JORGE CESAR ABRANTES DE FIGUEIREDO

Tese apresentada ao Curso de Pós-Graduação em Engenharia Elétrica da Universidade Federal da Paraíba, em cumprimento às exigências para obtenção do grau de Doutor em Ciências

MISAEL ELIAS DE MORAIS
SHI-KUO CHANG
Orientadores

CAMPINA GRANDE - PB
AGOSTO - 1994

Rede de Petri com Temporização Nebulosa

por

Jorge Cesar Abrantes de Figueiredo

Tese

Doutor

Universidade Federal da Paraíba

1994

F475r Figueiredo, Jorge César Abrantes de.
Rede de petri com temporização nebulosa / Jorge César Abrantes de Figueiredo. - Campina Grande, 1994.
139 f.

Tese (Doutorado em Engenharia Elétrica) - Universidade Federal da Paraíba, Centro de Ciências eTecnologia, 1994.
Referências.
"Orientação : Prof. Dr. Misael Elias de Morais, Prof. Dr. Shi-Kuo Chang".

1. Redes de Petri. 2. Temporização Nebulosa. 3. Redes de Computadores. 4. Tese - Engenharia Elétrica. I. Morais, Misael Elias de. II. Chang, Shi-Kuo. III. Universidade Federal da Paraíba - Campina Grande (PB). IV. Título

CDU 004.7(043)

Universidade Federal da Paraíba - UFPB
Centro de Ciências e Tecnologia - CCT
Coordenação de Pós-Graduação em Engenharia Elétrica - COPELE

PARECER FINAL DO JULGAMENTO DA TESE DO DOUTORANDO

JORGE CESAR ABRANTES DE FIGUEIREDO

TÍTULO: "Redes de Petri com Temporização Nebulosa"

CONCEITO: APROVADO

COMISSÃO EXAMINADORA:

Maria de Fátima Q. V. Turnell
PROFa. MARIA DE FÁTIMA QUEIROZ VIEIRA TURNELL, Ph.D., UFPB
Presidente da Comissão

MISAEL ELIAS DE MORAIS, Dr.-Ing., UFPB
Orientador

Shi-kuo Chang
SHI-KUO CHANG, Ph.D., Univ. of Pittsburgh
Co-Orientador

TADAO MURATA, Ph.D., Univ. of Illinois at Chicago

JORGE MOREIRA DE SOUZA, Dr.Etat.,CpQD/TELEBRÁS/UNICAMP

DAVID SIMONETTI BARBALHO, Dr., UFRN

ULRICH SCHIEL, Dr.rer.nat., UFPB

Campina Grande, 23 de agosto de 1994

HOMOLOGADO P/ COLEGIADO
EM: 29 / 08 / 94

Para Adriana, Rafael e Gabriela.

Resumo

Rede de Petri com Temporização Nebulosa

por Jorge Cesar Abrantes de Figueiredo

Nós apresentamos uma extensão para o modelo de redes de Petri, com o objetivo de caracterizar restrições temporais. Nesta extensão, os aspectos positivos das extensões determinísticas e estocásticas são combinados de uma forma complementar, ou seja, a rede de Petri estendida é adequada para modelar sistemas em tempo real e para fazer análise de desempenho de sistemas. A extensão, chamada de *Rede de Petri com Temporização Nebulosa - FTPN*, utiliza uma abordagem baseada na teoria dos conjuntos nebulosos. No modelo *FTPN*, as fichas carregam uma função nebulosa de tempo que indica a possibilidade de sua existência em um determinado lugar em um dado instante de tempo. Intervalos nebulosos de tempo ainda são associados com as transições, propiciando a representação de restrições temporais.

Para definir uma abordagem para análise temporal modular de sistemas complexos, o conceito de *FTPN* é integrado com uma ferramenta de estruturação chamada *G-Net*. Como resultado da integração, nós podemos dividir um sistema complexo em subsistemas que serão estudados isoladamente e os resultados combinados para determinar a solução global.

A ferramenta integrada, juntamente com uma metodologia de análise temporal modular, é utilizada para introduzir propriedades de tolerância a falhas dependentes do tempo no projeto de sistemas distribuídos em tempo real. Nós usamos as *Fuzzy Time G-Nets* para representar esquemas tolerantes a falhas e para considerar falhas, antecipadamente, nos sistemas distribuídos em tempo real. Além do mais, nós apresentamos uma representação gráfica, *TGIG*, para a interação entre *G-Nets*. Esse gráfico representa o tempo de execução para cada *G-Net*, levando em consideração a interação entre *G-Nets* em um sistema de *G-Nets*. A partir desse gráfico, alguns índices de desempenho podem ser facilmente obtidos.

Abstract

Fuzzy Time Petri Net

by Jorge Cesar Abrantes de Figueiredo

We present an extension for the Petri Net model in order to characterize timing constraints. In this extension, the positive aspects of the deterministic and stochastic approaches are combined in a complementary fashion, i.e., the proposed extended Petri net is amenable to model real-time systems and to make performance analysis of systems. The extension, called *Fuzzy Time Petri Net (FTPN)*, uses a fuzzy approach based in the fuzzy set theory. In the *FTPN* model, the tokens carry a fuzzy time function that characterizes the possibility of there being a token in a place in a given instant of time. Also, fuzzy time intervals are associated with transitions to provide ability to represent timing restrictions.

In order to define a modular approach to perform timing analysis of complex systems, the *FTPN* concept is integrated together with a structuring tool called *G-Net*. Thus, as result of the integration, we can divide a complex system in subsystems which are studied in isolation and the results later combined in order to compute the global solution for the entire system.

The integrated tool together with the modular timing analysis methodology is applied in order to introduce time-dependent fault-tolerant properties in the design of real-time distributed systems. We employ *Fuzzy Time G-Nets* to characterize Fault-tolerant schemes and to represent anticipated faults in real-time distributed systems. Moreover, we present a time graphical representation, $TGIG$, for the interaction between *G-Nets*. This time chart represents the time execution of each *G-Net* and takes in consideration the interaction between *G-Nets* in a *G-Net* system. Some performance indices may be easily derived from the graphical representation.

Agradecimentos

Essa Tese é o fim de uma longa jornada de trabalho e ela não seria possível sem a ajuda e a força positiva de muitas pessoas. Em primeiro lugar, gostaria de agradecer aos meus orientadores, Shi-Kuo Chang e Misael Elias de Morais, pelas orientações e discussões. Em especial, gostaria de agradecer a Shi-Kuo Chang que, durante meu período de pesquisa em Pittsburgh, me manteve no caminho certo para terminar essa Tese com seus comentários instrutivos, sua visão e pressão nos momentos importantes. Agradeço ainda as facilidades de acesso a computadores e biblioteca que ele me propiciou no CPDIS.

Pelos valorosos comentários e sugestões, agradeço aos membros da banca examinadora da minha defesa de Tese: Tadao Murata, Jorge Moreira de Souza, David Simonetti Barbalho e Ulrich Schiel.

Agradeço a Angelo Perkusich pelas discussões valorosas que tivemos nestes últimos anos. Sou grato ainda pela sua amizade e apoio em momentos difíceis que passei durante a minha temporada em Pittsburgh.

Agradeço a Albanita Guerra Araújo pelo apoio e pela revisão da versão em português da minha Tese. Agradeço também a Afonso Costa e Silva pela revisão e comentários de dois capítulos da versão em inglês da minha Tese.

Agradeço a Bernardo Lula Júnior pela confiança e apoio, esperando o fim do meu tempo de pesquisa em Pittsburgh.

Durante meu tempo em Pitt, eu tive a ajuda de muitos amigos que tornaram este período muito gratificante. Em particular, agradeço a Arli e Neusa Oliveira pelo constante incentivo e apoio. Gostaria de agradecer a amizade de duas pessoas especiais do CPDIS: C.T. Liu e Mary Conley. Agradeço ainda a Irma de Godoy, Ana Suely, Paulo Flôr e Mrs. B.

Agradeço o suporte financeiro do CNPq através da bolsa 201462/91-5.

Agradeço a meu pai Vicente pela constante orientação, amor e, principalmente, pelo exemplo de vida. Agradeço a minha mãe Mundica pelo carinho, amor e a fé que sempre teve em mim. Agradeço ainda a meus irmãos Giovannini, Jason e Fabrícia pelo amor e apoio.

Por fim, gostaria de dizer que essa Tese não seria possível sem o apoio e paciência da minha esposa Adriana. Eu devo essa Tese ao seu grande amor e à crença que ele teve em meu trabalho, fazendo com que superássemos todos os momentos difíceis dessa

longa jornada. Gostaria de agradecer também a meus filhos Rafael e Gabriela pela compreensão e pelos constantes momentos de alegria que me proporcionaram.

CONTEÚDO

LISTA DE FIGURAS

LISTA DE TABELAS

Capítulo 1
INTRODUÇÃO

1.1 Motivação

O desenvolvimento de sistemas grandes e complexos requer o uso de poderosas ferramentas de análise e modelagem no sentido de tratar a inerente complexidade destes sistemas. Devido a diversos fatores (firme fundamentação matemática e representação gráfica), as redes de Petri têm sido amplamente usadas na modelagem e análise de sistemas complexos. No caso de sistemas em tempo real, nós devemos considerar tanto o aspecto lógico como o temporal. Uma vez que o modelo original de redes de Petri não permite a representação de restrições temporais, diversas extensões foram propostas no sentido de incorporar aspectos temporais ao seu modelo. De uma forma geral, estas extensões são baseadas em uma abordagem determinística ou estocástica em que cada abordagem apresenta vantagens e limitações. De um lado, as extensões determinísticas são adequadas para modelar sistemas em tempo real mas são limitadas para fazer análise de desempenho de sistemas. Por outro lado, as extensões estocásticas são largamente usadas na avaliação de desempenho de sistemas mas não são apropriadas para a modelagem de sistemas em tempo real.

Muitas técnicas foram definidas para as extensões temporais de redes de Petri no sentido de analisar os aspectos temporais dos sistemas. No entanto, não existe uma técnica geral modular baseada em redes de Petri para a análise temporal dos sistemas, com o objetivo de reduzir a complexidade dos sistemas. Logo, a análise e a avaliação de desempenho de sistemas complexos pode ser impraticável devido ao problema de explosão de estados.

Este trabalho tem duplo propósito. Primeiro, nós propomos uma extensão para o modelo de redes de Petri para a caracterização de restrições temporais. Nele, os aspectos positivos das extensões baseadas nas abordagens determinísticas e estocásticas são combinados de uma forma complementar, i.e., a rede de Petri estendida proposta serve tanto para a modelagem de sistemas em tempo real como para a avaliação de desempenho de sistemas [30]. Esta extensão, chamada de *Redes de Petri com Temporização Nebulosa - Fuzzy Time Petri Net (FTPN)*, utiliza uma abordagem baseada na teoria dos conjuntos nebulosos introduzida por Zadeh [125]. Nesta abordagem nebulosa, cada ficha carrega

uma função nebulosa de tempo que caracteriza a possibilidade de sua existência em um determinado lugar em um dado instante de tempo. Intervalos nebulosos são ainda associados às transições capacitando a representação de restrições temporais. Estes intervalos de tempo associados às transições permitem a representação das três categorias de restrições temporais que são encontradas nos sistemas em tempo real [29]: restrições relacionadas com o tempo máximo, com o tempo mínimo e de duração. Esta extensão possibilita a avaliação de desempenho de sistemas em que, por exemplo, pode-se computar o tempo mínimo, máximo e mais provável de resposta necessário para atingir-se um determinado estado a partir de um estado inicial. Além disso, podem-se computar índices agregados de desempenho.

Segundo, nós integramos o modelo *FTPN* com uma ferramenta de estruturação chamada *G-Net*, com o objetivo de definir uma abordagem modular para efetuar a análise temporal de sistemas. Uma *G-Net* é definida como um ambiente para a especificação e prototipagem de sistemas complexos através do incorporamento de noção de módulo e estrutura de sistemas ao modelo de redes de Petri. Esta integração resulta nas *G-Nets com Temporização Nebulosa - Fuzzy Time G-Nets (FTG-Nets)* e permite dividir um sistema complexo em vários subsistemas que podem ser estudados separadamente e os resultados combinados para determinar a solução global do sistema.

Como aplicação, nós usamos a análise modular provida pelas *FTG-Nets* para introduzir propriedades de tolerância a falhas dependentes do tempo nos sistemas distribuídos em tempo real complexos. Como conseqüência da crescente popularidade do uso de sistemas de computação em aplicações críticas, aspectos relacionados com a tolerância a falhas de sistemas ganharam um grande impulso nos últimos anos. Um sistema é tolerante a falhas se ele mantém capacidade funcional e desempenho completo mesmo na presença delas. Logo, considerando um sistema distribuído em tempo real (SDTR), um alto grau de confiabilidade, disponibilidade e segurança são requeridos. O sistema deve garantir desempenho nos domínios do tempo e de valores como especificado [68, 90]. Nos sistemas em tempo real, a maioria das falhas estão relacionadas com erros de sincronização e de desempenho que se manifestam como falhas transitórias nos sistemas. Portanto, o projetista de um SDTR deve adotar uma abordagem que suporte a introdução de propriedades de tolerância a falhas dependentes e não dependentes do tempo em um componente do sistema. A integração entre *FTPN* e *G-Nets* permite a introdução de propriedades de tolerância a falhas dependentes do tempo em sistemas distribuídos em tempo real, bem como a representação de esquemas tolerantes a falhas dependentes do tempo.

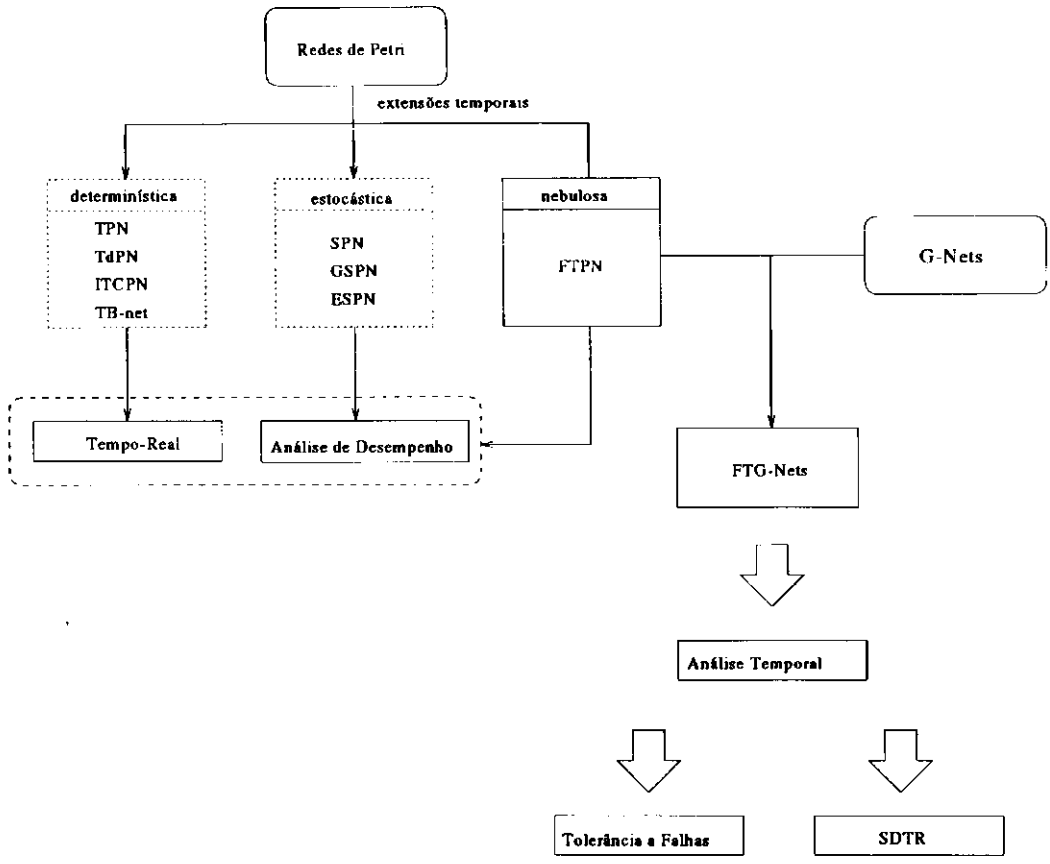

Figura 1.1: Sumário da Tese

A Figura 1.1 sumariza o objetivo deste trabalho. Na Figura 1.1, observa-se que as extensões temporais propostas para o modelo de redes de Petri, representadas pelos blocos pontilhados, são apropriados para áreas específicas. A abordagem determinística é boa para modelar sistemas em tempo real, enquanto que a abordagem estocástica serve para a avaliação de desempenho de sistemas. A extensão temporal proposta, *FTPN*, tem o objetivo de servir como base tanto para modelar sistemas em tempo real como para fazer avaliação de desempenho de sistemas. Além do mais, as *FTPNs* são combinadas com as *G-Nets* para analisar sistemas complexos. A ferramenta resultante da integração *FTG-Nets* é aplicada para introduzir propriedades de tolerância a falhas no projeto de sistemas distribuídos em tempo real. Nós também definimos uma metodologia para considerar falhas, antecipadamente, no projeto de sistemas distribuídos em tempo real.

1.2 Conceitos Básicos e Trabalhos Relacionados

A modelagem e análise de um sistema tem como objetivo criar e avaliar um projeto de um novo sistema, determinar projetos alternativos, garantir corretude e investigar possíveis melhoramentos em um sistema real. Diversas abordagens foram desenvolvidas para modelar e analisar sistemas. Cada abordagem apresenta aspectos positivos e limitações. Por exemplo, algumas abordagens são apropriadas para fazer análise qualitativa de sistemas mas são limitadas para a análise quantitativa. De uma forma geral, nós podemos distinguir os dois principais caminhos que dizem respeito à análise e modelagem de sistemas: técnicas informais e técnicas formais.

No caso das técnicas informais, nós podemos citar como exemplo algumas técnicas de diagramação que utilizam uma linguagem gráfica para descrever o fluxo de dados, controle de dados, etc [44, 124]. Estas técnicas resultam em uma descrição informal que não permite a análise quantitativa dos sistemas. Uma outra técnica informal é a simulação [3, 45, 53] que é uma técnica muito importante para analisar sistemas complexos. A principal característica da simulação é que ela não requer conhecimento matemático. Portanto, os resultados obtidos através da simulação podem ser facilmente entendidos por pessoas sem conhecimento técnico [3]. Entretanto, o uso de simulação não é suficiente para provar corretude de sistemas.

As técnicas formais são baseadas em uma sólida fundamentação matemática. Como exemplo de métodos formais para analisar e modelar sistemas, nós temos: modelos de fila, linguagens de especificação orientada ao modelo, álgebras de processo, lógica temporal e redes de Petri. Os modelos de fila são muito populares na área de análise de desempenho e são definidos como modelos analíticos de sistemas de computação usados para analisar contenção para recursos limitados ou para previsão de desempenho de sistemas [66]. As linguagens de especificação orientadas ao modelo, como por exemplo VDM [63] e Z [108], têm sido usadas na especificação de grandes sistemas comerciais mas falham por não tratarem com concorrência e tempo real [122]. As álgebras de processo, tais como CSP [57] e CCS [80], são boas para a modelagem de paralelismo e concorrência. Hansson [56] apresenta uma extensão para CCS com tempos discretos e probabilidade. Lógica temporal [94, 104] é um tipo especial de lógica modal que tem a capacidade de expressar propriedades de programas concorrentes. Lógica temporal também foi estendida no sentido de quantificar tempo [88].

Rede de Petri [85, 93, 99] é uma poderosa ferramenta gráfica e matemática que foi inicialmente desenvolvida por C. A. Petri no começo dos anos 60. A maior força das redes de Petri é a maneira como os aspectos básicos dos sistemas distribuídos são iden-

tificados tanto conceitual como matematicamente. As redes de Petri são uma poderosa ferramenta formal para a especificação e análise de sistemas caracterizados por serem concorrentes e assíncronos. As redes de Petri foram usadas na modelagem de diversos tipos de sistemas e vários tipos de extensões e modificações foram propostos no sentido de capacitá-las a suportar determinados requisitos em áreas de aplicações específicas, por exemplo, comunicação de protocolos, avaliação de desempenho, modelagem e análise de sistemas em tempo real, etc.

Diversas extensões foram propostas ao modelo de redes de Petri com o objetivo de considerar alguns aspectos que são encontrados nos sistemas reais. Dois tipos de extensões são relevantes: as extensões que permitem a representação de requisitos de tempo e as extensões relacionadas com a capacidade de modelagem funcional. Além do mais, as redes de Petri foram estendidas através de uma abordagem nebulosa com o intuito de tratar incertezas presentes na maioria dos sistemas reais.

As extensões relacionadas com a capacidade de modelagem funcional são representadas pelas redes de Petri de alto nível. Como exemplo destas extensões, citam-se entre outras as *Redes de Petri Coloridas* [61], as *Redes de Predicado/Transição* [48], *G-Nets* [36, 37] e *E-R Nets* [49].

Na seqüência, nós discutimos as extensões temporais e as redes de Petri nebulosas.

1.2.1 Tempo em Redes de Petri

Diferentes tipos de extensões foram propostos no sentido de capacitar as redes de Petri na caracterização de requisitos de tempo. Estas extensões utilizam basicamente dois tipos de abordagem: determinística [14, 49, 79, 96, 107, 123, 129] e estocástica [4, 55, 81, 87].

Na abordagem determinística, os requisitos de tempo são definidos como constantes e a análise, nestes casos, pode ser efetuada através de um método algébrico. Em geral, nesse tipo de abordagem, a caracterização das restrições de tempo modifica o comportamento qualitativo do modelo em relação ao correspondente modelo sem as restrições de tempo. Entretanto, a abordagem determinística é adequada para a modelagem de sistemas em tempo real em que a caracterização de limites de tempo é essencial. Esta abordagem é limitada para a avaliação de desempenho de sistemas em que, basicamente, determina-se o tempo de ciclo mínimo, i.e., o tempo necessário para atingir um determinado estado a partir de um estado inicial. As extensões determinísticas foram usadas em diversas áreas de aplicação como por exemplo, para modelar protocolos de comunicação [79, 78] e em sistemas flexíveis de manufatura [110, 111], etc. Diversas

metodologias de análise baseadas em grafos de alcançabilidade modificados foram propostas [13, 75, 95, 96, 113]. As extensões temporais determinísticas também foram aplicadas à classe de redes de Petri de alto nível para reduzir a complexidade de modelagem de sistemas complexos [50, 122]. As redes de alto nível temporais são adequadas para modelagem de sistemas complexos mas ainda são limitadas para a avaliação de desempenho de sistemas.

Na abordagem estocástica, os atrasos no disparo das transições são definidos através de variáveis aleatórias e o sistema pode ser analisado através de métodos probabilísticos. Na abordagem estocástica, o comportamento qualitativo do modelo estendido com a caracterização das restrições de tempo não difere do comportamento do correspondente modelo sem as considerações de tempo. A abordagem estocástica é utilizada, principalmente, na avaliação de desempenho de sistemas uma vez que a árvore de alcançabilidade de um sistema modelado por uma rede de Petri estocástica é isomórfica aos processos homogêneos de Markov [82]. Então, aplicando-se as propriedades Markovianas, índices agregados de desempenho podem ser facilmente determinados. Uma vez que no modelo original das redes de Petri estocásticas, os tempos de disparo asscociados às transições são restritos à uma função de densidade de probabilidade exponencial, diferentes variantes foram propostas ao modelo estocástico [3, 4, 42, 55]. Muitos outros trabalhos também propuseram a aplicação da abordagem estocástica para a análise de sistemas complexos [28, 76, 126].

1.2.2 Incerteza e Redes de Petri

Existem diferentes formas de representar incerteza sobre os conhecimentos que são comumente encontrados no mundo real. Uma das formas mais aceitas para representá-las é através do uso da teoria dos conjuntos nebulosos [125] que é baseada na idéia de generalização da teoria dos conjuntos ordinários e foi proposta por Zadeh em 1965.

Esta teoria dos conjuntos nebulosos foi extensivamente utilizada em diversas áreas e pode ser utilizada, por exemplo, para a representação e modelagem de imprecisão e incerteza do conhecimento temporal [26, 27, 41]. Existem dois tipos de incerteza temporal: imprecisão nas datas de eventos que são representados por números nebulosos e descrição nebulosa de tempo que geralmente é expressa em termos de predicados lingüísticos ou por quantidades nebulosas temporais. O conceito de tempos nebulosos foi usado para fazer estimação de tempo [86] e também foi aplicado ao problem PERT de achar os tempos de eventos e duração mínima de projetos [22, 74].

A integração de uma abordagem nebulosa e redes de Petri também foi alvo de muitos

trabalhos [19, 20, 21, 25, 73, 109, 121]. Estes tipos de redes de Petri nebulosas são adequados para aplicações específicas. Por exemplo, a rede de Petri nebulosa proposta por Looney [73] é uma rede de Petri modificada que é aplicada para o raciocínio baseado em regras nebulosas usando lógica proposicional. Chen et ali [25] propuseram um modelo de rede de Petri nebulosa para a representação de regras de produção nebulosas em um sistema baseado em regras.

As extensões temporais propostas ao modelo de redes de Petri não levam em consideração a incerteza e imprecisão dos aspectos temporais dos eventos. A primeira tentativa de considerar imprecisão foi proposta por Merlin [79] que representou os tempos de disparos como intervalos associados às transições. Valette e Cardoso [21, 121] introduziram o conceito de imprecisão e incerteza em redes de Petri para monitorar e controlar sistemas flexíveis de manufatura. Neste trabalho, aspectos sobre incerteza temporal são abordados mas não existe nenhuma consideração sobre a análise temporal.

1.3 Escopo da Tese

O principal objetivo deste trabalho é propor uma nova extensão temporal para o modelo de redes de Petri chamada de *Rede de Petri com Temporização Nebulosa(FTPN)*. Neste trabalho, nós definimos formalmente as *FTPN*s cuja principal característica é a capacidade em modelar sistemas em tempo real bem como de efetuar a análise de desempenho de sistemas. As *FTPN*s são apropriadas para modelar sistemas em tempo real porque os intervalos nebulosos de tempo permitem a representação dos diferentes tipos de restrições temporais que encontramos nos sistemas em tempo real. Na verdade, as *FTPN*s são mais gerais do que as extensões temporais determinísticas que foram propostas ao modelo de redes de Petri uma vez que é possível representá-las através de uma *FTPN* impondo algumas restrições aos intervalos de sensibilização e disparo e ajustando a caracterização nebulosa das fichas. Neste trabalho nós apresentamos a unificação das extensões temporais determinísticas através do modelo *FTPN*.

No mais, a função nebulosa de tempo associada às fichas possibilita a computação de alguns parâmetros de desempenho. Introduzimos uma metodologia para executar a análise temporal de sistemas baseado em um grafo de alcançabilidade modificado (grafo de alcançabilidade nebuloso) bem como um algoritmo para computar as funções nebulosas de tempo associadas às fichas. A função nebulosa de tempo é a base para a determinação de índices de desempenho e análise de alcançabilidade.

Com o objetivo de avaliar o desempenho de sistemas complexos, integramos as *FTPN*s com uma ferramenta de estruturação chamada *G-Net*. A integração provê meios

de dividir um sistema grande em subsistemas, analisar cada subsistema isoladamente e depois combinar os resultados para a obtenção dos resultados globais. Esta integração é realizada considerando-se uma abordagem de decomposição para as *G-Nets*. A análise temporal modular é baseada no grafo de alcançabilidade nebuloso.

