and Throughput Measures in Distributed Processing Models.* PhD thesis, UCLA, 1981.

[Mol82a] Michael K. Molloy. Performance analysis using stochastic petri nets. *IEEE Transactions on Computers*, C-31(9):913–917, 1982.

[Mol82b] M.K. Molloy. Performance analysis using stochastic petri nets. *IEEE Transactions on Computers*, c-31(9):913–917, September 1982.

[Mur89] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[MV91] G. Memmi and J. Vautherin. Analysing nets by the invariant method. In G. Rozenberg, editor, *High-Level Petri Nets: Theory and Application*, pages 247–336. Springer-Verlag, 1991.

[Nat80] S. Natkin. *Les Reseaux de Petri Stochastiques et leur Application a L'evaluation des Sistemes Informatiques.* PhD thesis, These de Docteur Ingegneur, CNAM, 1980.

[Pap65] A. Papoulis. *Probability, Random Variables and Stochastic Processes.* McGraw-Hill, New York, 1965.

[PBdFP91] A. Perkusich, T.C. Barros, J.C.A de Figueiredo, and M.L.B. Perkusich. A petri net based approach for knowledge base construction for fault analysis and control of discrete time systems. In *IEEE Industrial Electronics Society Conference IECON'91*, pages 1631–1636, Kobe, Japan, November 1991.

[PBP+91] A. Perkusich, T.C. Barros, M.L.B. Perkusich, D.S. Barbalho, and J.C.A. de Figueiredo. Knowledge based systems application to implement petri net models of discrete time systems. In *IFIP Working Conference on Dependability of Artificial Inteligence Systems*, Viena, Austria, May 1991.

[PdF95] A. Perkusich and J.C.A. de Figueiredo. A g-net based environment for logical and timing analysis of software system. In *Anais do SBES'95, Simpósio Brasileiro de Engenharia de Software*, pages 56–75, Recife, PE, October 1995.

[PdF97] A. Perkusich and J.C.A. de Figueiredo. G-nets: A petri net based approach for logical and timing analysis of complex software systems. *Journal of Systems and Software*, 39(1):39–59, 1997.

[PdFC94] A. Perkusich, J.C.A. de Figueiredo, and S.K Chang. Embedding fault-tolerant properties in the design of complex systems. *Journal of Systems and Software*, 2(25):23–37, 1994.

[PdFM93] A. Perkusich, J.C.A. de Figueiredo, and M.E. Morais. Projeto de sistemas em tempo real distribuídos com característica baseada em objetos e tolerância a falhas. In *Anais do XIX Seminário Integrado de Software e Hardware, SEMISH 93*, September 1993.

[Pet81] J.L. Peterson. *Petri Net Theory and Modeling of Systems.* Prentice-Hall, 1981.

[PPC96] A. Perkusich, M.L.B. Perkusich, and S.K Chang. G-nets: A petri net based approach for logical and timing analysis of complex software systems. *International Journal of Software Engineering and Knowledge Engineering*, 6(3):447–476, 1996.

[Ram74] C. Ramchandani. Analysis of asynchronous concurrent systems by petri nets. Technical Report Project MAC-TR120, M.I.T., Cambridge, MA, 1974.

[Rei85] W. Reisig. *Petri Nets: An Introduction.* Springer-Verlag, 1985.

[Rei87] W. Reisig. Place/transition systems. In W Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Central Models and their Properties, Proc. of 2nd Advanced Course on Petri Nets*, volume 254 of *Lecture Notes in Computer Science*, pages 117–141. Springer-Verlag, 1987.

[RH80] C.V. Ramamoorthy and G.S. Ho. Performance evaluation of asynchronous concurrent systems using petri nets. *IEEE Transactions on Software Engineering*, SE-6(5):440–449, 1980.

[Ric85] G. Richter. Clocks and their use for time modeling. *Information Systems*, pages 49–66, 1985.

[Sif80] J. Sifakis. Performance evaluation of systems using nets. In *Net Theory and Applications*, volume 84 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[Sil95] M. Silva. Interleaving functional and performance analysis of net models. In *Course Notes of The 2nd International Course on Petri Nets for Latin America*, Campina Grande, Pb, Brasil, 1995.

[SJ92] V.S. Srinivasan and M.A. Jafari. Fault detection/monitoring using time petri nets. submitted to IEEE Transactions on Systems, Man and Cybernetics, 1992.

[SL89] I. Suzuki and H. Lu. Temporal petri nets and their application to modeling and analysis of a handshake daisy chain arbiter. *IEEE Transactions on Computers*, 38(5):696–704, May 1989.

[SM91] Ye Souissi and G. Memmi. Composition of nets via a communication medium. In G. Rozemberg, editor, *Advances on Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*, pages 455–470. Springer-Verlag, 1991.

[SP] P.D. Stotts and T.W. Pratt. Coverability graphs for a class of synchronously executed unbounded petri net. to appear in Journal of Parallel and Distributed Computing.

[SP85] P.D. Stotts and T.W. Pratt. Hierarchical modeling of software systems with timed petri nets. In *1st Workshop on Timed Petri Nets*, pages 32 – 39, July 1985.

[Taz95] M. Tazza. Primitivas para modelagem e análise de sistemas de manufatura. *SBA Controle e Automação*, 6:1–12, 1995.