As *FTG-Nets* são adequadas para modelar e analisar sistemas distribuídos em tempo real pois as *FTPNs* são propícias para modelar sistemas em tempo real enquanto que *G-Net* é uma poderosa ferramenta para modelar e especificar sistemas distribuídos. A ferramenta integrada juntamente com uma metodologia de análise temporal modular são aplicadas para introduzir propriedades de tolerância a falhas dependentes no tempo no projeto de sistemas distribuídos em tempo real. Nós ainda aplicamos as *FTG-Nets* para caracterizar esquemas de tolerância a falhas e para considerar a antecipação de falhas em sistemas distribuídos em tempo real. Por fim, apresentamos uma representação temporal gráfica, *TGIG*, para a interação entre *G-Nets*. Esta representação gráfica temporal indica os tempos de execução de cada *G-Net*, considerando as interações entre *G-Nets* em um sistema de *G-Nets*. Alguns índices de desempenho também podem ser obtidos através deste gráfico.

Apresentamos ainda alguns resultados experimentais obtidos a partir de implementação do algoritmo de análise temporal baseado na integração entre *FTPN* e *G-Nets*.

1.4 Estrutura da Tese

O restante da Tese é organizado da seguinte forma: No Capítulo 2, nós apresentamos alguns trabalhos relacionados e conceitos básicos. No Capítulo 3, nós introduzimos as *FTPNs* e definimos as regras de computação da função nebulosa associada às fichas. No Capítulo 4, nós detalhamos a integração entre *FTPN* e *G-Nets*. O Capítulo seguinte está relacionado com a análise temporal das *FTPN*s. Nós propomos um grafo de alcançabilidade modificado juntamente com um algoritmo para computar as funções nebulosas de tempo. Nós definimos ainda, uma representação temporal gráfica para a interação e aspectos temporais entre *G-Nets*. Utilizamos um exemplo baseado no problema do produtor/consumidor para ilustrar a análise temporal. No Capítulo 6, aplicamos as *FTG-Nets* para a introdução de propriedades de tolerância a falhas dependentes no tempo no projeto de sistemas distribuídos em tempo real. Apresentamos ainda um exemplo baseado no problema de controle de veículos. Por fim, apresentamos as conclusões no capítulo 7.

Capítulo 2

CONCEITOS BÁSICOS E TRABALHOS RELACIONADOS

2.1 Introdução

Antes de introduzirmos o conceito de *FTPN* no Capítulo 3, vamos apresentar nesse capítulo alguns conceitos básicos e trabalhos relacionados. Esse capítulo é dividido em duas seções distintas. A primeira seção está relacionada com os conjuntos nebulosos, enquanto que a segunda seção está relacionada com as redes de Petri.

Além da introdução dos seus conceitos básicos e operações, na seção de conjuntos nebulosos discutimos o conceito de números e quantidades nebulosas bem como a representação de incerteza temporal através de conjuntos nebulosos.

Na seção de redes de Petri, apresentamos uma visão geral sobre as redes de Petri para servir de base para aqueles leitores que não estão familiarizados com este formalismo. Para maiores detalhes sobre os conceitos e propriedades das redes de Petri, o leitor pode se referir a [85, 93, 99]. O principal objeto dessa seção é a extensão temporal do modelo de redes de Petri. Discutiremos ainda os trabalhos que utilizaram uma combinação de redes de Petri com a teoria dos conjuntos nebulosos.

Uma vez que o Capítulo 6 está relacionado com a aplicação das *FTG-Nets* nas áreas de tolerância a falhas e sistemas distribuídos em tempo real, alguns conceitos e trabalhos relacionados pertinentes a essas aplicações serão apresentados separadamente no próprio Capítulo 6.

2.2 Teoria dos Conjuntos Nebulosos

A teoria dos conjuntos nebulosos é baseada na idéia de generalização dos conjuntos ordinários desenvolvida em 1965 por Lotfi Zadeh na Universidade da Califórnia, Berkeley. Desde então, milhares de artigos foram publicados sobre a teoria dos conjuntos nebulosos e suas aplicações. Através desta teoria, Zadeh tentou representar classes de objetos encontrados no mundo real que não possuem um critério preciso para definir o seu pertencimento a um determinado conjunto. Além do mais, é muito comum para

os humanos usar o conhecimento impreciso ou vago para executar ações complexas. Na seqüência dessa seção, apresentamos os principais conceitos da teoria dos conjuntos nebulosos.

2.2.1 Conceitos Básicos

Na teoria clássica dos conjuntos, um conjunto é definido como uma coleção de elementos ou objetos $x \in X$ que podem ser finitos, contáveis ou infinitos. Cada elemento pertence ou não a um determinado conjunto A, $A \in X$. Neste caso, é possível definir-se um elemento de um conjunto através de uma função característica μ, em que $\mu = 1$ indica que o determinado elemento pertence ao conjunto e $\mu = 0$ indica que o elemento não pertence ao conjunto [127].

Na teoria dos conjuntos nebulosos, os seus elementos também são representados por uma função característica μ. A diferença reside no fato de que a função característica μ permite vários graus de pertencimento para os elementos de um determinado conjunto, i.e., na teoria dos conjuntos nebulosos, o valor da função μ é estendido do par $\{0, 1\}$ para o intervalo $[0, 1]$.

Definição 2.1 *Se X é uma coleção de objetos denotados genericamente por x, então um conjunto nebuloso $\mathcal{A}$ em X é o conjunto de pares ordenados:*

$$\mathcal{A} = \{(x, \mu_{\mathcal{A}}(x)) | x \in X\} \tag{2.1}$$

$\mu_{\mathcal{A}}(x)$ é dita função de pertencimento ou grau de pertencimento de x em $\mathcal{A}$. A faixa de valores da função de pertencimento dos elementos de um conjunto nebuloso é um subconjunto dos números reais não-negativos cujo supremo é infinito.

Exemplo 1: Dado o conjunto de números reais $\mathbb{R}$, pode-se definir o conjunto nebuloso dos números reais, $\mathcal{A}$, que são consideravelmente maiores do que 10 por,

$$\mathcal{A} = \{(x, \mu_{\mathcal{A}}(x)) | x \in \mathbb{R}\}$$

em que,

$$\mu_{\mathcal{A}}(x) = \begin{cases} 0, & x \leq 10 \\ (1 + (x - 10)^{-2})^{-1}, & x > 10 \end{cases}$$

2.2.2 Operações Sobre os Conjuntos Nebulosos

Da mesma forma que na teoria clássica dos conjuntos, diversas operações podem ser definidas sobre os conjuntos nebulosos. Uma vez que os conjuntos nebulososo são prin-

cipalmente caracterizados pela função de pertencimento que é associada aos seus elementos, as operações sobre os conjuntos nebulosos são definidas através de suas funções de pertencimento. Na seqüência, nós apresentamos algumas das operações básicas sobre os conjuntos nebulosos. Para tanto, vamos considerar dois conjuntos nebulosos $\mathcal{A}$ e $\mathcal{B}$ em X, em que $\mu_{\mathcal{A}}(x)$ e $\mu_{\mathcal{B}}(x)$ representam suas respectivas funções de pertencimento. Então:

Intersecção: A função de pertencimento da intersecção de dois conjuntos nebulosos $\mathcal{A}$ e $\mathcal{B}$, $\mathcal{A} \cap \mathcal{B}$, é definida por

$$\mu_{\mathcal{A}\cap\mathcal{B}}(x) = min\{\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(x)\}, x \in X. \tag{2.2}$$

União: A função de pertencimento da união de dois conjuntos nebulosos $\mathcal{A}$ e $\mathcal{B}$, $\mathcal{A} \cup \mathcal{B}$, é definida por

$$\mu_{\mathcal{A}\cup\mathcal{B}}(x) = max\{\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(x)\}, x \in X. \tag{2.3}$$

Complemento: A função de pertencimento do complemento de um conjunto nebuloso $\mathcal{A}$, $\mu_{\overline{\mathcal{A}}}(x)$, é definida por

$$\mu_{\overline{\mathcal{A}}}(x) = 1 - \mu_{\mathcal{A}}(x). \tag{2.4}$$

Soma Algébrica: A função de pertencimento da soma algébrica de dois conjuntos nebulosos $\mathcal{A}$ e $\mathcal{B}$ é definida por

$$\mu_{\mathcal{A}+\mathcal{B}}(x) = \mu_{\mathcal{A}}(x) + \mu_{\mathcal{B}}(x) - \mu_{\mathcal{A}}(x).\mu_{\mathcal{B}}(x). \tag{2.5}$$

Soma Limitada: A função de pertencimento da soma limitada de dois conjuntos nebulosos $\mathcal{A}$ e $\mathcal{B}$ é definida por

$$\mu_{\mathcal{A}\oplus\mathcal{B}}(x) = min\{1, \mu_{\mathcal{A}}(x) + \mu_{\mathcal{B}}(x)\}. \tag{2.6}$$

Diferença Limitada: A função de pertencimento da diferença limitada de dois conjuntos nebulosos $\mathcal{A}$ e $\mathcal{B}$ é definida por

$$\mu_{\mathcal{A}\ominus\mathcal{B}}(x) = max\{0, \mu_{\mathcal{A}}(x) + \mu_{\mathcal{B}}(x) - 1\}. \tag{2.7}$$

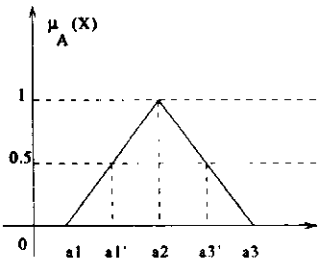

Figura 2.1: Um número nebuloso A

2.2.3 Números Nebulosos e Quantidades Nebulosas

De uma maneira informal, o conceito de um número nebuloso pode ser considerado uma extensão do conceito de intervalos de confidência [64]. Intervalo de confidência é uma forma de reduzir incerteza, utilizando limites inferior e superior, i.e., um intervalo de confidência é um intervalo fechado em $\mathbb{R}$ usado para representar um valor incerto. Dado um intervalo de confidência $[a_1, a_2]$, nós estamos certos de que o valor é maior ou igual a a_1 e menor ou igual a a_2. Um número nebuloso é definido a partir do conceito de intervalo de confidência associado a um outro conceito chamado de *nível de presunção*, i.e., diferentes intervalos de confidência podem ser determinados para representar um dado valor em diferentes níveis de presunção.

De uma maneira mais formal, um número nebuloso é um subconjunto nebuloso em $\mathbb{R}$ que é *normal* e *convexo*. Por *normal*, nós entendemos que o valor máximo do conjunto nebuloso em $\mathbb{R}$ é 1, i.e.,

$$\exists x \in \mathbb{R} : \forall_x \mu_A(x) = 1. \tag{2.8}$$

Convexo significa que um corte α (nível de presunção) que é paralelo ao eixo horizontal produz a propriedade de aninhamento (*nesting*), i.e.,

$$(\alpha\prime < \alpha \Longrightarrow (a_1^{(\alpha\prime)} \leq a_1^{(\alpha)}, a_2^{(\alpha\prime)} \geq a_2^{(\alpha)}). \tag{2.9}$$

em que a_1 e a_2 são os limites inferior e superior, respectivamente.

A Figura 2.1 mostra um número nebuloso A. É fácil ver que $a1 < a1^{'} < a2 < a3^{'} < a3$. As propriedades de normalidade e convexidade para um número nebuloso definidas, respectivamente, nas Equações 2.8 e 2.9 são verificadas. O número nebuloso A é normal pois seu valor máximo é 1, satisfazendo a Equação 2.8. Vamos considerar três diferentes cortes α nos níveis 0, 0.5 e 1. Para cada corte α, nós temos os seguintes intervalos de confidência: $A_{\alpha=0} = [a1, a3]$, $A_{\alpha=0.5} = [a1^{'}, a3^{'}]$ e $A_{\alpha=1} = [a2, a2]$. Para verificar a

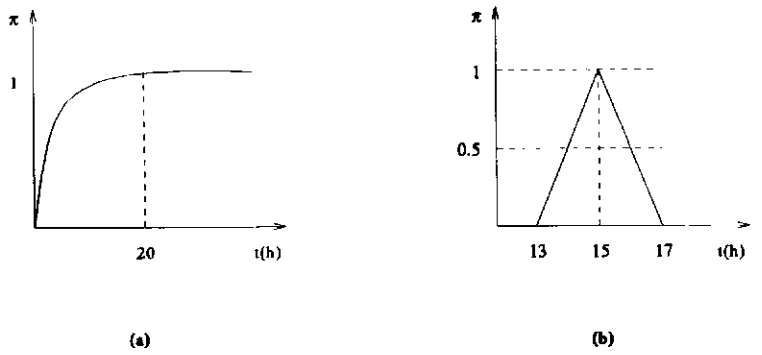

Figura 2.5: (a) Longo tempo, (b) Em torno de 15h

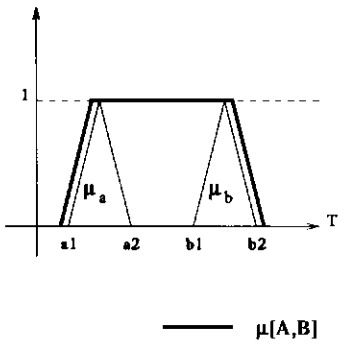

Figura 2.6: Intervalos nebulosos de tempo

duas datas a e b caracterizadas pelas funções de pertencimento μ_a e μ_b, respectivamente, como mostrado na Figura 2.6. A função de pertencimento que caracteriza o conjunto nebuloso $[A, B]$ é determinada através de Equação (2.12) e é representada pela linha mais cheia (ver Figura 2.6).

Alguns pesquisadores desenvolveram diferentes abordagens para tratar com incerteza temporal baseada nos conjuntos nebulosos. Muitos trabalhos apresentados combinam incerteza temporal nebulosa e redes de Petri [19, 20, 21], os quais serão discutidos posteriormente. Em [27], uma abordagem baseada em lógica nebulosa para tratar com incerteza temporal foi apresentada. A incerteza temporal é representada em termos de distribuição de possibilidades e as relações temporais entre dois elementos temporais nebulosos são propostas. Em um artigo mais recente, [26], Chen define um grafo temporal acíclico dirigido dito *Fuzzy Time Causal Like Network (FTCLN)* para representar o conhecimento sobre um processo ou um sistema. O raciocínio temporal sobre incertezas é efetuado pela evolução da rede.

2.3 Redes de Petri

Reisig [100] define redes de Petri da seguinte forma:

> Petri net is a collective term that, in the course of time, has come to designate a large number of system models, procedures, descriptive patterns, and techniques related to one another in the sense that they are all based on the same specific principle. In addition, there are systematic transitions between them which, themselves, make up part of Petri net theory.

Rede de Petri é uma ferramenta matemática e gráfica que foi primeiramente desenvolvida por C.A. Petri no início dos anos 60. Desde então, muitos pesquisadores melhoraram suas idéias e o estudo de redes de Petri[1] cresceu consideravelmente. A principal característica das redes de Petri é a maneira pela qual os aspectos de sistemas distribuídos são identificados tanto conceitual como matematicamente.

Na seqüência, nós apresentamos os conceitos básicos das redes de Petri bem como algumas extensões relevantes que foram propostas para representar alguns aspectos particulares que não podiam ser representados usando a definição original.

2.3.1 Conceitos Básicos

Uma rede de Petri pode ser informalmente definida como uma espécie de grafo dirigido ao qual associa-se uma marcação inicial [85]. O grafo de uma rede de Petri consiste em dois tipos de nós chamados *lugares* e *transições*, em que os arcos partem ou de um lugar para uma transição ou de uma transição para um lugar. Os lugares representam condições e geralmente são caracterizados graficamente por um círculo, enquanto que as transições representam eventos e são caracterizados no grafo por barras. Os lugares podem conter zero ou mais fichas (representados por pequenos círculos pretos). As fichas em redes de Petri modelam o comportamento dinâmico do sistema e sua distribuição, em um determinado momento, caracteriza uma marcação ou estado de uma rede de Petri.

Formalmente, as redes de Petri são definidas por uma 5-tupla *(P, T, I, O, M0)*, em que:

P é um conjunto finito de lugares.

[1] O estudo de redes de Petri desenvolveu-se em duas direções: teoria pura de redes de Petri e aplicação da teoria de redes de Petri.

T é um conjunto finito de transições.

$I : T \rightarrow P^{\infty}$ é a função que caracteriza os lugares de entrada das transições.

$O : T \rightarrow P^{\infty}$ é a função que caracteriza os lugares de saída das transições.

$M0 : P \rightarrow \mathbb{N}$ é uma função a partir do conjunto de lugares P para os inteiros não-negativos $\mathbb{N}$, representando a marcação inicial.

O comportamento dinâmico de um sistema modelado por uma rede de Petri é descrito em termos dos estados do sistema (marcações) e suas mudanças.

Definição 2.2 *Uma marcação M de uma rede de Petri é uma função definida a partir do conjunto de lugares P para os inteiros não negativos $\mathbb{N}$, $M : P \rightarrow \mathbb{N}$.*

A execução de uma rede de Petri é controlada pelo número e distribuição das fichas, ou seja, o comportamento dinâmico de um sistema modelado por uma rede de Petri é descrito em termos dos estados do sistema e suas mudanças, as quais são determinadas pelo fluxo das fichas na rede. Uma transição é dita sensibilizada se cada lugar de entrada contém tantas fichas quantos forem os arcos que os conectam às transições. Quando uma transição é sensibilizada ela pode disparar e, após o disparo, as fichas são retiradas dos lugares de entrada e são depositadas nos lugares de saída de acordo com o número de arcos que partem da transição para os lugares de saída.

As redes de Petri têm sido extensivamente usadas em diversas área da computação: avaliação de desempenho, comunicação de protocolos, modelagem e análise de sistemas distribuídos, etc. As redes de Petri são um bom formalismo para modelar e analisar muitos sistemas, especialmente aqueles que exibem uma das seguintes características: conflito, concorrência, junção, divisão, seqüência e sincronização. Esses aspectos podem ser facilmente representados através das redes de Petri como mostrado na Figura 2.7.

Na Figura 2.7(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 2.7(b) modela a concorrência, i.e., as transições *t1* e *t2* são ambas sensibilizadas e representam atividades que podem ser executadas concorrentemente. As Figuras 2.7(c) e 2.7(d) representam, respectivamente, as situações de junção e divisão. Na Figura 2.7(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 2.7(f).

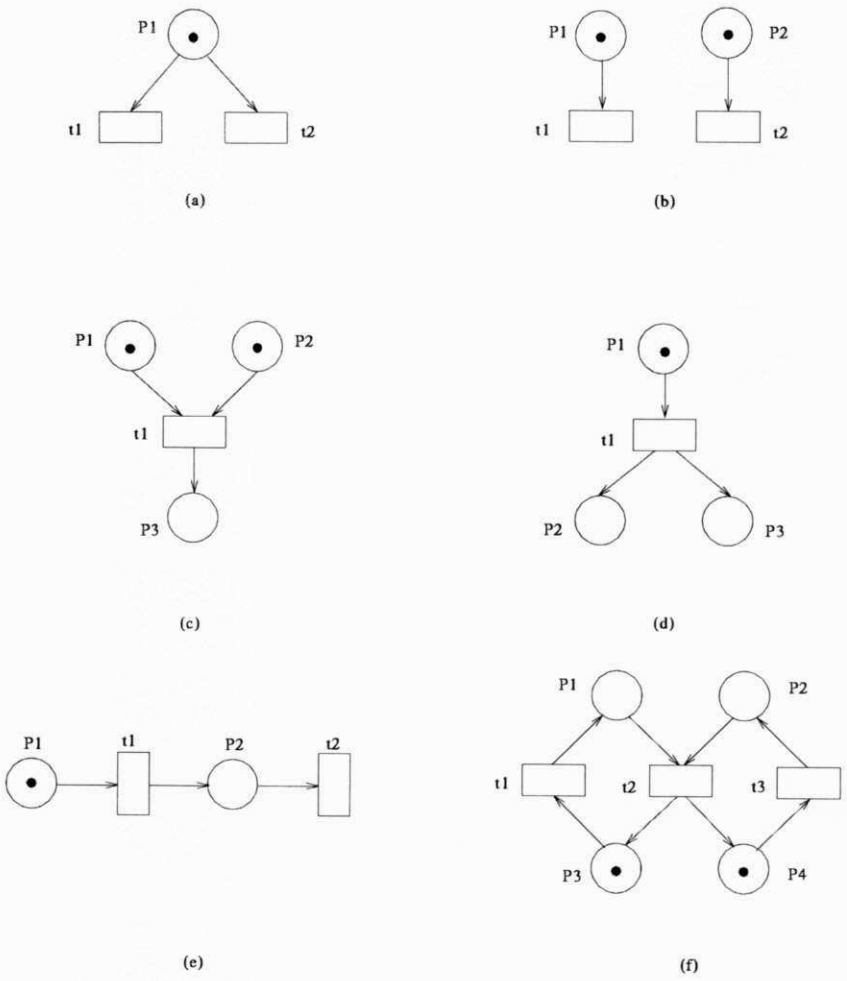

Figura 2.7: (a) Conflito, (b) Concorrência, (c) Junção, (d) Divisão, (e) Seqüência, (f) Sincronização

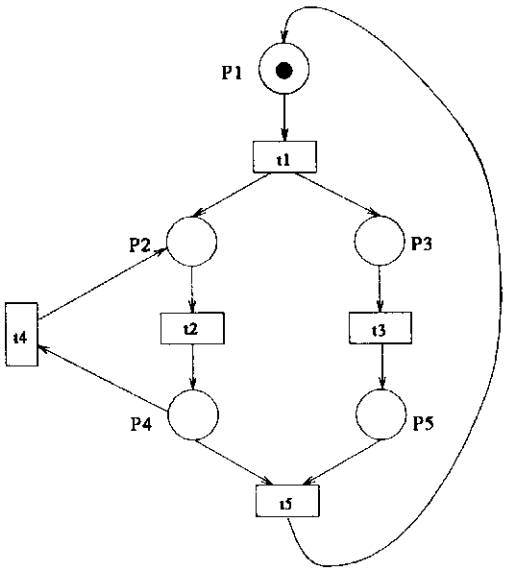

Figura 2.8: Uma rede de Petri

Mesmo considerando que as transiçãoes *t1* e *t3* possam ser executadas concorrentemente, o disparo da transição *t2* depende das fichas nos lugares *P1* e *P2*.

Para as redes de Petri, diferentes tipos de propriedades comportamentais como *liveness, safeness e boundedness* são definidas. Para detalhes sobre as propriedades das redes de Petri ver [85]. Os três métodos principais para analisar as redes de Petri são: técnicas de redução e decomposição, método da análise de invariantes e método da análise de alcançabilidade. O principal objetivo das técnicas de redução e decomposição é reduzir a complexidade da rede preservando suas propriedades. Murata [85] e Berthelot [12] discutem a decomposição de redes e os princípios de transformação. A análise de invariantes é baseada na análise estrutural da rede. Existem dois tipos de invariantes, os invariantes de transição (T-invariants) e os invariantes de lugar (S-invariants), os quais são definidos a partir da matriz de incidência da rede. A análise de alcançabilidade é baseada na enumeração de todas as marcações alcançáveis a partir da marcação inicial. Logo, a partir da marcação inicial novas marcações são geradas dependendo do disparo das transições. Esse processo resulta na chamada árvore de alcançabilidade. Entretanto, esta pode ser infinita mesmo para redes pequenas. Um grafo de alcançabilidade é uma árvore de alcançabilidade modificada cujos nodos representam estados alcançáveis para a rede e cujos arcos conectam nodos representando estados que podem ser diretamente alcançados. Na Figura 2.8 uma rede de Petri simples é apresentada. O seu correspondente grafo de alcançabilidade é mostrado na Figura 2.9. *M0 = (1,0,0,0,0)* é a marcação inicial que representa o estado inicial. Nesse caso, somente a

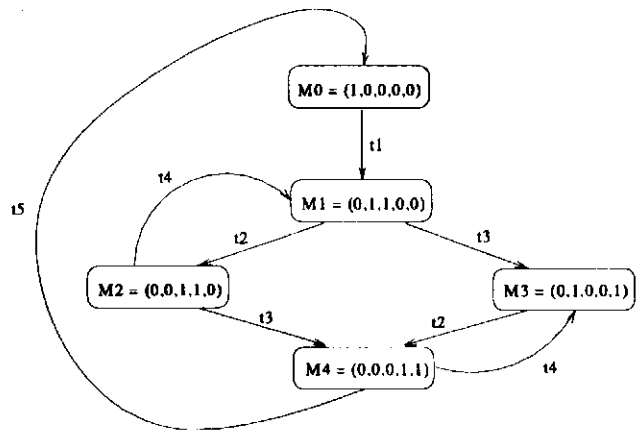

Figura 2.9: Grafo de alcançabilidade para a rede de Petri da Figura 2.8

transição *t1* está sensibilizada e o estado imediatamente alcançável é *M1 = (0,1,1,0,0)*. A análise de alcançabilidade pode também ser usada para determinar algumas propriedades temporais nos modelos temporais de redes de Petri como será apresentado posteriormente neste capítulo. No Capítulo 5, nós usamos um grafo de alcançabilidade diferente para executar a análise temporal de sistemas.

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 [85]. Os dois principais tipos de extensões de redes de Petri são discutidos na seqüência.

2.4 Redes de Petri de Alto Nível

Apesar de suas características, o modelo clássico das redes de Petri não é adequado para modelar muitos sistemas que são encontrados no mundo real devido ao problema da complexidade, i.e., as redes de Petri que descrevem sistemas reais tendem a ser complexas e extremamente grandes. No mais, o modelo clássico de redes de Petri não é completo o bastante para estudar o desempenho dos sistemas uma vez que nenhuma suposição é feita sobre a duração das atividades do sistema, i.e., a definição original do modelo não leva em consideração os aspectos temporais. Para superar esse problema, muitas extensões foram propostas ao modelo original. Dois tipos de extensões são relevantes: as extensões relacionadas com a caracterização de restrições temporais as quais serão estudadas na próxima seção e as extensões relacionadas com a capacidade de modelagem funcional.

As extensões relacionadas com a capacidade de modelagem funcional conduzem a

uma classe de redes de Petri chamadas de *Redes de Petri de Alto Nível* tais como: *redes de Petri coloridas (Colored Petri nets)* [61], *redes de Predicado/Transição (Predicate/Transition nets)*[48], *E-R nets* [49] e *G-Nets* [36]. Essas extensões aumentam o poder de modelagem provendo uma maneira mais compreensiva e compacta de construir modelos de sistemas. A classe das redes de Petri de alto nível difere das redes de Petri clássicas porque esta última permite apenas um tipo de ficha e a rede é plana.

As *redes de Predicado/Transição* são uma extensão das redes de Petri originais nas quais as fichas não são mais anônimas e o disparo das transições depende de certas condições relacionadas aos valores carregados pelas fichas. As *redes de Predicado/Transição* permitem a representação de sistemas em um nível mais alto de abstração do que as redes de Petri clássicas [51]. As regras de disparo que determinam o comportamento dinâmico de uma *rede de Predicado/Transição* são diferentes das regras clássicas de disparo de transições. Para disparar uma transição, além do número de fichas nos lugares de entrada, os valores das fichas devem satisfazer a fórmula associada com a transição. No mais, a capacidade dos lugares de saída não pode ser excedida. Depois do disparo da transição, as fichas são depositadas nos lugares de saída carregando valores que são especificados pelas fórmulas associadas com as transições.

Usando as *redes de Petri coloridas (CP-nets)*, é possível representar tipos e manipulações de dados complexos. Nesse caso, para cada ficha é associado um valor de dado chamado de cor da ficha que pode representar arbitrários tipos de dados complexos como por exemplo: inteiros, reais, registros, etc. O disparo das transições pode causar modificações nas cores das fichas.

As *redes Ambiente-Relação (E-R nets)* são um tipo de redes de alto nível no qual as fichas são relações entre variáveis e valores, e predicados e ações são associados com as transições. As *E-R nets* são semelhantes às *redes de Predicado/Transição* e o disparo de uma transição é determinado pela avaliação do predicado associado. Os ambientes representados nas fichas são dinamicamente modificados pelas ações, cujas execuções são partes do disparo da transição.

G-Net é um modelo de especificação executável multi-nível baseado em redes de Petri [36, 39]. Além das inerentes características do modelo de redes de Petri como concorrência, assincronia e não-determinismo, as *G-Nets* provêem características tais como modularidade e modificabilidade através de um mecanismo de abstrações para construir de forma incremental especificações de sistemas multi-nível. Uma vez que *G-Net* será utilizada para definir uma abordagem de análise temporal modular, maiores detalhes serão apresentados no Capítulo 4.

As redes de Petri de alto nível são muito importantes quando da modelagem e análise de sistemas complexos com o objetivo de evitar ou tratar o problema de explosão de estados. Algumas das técnicas de análise, como por exemplo, grafos de alcançabilidade e invariantes, foram estendidas para analisar redes de Petri de alto nível [61, 122]. Conceitos temporais também foram incorporados às redes de Petri de alto nível [50, 76, 123, 126] para fazer análise temporal de sistemas complexos. Na seção seguinte nós discutimos duas redes de Petri temporais de alto nível. No Capítulo 4, nós propomos uma extensão temporal para as *G-Nets* para efetuar a análise temporal modular de sistemas.

2.5 Extensões Temporais de Redes de Petri

Nas redes de Petri clássicas, a caracterização das propriedades temporais de um sistema não é possível, i.e., 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, nós apresentamos algumas das extensões de redes de Petri para a caracterização de restrições de tempo. Estas extensões utilizam uma abordagem determinística [49, 79, 96, 107] ou estocástica [4, 55, 82]. Uma abordagem diferente foi proposta por Suzuki [115]. 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 relaçãoes 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 lingua-

gem para descrever as restrições temporais. A linguagem utiliza uma variante da lógica temporal proposicional de tempo linear [69].

2.5.1 Extensões Temporais Determinísticas

As *Redes de Petri Temporizadas (Timed Petri Net - TdPN)* [96] 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 2.3 *Uma Rede de Petri Temporizada é uma 6-tupla (P, T, I, O, τ, M0) em que P, T, I, O e M0 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 transição na rede nos números naturais.*

As *TdPNs* foram usadas para fazer análise de desempenho de sistemas. Ramchandani [96] estudou o comportamento de estado das *redes de Petri temporizadas* e desenvolveu métodos para calcular a taxa de *throughput* para certas classes dessa rede. Zuberek [128, 129] 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 [95] 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 [79] 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 2.4 *Uma rede de Petri temporal é uma 6-tupla (P, T, I, O, E, M0) em que P, T, I, O e M0 são definidos como nas redes de Petri temporizadas. E é um intervalo de tempo $E : T \rightarrow [tmin, tmax]$, em que tmin e tmax $\in \mathbb{N}$ e $tmax \geq tmin$.*

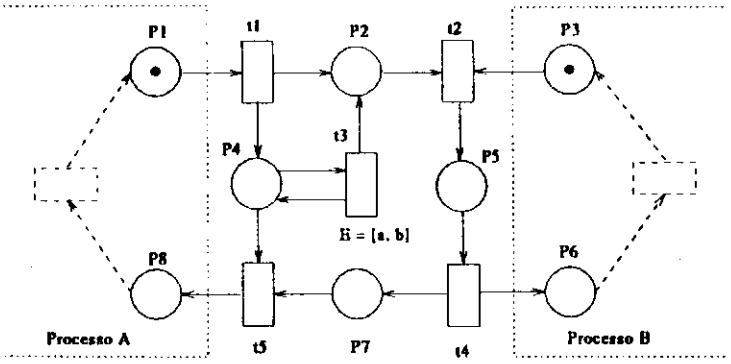

Figura 2.10: Rede de Petri temporal

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 [77, 78]. A Figura 2.10 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.

Berthomieu [13] 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 [111] mas em um contexto diferente. A análise funcional e de desempenho de uma *TPN* foi investigada por Majmudar [75].

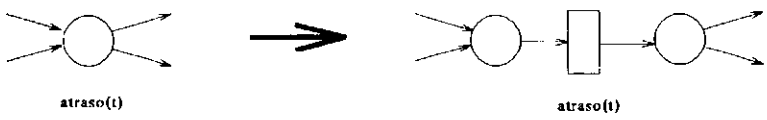

Figura 2.11: Transformação do atraso de lugar em atraso de transição

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* [107], 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 2.5 *Uma rede de lugar-transição temporizada é uma 6-tupla (P, T, I, O, τ, M0) em que P, T, I, O e M0 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.*

Em [107], 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. A Figura 2.11 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.

Em [114], *TdPTN* foi usada para representar computações concorrentes no modelo de um sistema de software. Em [113], um grafo de alcançabilidade modificado é definido para o modelo *TdPTN* para suportar uma variedade de estudos de desempenho de computações em tempo real.

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.

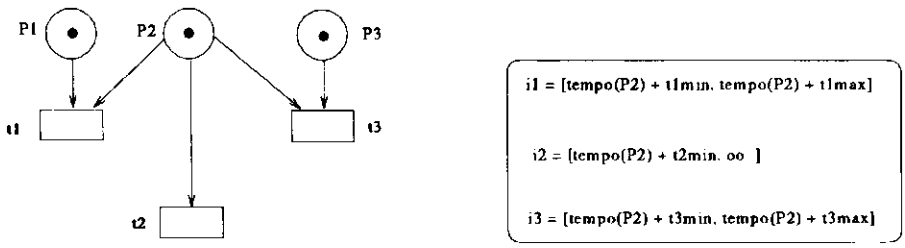

Figura 2.12: Exemplo de uma rede TB

Em [49], as *Redes Básicas de Tempo (Time Basic Nets – TB nets)* são propostas. As *Redes Básicas de Tempo* consistem, basicamente, na associação de um valor de tempo (*timestamp*) a cada ficha, representando o tempo de disparo da transição que a gerou. 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 2.12, 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 2.12, os intervalos *i1*, *i2* e *i3* estão relacionados com as 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 podre.

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. Ghezzi usa um tipo de rede de Petri de alto nível chamado de *E-R net* [50] e também mostra que as *Redes Básicas de Tempo* unificam e generalizam todos os métodos descritos anteriormente.

Uma *Rede de Petri Colorida Temporizada por Intervalos (Interval Timed Colored Petri Net (ITCPN))* [122, 123] é uma *Rede de Petri Colorida* estendida com tempo. O tempo está caracterizado nas fichas e um intervalo está associado com as transições

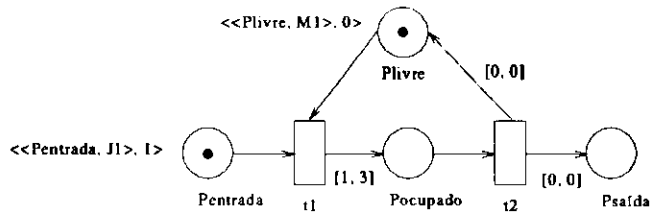

Figura 2.13: ITCPN

para determinar um atraso para cada ficha produzida. No modelo *ITCPN*, uma ficha tem quatro atributos: uma identidade, uma posição ou localização, um valor e um *timestamp*. Uma transição é dita sensibilizada se todos os lugares de entrada contêm pelo menos o número especificado de fichas. Uma transição sensibilizada pode disparar em tempo t se todas as fichas a serem consumidas têm um *timestamp* cujo valor é menor do que t. O tempo de sensibilização de uma transição é o maior *timestamp* das fichas a serem consumidas. O disparo de uma transição significa o consumo de fichas dos lugares de entrada e a produção de fichas nos lugares de saída. O *timestamp* das fichas produzidas é pelo menos igual ao tempo de disparo. A diferença entre o tempo de disparo e o *timestamp* das fichas produzidas é chamada de atraso de disparo. Esse atraso é especificado por um intervalo associado às transições. Na Figura 2.13, uma *ITCPN* é mostrada representando um chão de fábrica onde tarefas chegam através do lugar *Pentrada* e são executadas através do lugar *Psaída*. As máquinas que compreendem o chão de fábrica são representadas por fichas. Existem duas classes de cores: máquinas (por exemplo *M1*, na ficha localizada no lugar *Plivre*) e tipo de tarefa (por exemplo *J1*, na ficha localizada no lugar *Pentrada*). O tempo de sensibilização de uma transição é o maior *timestamp* das fichas a serem consumidas (no caso do disparo de *t1*, o tempo de sensibilização é 1). O atraso é um valor arbitrário entre 1 e 3. Em [122], alguns métodos de análise para o modelo *ITCPN* foram apresentados bem como algumas aplicações.

2.5.2 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 [3]. Entre a classe dos processos estocásticos, nós podemos citar os processos Markovianos [66]. 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

[81] e Natkin [87] 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 [82]. 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.

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*) [4], redes de Petri estocásticas estendidas (*Extended Stochastic Petri Nets - ESPN*) [42] e redes de Petri estocásticas regenerativas (*Regenerative Stochastic Petri Nets*) [55]. A extensão de *SPN* mais conhecida e usada é a *GSPN* [3, 4] 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 [76] e [126].

2.6 Redes de Petri Nebulosas

Para tratar ou modelar as incertezas e imprecisões que são encontradas em muitos sistemas do mundo real, a integração das redes de Petri com lógica nebulosa foi proposta. Diferentes tipos de redes de Petri nebulosas foram propostos para resolver problemas em diferentes tipos de aplicações. Na seqüência, nós resumimos as diferentes redes de Petri nebulosas propostas.

As *Redes de Petri Nebulosas (FPN)*, introduzidas por Looney [73], são uma rede de Petri modificada aplicada para o raciocínio nebuloso baseado em regras utilizando

lógica proposicional. As fichas representam condições e, para cada ficha, é assinalado um valor nebuloso denotando o grau de crença de veracidade da correspondente condição. Nesse modelo *FPN*, duas modificações são consideradas: não existem ciclos e, baseado em uma característica específica do raciocínio lógico, as fichas não são consumidas depois do disparo de uma transição. Um algoritmo para raciocinar sobre o vetor de estados-verdade nebuloso baseado nas operações interativas conjuntivas e disjuntivas é apresentado. Para efetuar o raciocínio nebuloso, as fichas verdadeiras e os componentes do estado transição são atualizados até que se alcance uma convergência, i.e., os valores-verdade não mudem mais.

Chen [25] propôs um modelo de rede de Petri nebuloso para representar regras de produções nebulosas de um sistema baseado em regras. Uma regra de produção nebulosa é uma regra que descreve a relação nebulosa entre duas proposições. Quatro tipos de composição de regras nebulosas são estudados e um algoritmo é apresentado para efetuar o raciocínio nebuloso, i.e., determinar se existe uma relação antecedente-conseqüente a partir de uma proposição d_s para uma proposição d_j. O algoritmo de raciocínio nebuloso é um algoritmo interativo que gera automaticamente todos os caminhos partindo de um lugar p_s (representando d_s) para um lugar alvo p_j (representando d_j) e, se o valor da ficha no lugar p_s é conhecido, então o valor da ficha no lugar alvo p_j pode ser avaliado.

Srinivasan [109] também definiu um modelo de rede de Petri nebuloso para ser usado no campo do raciocínio aproximado. A rede de Petri nebulosa, nesse caso, observa dois aspectos de *fuzificação* de um dado modelo: representação de informação, e controle da nebulosidade de uma informação. O controle de nebulosidade de uma informação é importante na modelagem da aquisição de informação para diminuir a imprecisão. Logo, todos os componentes padrões de redes de Petri são redefinidos para suportar nebulosidade. Cada lugar tem um predicado ou uma propriedade a ele associado. Uma função de pertencimento é associada às fichas, determinando o grau de pertencimento em um lugar particular, ou o valor-verdade daquela proposição. Uma transição nebulosa corresponde a um *if-then* das regras de produção nebulosas. Os arcos são também *fuzificados* para especificar o valor linguístico da correspondente ficha de entrada/saída.

Em Cao e Sanderson [20], uma rede de Petri nebulosa é definida utilizando três tipos de variáveis nebulosas: as variáveis nebulosas locais, as variáveis de marcação nebulosas e as variáveis nebulosas globais. Esses diferentes tipos de variáveis nebulosas são usados para modelar incerteza baseado em diferentes aspectos de uma informação nebulosa. As variáveis nebulosas locais modelam as informações que afetam localmente uma operação e são associadas aos lugares. As variáveis de marcação nebulosas são também

associadas aos lugares e representam o estado do sistema. As variáveis nebulosas globais são associadas às fichas e representam uma característica variável de uma tarefa global. Esses três tipos diferentes de variáveis são operados ou carregados ao longo da rede. As variáveis locais nebulosas são estudadas em detalhes e três casos diferentes são considerados: as variáveis nebulosas locais não são modificadas pelas transições, as variáveis nebulosas locais são modificadas pelas transições e o disparo da transição depende das variáveis nebulosas locais dos lugares de entrada. No primeiro caso, depois do disparo de uma transição, as variáveis nebulosas locais dos lugares de saída permanecem inalteradas. No segundo caso, depois do disparo da transição, novas variáveis nebulosas locais são associadas aos lugares de saída. No último caso, o disparo de uma transição não implica que todos os lugares de saída receberão uma ficha. Propriedades tais como alcançabilidade, vivacidade, *boundedness*, e *reversibility* são analisadas para os três casos.

Em um trabalho anterior, Cao e Sanderson [19] aplicaram as variáveis nebulosas globais em uma rede de Petri nebulosa para a representação e planejamento de seqüências de operações utilizando estratégias de raciocínio nebuloso.

Valette e Cardoso [21, 121] introduziram os conceitos de incerteza e imprecisão em redes de Petri para monitorar e controlar um sistema flexível de manufatura (FMS). Um diferente tipo de rede de Petri de alto nível chamado *Rede de Petri a Objetos* [106] é usado para descrever os mecanismos de coordenação do controle FMS. Incerteza e imprecisão aparecem quando a presença de um objeto (ficha) é conhecida mas sua localização é imprecisa em termos de um conjunto de lugares alternativos. A imprecisão da localização do objeto é determinada por uma distribuição de possibilidades que delimita as possíveis localizações do objeto. Nesse trabalho, as fichas são tuplas de objetos e os lugares são predicados representando o estado de alguns objetos. Uma distribuição de possibilidades é assinalada para cada tupla de objetos que determina os possíveis lugares os quais as instâncias do objeto pertencem. As transições podem ser disparadas de duas formas: o disparo clássico de uma transição (a localização das fichas é certa) ou pseudo-disparo. Quando um pseudo-disparo é efetuado, novas distribuições de possibilidades para algumas instâncias de objetos são computadas e a imprecisão sobre elas cresce. Então, fichas são depositadas nos lugares de saída das transições mas as fichas dos lugares de entrada não são removidas. A imprecisão dos objetos pode ser reduzida se uma mensagem é recebida sobre o estado real do sistema, i.e., o conhecimento sobre o sistema é atualizado.

Capítulo 3

FTPN

3.1 Introdução

Como discutido no capítulo anterior, as redes de Petri foram amplamente usadas para a modelagem e análise de sistemas concorrentes. Muitas razões contribuíram para essa larga aceitação: a sólida base matemática, a simplicidade do modelo, a imediata representação gráfica e a facilidade de modelar os aspectos paralelos e distribuídos. Entretanto, a definição original das redes de Petri falha na representação de aspectos temporais que são essenciais em muitos casos, como por exemplo, na modelagem de sistemas em tempo real. Diversas extensões foram propostas (ver Capítulo 2) com o objetivo de superar esta limitação. Existem duas abordagens principais: determinística e estocástica. Cada abordagem apresenta suas vantagens e limitações. As extensões determinísticas são basicamente utilizadas para modelar sistemas em tempo real em que as restrições temporais são representadas por valores determinísticos. Entretanto, elas são limitadas para fazer análise de desempenho e para a representação de incertezas que são comumentes encontradas nos sistemas reais. As extensões estocásticas são largamente utilizadas para fazer análise de desempenho de sistemas e são principalmente utilizadas para determinar alguns índices de desempenho agregados. No entanto, elas falham em modelar os sistemas em tempo real no qual a representação dos limites de tempo é fundamental [49]. Além do que, utilizando a abordagem estocástica é muito difícil calcular índices individuais de desempenho, como por exemplo, o tempo de ciclo mínimo ou o atraso para se alcançar uma determinada marcação.

Nesse capítulo, introduzimos o modelo *Rede de Petri com Temporização Nebulosa (FTPN)*. As *FTPNs* são uma extensão de redes de Petri que utilizam uma abordagem nebulosa para introduzir tempo em redes de Petri. O modelo tem a intenção de servir como uma ferramenta que seja adequada tanto para a modelagem de sistemas em tempo real como para fazer avaliação de desempenho de sistemas, i.e., esse modelo combina a capacidade de modelar sistemas em tempo real das extensões determinísticas com a habilidade de fazer avaliação de desempenho das extensões estocásticas. Como será mostrado adiante, isso é possível através da combinação entre intervalos de tempo e

a teoria dos conjuntos nebulosos. Os intervalos de tempo permitem a representação dos limites de tempo que são necessários para modelar sistemas em tempo real e os valores nebulosos permitem a estimação de alguns índices de desempenho. O modelo *FTPN* é mais geral do que as extensões temporais determinísticas pois, a partir da *FTPN*, é possível derivar as extensões determinísticas ajustando os valores das funções de pertencimento dos intervalos nebulosos. Apresentamos a unificação dessas extensões em termos de $FTPN$ no final desse capítulo.

No modelo *FTPN*, as fichas carregam uma função nebulosa de tempo (FTF) representando a possibilidade de existir uma ficha em um determinado lugar a partir de um ponto de referência, i.e., a escala do tempo. Além do mais, dois intervalos nebulosos E (intervalo de sensibilização) e D (intervalo de disparo), são associados à cada transição. A função nebulosa de tempo associada com as fichas e os intervalos nebulosos de tempo associados com as transições determinam o comportamento dinâmico de um *FTPN*. A definição de dois parâmetros para representar os aspectos temporais foi primeiramente proposta po Razouk [98]. Em seu trabalho, Razouk definiu dois valores fixos (sensibilização e disparo) para representar tempo. Em nosso trabalho, nós usamos o conceito similar para os intervalos de sensibilização e disparo mas, ao invés de valores fixos, nós usamos intervalos nebulosos devido à inerente imprecisão que encontramos nos sistemas reais.

Em suma, neste capítulo, apresentamos a definição formal das *FTPNs* e discutimos o comportamente dinâmico das mesmas. Para tanto, descrevemos as regras de computação das FTFs das novas fichas que são criadas durante a execução da rede. Além do mais, apresentaremos alguns exemplos e mostaremos como o modelo $FTPN$ unifica as extensões temporais determinísticas propostas ao modelo de redes de Petri.

3.2 Definição e Regra de Disparo

Uma *FTPN* é um grafo dirigido, bi-partido com dois tipos diferentes de nodos chamados de lugares e transições. Os lugares são graficamente representados por círculos e as transições por barras. Os lugares e as transições são conectados por um arco dirigido. Um lugar P é dito um lugar de entrada de uma transição t, se existe um arco dirigido que parte de P para t. Da mesma forma, um lugar P é dito lugar de saída de uma transição t, se existe um arco dirigido que parte de t para P. No sentido de caracterizar os aspectos ou restrições temporais de um evento, dois intervalos nebulosos de tempo são associados com cada transição. O intervalo de sensibilização E representa os tempos mínimo e máximo decorridos entre o instante de sua sensibilização e o seu disparo. O

intervalo de disparo D representa a duração do disparo, i.e., o atraso necessário para executar a ação representada pela correspondente transição. Os lugares podem conter uma ou mais fichas. Para cada ficha é associada uma função nebulosa de tempo, FTF, representando a possibilidade de sua existência em um determinado lugar em um dado instante de tempo, i.e., as fichas têm um período de validade em um lugar e, após esse período, elas não contribuem para o disparo de transições.

No que se segue, introduzimos alguns conceitos importantes para a definição formal de uma *FTPN*.

Definição 3.1 *TS é o conjunto da escala de tempo. $TS = \{x \in R \mid x \geq 0\}$, i.e., o conjunto de todos os números reais não-negativos.*

Para definir um intervalo nebuloso, consideramos duas datas nebulosas na escala de tempo TS. As datas nebulosas são números nebulosos, i.e., são subconjuntos nebulosos normais e convexos (ver Capítulo 2). Um intervalo nebuloso é definido como um conjunto de pontos de tempo que estão entre duas datas nebulosas. Nossa definição de intervalos nebulosos é semelhante à definição apresentada por Dubois [41]. O intervalo nebuloso delimitado por duas datas nebulosas pode ser considerado como um número nebuloso pois, de acordo com a definição adiante, o subconjunto nebuloso que representa o intervalo nebuloso é normal e convexo.

Definição 3.2 *Considere y e z duas datas nebulosas na escala de tempo TS, i.e., $y, z \in TS, \forall \mu_y(t), \mu_z(t) \in [0,1]$. Um intervalo nebuloso de tempo, $FINT = \{[y,z] \in TS \times TS \mid z \geq y\}$, de pontos de tempo que são maiores que y e menores do que z, é definido pela função de pertencimento $\mu(FINT) = \mu_{[y,z]}(t) = sup\ min(\pi_y(s), \pi_z(s'))$ onde $t, s, s' \in TS$ e $\pi_y(s), \pi_z(s')$ delimita o conjunto nebuloso dos possíveis valores de y e z, respectivamente.*

Uma FTF é também definida como uma data má conhecida que é atribuída para cada ficha em um lugar. As FTFs das fichas são combinadas para determinar ou atualizar as FTFs das fichas subseqüentes.

Definição 3.3 *Suponha que P é um lugar e x é uma data má conhecida representando o valor de tempo que define a existência de uma ficha no lugar P. Então, a função nebulosa de tempo, associada com a ficha no lugar P é $FTF_x(P) = \pi_x(TS)$, em que $\pi_x(TS)$ delimita o conjunto nebuloso dos possíveis valores de x.*

Definição 3.4 *Uma ficha nebulosa é uma ficha que carrega uma função nebulosa de tempo.*

Na seqüência, nós definimos formalmente as *FTPNs*.

Definição 3.5 *As FTPNs são definidas formalmente pela 7-tupla (P, T, I, O, $\mathcal{E}$, $\mathcal{D}$, M0), em que:*

P é um conjunto finito de lugares.

T é um conjunto finito de transições.

$I : T \rightarrow P^{\infty}$ é a função de entrada das transições.

$O : T \rightarrow P^{\infty}$ é a função de saída das transições.

$\mathcal{E} : T \rightarrow E([e_1, e_2])$ é o conjunto dos intervalos nebulosos de sensibilização, em que E é um $FINT$.

$\mathcal{D} : T \rightarrow D([d_1, d_2])$ é o conjunto dos intervalos nebulosos de disparo, em que D é um $FINT$.

$M0 : P \rightarrow FTF_{\tau}(P)$ é a marcação inicial da rede, em que $FTF_{\tau}(P)$ é uma função nebulosa de tempo caracterizando a distribuição de possibilidades de existir uma ficha em um lugar P em um dado instante τ, em que $\tau \in TS$.

Semelhante à definição de redes de Petri, $P \cap T = \emptyset$ e $P \cup T \neq \emptyset$. No mais, por uma questão de simplicidade, vamos considerar que o conjunto de lugares de entrada de uma transição t, $t \in T$, é representado por $\bullet t$ e o conjunto dos lugares de saída de uma transição t é $t\bullet$.

Uma vez que nesse modelo cada ficha carrega uma FTF que é definida após o disparo de uma transição, o comportamento dinâmico de uma $FTPN$ difere do comportamento dinâmico de uma rede de Petri atemporal. O comportamento dinâmico de uma $FTPN$ depende das FTFs das fichas e dos intervalos nebulosos os quais modificam as conhecidas regras de disparo. As regras de disparo do modelo $FTPN$ são dadas por:

a) Uma transição t é dita sensibilizada se existe, pelo menos, uma ficha em cada lugar de entrada (assume-se que os arcos têm peso 1) em um mesmo ponto na escala do tempo, isto é, dada uma transição t e o conjunto de seus lugares de entrada $(P_1, P_2, ..., P_n)$ com as respectivas funções nebulosas de tempo $FTF(P_1), FTF(P_2), ..., FTF(P_n)$, a transição t é sensibilizada se e somente se:

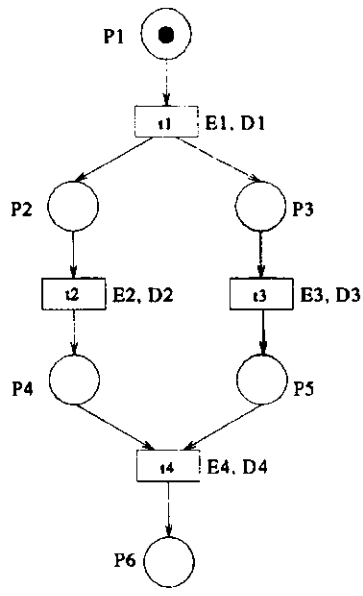

Figura 3.1: Uma *FTPN* simples

$$(FTF(P_1) \cap FTF(P_2) \cap ... \cap FTF(P_n)) \neq 0 \tag{3.1}$$

b) Se, em um dado momento, mais de uma transição está sensibilizada uma escolha não determinística determina qual a próxima transição a disparar. Entretanto, é importante lembrar que, em alguns casos dependendo de seus limites, os intervalos de sensibilização podem contribuir para a escolha da próxima transição a disparar. Essa escolha não-determinística reflete a situação de time-out como explicaremos adiante.

c) A transição permanece sensibilizada por um período E antes de disparar.

d) Uma transição começa a disparar e continua por um período D.

e) Após o disparo, as fichas são removidas dos lugares de entrada e uma ficha que carrega o novo valor da função nebulosa de tempo é depositada em cada lugar de saída da transição. Esta nova função é obtida a partir das funções nebulosas de tempo carregadas pelas fichas dos lugares de entrada e dos intervalos nebulosos de tempo associados à transição. Descrevemos a computação das novas funções nebulosas de tempo na Seção 3.3.

$P = \{P1, P2, P3, P4, P5, P6\}$

$T = \{t1, t2, t3, t4\}$

$\bullet t1 = \{P1\}$, $t1\bullet = \{P2, P3\}$

$\bullet t2 = \{P2\}$, $t2\bullet = \{P4\}$

$\bullet t3 = \{P3\}$, $t3\bullet = \{P5\}$

$\bullet t4 = \{P4, P5\}$, $t4\bullet = \{P6\}$

$d(t1) = D1$, $e(t1) = E1$

$d(t2) = D2$, $e(t2) = E2$

$d(t3) = D3$, $e(t3) = E3$

$d(t4) = D4$, $e(t4) = E4$

$M0 = (1, 0, 0, 0, 0, 0)$

$FTF(P1)$

Figura 3.2: Definição da $FTPN$ mostrada na Figura 3.1

Vamos introduzir um exemplo simples para ilustrar o conceito de *FTPN*, Figura 3.1. A rede é composta por seis lugares e quatro transições. De acordo com a definição de *FTPN*, dois intervalos nebulosos de tempo são associados com cada transição. Logo, no exemplo, o intervalo de sensibilização $E1$ e o intervalo de disparo $D1$ são associados à transição *t1*, os intervalos $E2$ e $D2$ são associados à *t2*, e assim por diante. Nós assumimos uma marcação inicial de uma ficha no lugar *P1* e zero fichas nos demais lugares. Uma função nebulosa de tempo, $FTF(P1)$, é carregada pela ficha no lugar *P1* que determina a possibilidade de se encontrar no lugar *P1*. Formalmente, a $FTPN$ mostrada na Figura 3.1 é definida como na Figura 3.2.