[Val79] R. Valette. Analysis of petri nets by stepwise refinements. *Journal of Computer and Systems Sciences*, 18:35–46, 1979.

[VCD85] R. Valette, M. Corvousier, and C. Desclaux. Putting petri nets to work for controlling flexible manufacturing systems. In *Proc. of IEEE International Symposium on Circuits and Systems, ISCAS 85*, Kyoto, Japan, 1985.

[ZD93] M. Zhou and F. Dicesare. *Petri Net Syntesis for Discrete Event Control os Manufacturing Systems.* Kluwer Academic Publishers, New Jersey, USA, 1993.

[Zen85] A. Zenie. Colored stochastic petri nets. In *1st Workshop on Timed Petri Nets*, July 1985.

[Zub80] W.M. Zuberek. Timed petri nets and preliminary performance evaluation. In *7th Annual Symposium on Computer Architecture*, pages 88–96, May 1980.

[Zub91] W.M. Zuberek. Timed petri nets: Definitions, properties, and applications. *Microelectronics and Reliability*, 31(4):627 – 644, 1991.

[ZZ94] R. Zurawski and M. Zhou. Petri nets and industrial applications: A tutorial. *IEEE Transactions on Industrial Eletronics*, 41(6):567–581, 1994.

Anexo A - Arquivo de Ajuda

A ajuda *On-line* na ferramenta ManNet utiliza um programa executável e um *Arquivo de Ajuda.* Quando ativada a ajuda uma janela é apresentada, a qual está dividida em duas partes: a superior, contendo um conjunto de controle; e a inferior, que contém as informações gerais sobre o uso da ferramenta ManNet.

Os botões de controle permitem localizar informações através de palavras-chave e por índice, e a impressão destas, entre outras opções, comuns aos programas atualmente baseados na interface *Windows.*

As informações de ajuda são mensagens que, organizadas em várias páginas ou janelas, são selecionadas através de comandos nestas contidos, além do texto em si. Estes comandos, sensíveis ao contexto, uma vez selecionados através do *mouse*, permitem visualizar todas as janelas existentes, uma a uma, *navegando-se* entre elas com muita facilidade.

Os comandos são distintos por estarem representados por *ícones* ou por palavras estampadas em cor ou intensidade mais visível do que as de texto. Para se obter informações sobre um comando ou procedimento apresentado no vídeo (e ativar uma nova janela), deve-se posicionar o cursor sobre o mesmo e pressionar o botão esquerdo do *mouse* (*LeftClick*). A seguir apresentamos o conteúdo de suas principais janelas e o que realizam seus principais menus, sub-menus e comandos.

Conteúdo: (Descreve os menus da janela principal da Ferramenta ManNet)

- **Comandos**

Arquivo Marcações Estrutura Simulação ?

- **Barra de Ferramentas:**

  Ações e Comandos Correspondentes

Procedimentos: Saindo Usando o Gerenciador de Arquivos

Teclado: Teclas e Atalhos

Sub-menu *Arquivo*: O sub-menu *Arquivo* apresenta comandos que possibilitam operações com arquivos de modelos. São seus comandos: *Novo*, para criar um novo arquivo. *Abrir*, para abrir um arquivo existente em disco. *Fechar*, para encerrar a edição do arquivo corrente. *Salvar*, para salvar o arquivo corrente se seu conteúdo apresenta mudanças. *Salvar Como*, para salvar o corrente arquivo com um novo nome. *Sair*, para finalizar a ferramenta ManNet.

Sub-menu *Marcações*: O sub-menu *Marcações* permite a visualização da marcação *Atual* (ou corrente) da rede de Petri, a edição e trocas entre as marcações inicial e de teste, e a obtenção da próxima marcação (após o disparo de uma transição ou de uma sequência de transições. Lembramos que, neste caso, a rede de Petri assume (e mantém) a marcação resultante (*Atual)* se e somente se a transição ou sequência for disparada com sucesso, e que o sub-menu *Marcação | Final* pode ser utilizado na obtenção da Árvore de Alcançabilidade das redes de Petri.

Sub-menu *Estrutura*: Este sub-menu permite apresentar a matriz de incidência e a árvore de cobertura em janelas/arquivos do tipo texto. A árvore de cobertura pode ser apresentada no formato Murata[Mur89] ou no formato ManNet[1][LdFP96], que apresenta veto res que representam os estados do *Grafo de Alcançabilidade* das redes de Petri, entre outras informações.

As janelas criadas poderão ter seus conteúdos salvos em disco, através do sub-menu *Pop Up*, que é apresentado quando um comando de *RightClick*, ou toque no botão direito do *mouse* dando acesso ao comando *Salvar*, ou ainda, ao comando *Abrir* e ao *Imprimir*.

Sub-Menu Simulação: O menu Simulação permite configurar parâmetros necessários à simulação das redes de Petri, iniciá -lo, proceder a coleta de dados e apresentar as estatísticas obtidas.

? O menu ? permite acessar comandos de ajuda da ferramenta e a janela de diálogo Sobre ManNet.

[1]Neste formato, a cada ma rcação ou estado é associado um vetor com campos, onde são registradas as transições cujos disparos levaram-na a este estado, a ordem da marcação corrente, da(s) antecedente(s) e seu tipo.