Considerando a definição de $FTPN$ e as regras de disparo apresentadas anteriormente, a partir do exemplo da Figura 3.1 temos que a disponibilidade da ficha no lugar *P1* para sensibilzar a transição *t1* é determinada pela função nebulosa de tempo $FTF(P1)$, i.e., $\bigcap FTF(\bullet t1) = FTF(P1)$. Se a ficha está disponível, a transição *t1* só fica sensibilizada depois de um período de tempo e que é determinado pelo intervalo nebuloso de sensibilização $E1$. A transição *t1* começa a disparar e permanece a disparar

por um periodo d que é determinado pelo intervalo de disparo $D1$. Então, uma ficha é depositada nos lugares *P2* e *P3* com novas FTFs, $FTF(P2)$ e $FTF(P3)$, respectivamente. As funções $FTF(P2)$ e $FTF(P3)$ são idênticas pois as fichas depositadas nos lugares *P2* e *P3* foram criadas pelo disparo da mesma transição. Da mesma forma, as fichas saem do lugar *P2* para o lugar *P4* e do lugar *P3* para o lugar *P5*. Nesse estágio, a marcação corrente é composta de uma ficha no lugar *P4* e uma outra ficha no lugar *P5* com funções nebulosas de tempo $FTF(P4)$ e $FTF(P5)$, respectivamente. Então, de acordo com as regras de disparo descritas acima, a transição *t4* pode disparar se e somente se existir a possibilidade da presença de uma ficha nos lugares *P4* e *P5* no mesmo instante de tempo, i.e., $FTF(P4) \cap FTF(P5) \neq \emptyset$. Se essa condição é satisfeita, *t4* dispara e uma ficha é depositada no lugar *P6* carregando uma nova função nebulosa de tempo.

3.3 Regra de Computação da Função Nebulosa

Nessa seção, nós apresentamos como computar as FTFs das fichas criadas durante a execução de um *FTPN*. A função nebulosa de tempo associada à ficha de saída criada após o disparo de uma transição depende das funções nebulosas de tempo das fichas de entrada e dos intervalos nebulosos de tempo associados com a transição disparada. Em geral, essa computação pode ser efetuada em dois passos. No primeiro passo, as funções nebulosas de tempo das fichas de entrada são combinadas para verificar a possibilidade de existência de fichas nos lugares de entrada em um mesmo instante de tempo. No segundo passo, a função nebulosa de tempo resultante da combinação efetuada no passo um é composta com os intervalos nebulosos de tempo da transição disparada. No restante dessa seção, nós detalhamos esses dois passos.

3.3.1 Passo 1: Computação das Fichas Nebulosas

Considere a escala de tempo TS, definida na Seção 3.2, como conjunto de referência. A possibilidade da disponibilidade de uma ficha em um lugar P em um instante de tempo τ, $\tau \in TS$, é dado pela função característica ou função de pertencimento μ:

$$\forall \tau \in TS : \mu_P(\tau) \in [0, 1] \tag{3.2}$$

Consideramos dois casos na obtenção da função nebulosa de tempo resultante a partir das fichas dos lugares de entrada. Vamos considerar a situação mostrada na Figura 3.3 e supor que $FTF(P1)$ é a função nebulosa de tempo associada à ficha no

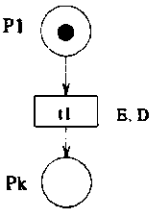

Figura 3.3: Um lugar de entrada

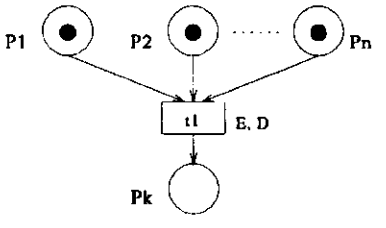

Figura 3.4: n lugares de entrada

lugar *P1*. Se $FTF(P1)$ representa a possibilidade de existir uma ficha no lugar *P1* e seu valor é determinado pela função de pertencimento $\mu_{P1}(\tau)$, a função nebulosa de tempo resultante $\mu_R(\tau)$ é

$$\mu_R(\tau) = \mu_{P1}(\tau) \tag{3.3}$$

O segundo caso é quando a transição tem mais de um lugar de entrada, como mostrado na Figura 3.4. Nesse caso, para disparar uma transição, a função nebulosa de tempo resultante, $\mu_R(\tau)$, é a intersecção das funções nebulosas de tempo das fichas de entrada. Na Figura 3.4, $FTF(P1)$, $FTF(P2)$, ..., $FTF(Pn)$ são as funções nebulosas de tempo carregadas pelas fichas nos lugares *P1*, *P2*, ..., *Pn*, respectivamente. Vamos considerar que $FTF(P1)$, $FTF(P2)$, ..., $FTF(Pn)$ são representadas pelas funções de pertencimento $\mu_{P1}(\tau), \mu_{P2}(\tau), ..., \mu_{Pn}(\tau)$, respectivamente. A função nebulosa de tempo resultante, $\mu_R(\tau)$, é dada por:

$$\mu_R(\tau) = \bigcap_{i=0}^{n} \mu_{Pn}(\tau) = MIN(\mu_{P1}(\tau), \mu_{P2}(\tau), ..., \mu_{Pn}(\tau)). \tag{3.4}$$

Definição 3.6 *O operador* $\diamond$ *é definido como a intersecção das funções de pertencimento de seus operandos (executa o* **Passo 1***).*

3.3.2 Passo 2: Função Nebulosa de Tempo Final

Nesse passo, a função nebulosa de tempo resultante obtida a partir das fichas nebulosas de entrada é combinada com os intervalos nebulosos de tempo associados com a transição disparada. Suponha que $\mu_R(\tau)$ representa a função nebulosa de tempo resultante considerando as fichas nebulosas de entrada para a transição *t1* na Figura 3.4. Vamos supor também que o intervalo de disparo D associado à transição *t1* é dado pelo intervalo $[0, 0]$ (disparo instantâneo). Nesse caso, quando uma ficha alcança o lugar *Pk*, a função de pertencimento representando a função nebulosa de tempo associada à ficha no lugar *Pk*, $\mu_{Pk}(\tau)$, é a combinação entre a função nebulosa de tempo resultante $\mu_R(\tau)$ e o intervalo de sensibilização E associado à transição *t1*. Essa combinação é determinada pela adição de variáveis nebulosas utilizando as funções de pertencimento:

$$\mu_{(A(+)B)}(z) = MAX_{z=x+y} MIN(\mu_A(x), \mu_B(y)). \tag{3.5}$$

em que z, x e $y \in TS$, $\mu_A(x)$, $\mu_B(y)$ e $\mu_{(A(+)B)}(z)$ representa a função nebulosa de tempo resultante, o intervalo de sensibilização e a função nebulosa de tempo final associada à ficha no lugar *Pk*, respectivamente.

Para considerar o intervalo de disparo D associado à transição, o mesmo método de adição de variáveis nebulosas é usado, i.e., nesse caso a função nebulosa de tempo final é a combinação entre a função nebulosa de tempo resultante e os intervalos nebulosos de tempo de sensibilização e disparo, E e D, respectivamente.

Definição 3.7 *O operador $\oslash$ representa a adição de variáveis nebulosas (executa o* **Passo 2***).*

Retornando ao exemplo da Figura 3.1, se nós simularmos o comportamente dinâmico da rede mostrada, nós determinamos as seguintes equações para computar as demais $FTFs$. Por uma questão de brevidade, vamos supor que $F1 = FTF(P1)$, $F2 = FTF(P2)$ e assim por diante. A partir das Equações 3.6, 3.7, 3.8 e 3.9, as seguintes $FTFs$ são computadas:

$$F_2 = F_3 = F_1 \oslash E_1 \oslash D_1. \tag{3.6}$$

$$F_4 = F_2 \oslash E_2 \oslash D_2. \tag{3.7}$$

$$F_5 = F_3 \oslash E_3 \oslash D_3. \tag{3.8}$$

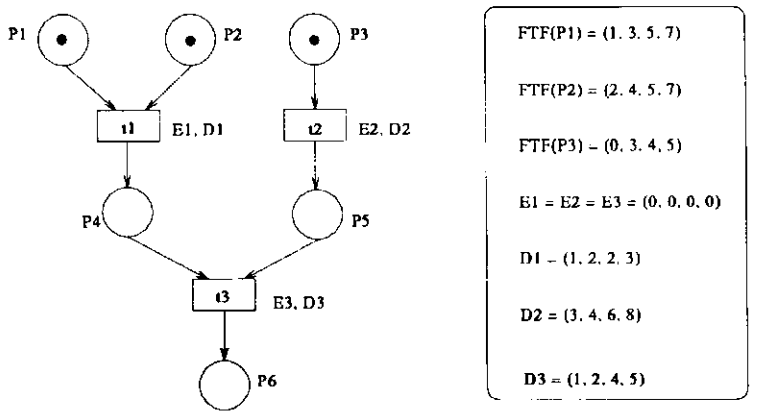

Figura 3.5: Uma *FTPN*

$$F_6 = (F_4 \diamond F_5) \oslash E_4 \oslash D_4. \tag{3.9}$$

Uma vez que as fichas nos lugares *P2* e *P3* são criadas pelo disparo de uma única transição, *t1*, os valores das FTFs associados às fichas criadas são iguais, como observado na Equação 3.6. As Equações 3.7 e 3.8 determinam os valores das FTFs para as fichas nos lugares *P4* e *P5*, respectivamente. A Equação 3.9 computa a FTF carregada pela ficha no lugar *P6*. Se a função combinada $(F_4 \diamond F_5)$ é nula, a ficha não alcança o lugar *P6*.

Um exemplo diferente é mostrado na Figura 3.5. Nesse exemplo, as FTFs iniciais e os intervalos nebulosos de tempo são definidos como números nebulosos trapezoidais. Os intervalos de sensibilização são definidos pela quádrupla *(0, 0, 0, 0)* e representam a sensibilização instantânea das transições. Na Figura 3.6, são apresentados os números nebulosos representando os valores das FTFs iniciais e os intervalos de disparo. A Figura 3.6(a) mostra a representação gráfica da função nebulosa de tempo associada com a ficha no lugar *P1*, que é dada pela quádrupla *FTF(P1) = (1, 3, 5, 7)*. As Figuras 3.6(b), 3.6(c), 3.6(d), 3.6(e) e 3.6(f) representam, respectivamente, as funções nebulosas de tempo associadas com os lugares *P2* e *P3*, e os intervalos de disparo associados às transições *t1*, *t2* e *t3*.

De acordo com as regras de disparo, as transições *t1* e *t2* podem disparar. O disparo de *t1* significa que as fichas de entrada são combinadas resultando no número nebuloso definido pela quádrupla *(2, 4, 5, 7)* que é posteriormente combinado com o intervalo de disparo *D1*. A combinação final determina a função nebulosa de tempo associada com a ficha no lugar *P4* cujo valor é definido pela quádrupla *(3, 6, 7, 10)*. As funções nebulosas de tempo das fichas nos lugares *P4*, *P5* e *P6* são computadas de forma similar

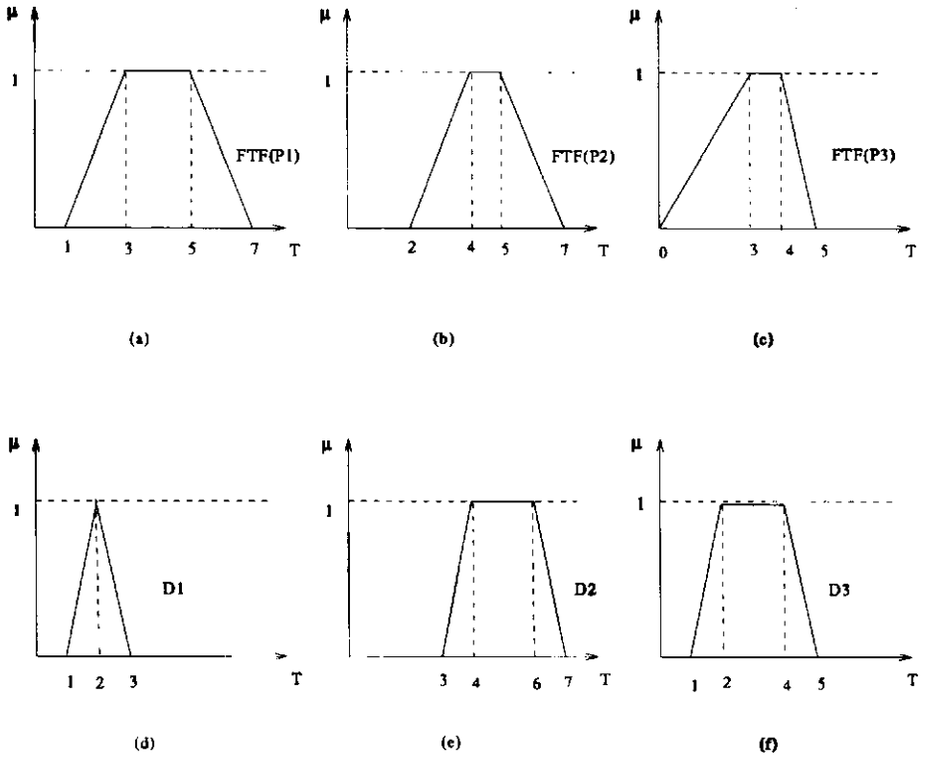

Figura 3.6: Caracterização nebulosa da *FTPN* da Figura 3.5

e são definidas, respectivamente, pelas quádruplas *FTF(P4) = (3, 6, 7, 10)*, *FTF(P5) = (3, 7, 10, 12)* e *FTF(P6) = (4, 9, 11, 15)*.

Em suma, para computar as funções nebulosas de tempo das fichas subseqüentes, nós temos primeiro que combinar as funções nebulosas de tempo das fichas de entrada e depois combinar o resultado com os intervalos de tempo. É importante notar que, a intersecção de dois subconjuntos nebulosos representando a *FTF* das fichas de entrada pode ser um subconjunto nebuloso que não seja convexo ou normal. Logo, para aplicar a técnica descrita no segundo passo é necessário, quando for o caso, normalizar tais subconjuntos nebulosos.

3.4 Considerações Sobre Intervalos Nebulosos de Tempo

A existência de dois intervalos nebulosos de tempo associados às transições é necessária para permitir a representação de dois diferentes conceitos temporais: *time-outs* e atrasos de processamento. De um lado, o intervalo de sensibilização é bom para modelar *time-outs*. Se a transição está sensibilizada então, depois de decorrido um determinado tempo representado pelo valor do limite superior do intervalo, ela pode disparar. Na Figura

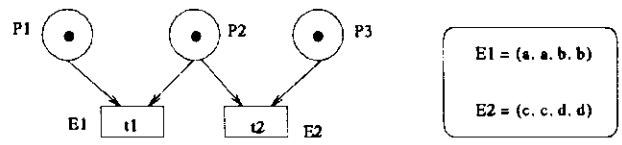

Figura 3.7: Uma situação de *time-out* modelada por uma *FTPN*

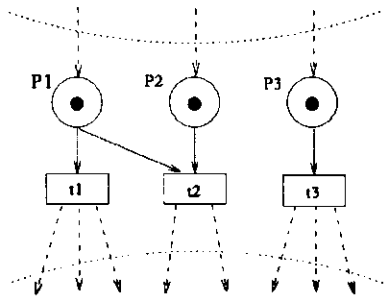

Figura 3.8: Parte de uma rede

3.7, mostramos um pedaço de uma *FTPN* modelando uma situação de *time-out*. Os intervalos de disparo e os valores das FTFs iniciais não são representados na Figura 3.7. Os intervalos de sensibilização $E1$ e $E2$ associados, respectivamente, com as transições *t1* e *t2* são definidos como números nebulosos que são representados pelas quádruplas $E1 = (a, a, b, b)$ e $E2 = (c, c, d, d)$. Escolhemos representar os intervalos nebulosos de tempo por números nebulosos trapezoidais uma vez que eles são mais gerais do que os intervalos *crisp* e números nebulosos triangulares. O intervalo de sensibilização é importante porque, em alguns casos, quando temos conflito, ele pode decidir que transição disparará. Retornando para a situação de *time-out* mostrada na Figura 3.7, se o limite inferior do intervalo de sensibilização associado com a transição *t2* é maior do que o limite superior do intervalo de sensibilização associado com a transição *t1*, $(c > b)$, a transição *t2* nunca disparará pois, no pior caso, *t1* deve disparar decorrido um período de tempo igual a b.

O intervalo de sensibilização também é adequado para representar prioridade de atividades. Na realidade, a priorização de atividades é uma generalização da situação de time-out. Vamos supor que, em um dado momento, nós temos três transições sensibilizadas como mostrado na Figura 3.8. Logo, considerando a seguinte ordem de prioridade, $PRIOR(t3) > PRIOR(t2) > PRIOR(t1)$, *t3* deve disparar primeiro, seguido do disparo de *t2* e de *t1*. Para garantir que *t3* dispare primeiro, podemos ajustar o conjunto de intervalos de sensibilização das transições *t1*, *t2* e *t3* como mostrado na Figura 3.9. O intervalo de sensibilização $E = [1, 1]$ associado com a transição *t3* indica que a transição

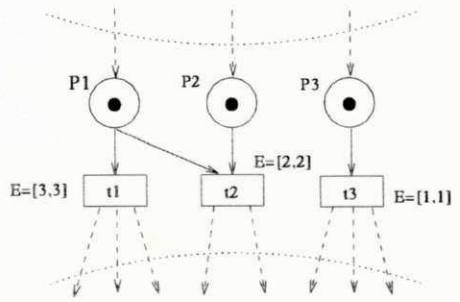

Figura 3.9: Priorização das transições da rede mostrada na Figura 3.8

t3 deve disparar primeiro do que as transições *t2* e *t1* cujos intervalos de sensibilização a elas associados são $E = [2, 2]$ e $E = [3, 3]$, respectivamente. Note que, se considerarmos uma ficha em cada lugar, a transição *t1* nunca disparará pois *t2* tem prioridade sobre *t1* e ambas as transições compartilham um lugar de entrada comum. Logo, após o disparo de *t2*, a transição *t1* não estará mais sensibilizada.

Por outro lado, o intervalo de disparo é adequado para modelar atrasos de processamento de atividades. Isso corresponde à duração da atividade ou evento.

Esses dois tipos de intervalos são importantes quando tratando com sistemas em tempo real uma vez que, nesse tipo de sistema, diferentes restrições temporais são definidas [29, 112]. Existem três categorias gerais de restrições temporais: restrições de tempo máximo, restrições de tempo mínimo e restrições de duração. As restrições de tempo máximo estão relacionadas com o tempo máximo que deve ocorrer entre dois eventos. A restrição de tempo mínimo indica um tempo mínimo que deve ocorrer entre dois eventos. Finalmente, as restrições de duração indicam para que total de tempo um evento deve ocorrer. Os intervalos nebulosos de sensibilização e disparo servirão como base para especificar essas três categorias de restrições temporais. Essas restrições temporais podem ser chamadas de restrições temporais de desempenho porque elas influenciarão nos tempos de resposta dos sistemas. Logo, os intervalos de sensibilização e disparo podem ser usados para determinar alguns índices de desempenho interessantes como mostraremos no Capítulo 5.

3.5 Requisitos para Considerar o Tempo

Como foi exaustivamente discutido nas seções anteriores, a função nebulosa de tempo carregada pelas fichas representa a possibilidade de uma ficha em um dado lugar, i.e., a possibilidade de uma dada condição ser verdadeira considerando a escala do tempo. Quando estamos tratando com o tempo, temos que fazer algumas considerações para

manter a sua noção intuitiva. Uma vez que estamos considerando redes de Petri nas quais temos uma natural ordem de eventos que é caracterizada pelas transições e suas dependências na rede, a propriedade de monotonicidade deve ser satisfeita.

Vamos primeiro considerar o conceito da relação de ordem $\leq$ entre dois intervalos dependentes.

Definição 3.8 *Vamos supor dois intervalos* $[a1, a2]$ *e* $[b1, b2]$ *onde,* $a1 \leq a2$ *e* $b1 \leq b2$. $[a1, a2] \leq [b1, b2] \Leftrightarrow a1 \leq b1 \wedge a2 \leq b2$.

Podemos definir marcação e estado em uma $FTPN$ da seguinte forma:

Definição 3.9 *Uma marcação* m *de uma FTPN é uma função* $m : P \rightarrow \{0, 1, 2, ...\}$.

Definição 3.10 *Um estado* S *de uma FTPN é definida como a distribuição de fichas ou sua marcação* $S = \{m(p1), m(p2), ..., m(pn)\}$ *cada uma conduzindo uma função nebulosa de tempo.*

De acordo com a definição de $FTPN$, cada ficha carrega uma função nebulosa de tempo FTF.

Definição 3.11 *O tempo nebuloso para um determinado estado* s_i, $ft(s_i)$, *é a intersecção das funções nebulosas de tempo associadas com as fichas que pertencem a* s, *i.e.,* $ft(s_i) = FTF(p1) \cap FTF(p2) \cap .. \cap FTF(pn)$ *em que,* $s_i = [m(p1), m(p2), ..., m(pn)]$.

Os conceitos do *menor tempo possível* (*earliest possible time*) e *maior tempo possível* (*latest possible time*) podem ser diretamente definidos a partir do tempo nebuloso dos estados. Uma vez que estamos considerando números nebulosos, *ept* e *lpt* são definidos para cada nível de presunção (veja o Capítulo 2). Então, para cada nível de presunção, definimos *ept* e *lpt* da seguinte forma:

Definição 3.12 $ept(s_i) = min\{f\}$ *em que,* $f \in ft(s_i)$.

Definição 3.13 $lpt(s_i) = max\{f\}$ *em que,* $f \in ft(s_i)$.

Axioma 3.1 *Para qualquer disparo, a FTF de todas as fichas que foram criadas são iguais.*

O Axioma 3.1 reflete a definição da regra de disparo.

Axioma 3.2 *As FTFs carregadas pelas fichas que foram criadas são idênticas e maiores do que o tempo nebuloso do estado anterior.*

O Axioma 3.2 é garantido pela definição do intervalo de disparo.

Teorema 3.1 *Se* s_i, $s_j \in S$ *e* $s_i \rightarrow s_j$, *então* $ft(s_i) \leq ft(s_j)$.

Prova

Em nossa prova, sem perda de generalidade, estamos considerando apenas um nível de presunção. Isso é possível porque os números nebulosos são subconjuntos nebulosos que são convexos e normais. Após o disparo de uma transição, fichas serão removidas e fichas serão adicionadas. De acordo com a definição de *FTPN*, os intervalos de tempo associados com as transições são restritos aos valores positivos e, a partir do Axioma 1, todas as fichas adicionadas carregam a mesma FTF que será, pela definição 3.8, maior do que $ft(s_i)$. Então, se as novas fichas adicionadas são maiores do que o tempo nebuloso do estado anterior, é trivial ver que $ept(s_i) \leq ept(s_j)$ e $lpt(s_i) \leq lpt(s_j)$ e, conseqüentemente, $ft(s_i) \leq ft(s_j)$.

□

3.6 Generalização das Extensões Temporais Determinísticas

A maior característica do modelo $FTPN$ é a sua capacidade de modelar sistemas em tempo real e de fazer avaliação de desempenho de sistemas. Na verdade, o modelo $FTPN$ é mais geral do que as extensões temporais determinísticas que foram propostas para o modelo de redes de Petri pois, podemos representá-las por meio de uma *FTPN*. Essas extensões podem ser modeladas através de uma $FTPN$ impondo algumas restrições para os intervalos de sensibilização e disparo ajustando adequadamente suas caracterizações nebulosas. Como mostrado no Capítulo 2, as redes de Petri foram estendidas no sentido de incorporar o conceito de tempo utilizando diversas abordagens. Focalizando nas extensões determinísticas, as redes de Petri foram estendidas associando basicamente valores de tempo com as transições e lugares. Os valores de tempo podem ser representadas por intervalos ou pontos fixos. No Capítulo 2, descrevemos as três extensões determinísticas clássicas. Nessa seção, mostramos como algumas extensões como *TPN* [79] e *TdPN* [96] podem ser modeladas através de *FTPN*. O modelo *TdPTN* não é discutido uma vez que esse modelo é equivalente ao modelo *TdPN*.

Na unificação, consideramos dois aspectos que distinguem $FTPN$ das outras extensões temporais: a função nebulosa de tempo associada com as fichas e os intervalos nebulosos de tempo associados com as transições. Nas extensões clássicas temporais,

as fichas não possuem um período de validade ou disponibilidade e, uma vez que chegam em um lugar, elas permanecem nele até que sejam removidas pelo disparo de uma transição.

Lema 3.1 *Uma ficha em uma rede de Petri clássica pode ser representada através de uma ficha nebulosa definida no modelo FTPN.*

Prova

Para representar uma ficha clássica através de uma ficha nebulosa, é necessário considerar que uma ficha não tem um período de validade. Vamos supor que uma ficha chega a um lugar em um tempo τ. É fácil ver que, a FTF correspondendo à ficha tradicional é representada pelo intervalo nebuloso $[\tau, \infty]$, onde $\mu(x) = 1$, $\forall x | x > \tau$. Para as fichas iniciais $\tau = 0$.

□

Lema 3.2 *Um intervalo nebuloso de disparo pode modelar uma duração de tempo fixa.*

Prova

Isso pode ser alcançado ajustando os limites superior e inferior dos intervalos nebulosos de disparo para o valor da duração fixa. Nesse caso, não existe imprecisão sobre os valores. Logo, $D = [d, d]$, $\mu(d) = 1$.

□

O modelo de *rede de Petri temporal (TPN)* proposto por Merlin é caracterizado pela atribuição de um intervalo de tempo $TI = [t_{min}, t_{max}]$ para cada transição.

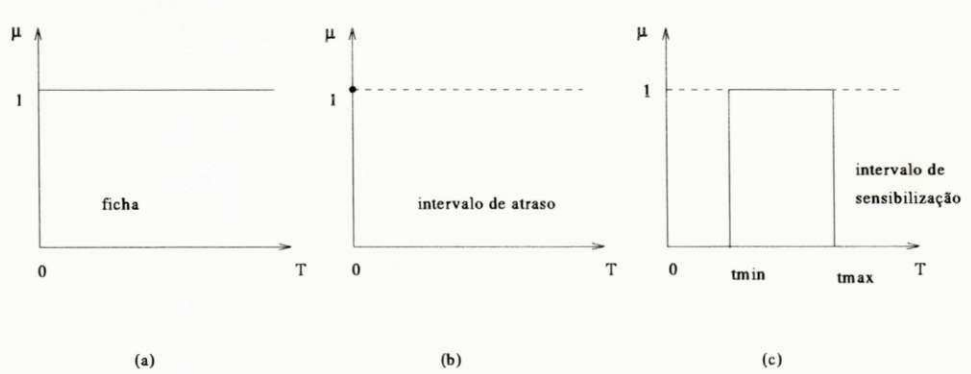

Figura 3.10: Generalização de uma *TPN*

Lema 3.3 *O intervalo de tempo TI associado com as transições no modelo TPN pode ser modelado pelo intervalo nebuloso de sensibilização E do modelo FTPN.*

Prova

A definição dos dois intervalos são equivalentes, i.e., eles representam o mesmo tipo de restrição temporal. Eles têm um tempo mínimo que deve decorrer antes de sensibilizar a transição, bem como um valor máximo que a transição deve esperar para disparar, após o qual a transição deve disparar. A diferença é que no intervalo nebuloso existem diferentes graus de pertencimento associados aos pontos dos intervalos. Então, o intervalo $TI = [t_{min}, t_{max}]$ pode ser representado pelo intervalo de sensibilização assumindo que todos os pontos do intervalo têm o mesmo grau de possibilidade, i.e., $E = [t_{min}, t_{max}]$, $\mu(t) = 1, \forall T, t_{min} \leq T \leq t_{max}$.

□

Teorema 3.2 *Uma TPN pode ser modelada por uma FTPN.*

Prova

De acordo com a Definição 2.4, a *TPN* é caracterizada por dois valores de tempo *tmin* e *tmax*. Esses valores de tempo são relativos ao momento τ no qual todos os lugares do conjunto de lugares de entrada da transição t estão marcados. Assumindo que isso acontece no tempo igual a τ, então, a transição t não pode disparar antes de $\tau + t_{min}$, e deve disparar dentro de $\tau + t_{max}$. Esse conceito pode ser facilmente representado pelo intervalo de sensibilização. A partir dos Lemas 3.1 e 3.3, garantimos que podemos representar as fichas e os intervalos de tempo do modelo *TPN*. É necessário considerar que não existe atraso no disparo da transição. Isso é diretamente obtido através do Lema 3.2 ajustando o valor da duração para 0. Isso é,

1. $E = [tmin, tmax]$, $\mu(e) = 1$, $tmin \leq e \leq tmax$.

2. $D = [0, 0]$, $\mu(0) = 1$.

□

A Figura 3.10 mostra como a função nebulosa de tempo e os intervalos nebulosos de tempo são caracterizados para representar uma *TPN*. A Figura 3.10(a) mostra a caracterização das funções nebulosas de tempo das fichas iniciais. Na Figura 3.11(a), uma *TPN* simples é mostrada enquanto que na Figura 3.11(b), sua correspondente

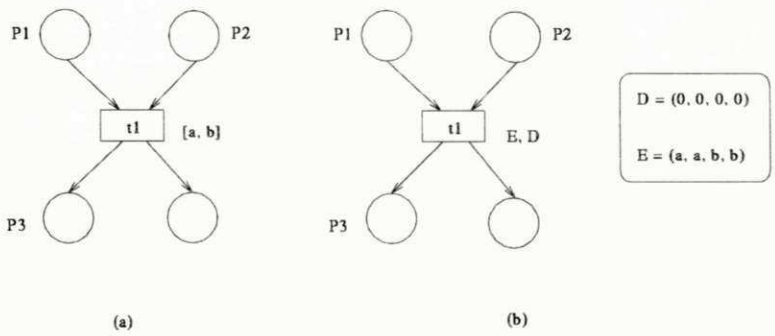

Figura 3.11: (a) *TPN*, (b) *FTPN* correspondente

FTPN é mostrada. Os intervalos de sensibilização e disparo são representados por números nebulosos trapezoidais os quais são denotados por quádruplas $D = (0, 0, 0, 0)$ (o atraso é nulo) e $E = (0, 0, 0, 0)$ (o intervalo de sensibilização é um intervalo *crisp*).

Teorema 3.3 *Uma TdPN pode ser modelada por uma FTPN.*

Prova

De acordo com a Definição 2.3, uma duração fixa d é atribuída para cada transição. Também, uma vez que as fichas de entrada chegam aos lugares de entrada, a transição é sensibilizada intantaneamente. Essas duas condições podem ser expressadas através de uma *FTPN* (Lema 3.2 e Lema 3.3). Isso é,

1. $E = [0, 0]$, $\mu(0) = 1$.

2. $D = [d, d]$, $\mu(d) = 1$.

□

Da mesma forma, a representação gráfica para os valores de uma *FTPN* modelando uma *TdPN* é mostrada na Figura 3.12.

Teorema 3.4 *Uma rede de Petri clássica (não-temporal) pode ser modelada através de uma FTPN.*

Prova

Esse Teorema pode ser provado através dos Lemas 3.1, 3.2 e 3.3. O Lema 3.1 indica a modelagem da ficha clássica. Através dos Lemas 3.2 e 3.3, garantimos que não existe nenhuma dependência temporal na regra de disparo, i.e., os intervalos nebulosos são representados por:

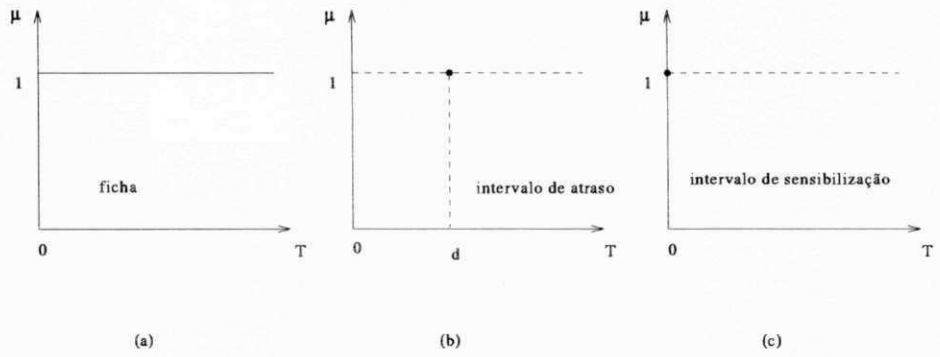

Figura 3.12: Generalização da *TdPN*

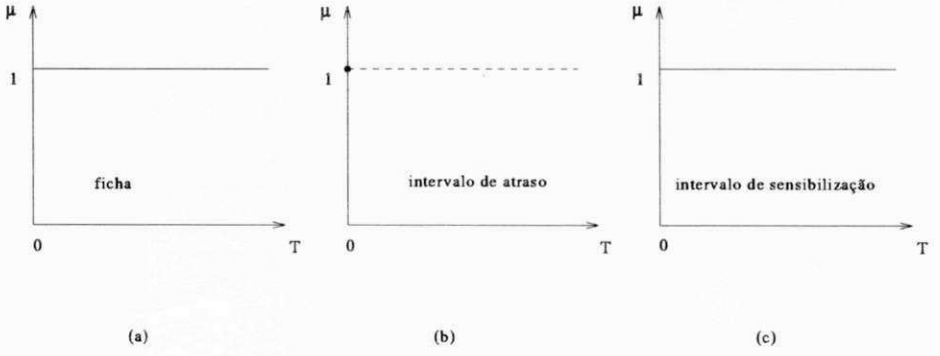

Figura 3.13: Generalização de uma rede de Petri clássica

1. $D = [0, 0]$ com valor de pertencimento $\mu(0) = 1$.

2. $E = [0, \infty]$, $\mu(t) = 1, \forall t, 0 \leq t \leq \infty$.

□

A caracterização dos elementos nebulosos para representar uma rede de Petri clássica é mostrada na Figura 3.13.

Capítulo 4

SISTEMAS DE G-NETS COM TEMPORIZAÇÃO NEBULOSA

4.1 Introdução

A modelagem e análise de grande sistemas complexos utilizando redes de Petri pode ser intratável devido ao problema de explosão de estados. Essa dificuldade está relacionada tanto ao aspecto comportamental quanto ao aspecto temporal. Centralizando na análise temporal, como dito anteriormente, as redes de Petri foram utilizadas em diferentes áreas de aplicação. Entretanto, devido ao problema de explosão de estados, muitas limitações são encontradas quando tratamos com sistemas complexos. Logo, para analisar ou avaliar desempenho de sistemas complexos, é necessário uma ferramenta de estruturação que permita a redução de sua complexidade. Além do mais, se estamos tratando com sistemas em tempo real, esta ferramenta de estruturação deve ser capaz de lidar com os aspectos temporais, seja através de sua extensão ou pela incorporação de alguma técnica ou metodologia específica, que permita a representação e análise dos aspectos temporais.

As redes de Petri de Alto nível [36, 47, 50, 61] são uma classe de extensões de redes de Petri propostas com o objetivo de reduzir a complexidade de modelagem e análise de sistemas complexos. Este tipo de redes de Petri propicia uma maneira simples e elegante de representar detalhadamente os aspectos funcionais de um sistema. Entretanto, as redes de Petri de alto nível não possuem mecanismos para representar tempo. Muitos esforços foram feitos para estender os modelos de redes de Petri de alto nível para permitir a representação e análise de sistemas dependentes do tempo. As *redes de Petri estocásticas de alto nível* [76] e as *redes de Petri estocásticas coloridas* [126] são redes de Petri de alto nível estendidas integrando o conceito de *redes de Petri estocásticas* dentro das *redes de Predicado/Transição* e *redes de Petri coloridas*, respectivamente. Essas redes estocásticas de alto nível permitem a computação de índices de desempenho de sistemas complexos através da redução do espaço de estados. Entretanto, mesmo considerando essa redução do espaço de estados, no caso de sistemas grandes, sua avaliação permanece complexa uma vez que o conceito de modularidade não está inerente a estes

tipos de redes, i.e., não foi definido como dividir o sistema em partes, estudá-las em separado e depois combinar os resultados para chegar a uma solução global. As redes de Petri de alto nível também foram estendidas utilizando técnicas determinísticas. Utilizando as *redes E-R*, uma classe especial de redes de Petri chamada de *redes TB* [50] foi proposta. Essa combinação é adequada para modelar sistemas em tempo real mas é limitada para efetuar análise temporal de sistemas. Uma outra extensão determinística é representada pelo modelo *ITCPN* [122]. Diferentes métodos de análise foram propostos para o modelo *ITCPN* mas não existe nenhuma metodologia de análise que utilize uma abordagem modular. Também, o modelo *ITCPN* não é suficientemente poderoso para representar todos os possíveis tipos de restrições temporais que encontramos nos sistemas em tempo real.

G-Net é um modelo de especificação executável multi-nível baseado em redes de Petri que incorpora os conceitos de módulo e sistema estruturado. Um sistema de *G-Nets* é uma composição de uma especificação de sistemas baseados em *G-Nets*. Uma vez que a estrutura de *G-Nets* utiliza os conceitos do paradigma orientado a objeto [18, 58], a interação entre *G-Nets*, em um sistema de *G-Nets*, é realizada através de interfaces bem definidas como será apresentado mais adiante.

Nesse capítulo, a estrutura de *G-Net* e o conceito de $FTPN$ são integrados com o objetivo de permitir a análise de requisitos temporais de sistemas complexos com restrições em tempo real. Essa integração resulta nas *FTG-Nets*. Logo, considerando um sistema de *G-Nets*, é possível estudar os requisitos temporais de cada *G-Net* isoladamente e depois combinar esses resultados para determinar as propriedades globais do sistema. Para executar a análise temporal, consideraremos um sistema de *G-Nets* decomposto acrescido de restrições temporais. Essas restrições temporais são representadas por intervalos nebulosos de tempo associados às transições.

Na seqüência, informalmente, introduzimos as *G-Nets* bem como uma abordagem de decomposição para as *G-Nets*. Apresentamos ainda sua integração com o modelo $FTPN$ e mostramos um exemplo baseado no problema do produtor/consumidor. No próximo capítulo, discutiremos a análise temporal utilizando esta ferramenta integrada.

4.2 G-Nets e Sistemas de G-Nets

G-Net é uma estrutura para a especificação e prototipagem de sistemas de software complexos. A notação de *G-Net* incorpora as noções de módulo e estrutura de sistemas em redes de Petri [39]. Nessa seção, introduzimos os conceitos de *G-Nets* juntamente com uma abordagem de decomposição.

4.2.1 Conceitos Básicos

Em um sistema de *G-Nets*, uma *G-Net* é vista como um módulo funcional ou um objeto e suas operações associadas, ditas métodos [36, 37, 38]. Um sistema de *G-Nets* é definido pela tripla $GNS = (TS, GS, AS)$ em que, TS é uma coleção de fichas dinamicamente geradas durante a execução do sistema, GS é um conjunto de *G-Nets* e, AS é um conjunto descentralizado de agentes computacionais concorrentes executando uma especificação baseada em *G-Nets*. As *G-Nets* em um sistema de *G-Nets* são fracamente acopladas porque elas não compartilham variáveis e a interação entre elas é efetuada apenas através de interfaces bem definidas. Uma outra característica interessante de *G-Net* é que ela permite mais de uma invocação simultaneamente. Isto pode ocorrer devido à unicidade da estrutura da ficha como mostraremos mais adiante.

De uma forma geral, uma *G-Net* é composta de duas partes: GSP e IS. GSP é um lugar especial chamado *lugar genérico de chaveamento*, que provê uma abstração para a *G-Net* e serve como interface entre a *G-Net* e os outros módulos. IS é a *estrutura interna* de uma *G-Net* que é uma rede de Petri modificada.

Um GSP é unicamente identificado por um nome, $G.GSP.NID$, o qual abstrai um conjunto de métodos, denotados por $G.GSP.MS$. Os métodos definem as formas pelas quais a estrutura interna, $G.IS$, pode ser executada. Em outras palavras, o conjunto de métodos indica todas as possíveis marcações iniciais que podem ser definidas, as quais resultam em diferentes execuções da estrutura interna. Cada método *mtd*, $mtd \in G.GSP.MS$ é definido pela tupla mtd = (m_nome, m_argumentos, m_iniciador) em que, m_nome é um conjunto de nomes para os métodos, m_argumentos é um conjunto de variáveis especificando os parâmetros de entrada para o método e m_iniciador especifica a marcação inicial para $G.IS$ definida pelo método.

A estrutura interna de uma G-Net pode assumir diferentes formatos. O conjunto de lugares P, $P \in G.IS$ é definido por $P = (\mathcal{ISP}, \mathcal{NP}, \mathcal{GP})$ em que, $\mathcal{ISP}$ é um subconjunto de lugares de chaveamento para instanciação (Instantiated Switching Place,) $\mathcal{NP}$ é um conjunto de lugares normais (Normal Places) e $\mathcal{GP}$ é um conjunto de lugares alvo (Goal Places).

Um $isp_i \in \mathcal{ISP}$ é um mecanismo usado em sistemas de *G-Nets* para implementar a conexão entre *G-Nets*. Um *isp* é graficamente representado por uma elipse e é definido pela quádrupla: *isp = (NID, mtd,* ação_antes, ação_após) em que, NID é um identificador único para a rede invocada, *mtd* é o método usado, ação_antes e ação_após são duas ações primitivas, cuja função é atualizar a *seqüência de propagação da ficha.*

Um *lugar normal*, $NP_i \in \mathcal{NP}$, em uma *G-Net* contém as regras para a manipulação

de fichas. Um *lugar normal* é representado no grafo por um círculo e o fluxo das fichas na rede modela o comportamento dinâmico do sistema. Quando uma ficha é depositada em um NP, a ação a ela associada, cujos parâmetros estão definidos no campo msg, é executada. O resultado da ação é associado ao campo msg da ficha e a *cor de desvio da ficha* é definida, de acordo com este resultado. O campo *scolor* da ficha é atualizado para indicar a disponibilidade da ficha (detalhes sobre a estrutura da ficha serão discutidos adiante). Ainda, um valor $K(P)$ chamado de capacidade pode ser associado a cada lugar normal. $K(P)$ define o número máximo de fichas que podem ocupar um lugar no mesmo instante de tempo.

Um lugar alvo, $GP_i \in \mathcal{GP}$, pertence à marcação final de $G.IS$. Para todo método, mtd, definido para uma *G-Net*, lugares alvos devem ser alcançáveis de modo a poder retornar a informação para a rede que a invocou. Um *lugar alvo* é graficamente representado por círculos duplos.

De forma similar ao modelo de redes de Petri, o fluxo das fichas em uma *G-Net* representa o comportamento dinâmico da *G-Net*. Uma ficha em uma *G-Net* é uma tupla de *ítens* da forma $\langle seq, scor, d_1, \cdots, d_q \rangle$, em que seq é a *seqüência de propagação da ficha*, $scor \in$ (antes, após) é chamada *cor de estado da ficha*, e d_i, $i \in \mathbb{N}$, é a mensagem carregada pela ficha. seq tem o seguinte formato $\langle NID, isp, PID \rangle$, em que NID é a identificação da *G-Net*, isp é o nome de um ISP e PID é uma identificação de processo. A ficha só está disponível quando $scor$={após}. Quando uma nova ficha é recebida em um lugar, sua *cor de estado* é fixada para ser antes. Após a execução das ações primitivas definidas para o lugar, a *cor do estado* da ficha é atualizado para depois, indicando que a ficha está pronta para ser usada em subseqüentes disparos de transições. A estrutura da ficha no modelo de *G-Net* é muito importante. Por exemplo, as fichas que pertencem a uma mesma instância de execução de um *G-Net* têm uma seqüência de propagação única. A informação da seqüência de propagação determina qual *G-Net* foi invocada e qual *G-Net* que invocou.

Para facilitar a compreensão dos conceitos de *G-Net*, apresentaremos dois exemplos em que os conceitos de *G-Nets* são discutidos. A notação de *G-Net* permite a especificação de configurações recursivas e hierárquicas. A Figura 4.1 mostra um sistema de *G-Nets* composto por duas *G-Nets*, $G1$ e $G2$. Nesse exemplo, as duas *G-Nets* são conectadas pelo *isp*. A Figura 4.2 mostra uma *G-Net*, especificando a função recursiva $f(n) = 2^{f(n-1)}$; $f(0) = 1$, em que $n \in \mathbb{N}$. A *G-Net* tem um método, m_1, com entrada n e marcação inicial de uma ficha em *P1*. Se $n > 0$, a transição *t1* se sensibiliza e dispara. A primitiva de *P2* decrementa de 1 o valor de n. A primitiva de *P3* é um *isp*

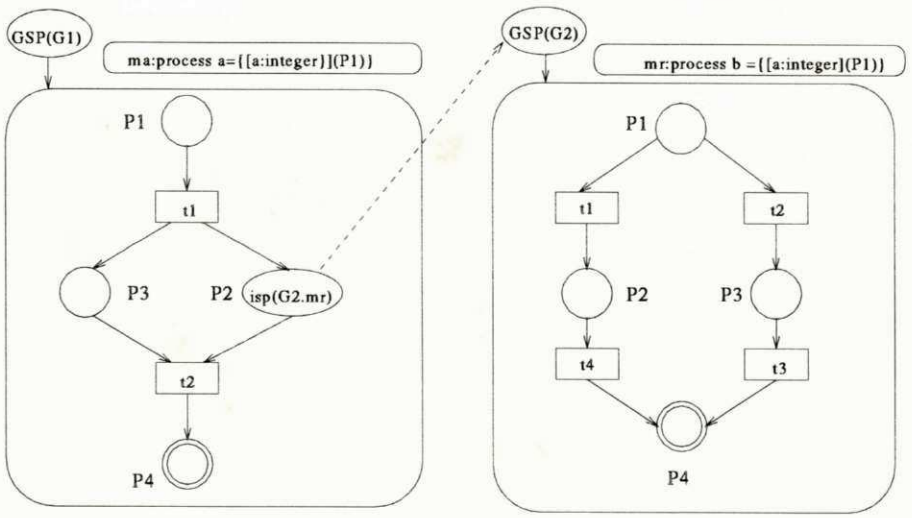

Figura 4.1: Comunicação entre *G-Nets*

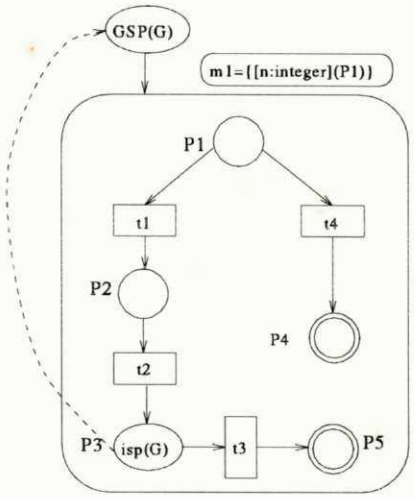

Figura 4.2: Especificação de uma função recursiva

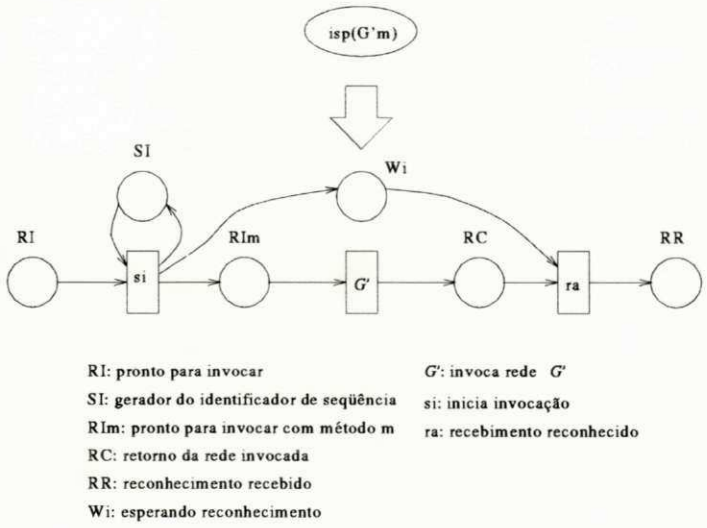

Figura 4.3: Decomposição de um $isp(G'm)$

que recursivamente invoca a *G-Net* para computar $f(n-1)$. A primitiva de *P5* calcula $f(n-1) \times 2$. Se a entrada $n = 0$, a transição *t4* dispara, e a primitiva de *P4* retorna 1 como resultado.

Para maiores detalhes sobre *G-Nets* e sistemas de *G-Net* o leitor pode se referir a [36, 37, 38, 39].

4.2.2 Decomposição de G-Nets

Em [89], uma decomposição para *G-Nets* é apresentada com o objetivo de definir uma metodologia de análise modular para sistemas de *G-nets*. Uma vez que a análise modular é baseada na análise dos módulos que interagem, a decomposição é concentrada nos elementos de interface das *G-Nets*, i.e., *isp* e *GSP*.

Um $isp(G'm)$, representando a invocação da rede G' utilizando o método m é decomposto como mostrado na Figura 4.3, e descrito da seguinte forma:

O lugar *RI* recebe a ficha a partir do disparo da transição de entrada, que não está respresentada na Figura 4.3. Então, a transição *si* dispara e o campo *seq* da ficha é atualizado baseado no valor inteiro n, em que $n \in \mathbb{N}$, da ficha no lugar *SI*, a qual é incrementada de 1 toda vez que *si* dispara. A ficha de saída será depositada em *RIm*. Esta ficha terá um identificador de seqüência único, juntamente com um conjunto de fichas definindo a marcação inicial da rede *G'*. Além do mais, uma ficha será depositada no lugar *Wi*. A função deste lugar é manter informação sobre o estado da rede que pode ser usada para tolerância a falhas e recuperação de erros, como apresentado em [90]. A

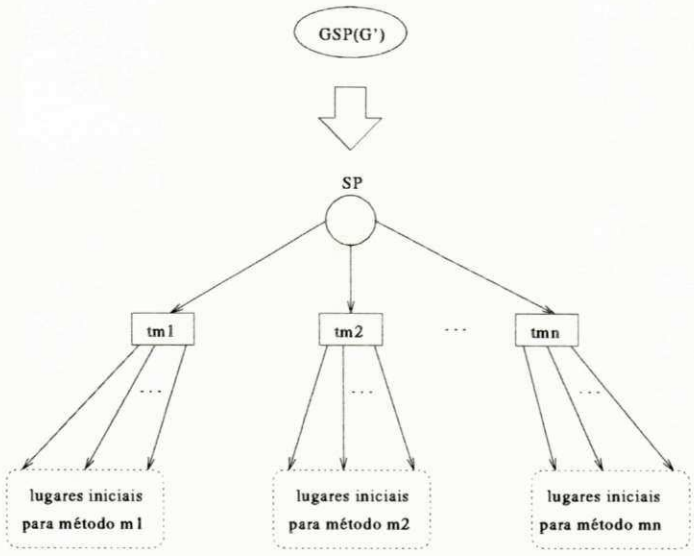

Figura 4.4: Decomposição de um $GSP(G')$

transição G' pode disparar e seu disparo corresponde à invocação da rede G'. Quando a rede G' atinge um lugar alvo, as fichas que retornam são depositadas no lugar RC e a transição ra dispara resultando em uma ficha no lugar RR.

No caso do $GSP(G')$, a decomposição é mostrada na Figura 4.4. O lugar inicial SP recebe todas as fichas correspondentes à marcação inicial relativa ao método m. A transição que corresponde ao método especificado dispara e as fichas são então depositadas no conjunto de lugares iniciais, representados na Figura 4.4 pelas linhas pontilhadas.

Maiores detalhes sobre a decomposição e análise de *G-Nets* podem ser encontrados em [89, 91].

4.3 Aspectos da Integração

Nesta seção, descrevemos a integração entre *G-Net* e *FTPN*. O objetivo dessa integração é permitir a análise temporal modular para sistemas complexos. Cada *G-Net*, em um sistema de *G-Nets*, é estudada isoladamente e os resultados são combinados para determinar a solução final. Uma vez que o conceito original de *G-Net* não leva em consideração os aspectos temporais e, devido à flexibilidade de sua estrutura interna, consideraremos a estrutura interna como sendo uma *FTPN*. Para tanto, a integração é efetuada considerando-se a *G-Net* decomposta. Na análise modular, o ponto mais importante é saber, como combinar de forma correta, os resultados obtidos nas análises individuais de cada *G-Net*. Esta combinação é determinada através da maneira pela

qual elas estão conectadas. No caso de sistemas de *G-Nets*, isso é feito pelos *isp* e *GSP* como mostrado na última seção. Na seqüência, apresentamos os detalhes sobre a integração.

Quando uma *G-Net* invoca uma outra *G-Net* com um determinado método m, esse método define a marcação inicial da rede invocada. Uma vez que a estrutura interna é uma *FTPN*, a ficha que chega em um *isp* é uma ficha nebulosa; então, a marcação inicial da rede invocada é composta por fichas nebulosas. Devido às propriedades de comutatividade e distributividade da teoria dos conjuntos nebulosos é possível estudar a *G-Net* invocada, considerando que as fichas da marcação inicial da rede invocada não são nebulosas, para depois combinar com as funções nebulosas associadas às fichas. Neste caso, o intervalo nebuloso que corresponde ao tempo de execução mínimo e máximo da rede invocada é computado, combinando os intervalos de sensibilização e disparo associados às transições, como definido no Capítulo 3. A computação deste intervalo nebuloso termina quando um *lugar alvo* é alcançado. Finalmente, esse intervalo nebuloso é combinado com as funções nebulosas de tempo carregadas pela ficha do *isp*. A análise da rede invocada depende do método utilizado na invocação. Entretanto, como veremos no próximo capítulo, para garantir resultados corretos, temos que fazer algumas suposições como por exemplo, que o conjunto de transições de diferentes métodos seja livre de conflito[1].

Para definir a *FTG-Net*, a decomposição do *isp* e *GSP* (como apresentada na Seção 4.2.2) é integrada com os intervalos nebulosos de tempo associados com as transições. Algumas considerações devem ser feitas para permitir a integração e, conseqüentemente, a análise de desempenho. Essas considerações estão relacionadas com alguns lugares e transições especiais que foram definidos no modelo composto para garantir algumas propriedades da definição original de *G-nets*.

Do ponto de vista da análise temporal, o que é importante conhecer é a ficha nebulosa no *isp* e a análise da rede invocada. Logo, a integração do *isp* decomposto com os intervalos de tempo é considerada da seguinte forma:

1. Os lugares *SI* e *Wi* no *isp* decomposto não recebem fichas nebulosas;

2. Os intervalos associados às transições *si* e *ra* são ambos $[0, 0]$, i.e., o disparo destas transições é instantâneo;

[1] O conjunto de transições de dois métodos diferentes é livre de conflito se o disparo das transições em um método não depende do disparo de qualquer transição no outro método

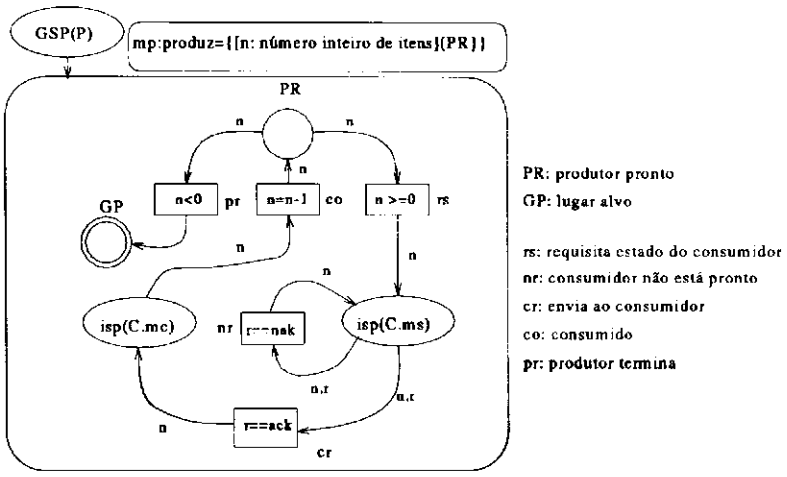

Figura 4.5: *G-Net* modelando o produtor

3. O intervalo de tempo associado à transição G' corresponde aos tempos de execução mínimo e máximo da rede invocada;

4. O lugar *RIm* mantém a função nebulosa de tempo da ficha do *isp*;

5. A ficha depositada no lugar *RR* após o disparo de *ra* é uma ficha nebulosa que carrega a função nebulosa de tempo resultante, considerando a rede invocada.

Na realidade, para fins de análise temporal, o *isp* decomposto integrado com os intervalos nebulosos de tempo pode ser simplificado. Isso ocorre porque assumimos as 5 considerações enumeradas acima. Retornaremos a esse ponto no Capítulo 5 quando discutiremos a análise temporal nebulosa.

Para o *GSP* decomposto, nenhuma consideração especial é definida. A transição que é conectada ao lugar inicial *SP* pode ser usada no sentido de representar o atraso de comunicação entre as *G-Nets* comunicantes. Neste caso, associa-se às transições *tm1*, *tm2*, ..., e *tmn* um intervalo de atraso nebuloso para representar o atraso na comunicação. É óbvio que os intervalos de atraso nebulosos são idênticos uma vez que essas transições representam uma mesma comunicação e são definidas apenas para determinar que método será executado.

4.4 O Exemplo do Produtor/Consumidor

Para ilustrar a integração, introduzimos um exemplo baseado no problema do produtor/consumidor. Na Figura 4.5, o modelo *G-Net*, $G(P)$, para o produtor é apresentado.

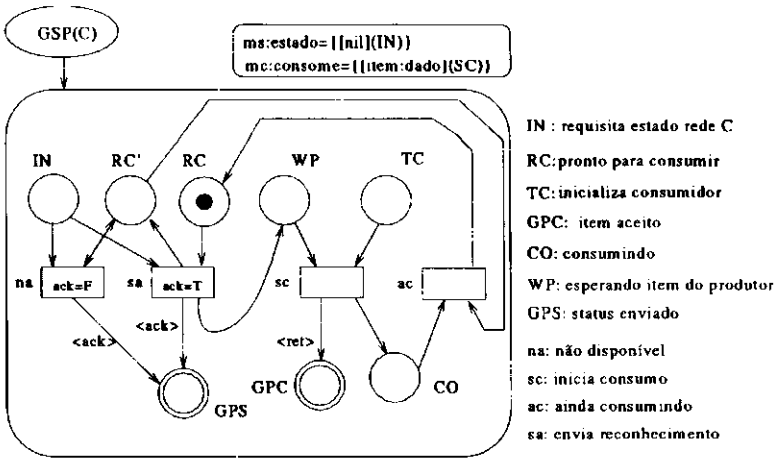

Figura 4.6: *G-Net* modelando o consumidor

Para esta rede, define-se um método (*mp*) que é o de produzir n ítens para serem consumidos. Quando $G(P)$ é invocada através de *GSP(P)*, uma ficha juntamente com um campo n, indicando o número de ítens a serem produzidos é depositado no lugar *PR*. Neste lugar, o valor de n é decrementado de 1. Se $n < 0$, a transição *pr* dispara e a invocação de $G(P)$ termina, uma vez que o *lugar alvo GP* é alcançado. Quando $n \geq 0$, a transição *rs* dispara, uma ficha é depositada no lugar *isp(C.ms)*, e a rede C (consumidor) é invocada utizando o método *ms*. Com a invocação, o estado atual do consumidor é verificado que pode ser consumindo ou pronto para consumir um ítem. Caso o consumidor não esteja pronto para consumir, a transição *nr* dispara, e o consumidor será novamente inquerido a respeito de seu estado atual. Se o consumidor está pronto para consumir, a transição *cr* dispara. Depois do disparo de *cr*, uma ficha é depositada no lugar *isp(C.mc)*. A rede C é então invocada, utilizando o método *mc* juntamente com o ítem a ser consumido. Uma vez que a ficha é retornada de C, a transição *co* dispara e uma ficha é novamente depositada no lugar *PR*.

A *G-Net* para o consumidor, $G(C)$, é apresentada na Figura 4.6. Dois métodos são definidos para $G(C)$: *método estado* (*ms*) e *método consome* (*mc*). Quando $G(C)$ é invocada com o método *ms*, uma ficha é depositada no lugar *IN*. Dependendo do estado de $G(C)$, o qual pode ser tanto consumindo como pronto para consumir, as transições *na* ou *sa* poderão disparar. A escolha entre *na* ou *sa* é baseada nos estados representados pelos lugares *RC* (pronto para consumir) e *CO* (consumindo). Se *RC* está marcado, então a transição *sa* poderá disparar e uma ficha com o campo *reconhecido, ack*, associado a ela será retornada após o *lugar alvo GP* ser atingido. De outra forma, um campo *não reconhecido, nak*, associado à ficha será retornado. Depois do disparo de

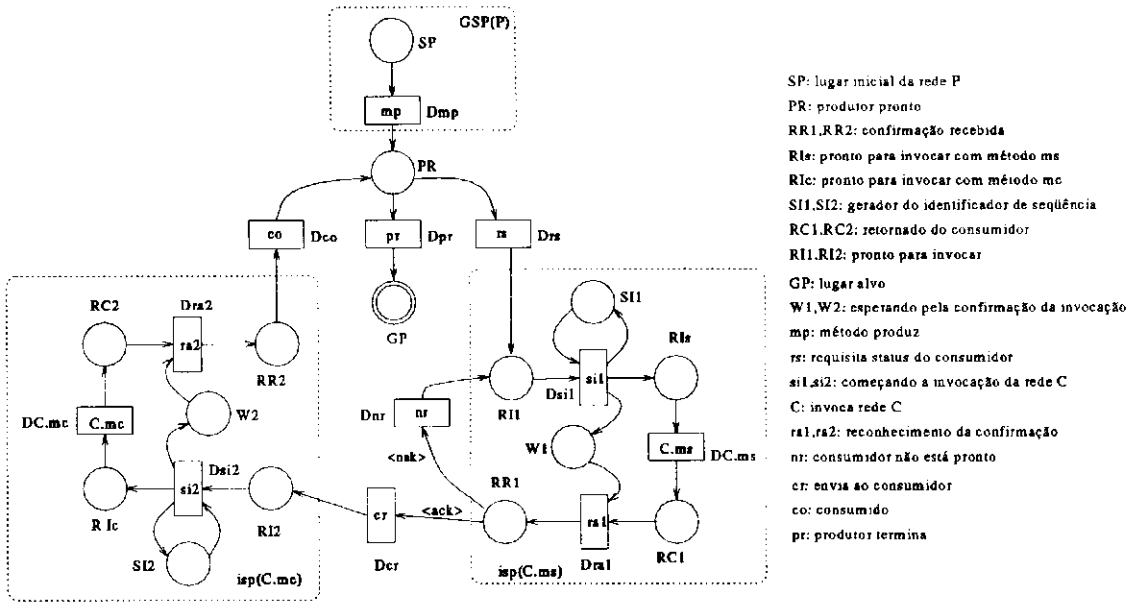

Figura 4.7: *G-Net* decomposta, $Gd(P)$, para o produtor incluindo tempo

sa, uma ficha é colocada nos lugares *WP* e *RC'* indicando que o consumidor está pronto para consumir. Quando $G(C)$ é invocada com o método *mc*, uma ficha é depositada no lugar *TC* (assume-se que o produtor só invoca $G(C)$ com método *mc* após invocá-la com método *ms*). Uma vez que *WP* foi previamente marcado, após uma execução com sucesso do método *mc*, a transição *sc* pode disparar e o lugar *CO* será marcado. Isto ocorre pois a transição *sc* dispara e marca o lugar *CO* e o *lugar alvo GP*. Mas deve-se notar que a transição *ac* pode ou não disparar, antes de o produtor receber a ficha de retorno através de *GP*. É importante enfatizar que este tipo de situação foi criada de modo que o produtor pudesse ser executado concorrentemente com o consumidor. Se $G(C)$ terminasse sua execução somente após o disparo de *ac*, não haveria razão para o método status, uma vez que $G(P)$ nunca executaria dois acessos seguidos a $G(C)$. Após o disparo de *ac*, uma ficha é removida de *CO* e *RC'*, depositada em *RC* e $G(C)$ está pronta para consumir outro ítem.

O sistema de *FTG-Nets*, representando o problema do produtor/consumidor é composto por uma *FTG-Net*, modelando o produtor e uma *FTG-Net* modelando o consumidor. Essas *FTG-Nets* são obtidas a partir da decomposição das *G-Nets* originais e pela introdução dos intervalos nebulosos de tempo. As *G-Nets* para o consumidor e para o produtor são decompostas como mostrado nas Figuras 4.7 e 4.8, respectivamente. Para melhorar a visualização das *FTG-Nets*, vamos omitir os rótulos nos arcos e inscrições nas transições. Vamos considerar que $Gd(P)$ e $Gd(C)$ são as *G-Nets* decompostas re-

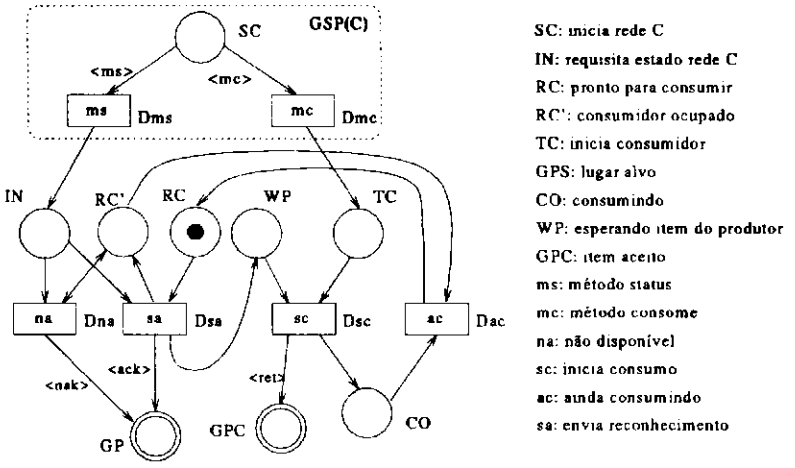

Figura 4.8: *G-Net* decomposta, $Gd(C)$, para o consumidor incluindo tempo

sultantes para o produtor e para o consumidor, respectivamente. Ainda, os intervalos nebulosos de tempo são associados com cada transição da *G-Net* decomposta. Por uma questão de simplicidade, os intervalos nebulosos de sensibilização não são apresentados na Figura 4.7 4.8. Logo, na Figura 4.7, *DC.ms* é o intervalo de disparo associado à transição *C.ms*, *Dra1* é o intervalo de disparo para a transição *ra1*, e assim por diante.

Quando a *G-Net* decomposta *Gd(C)* é invocada usando um método ms, dois estados são possíveis: *Gd(C)* ou está consumindo (ficha em lugar *CO* e *RC'*) ou está pronta para consumir (ficha no lugar *RC*). Considerando o estado pronto para consumir, a *G-Net* decomposta *Gd(C)* é invocada duas vezes: primeiro com o método ms e depois, com o método mc. Aplicando as regras de computação nebulosa descritas no último capítulo, as seguintes *FTFs* são determinadas para a *G-Net*, representando o consumidor, considerando o método ms:

$$FIN = FSC \oslash Dms. \tag{4.1}$$

$$FWP = FGP = (FIN \diamond FRC) \oslash Dsa. \tag{4.2}$$

em que, *FIN*, *FSC*, *FWP*, *FGP* e *FRC* são as funções nebulosas de tempo associadas às fichas nos lugares *IN*, *SC*, *WP*, *GP* e *RC*, respectivamente.

Da mesma forma, considerando a invocação de $Gd(C)$ utilizando o método mc, as seguintes equações de funções nebulosas de tempo são determinadas:

$$FTC = FSC \oslash Dmc. \tag{4.3}$$

$$FCO = FGPC = (FWP \diamond FTC) \oslash Dsc. \tag{4.4}$$

Neste exemplo, para efetuar de forma modular a análise temporal, temos que analisar em separado a *G-Net* representando o consumidor, considerando os métodos *ms* e *mc*. Então, os resultados são embutidos no produtor que é analisado para se chegar a uma resultado final. No próximo capítulo, apresentamos uma técnica para computar as FTFs baseada em um grafo de alcançabilidade modificado. Essa técnica é então aplicada ao exemplo do produtor/consumidor.

Capítulo 5

ANÁLISE TEMPORAL

5.1 Introdução

De uma forma geral, a área de análise de desempenho pode ser dividida em duas categorias distintas: medida e modelagem [3]. A medida é muito útil quando estudando o desempenho de sistemas reais, enquanto que a modelagem é importante para estudar o desempenho quando do projeto do sistema. Essa categoria compreende três diferentes campos. O primeiro campo, dito medição, apresenta como resultado, o desempenho de sistemas sob condições específicas, i.e., as medições são executadas sobre uma operação real do sistema. A principal limitação deste tipo de estudo de desempenho é a falta de generalidade, uma vez que elas são determinadas para um dado conjunto de condições. Logo, é praticamente impossível comparar dois sistemas, utilizando medição porque é muito difícil garantir que as medições de dois sistemas são efetuadas sob as mesmas condições. O segundo campo, chamado de benchmarks [45, 59, 83], foi desenvolvido com o objetivo de propiciar uma maneira de comparar dois sistemas de uma forma justa. O princípio básico de benchmarks é prover uma sobrecarga artificial para o sistema no sentido de permitir a comparação entre sistemas, garantindo condições de operação equivalentes. O terceiro campo é chamado prototipagem. A prototipagem é uma aproximação do sistema real que pode ser usada para estudar o desempenho de sistemas quando eles não estão disponíveis para estudo.

As técnicas descritas acima são boas para serem aplicadas quando o sistema está completamente disponível ou quando temos um protótipo do sistema. No entanto, em muitos casos, é muito importante estudar o desempenho de sistemas durante os primeiros estágios do projeto. Utilizando os métodos acima, não é possível estudar o desempenho durante as fases iniciais do projeto. Isto é efetuado por alguns métodos na categoria de modelagem. Nessa categoria, o desempenho pode ser estudado utilizando modelos de simulação ou modelos analíticos.

De um lado, a simulação não requer conhecimento matemático e consiste em analisar o sistema conduzindo-se alguns experimentos. No modelo de simulação, a descrição é dada por meio de um programa de computador. Uma vez que a simulação não de-

pende de técnicas matemáticas, ela é considerada uma poderosa ferramenta de análise, especialmente, porque seus resultados podem ser facilmente entendidos por pessoas sem conhecimento técnico. Entretanto, em alguns casos, a simulação é cara em termos de tempo de computador necessário para obter resultados confiáveis [122], i.e., os modelos de simulação são geralmente muito detalhados e por conseguinte o tempo para desenvolvê-los pode ser considerável [53].

Por outro lado, os modelos analíticos são caracterizados pela utilização de ferramentas ou modelos formais. Entre eles, podemos incluir modelos de fila e redes de Petri. Em nosso caso, estamos utilizando as redes de Petri como modelo formal. Redes de Petri têm sido extensivamente utilizadas para modelar e analisar sistemas considerando aspectos temporais e não-temporais. Existem diferentes técnicas para efetuar análise temporal no modelo de redes de Petri. Dependendo da técnica aplicada, diferentes índices de desempenho podem ser computados. Focando nos aspectos temporais dos sistemas, as duas técnicas de análise mais comuns são: análise de alcançabilidade e análise Markoviana.

A técnica de análise de alcançabilidade é baseada na construção do grafo de alcançabilidade ou árvore de alcançabilidade. Como discutido no Capítulo 2, o grafo de alcançabilidade representa todos os possíveis estados ou trilhas de execução de um sistema. A técnica de análise de alcançabilidade é amplamente utilizada no caso das extensões determinísticas de redes de Petri [13, 98, 111, 123, 128]. Então, a partir do grafo de alcançabilidade, diversos índices de tempo podem ser computados, como por exemplo, o atraso para alcançar um determinado estado. A maior limitação desta técnica é o bem conhecido problema da explosão de estados, i.e., mesmo para pequenas redes o grafo de alcançabilidade pode ser grande.

A técnica de análise Markoviana é empregada em certos tipos de redes de Petri chamadas de redes de Petri estocásticas. A rede de Petri estocástica é transformada em uma cadeia de Markov de tempo contínuo e, aplicando-se as conhecidas técnicas Markovianas, é possível computar alguns índices de desempenho (steady-state), como por exemplo, a probabilidade de um determinado estado. Na verdade, na análise de uma rede de Petri estocástica, algumas técnicas de alcançabilidade são também usadas porque o conjunto de alcançabilidade dos estados é determinado para derivar a correspondente cadeia de Markov [81]. Embora o uso das técnicas de Markov possa facilitar a computação de alguns índices de desempenho, o problema da explosão de estados ainda permanece quando tratamos de sistemas complexos.

Neste capítulo, apresentamos uma abordagem diferente para efetuar análise temporal

baseada na técnica de análise de alcançabilidade. Basicamente, a abordagem consiste na construção de um grafo de alcançabilidade modificado, considerando o modelo de *FTPN*. Este grafo de alcançabilidade modificado é chamado *Grafo de Alcançabilidade Nebuloso (FRG)*. Uma vez que o *FRG* é baseado na técnica de análise de alcançabilidade, no caso de sistemas complexos, o problema da explosão de estados persiste. Para superar este problema ou pelo menos gerenciá-lo, esta abordagem de análise temporal é estendida para o modelo de *FTG-Net* [31, 32, 33]. Além do mais, um algoritmo de análise temporal baseado no *FRG* é descrito bem como sua análise de complexidade.

Também introduzimos uma representação gráfica para a visualização do comportamento temporal de um sistema de *G-Net*. Este diagrama de tempo chamado *Gráfico de Interação entre G-Nets com Temporização (TGIG)* mostra os aspectos temporais e de interação entre *G-Nets*. O *TGIG* pode também ser usado para determinar alguns índices de desempenho. Tanto o *FRG* como o *TGIG* são exemplificados através do exemplo do produtor/consumidor estendido com valores numéricos.

5.2 Grafo de Alcançabilidade Nebuloso

Uma das mais conhecidas técnicas para analisar redes de Petri é a técnica da árvore de alcançabilidade que é uma ferramenta muito útil para muitas propriedades das redes de Petri. Entretanto, sua aplicação é, em geral, limitada pelo fato de que, na maioria dos casos, a árvore de alcançabilidade de uma rede de Petri é uma árvore infinita para uma dada marcação inicial. Uma maneira muito simples de reduzir a árvore de alcançabilidade é construir um grafo tal que seus estados correspondem a um único nodo no grafo. O grafo de alcançabilidade é um grafo dirigido que enumera todos os estados alcançáveis do sistema, e descreve todas as possíveis transições entre estados [98]. Infelizmente, o problema de explosão de estados continua presente, especialmente quando existem mais de uma ficha em um lugar. Em nosso trabalho, consideramos que as redes são seguras a nível 1 (*1-safe*), i.e., redes que não suportam mais de uma ficha em um lugar.

A técnica de análise baseada no grafo de alcançabilidade foi muito utilizada para provar diferentes tipos de propriedades. No caso das redes de Petri, que não consideram aspectos temporais, propriedades relacionadas com a corretude (*correctness*) podem ser facilmente provadas [85, 93]. Propriedades temporais e avaliação de desempenho de sistemas podem também ser efetuadas pela análise de um grafo de alcançabilidade modificado que é uma extensão do grafo de alcançabilidade regular incorporando-se o aspecto tempo. Existem muitos trabalhos que usam a análise de alcançabilidade para

derivar propriedades temporais. Em Zuberek [128], o *Grafo de Alcançabilidade Temporizada (Timed Reachability Graph - TRG)* é definido pela adição de um componente de tempo para cada estado. Mais precisamente, cada estado é composto por uma marcação e uma lista de tempos de disparos remanescentes das transições os quais são relacionados com as transições em processo de disparo. Razouk [98] adotou o conceito de *TRG* para um tipo diferente de rede de Petri temporal com o objetivo de efetuar análise temporal. Berthomieu e Menasche [13] definiram um grafo de alcançabilidade diferente em que os nodos são classes de estados, representando o conjunto de todos os estados alcançáveis a partir de uma marcação inicial. Uma idéia semelhante foi adotada em [111]. Aalst [123] definiu uma abordagem baseada na análise através do grafo de alcançabilidade chamada de *Modified Transition System Reduction Technique (MTSRT)*. O método *MTSRT* é aplicada às *ITCPNs*.

Na seqüência, introduzimos um grafo de alcançabilidade modificado chamado *Grafo de Alcançabilidade Nebuloso* que será utilizado na análise temporal das *FTPNs*.

5.2.1 Definição e Construção do Grafo de Alcançabilidade Nebuloso

O grafo de alcançabilidade nebuloso é um grafo de alcançabilidade modificado que incorpora os conceitos temporais definidos no modelo *FTPN*. A idéia básica é ampliar o conceito de grafo de alcançabilidade para incluir as ($FTFs$) carregadas pelas fichas, que são computadas depois de cada disparo de transição, começando da marcação inicial. O grafo de alcançabilidade modificado ainda considera os intervalos nebulosos de tempo associados com as transições. Então, o grafo nebuloso de alcançabilidade é caracterizado por:

1) Assinalar uma FTF para cada ficha nos lugares em um dado estado. Isso segue a definição do modelo *FTPN* em que as fichas carregam funções nebulosas características.

2) Associando a cada arco do grafo os intervalos de tempo associado com as correspondentes transições disparadas, representando as restrições de tempo quando do disparo de uma transição no modelo *FTPN*.

Antes de introduzirmos, formalmente, o conceito do grafo nebuloso de alcançabilidade, introduzimos o conceito de conjunto de função de tempo nebuloso de um estado ($FSET$). De acordo com a Definição 3.10, um estado em uma $FTPN$ é definido como sua marcação $S = \{m(p1), m(p2), ..., m(pn)\}$.

1. Dado um estado S
2. Determinar o conjunto, FT, das transições que podem disparar
3. *For all* transições $t \in FT$ *do*
4. Gerar um estado sucessor S' a partir de S depois de disparar t
5. Associar ao arco que conect e a S' os intervalos de tempo correspondentes
6. Associar aos nodos o correspondente $FSET$
7. *enddo*

Figura 5.1: Procedimento para computar os sucessores de um estado

Definição 5.1 *Um conjunto de função nebulosa de tempo, $FSET$, é o conjunto de $FTFs$ para um dado estado, i.e., $FSET(S) = \{FTF(p1), FTF(p2), ..., FTF(pn)\}$.*

Definição 5.2 *O grafo de alcançabilidade nebuloso, FRG, é definido como um grafo rotulado $FRG = (N, A)$. N é chamado conjunto dos nodos, ou seja, é o conjunto de todos os possíveis estados ou marcações. Um $FSET$ relacionado ao estado representado pelo nodo é a ele associado. A é dito conjunto de arcos que é o conjunto dos arcos rotulados com os intervalos de tempo nebulosos representando todos os possíveis disparos de transições.*

Definição 5.3 *Um dado estado S é definido como um estado imediatamente alcançável (irs) a partir de um estado inicial $S0$, se S é alcançável de $S0$ após o disparo de uma única transição.*

Definição 5.4 *Um conjunto de estados imediatamente alcançáveis para um dado estado S, $IRS(S)$, é o conjunto de todos os estados imediatamente alcançáveis de S.*

O grafo de alcançabilidade nebuloso é construído recursivamente, calculando os sucessores de todos estados alcançados, começando do estado inicial. O procedimento para a computação dos sucessores de um estado é mostrado na Figura 5.1.

Para ilustrar a construção do grafo de alcançabilidade nebuloso vamos considerar a $FTPN$ introduzida no Capítulo 3. Para uma melhor leitura, o exemplo é mostrado novamente na Figura 5.2(a). Assumindo uma marcação inicial de uma ficha nebulosa no lugar $P1$ com função nebulosa de tempo $FTF(P_1)$, existem seis possíveis estados alcançáveis para esta rede como mostrado na Tabela 5.1. O grafo de alcançabilidade nebuloso para a rede da Figura 5.2(a) é construído de acordo com o procedimento apresentado acima e é mostrado na Figura 5.2(b). De acordo com a sua definição, para cada nodo (representando estado) é associado o seu correspondente $FSET$. Logo, para o

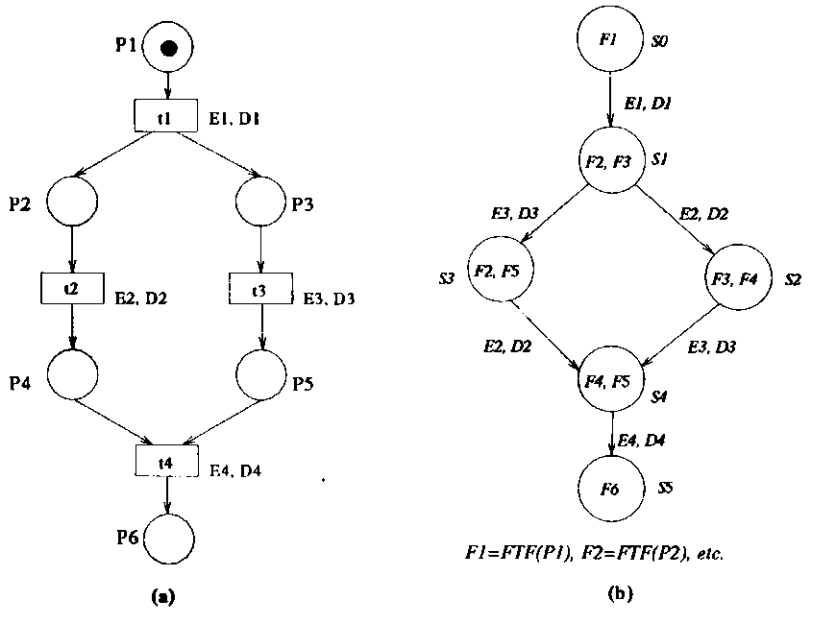

Figura 5.2: (a) Rede simples e, (b) seu grafo de alcançabilidade

nodo que representa o estado *S0* que é composto por uma ficha no lugar *P1*, é associado o conjunto de função nebulosa de tempo $\{F1\}$[1]. Semelhantemente, $\{F2, F3\}$ é associado com o estado *S1*, $\{F3, F4\}$ com o estado *S2*, $\{F2, F5\}$ com o estado *S3*, $\{F4, F5\}$ com o estado *S4* e $\{F6\}$ com o estado *S5*. Além do mais, os intervalos nebulosos de tempo são assinalados com os arcos. Por exemplo, considerando o estado *S0*, apenas uma transição pode disparar (transição *t1*) e seu disparo leva a alcançar o estado *S1*. Isso é observado no grafo de alcançabilidade pelo arco que conecta os estados *S0* e o estado *S1*, onde *E1* e *D1* são os intervalos de tempo nebulosos associados à transição *t1*. Os intervalos nebulosos de tempo *E2* e *D2* são associados ao arco conectando *S1* a *s2* e *S3* a *S4*, os intervalos nebulosos de tempo *E3* e *D3* são associados aos arcos conectando *S1* a *S3* e *S2* a *S4*. Finalmente, os intervalos nebulosos de tempo *E4* e *D4* são associados com o arco conectando *S4* a *S5*.

5.3 Aspectos da Análise Temporal

A análise temporal é baseada nas funções nebulosas de tempo associadas com as fichas. A idéia é usar o grafo de alcançabilidade nebuloso onde temos o completo espaço de estados, no sentido de computar as FTFs. Assumimos que as FTFs associadas com

[1] Por uma questão de simplicidade nós representamos $FTF(P1)$ por $F1$, $FTF(P2)$ por $F2$ e assim por diante.

	LUGARES					
ESTADOS	P1	P2	P3	P4	P5	P6
S0	1	0	0	0	0	0
S1	0	1	1	0	0	0
S2	0	0	1	1	0	0
S3	0	1	0	0	1	0
S4	0	0	0	1	1	0
S5	0	0	0	0	0	1

Tabela 5.1: Possíveis estados para a rede mostrada na Figura 5.2(a)

as fichas inicias são conhecidas[2] e definidas no grafo de alcançabilidade nebuloso pelo $FSET$ atribuido à raiz (estado inicial). Logo, a partir do grafo de alcançabilidade nebuloso e das funções nebulosas de tempo iniciais, podemos computar as FTFs nos demais estados do grafo de alcançabilidade nebuloso. De uma maneira informal, para computar essas funções, é necessário identificar as diferenças entre as $FSET$s dos estados imediatamente alcançáveis. Considerando-se dois estados conectados no FRG e os seus respectivos $FSET$s, temos que considerar dois casos durante a análise:

1. elementos comuns entre os $FSET$s.

2. elementos diferentes entre os $FSET$s.

Focalizando no comportamento dinâmico de uma $FTPN$, o ítem 1 indica que as fichas não foram removidas após o disparo de uma transição. O ítem 2 pode representar dois aspectos: as fichas removidas dos lugares de entrada e as fichas depositadas nos lugares de saída. De acordo com as regras de computação da função nebulosa apresentadas no Capítulo 3, as fichas de entrada são combinadas para determinar a FTF das fichas de saída. Na seqüência, nós apresentamos as regras de análise temporal que são baseadas na análise do FRG.

Sejam S e S' dois estados no FRG com respectivos conjuntos de função nebulosa de tempo $FSET(S)$ e $FSET(S')$, em que $S' \in IRS(S)$. Suponha que n é o número de lugares da $FTPN$ e a é o arco conectando S a S'.

Definição 5.5 *O conjunto das FTFs comuns, SCF, é o conjunto de todos os elementos* $FTF(p_i)$, *em que* $FTF(p_i) \in FSET(S), FSET(S')$ *e* $1 \leq i \leq n$.

Definição 5.6 *O conjunto das FTFs geradas, SGF, é o conjunto de todos os elementos* $FTF(p_j)$, *em que* $FTF(p_j) \in FSET(S')$, $FTF(p_j) \notin FSET(S)$ *e* $1 \leq j \leq n$.

[2] A FTF inicial pode ser ajustada para computar específicos índices ou pode representar diferentes tipos de situações.

Definição 5.7 *O conjunto das FTFs influentes, SIF, é o conjunto de todos os elementos* $FTF(p_k)$, *em que* $FTF(p_k) \in FSET(S)$, $FTF(p_k) \notin FSET(S')$ *e* $1 \leq k \leq n$.

A análise temporal é executada baseada nas seguintes regras:

R1: Os valores das FTFs para os elementos de SCF não são modificados.

R2: Combinar os valores das FTFs dos elementos em SIF, utilizando a operação $\diamond$ definida no Capítulo 3.

R3: O valor obtido na regra **R2** é combinado com os intervalos nebulosos de tempo associados ao arco a, utilizando a operação $\oslash$ definida no Capítulo 3. O resultado é o valor da FTF para todos os elementos de SGF.

No FRG mostrado na Figura 5.2(b), o estado inicial *s0* é composto por uma ficha no lugar *P1*. Assumindo que a FTF inicial, $F1$, é conhecida e seguindo as regras apresentadas acima, podemos computar as demais FTFs. As FTFS restantes são:

a) Do estado S0 para S1:

Nesse caso, é fácil ver que $SGF = \{F2, F3\}$, $SIF = \{F1\}$ e $E1, D1$ são os intervalos nebulosos associados ao arco, conectando *S0* a *S1*. Então, temos:

$$F_2 = F_3 = F_1 \oslash E_1 \oslash D_1. \tag{5.1}$$

b) Do estado S1 para S2:

Nesse caso, $SGF = \{F4\}$, $SIF = \{F2\}$, $SCF = \{F3\}$ e $E2, D2$ são os intervalos nebulosos associados ao arco que conecta *S1* a *S2*. Então, temos:

$$F_4 = F_2 \oslash E_2 \oslash D_2. \tag{5.2}$$

Computação semelhante é executada para as demais mudanças de estado.

c) Do estado S1 para S3:

$$F_5 = F_3 \oslash E_3 \oslash D_3. \tag{5.3}$$

d) Do estado S2 para S4:

$$F_5 = F_3 \oslash E_3 \oslash D_3. \tag{5.4}$$

e) Do estado S3 para S4:

$$F_4 = F_2 \oslash E_2 \oslash D_2. \tag{5.5}$$

f) Do estado S4 para S5:

$$F_6 = (F_4 \diamond F_5) \oslash E_4 \oslash D_4. \tag{5.6}$$

Note que, os resultados obtidos em 5.3 e 5.4 são idênticos bem como os resultados em 5.2 e 5.5. Estes valores estão corretos pois eles representam a FTF da mesma ficha que pertence a dois estados distintos.

5.4 Índices de Desempenho

Existem diferentes tipos de índices de desempenho os quais podem ser classificados como individuais ou agregados. Os índices individuais estão relacionados a um único componente ou ponto de referência. Os índices agregados são computados baseados em parâmetros de diversos componentes. Diferentes métodos de análise de desempenho computam diferentes índices de desempenho. No caso das redes de Petri, os índices de desempenho podem ser obtidos a partir das extensões determinísticas e estocásticas. De cada tipo de extensão, índices distintos são calculados. De um lado, os índices de desempenho mais comuns computados a partir das extensões determinísticas estão relacionados com índices de *throughput*, i.e., o atraso para atravessar uma sub-rede ou para computar o tempo de ciclo mínimo [122]. Do outro lado, na abordagem estocástica, os índices de desempenho estão mais relacionados com os índices agregados [3]. Nesse caso, os índices mais comuns são: a probabilidade de um evento que é definida através da marcação dos lugares e o número médio de fichas em um lugar. Entretanto, utilizando o método estocástico, é muito difícil de se determinar alguns índices de desempenho individual.

Utilizando o método descrito na seção anterior, podemos determinar alguns índices de *throughput* bem como índices agregados e de alcançabilidade. É fácil de determinar a possibilidade de uma ficha em um lugar em um dado instante de tempo. Isso é diretamente obtido calculando-se a FTF associada com a ficha no lugar. Um outro índice de desempenho que pode ser computado é o tempo de atraso necessário para atingir um dado lugar ou estado a partir de um lugar ou estado inicial, respectivamente. Para tanto, temos que iniciar com zero as FTFs das fichas iniciais, i.e., as FTFs iniciais são representadas pelo intervalo $[0, 0]$ com função de pertencimento $\mu(0) = 1$.

Isso significa que começamos a computar o atraso a partir do instante do disparo da primeira transição. É importante observar que, para computar esse índice, temos que considerar que os lugares de entrada são independentes pois, uma vez considerando os intervalos de tempo, a informação imprecisa sobre o tempo de disparo pode levar a resultados errôneos.

Também, os índices de desempenho agregados relacionados com a possibilidade de um dado estado podem ser computados depois de se calcular as FTFs das fichas que compõem o estado ou marcação. A possibilidade de um estado é definida pela intersecção das FTFs[3]. Por exemplo, a possibilidade do estado *S2* no exemplo da Figura 5.2 é determinada pela intersecção entre $F3$ e $F4$.

Por fim, avaliação de alcançabilidade pode ser efetuada. Nesse caso, as restrições temporais são avaliadas para conhecer se seus valores estão corretos ou bem definidos. Vamos considerar novamente o exemplo da Figura 5.2. A análise de alcançabilidade pode determinar se o lugar *P6* ou estado *S5* são alcançáveis. Se $F4 \cap F5$ é vazio, o estado *S5* nunca é alcançado. Isso pode ser usado para definir corretamente as restrições de tempo durante a fase de projeto. Esse tipo de avaliação pode ser utilizado ainda para indicar se um estado não desejado foi alcançado ou se estados sem segurança são possíveis de serem alcançados.

5.5 Análise Temporal Utilizando G-Nets

No Capítulo anterior, vimos a necessidade de uma ferramenta estrutural para fazer análise de sistemas complexos. Além do mais, apresentamos uma integração entre $FTPN$ e *G-Net*. A integração foi efetuada, utilizando um modelo de *G-Net* decomposto onde algumas suposições foram consideradas para combinar os dois modelos. Logo, a ferramenta integrada permite o estudo de sistemas complexos dividindo o sistema em subsistemas, estudando separadamente cada subsistema e combinando os resultados parciais para determinar a solução global. Nesta seção, aplicamos a técnica de análise do grafo de alcançabilidade nebuloso para o exemplo do produtor/consumidor no sentido de ilustrar a análise temporal modular.

Além das suposições de integração apresentadas no capítulo anterior, temos que considerar que o conjunto das transições de dois métodos diferentes são livres de conflito, i.e., os disparos das transições de dois métodos distintos podem ser concorrentemente executadas sem interferir na execução normal de cada método. Isto é fundamental para

[3] Isso já foi definido na Seção 3.5 (veja Definição 3.11).

se obter valores corretos na análise de uma *FTG-Net*, porque podemos assumir independência de execução de processos. Na seqüência, definimos o conjunto das transições livres de conflito.

Definição 5.8 *Seja IS a estrutura interna de uma G-Net decomposta Gd. Sejam* $\mathcal{T}_{m1}$ *e* $\mathcal{T}_{m2}$ *dois conjuntos de transições pertencentes aos métodos* $method_1$ *e* $method_2$*, respectivamente. Estes dois conjuntos de transições são ditos livres de conflito se e somente se*

$$\bullet t \cap \bullet t' = \emptyset, \text{ with } t \neq t', \forall t \in \mathcal{T}_{m1} \text{ e } \forall t' \in \mathcal{T}_{m2}.$$

Existem dois tipos de conjuntos *livres de conflito*: estático e dinâmico. *Livre de conflito estático* significa que não existe lugar comum que seja entrada para transições que pertença ao conjunto de transições de diferentes métodos. *Livre de conflito dinâmico* indica que o conjunto de transições não está em conflito para um determinado comportamento dinâmico, mesmo que elas estejam em conflito estático.

Com o objetivo de efetuar a análise temporal modular de um sistema representado por um sistema de *G-Nets*[4], temos que considerar as seguintes etapas:

1. Analisar a *G-Net* invocada. Se esta invoca uma outra *G-Net*, a última deve ser analisada primeiro.

2. Embutir o resultado na *G-Net* que invocou associando o resultado ao *isp* decomposto.

3. Analisar a *G-Net* que invocou.

Como resultado da análise da *G-Net* invocada, estamos interessados em determinar o atraso para executar o método utilizado. Logo, como mostramos na Seção 5.4, temos que fixar a FTF inicial de acordo, i.e., a FTF inicial é iniciada com $[0, 0]$. O atraso para executar a rede invocada é representado pela FTF carregada pela ficha no lugar alvo da *G-Net* invocada.

Retornando para o exemplo do produtor/consumidor introduzido no capítulo anterior, antes de analisarmos o produtor, temos que primeiro analisar o consumidor. Considerando o estado pronto para consumir, a *G-Net* decomposta $Gd(C)$ é invocada duas vezes. Primeiro, $Gd(C)$ é invocada com o método *ms* e depois com o método

[4] Daqui por diante o termo *G-Net* significa a *G-Net* decomposta com intervalos nebulosos de tempo.

	LUGARES								
ESTADOS	SC	IN	RC	RC'	WP	TC	CO	GP	GPC
S0	1	0	1	0	0	0	0	0	0
S1	0	1	1	0	0	0	0	0	0
S2	0	0	0	1	1	0	0	1	0

Tabela 5.2: Possíveis estados para a invocação de $Gd(C)$ com o método ms

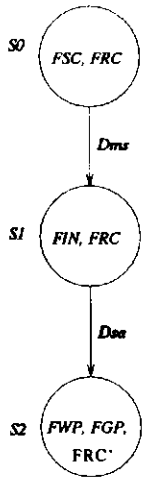

Figura 5.3: FRG para o consumidor considerando o método ms

mc. Cada invocação é estudada isoladamente e os resultados embutidos em $Gd(P)$, i.e., eles são associados com as transições $C.ms$ e $C.mc$ (veja Figura 4.7), respectivamente. No caso do método ms, três estados são alcançáveis como mostrado na Tabela 5.2. A Figura 5.3 mostra o grafo de alcançabilidade nebuloso para a *G-Net* decomposta $Gd(C)$ invocada com o método ms. De forma semelhante, como apresentado na Seção 5.3, as seguintes FTFs são computadas:

$$FIN = FSC \oslash Dms. \tag{5.7}$$

$$FWP = FGP = FRC' = (FIN \diamond FRC) \oslash Dsa. \tag{5.8}$$

O intervalo nebuloso correspondente a esta invocação é dada pela função nebulosa de tempo carregada pela ficha no lugar alvo GP. Considerando que os valores das FTFs da marcação inicial não são nebulosos ($FSC = [0, 0]$), a partir das Expressões (5.7) e (5.8) computamos a FTF associada com a ficha em GP (Expressão (5.9)). O intervalo nebuloso é associado à transição $C.ms$ em $Gd(P)$.

$$FGP = Dms \oslash Dsa. \tag{5.9}$$

	LUGARES								
ESTADOS	SC	IN	RC	RC'	WP	TC	CO	GP	GPC
S0	1	0	0	1	1	0	0	0	0
S1	0	0	0	1	1	1	0	0	0
S2	0	0	0	1	0	0	1	0	1

Tabela 5.3: Possíveis estados para a invocação de $Gd(C)$ com o método mc

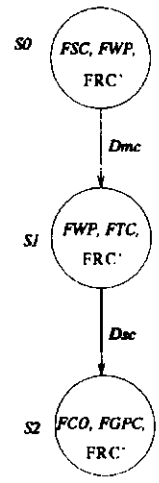

Figura 5.4: FRG para o consumidor considerando o método mc

A Tabela 5.3 e a Figura 5.4 mostram os possíveis estados alcançáveis e o grafo de alcançabilidade nebuloso para a invocação de $Gd(C)$, utilizando o método mc, respectivamente. Da mesma forma, as equações das funções nebulosas de tempo são determinadas e o intervalo nebuloso correspondente a esta invocação, expressão (5.12), é associado à transição $C.mc$ em $Gd(P)$.

$$FTC = FSC \oslash Dmc. \tag{5.10}$$

$$FCO = FGPC = (FWP \diamond FTC) \oslash Dsc. \tag{5.11}$$

$$FGPC = ((Dms \oslash Dsa) \diamond Dmc) \oslash Dsc. \tag{5.12}$$

Considerando o estado pronto para consumir e os intervalos nebulosos de tempo determinados acima, cada invocação de $Gd(C)$ é estudada isoladamente e as propriedades temporais de $Gd(P)$ podem então ser determinadas. A Tabela 5.4 mostra os possíveis estados alcançáveis para o produtor e a Figura 5.5 mostra seu grafo de alcançabilidade

ESTADOS	LUGARES														
	SP	PR	RI1	SI1	RIs	RC1	W1	RR1	RI2	SI2	RIc	RC2	W2	RR2	GP
S0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0
S1	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0
S2	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0
S3	0	0	0	1	1	0	1	0	0	1	0	0	0	0	0
S4	0	0	0	1	0	1	1	0	0	1	0	0	0	0	0
S5	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0
S6	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0
S7	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0
S8	0	0	0	1	0	0	0	0	0	1	0	1	1	0	0
S9	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0
S10	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1

Tabela 5.4: Possíveis estados para $Gd(P)$ considerando as invocações

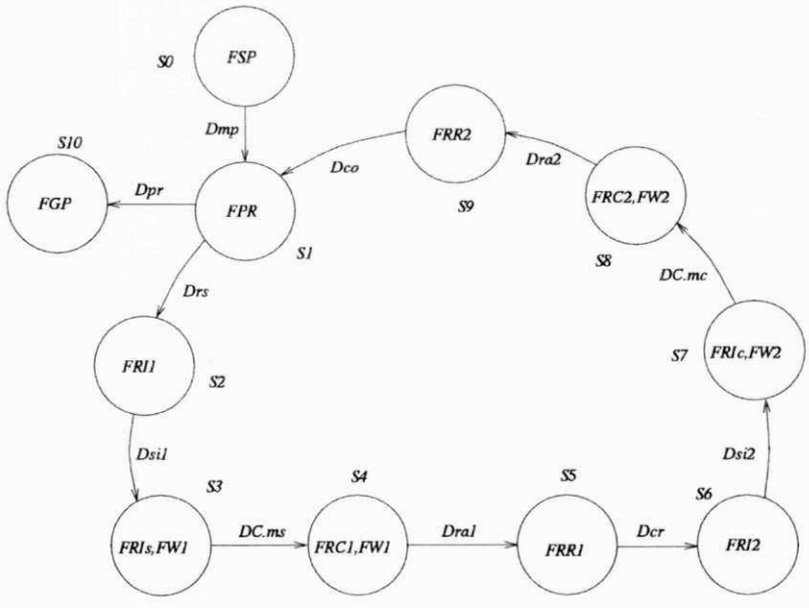

Figura 5.5: *FRG* para o produtor

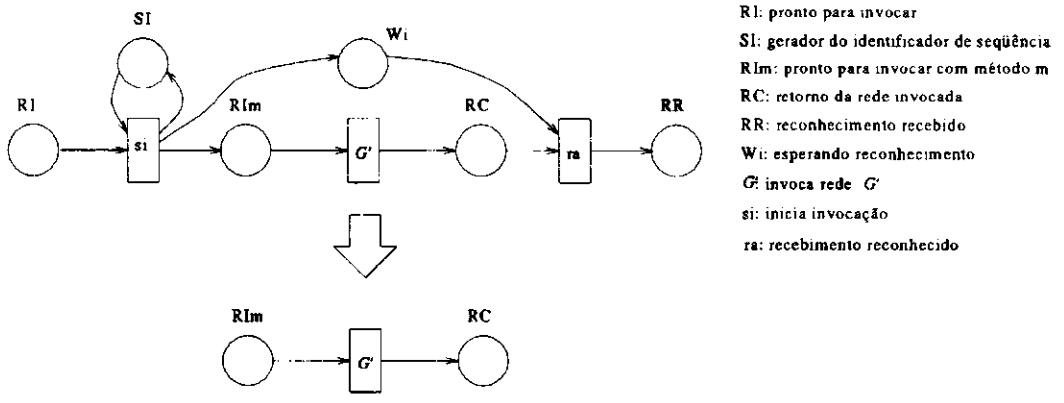

Figura 5.6: *isp* decomposto simplificado com o propósito de análise temporal

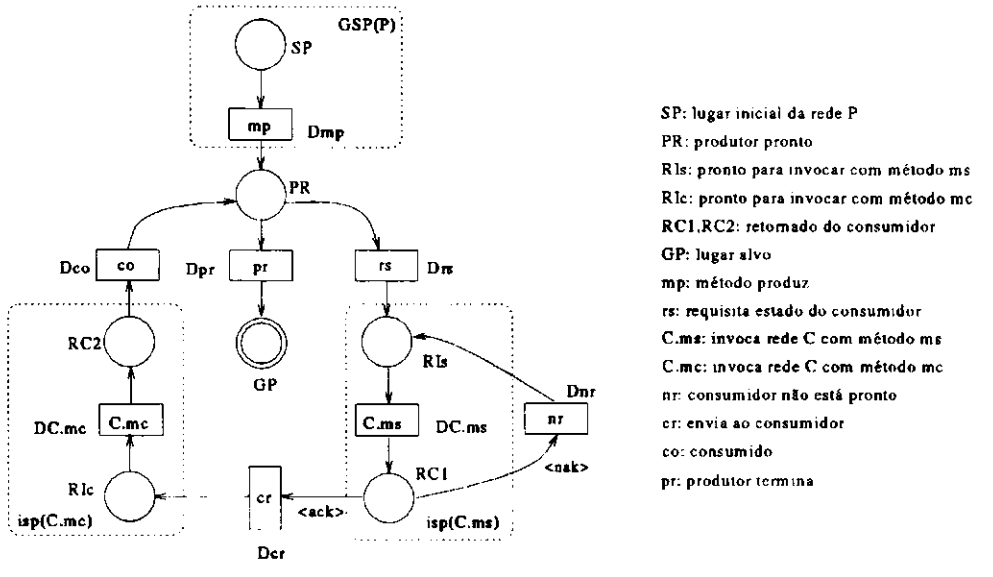

Figura 5.7: *G-Net* decomposta simplificada para o produtor

nebuloso. As funções nebulosas de tempo para o produtor são identicamente determinadas.

No caso da análise temporal, o *isp* decomposto pode ser simplificado como mostrado na Figura 5.6, em que *G'* é a transição que representa a execução da rede invocada. Isso é verdade porque estamos apenas interessados no tempo de execução da rede invocada cujo valor é associado à *G'*. Ainda consideramos que não existem restrições temporais associadas com as transições no *GSP* decomposto.

Então, usando a versão simplificada do *isp* decomposto, a *G-Net* simplificada, representando o produtor é mostrada na Figura 5.7. Os *isp*s simplificados são representados pelas linhas pontilhadas. O *isp* relativo ao método *ms* é composto pelos lugares *RIs* e

	LUGARES					
ESTADOS	PR	RIs	RC1	RIc	RC2	GP
S0	1	0	0	0	0	0
S1'	0	1	0	0	0	0
S1"	0	0	1	0	0	0
S2'	0	0	0	1	0	0
S2"	0	0	0	0	1	0
S3	0	0	0	0	0	1

Tabela 5.5: Possíveis estados para a *G-Net* simplificada para o produtor

RC1, e pela trasnsição *C.ms*. O *isp* relativo ao método *mc* é composto pelos lugares *RIc* e *RC2*, e pela transição *C.mc*. As transições *C.ms* e *C.mc* representam a invocação do consumidor com os métodos *ms* e *mc*, respectivamente.

Uma vez que não estamos considerando qualquer tipo de restrição temporal sobre a transição *mp*, o estado inicial é composto por uma ficha no lugar *PR*. Para essa marcação inicial, existem seis estados possíveis como mostrado na Tabela 5.5. O grafo de alcançabilidade nebuloso completo para a rede simplificada é mostrado na Figura 5.8.

A análise da *G-Net* simplificada pode ser executada da mesma forma como definida anteriormente. Embora o *isp* decomposto simplificado seja suficiente para a análise temporal, o *isp* decomposto original é necessário para muitas aplicações como nas áreas de tolerância a falhas e sistemas distribuídos em tempo real [24]. Na seção seguinte, apresentamos um algoritmo de análise temporal e sua aplicação no problema do produtor/consumidor acrescido de valores numéricos.

5.6 Algoritmo de Análise Temporal

Nessa seção, detalhamos o algoritmo para efetuar a análise temporal baseada no grafo de alcançabilidade nebuloso. O algoritmo reflete a técnica de análise de alcançabilidade introduzida no início deste capítulo, i.e., para computar as demais FTFs, é necessário observar os estados e combinar os intervalos nebulosos adequadamente. Antes de apresentar o algoritmo, vamos introduzir algumas definições básicas para facilitar o entendimento do algoritmo.

Baseada nas Definições 5.1 e 5.4, a Tabela 5.6 resume o grafo de alcançabilidade nebuloso mostrado na Figura 5.2(b). $FSET$ é o conjunto das funções nebulosas de tempo para cada estado e IRS indica o conjunto dos estados imediatamente alcançáveis para cada estado.

No algoritmo que estamos propondo, consideramos redes *1-safe*. Por uma questão de

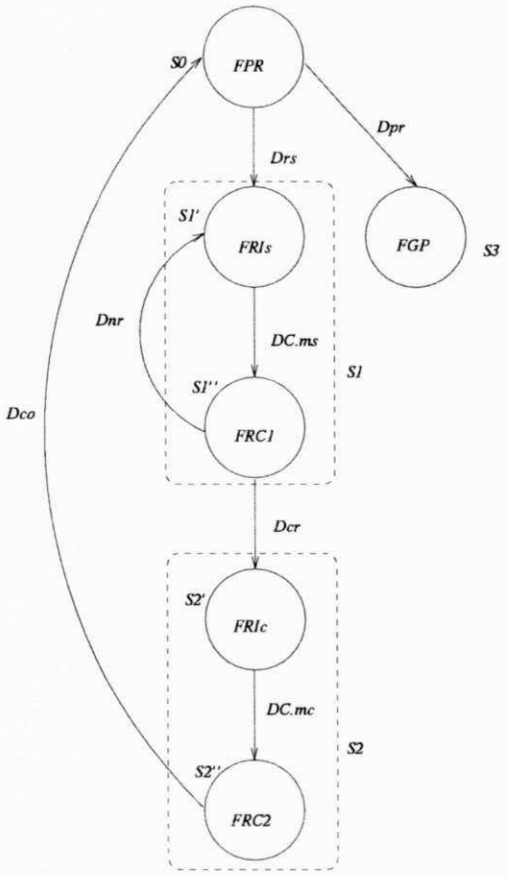

Figura 5.8: *FRG* para o produtor decomposto simplificado

ESTADOS	*FSET*	*IRS*
S0	F1	(E1, t1)
S1	F2, F3	(S2, t2), (S3, t3)
S2	F3, F4	(S4, t3)
S3	F2, F4	(S4, t2)
S4	F4, F5	(S5, t4)
S5	F6	–

Tabela 5.6: Sumário do *FRG* mostrado na Figura 5.2(b)

```
1.  VS = S0
2.  For all s ∈ PS do
3.      SF0 = FSET(s)
4.      B = IRS(s)
5.      For all b ∈ B do
6.          if b.state ∈ VS
7.          then
8.              "Estado visitado"
9.          else
10.             SF1 = FSET(b)
11.             For all sf ∈ SF1 do
12.                 if (sf ∈ SF0)
13.                 then
14.                     SF0 = SF0 − sf
15.                     SF1 = SF1 − sf
16.                 endif
17.             enddo
18.             Int = ⋂SF0
19.             For all sf ∈ SF1 do
20.                 if sf ∈ IF
21.                 then
22.                     "FTF computada anteriormente"
23.                 else
24.                     sf.value1 = b.value1 + b.value3 + Int.value1
25.                     sf.value2 = b.value2 + b.value4 + Int.value2
26.                     IF = IF + sf
27.                 endif
28.             enddo
29.             VS = VS + b.state
30.         endif
31.     enddo
32. enddo
```

Figura 5.9: Algoritmo de análise temporal

simplicidade, ainda assumimos que os intervalos são caracterizados por valores *crisp*[5] e que não existem ciclos na $FTPN$[6]. Além do estado inicial (S0) e o conjunto das funções nebulosas de tempo (IF), o algoritmo recebe como entrada o conjunto de todos os possíveis estados (PS), o conjunto de estados imediatamente alcançáveis para cada estado (IRS) e o conjunto de função nebulosa de tempo para cada estado ($FSET$), os quais são derivados do grafo de alcançabilidade. Como saída, todos os valores das FTFs são computados. O algoritmo detalhado é mostrado na Figura 5.9, em que VS é o conjunto dos estados visitados, B é o conjunto dos estados imediatamente alcançáveis, $SF0$ e $SF1$ são, respectivamente, o conjunto das funções nebulosas de tempo para o estado e seu estado imediatamente sucessor.

Inicialmente, o conjunto de estados visitados VS é restringido ao estado inicial IS (linha 1). Então, para cada estado, temos que compará-lo com seus sucessores. Isso é feito, considerando o $FSET$ do estado corrente (linha 3) e o $FSET$ do seu sucessor (linha 10). Para otimização, todos os sucessores são verificados se já foram analisados (linha 6). Os dois $FSET$s são comparados e os elementos comuns são removidos dos conjuntos (linhas 11– 17)[7]. Então, os elementos restantes do FSET do estado corrente são combinados de acordo com a regra **R2** (linha 18). O resultado é então combinado com os intervalos nebulosos (linhas 24 e 25) (regra **R3**). Uma vez que nós estamos considerando valores *crisp*, o intervalo de sensibilização de um estado sucessor b é representado no algoritmo pelo intervalo $[b.value1, b.value2]$. O intervalo de disparo é representado por $[b.value3, b.value4]$. $[Int.value1, Int.value2]$ representa a combinação das fichas nebulosas de entrada. O resultado final, $[sf.value1, sf.value2]$, é armazenado para futuras computações (linha 26). O conjunto dos estados visitados é atualizado e o ciclo é repetido até terminar a computação de todas as FTFs.

5.6.1 Complexidade do Algoritmo

Os algoritmos podem ser avaliados por uma variadade de critérios como, por exemplo, a quantidade de dados de entrada (complexidade de tamanho) ou o tempo necessário para o algoritmo, expresso como uma função do tamanho do problema (complexidade de tempo) [2]. Para computar a complexidade de tempo do algoritmo de análise temporal, temos que considerar os quatro laços do algoritmo (linhas 2 – 32, 5 – 31, 11 – 17 e 19

[5] Para considerar valores nebulosos ao invés de valores *crisp*, temos que implementar algumas funções nebulosas para computar a combinação de intervalos nebulosos.

[6] Nesse caso, é necessário fazer algumas asserções para extrair os índices requeridos.

[7] Isso é equivalente à computação de SGF e SIF (ver Definições 5.6 e 5.7)

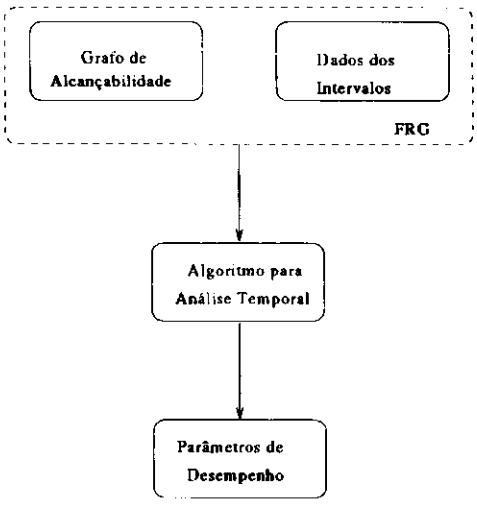

Figura 5.10: Diagrama de blocos para a análise temporal

–28). Considerando os dois laços mais internos, o valor da eficiência para o primeiro laço (linhas 11 – 17) é sf e o valor da eficiência para o segundo laço (linhas 19 – 28) não é maior do que sf. Como um laço segue o outro, nessa análise de complexidade podemos considerar apenas um desses dois laços e a eficiência deles é, no pior caso, sf. Agora, ficamos com três laços aninhados e as operações críticas para essa análise são aquelas efetuadas no laço mais interno. Conseqüentemente, a complexidade do algoritmo é $O(s \times b \times sf)$ em que, s é o número de estados, b é o número de estados imediatamente alcançáveis para um dado estado e sf é o número de FTFs para um dado estado imediatamente alcançável.

Para determinar o limite superior da complexidade de tempo do algoritmo de análise temporal, temos que fazer algumas considerações baseadas no pior caso. Uma vez que o valor de sf está relacionado ao número de lugares (p) e as redes são *1-safe*, o pior caso do valor de sf é p. b representa o número de estados imediatamente alcançáveis e como nós não estamos considerando ciclos, o pior caso para o valor de b é $S - 1$. Então o limite superior da complexidade de tempo é $O((S^2 - S) \times p)$. O termo S^2 é dominante sobre S então, o limite superior da complexidade de tempo é $O(p \times S^2)$ em que p é o número de lugares e S é o número de estados.

5.6.2 Implementação e Exemplificação

Com o objetivo de determinar automaticamente alguns índices de desempenho, o algoritmo proposto para computar as FTFs de uma $FTPN$ foi implementado utilizando a linguagem C. Assumimos que o grafo de alcançabilidade nebuloso é conhecido e ser-

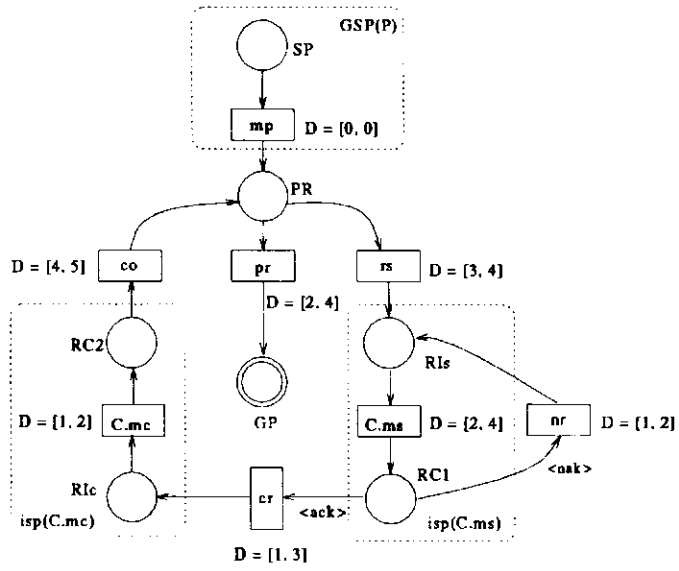

Figura 5.11: *G-Net* modelando o produtor decomposto com valores numéricos

virá como entrada para o algoritmo de análise temporal. Na realidade, o algoritmo implementado está integrado juntamente com um simulador de *G-Nets* e um pacote para efetuar a análise comportamental de sistemas de *G-Net*, compondo um *ambiente para análise de G-Nets* (*G-Net workbench*). Rodando o simulador, é possível obter-se o grafo de alcançabilidade da *G-Net*. Informações de tempo nebuloso (intervalos de tempo nebulosos e FTFs iniciais) armazenadas em arquivos são combinadas com o grafo de alcançabilidade determinando o correspondente FRG. Aplicando-se o algoritmo de análise temporal à FRG, computamos índices de desempenho (FTF para as fichas restantes). O diagrama de blocos para a análise temporal automática é mostrado na Figura 5.10. A descrição detalhada da *G-Net workbench* está fora do escôpo dessa Tese e nos concentraremos apenas na parte de análise temporal.

No exemplo do produtor/consumidor, o grafo de alcançabilidade nebuloso para o produtor, como mostrado na Figura 5.8, apresenta dois ciclos. No entanto, de acordo com a definição do algoritmo de análise temporal, a análise pode ser efetuada se considerarmos redes *1-safe* e que não contenham ciclos. Para utilizar o algoritmo, é necessário fazer algumas suposições para remover os ciclos do grafo de alcançabilidade. Isso é feito considerando um certo comportamento do consumidor. Em nosso exemplo, estamos assumindo que o consumidor, quando invocado com o método *ms*, retorna dois *nak*s antes de retornar um *ack*. Consideramos ainda que o produtor produz apenas um ítem. Logo, o consumidor é invocado três vezes com o método *ms* e uma vez com o método *mc*. A Figura 5.11 mostra a *G-Net* decomposta, modelando o produtor que é aumentada com

ESTADOS	*FSET*	*IRS*
S0	FPR	(S1, rs)
S1	FRIs/FRC1	(S1', nr)
S1'	FRIs/FRC1	(S1", nr)
S1"	FRIs/FRC1	(S2, cr)
S2	FRIc/FRC2	(S0', co)
S0'	FPR	(S3, pr)
S3	FGP	–

Tabela 5.7: Sumário do *FRG* mostrada na Figura 5.12

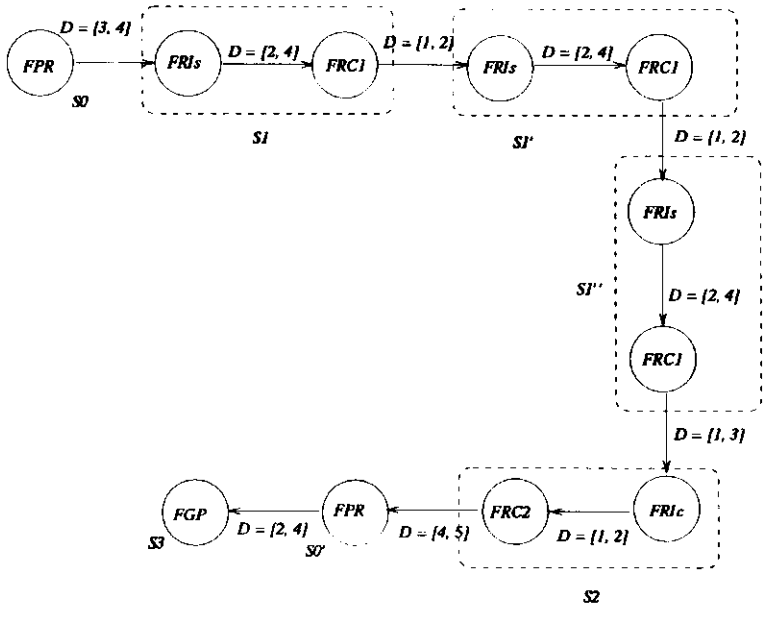

Figura 5.12: *FRG* para a *G-Net* mostrada na Figura 5.11

valores numéricos. Os intervalos de disparo são associados com as transições. Os tempos de execução para as invocações são associados com as transições no *isp* decomposto simplificado. Por exemplo, o tempo de execução para a *G-Net* $G(C)$ com o método ms é determinado pelo intervalo $D = [2, 4]$ associado à transição $C.ms$ na Figura 5.11.

O grafo de alcançabilidade nebuloso parcial, que corresponde ao grafo computacional, é gerado expandindo-se o grafo de alcançabilidade nebuloso apresentado na Figura 5.8 e é apresentado na Figura 5.12. Os valores numéricos para os intervalos de disparo são também representadas. A Tabela 5.7 resume o grafo de alcançabilidade nebuloso mostrado na Figura 5.11 e serve como entrada para ao algoritmo de análise temporal.

A Figura 5.13 apresenta a cópia da tela da análise temporal, utilizando o algoritmo implementado. Como resultado, as FTFs são determinadas e, conseqüentemente, o atraso para alcançar o lugar alvo (tempo de execução da rede). Os valores das FTFs são apresentados na tela *main* (esquerda-abaixo).

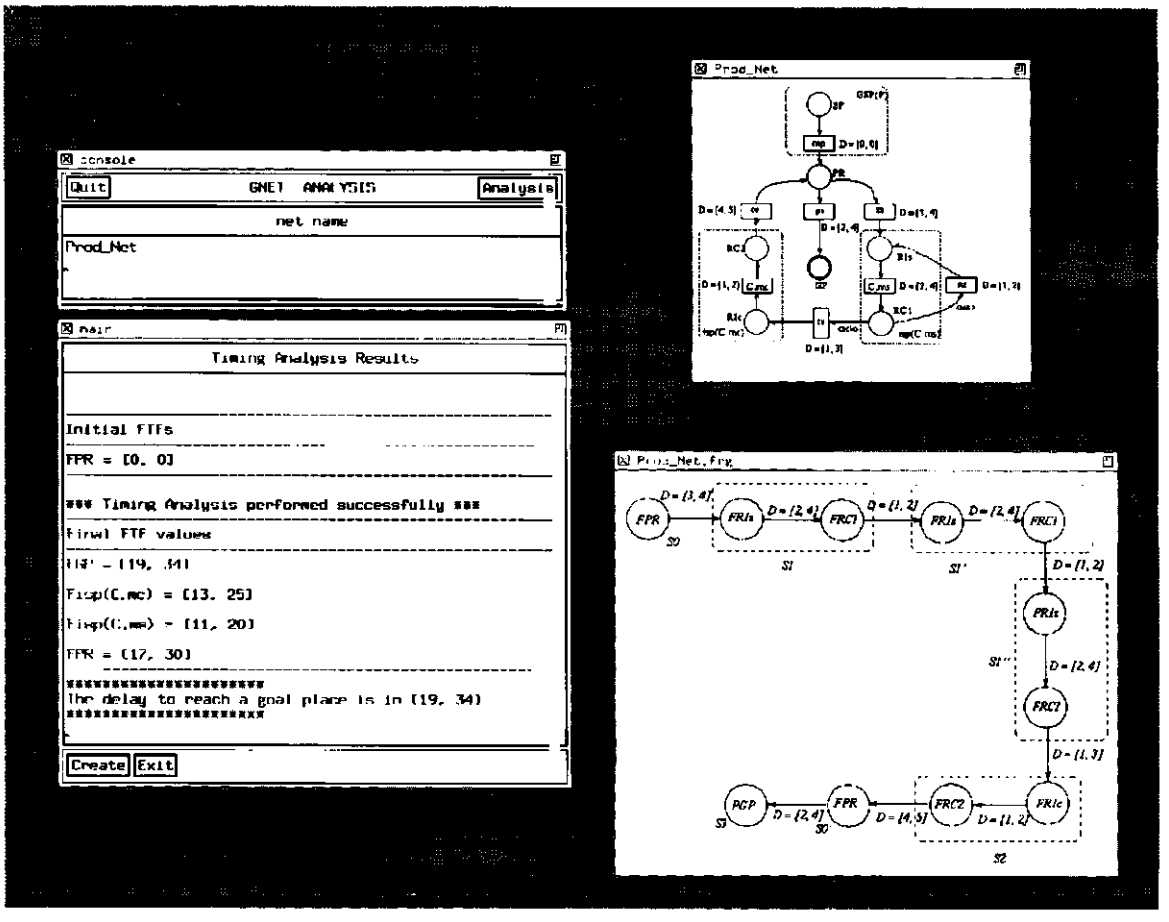

Figura 5.13: Cópia da tela para a análise temporal

5.7 Representação Gráfica para a Visualização do Comportamento Temporal

A análise temporal modular mostrada na última seção é muito importante quando estamos tratando com sistemas complexos e distribuídos. A análise temporal é concentrada nos elementos que são responsáveis pela interação entre módulos. Nessa seção, introduzimos um diagrama de tempo para mostrar, graficamente, o comportamento de execução no tempo de um sistema, focalizando na interação entre seus módulos. O diagrama de tempo chamado de *Gráfico de Interação entre G-Nets com Temporização (TGIG)* é um gráfico bi-dimensional que apresenta os aspectos de tempo e de interação entre *G-Nets* em um sistema de *G-Nets*. O *TGIG* pode ser usado como uma abordagem alternativa para determinar alguns índices de desempenho como veremos na seqüência dessa seção.

No eixo vertical, representamos as *G-Nets* e os seus correspondentes métodos en-

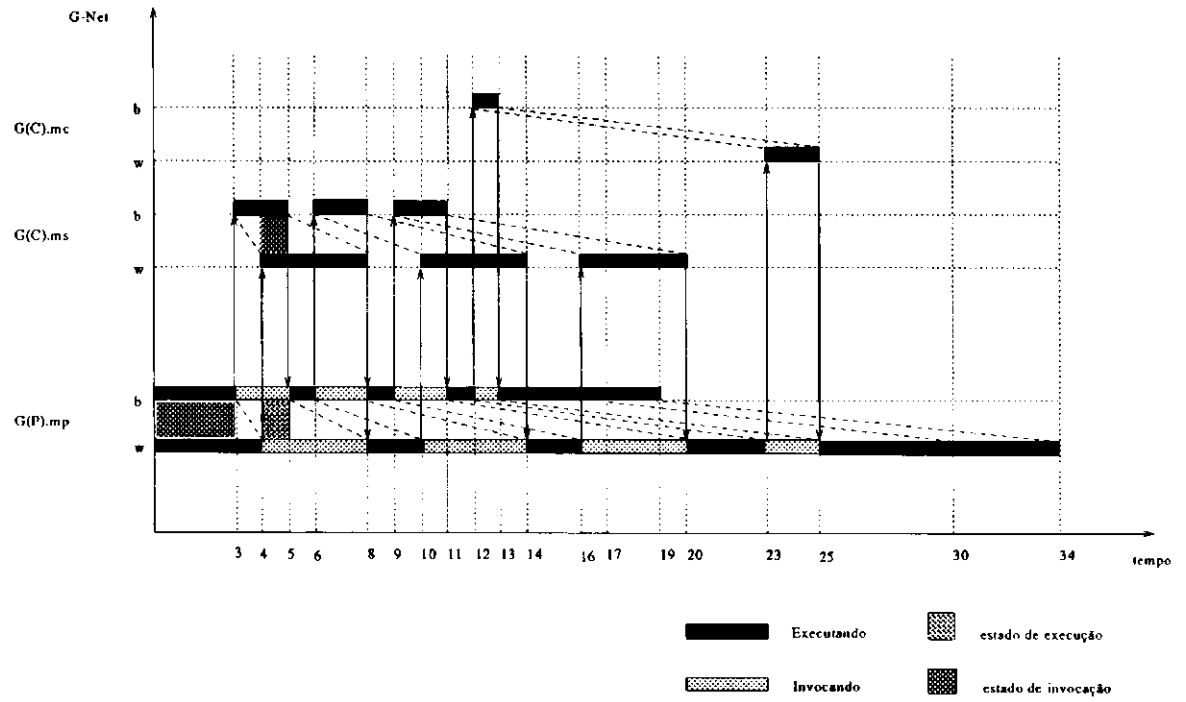

Figura 5.14: *TGIG* para o exemplo do produtor/consumidor

quanto que no eixo horizontal representamos os estados das *G-Nets* e os respectivos tempos de duração. Para cada *G-Net*, estamos considerando dois tipos de estados: executando e invocando. Também, para cada *G-Net* e um específico método[8], consideramos o melhor e pior casos. As invocações das *G-Nets* são denotadas por um arco dirigido conectando duas diferentes *G-Nets*. A direção do arco define que *G-Net* está invocando e que *G-Net* está sendo invocada. O arco dirigido ainda representa o fim da invocação. Para ilustrar o *TGIG*, vamos usar o exemplo do produtor/consumidor.

Na Figura 5.14, apresenta-se o *TGIG* para o exemplo do produtor/consumidor. Estamos considerando os intervalos de tempo de disparo como mostrado na Figura 5.11 e assumimos que o consumidor é invocado três vezes com o método ms. Vamos supor que o consumidor retorna *nak*, *nak* e *ack* para a primeira, segunda e terceira invocação, respectivamente. Além do mais, o consumidor é invocado uma vez com o método mc. Também estamos considerando que a marcação inicial é dada por uma ficha no lugar PR e $FTF(PR) = [0, 0]$. Logo, na coluna vertical, temos que representar três diferentes combinações *G-Net/método*. Para cada uma, mostramos os melhores e piores casos, denotados pelas letras b e w na coluna vertical. Nas linhas horizontais, para cada *G-Net/método*, representamos seus estados e duração. As barras escuras nas li-

[8] Nós nos referimos à execução de uma *G-Net* com um método específico pela notação *G-Net/método*.

nhas horizontais denotam que a *G-Net* está executando e as barras claras indicam que a *G-Net* está invocando. De acordo com a marcação inicial, apenas uma transição está sensibilizada e o seu disparo tem uma duração que é, no melhor caso, igual a 3 e, no pior caso, igual a 4. Isso é representado no *TGIG* por uma barra escura na linha b de 0 a 3 e na linha w de 0 a 4. Após isso, uma ficha é depositada no *isp* que invoca a rede que modela o consumidor. Isso é representado no *TGIG* por um arco dirigido conectando as linhas b e w da *G-Net* $G(C).ms$. Um arco dirigido conectando a *G-Net* $G(C).ms$ à *G-Net* $G(P).mp$ indica o fim da invocação. É interessante notar que enquanto a *G-Net* invocada está executando (barra escura), a *G-Net* que invocou permanece no estado invocando (barra clara). Logo, para o melhor caso, entre tempo 3 e tempo 4 a *G-Net* $G(C).ms$ está no estado executando e a *G-Net* $G(P).mp$ está no estado invocando. Não estamos considerando atraso de comunicação durante as invocações mas isso pode ser facilmente introduzido em nosso *TGIG*. Nesse caso, ao invés de arcos dirigidos verticais, teríamos arcos inclinados onde o deslocamento horizontal indicaria o atraso. As linhas tracejadas inclinadas conectando os melhores e piores casos para cada estado, i.e., as linhas tracejadas entre tempo 3 e 4, e tempos 5 e 8 relacionados com $G(P).mp$ delimitam os melhores e piores casos para a primeira invocação de $G(C).ms$.

Diferentes tipos de informação podem ser obtidas a partir do *TGIG*. Podemos computar o melhor e pior tempo de execução de um *G-Net* (19 e 34 para $G(P).mp$). O tempo de execução total e o tempo de invocação para uma dada *G-Net*. Isso pode ser obtido pela soma dos intervalos de tempo na qual a *G-Net* permaneceu no estado executando ou invocando. Além do mais, combinando a representação dos melhores e piores casos, podemos determinar intervalos de tempo no qual uma dada *G-Net* está com certeza em um estado de execução ou invocação. Isso é determinado traçando linhas verticais entre as linhas de melhor e pior casos. Se as cores das barras são iguais e elas representam o mesmo estado (as linhas verticais tracejadas não interceptam as linhas tracejadas inclinadas), podemos dizer que a *G-Net* está, com certeza, em um estado de execução ou invocação, dependendo da cor das barras (escura ou clara). Por exemplo, considerando a *G-Net* $G(P).mp$, entre os tempos 0 e 3, $G(P).mp$ está, com certeza, no estado de execução e entre 4 e 5, $G(P).mp$ está em um estado de invocação.

A construção de um *TGIG* é muito simples mas, em alguns casos, pode ser relativamente complexa. Isso acontece quando temos transições sensibilizadas concorrentemente, i.e., quando mais de uma transição dispara no mesmo instante de tempo pois, temos que representar o melhor e pior caso para cada transição disparada. Portanto, se o número de transições sensibilizadas concorrentes é grande, a construção do *TGIG*

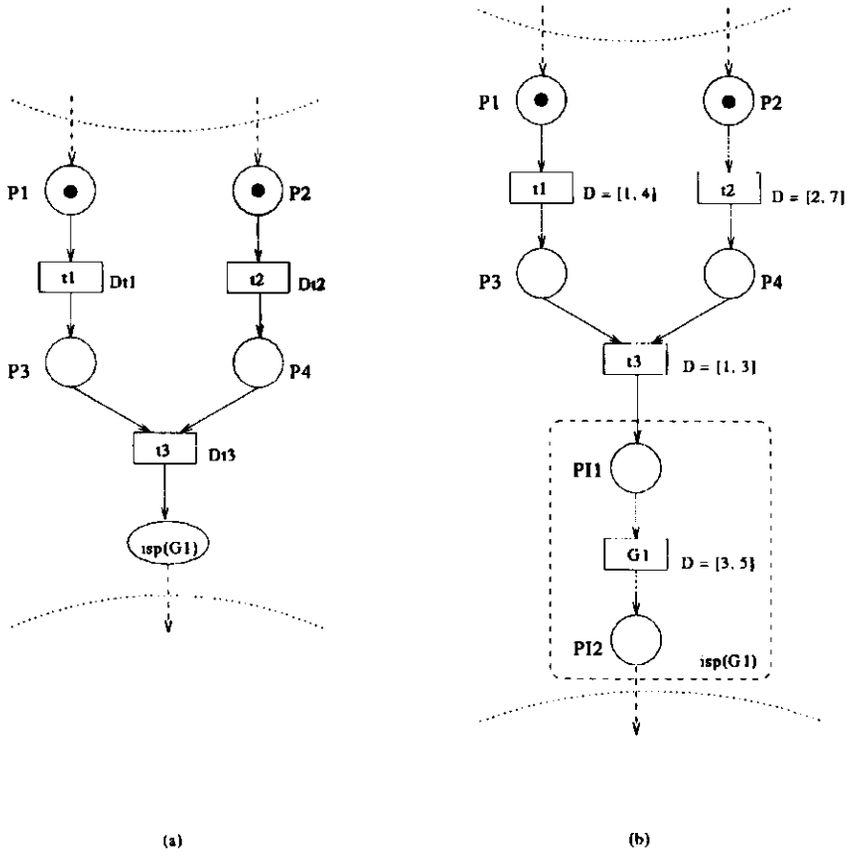

Figura 5.15: Pedaço da estrutura interna de uma *G-Net* com concorrência

pode ser impraticável.

Para ilustrar como tratar este problema, vamos supor um pedaço da estrutura interna de uma *G-Net* $G0$ como mostrado na Figura 5.15(a). Esta figura indica que, em um dado instante da execução da *G-Net* $G0$, temos duas transições sensibilizadas concorrentes: *t1* e *t2*. Depois de disparar essas duas transições, uma ficha é depositada nos lugares *P3* e *P4*. Então, a transição *t3* está sensibilizada e, depois de disparada, uma ficha é depositada no $isp(G1)$ e $G1$ é invocada. Na Figura 5.15(b), a rede da Figura 5.15(a) é estendida com valores numéricos. A Figura 5.16 mostra o *TGIG* para o pedaço de rede mostrado na Figura 5.15. Estamos supondo que as fichas nebulosas iniciais são iniciadas em $[0, 0]$. Note que, para $G0$, nós representamos o disparo concorrente das transições *t1* e *t2*, duplicando as linhas horizontais correspondente à $G0$. Logo, para melhor caso de tempo de execução, para a transição *t1* temos a barra escura entre os tempos 0 e 1 e para a transição *t2* a barra escura está entre 0 e 2. Uma vez que o disparo da transição *t3* depende das fichas geradas pelos disparos das transições *t1* e *t2*, as linhas horizontais são combinadas para determinar apenas um único melhor caso e

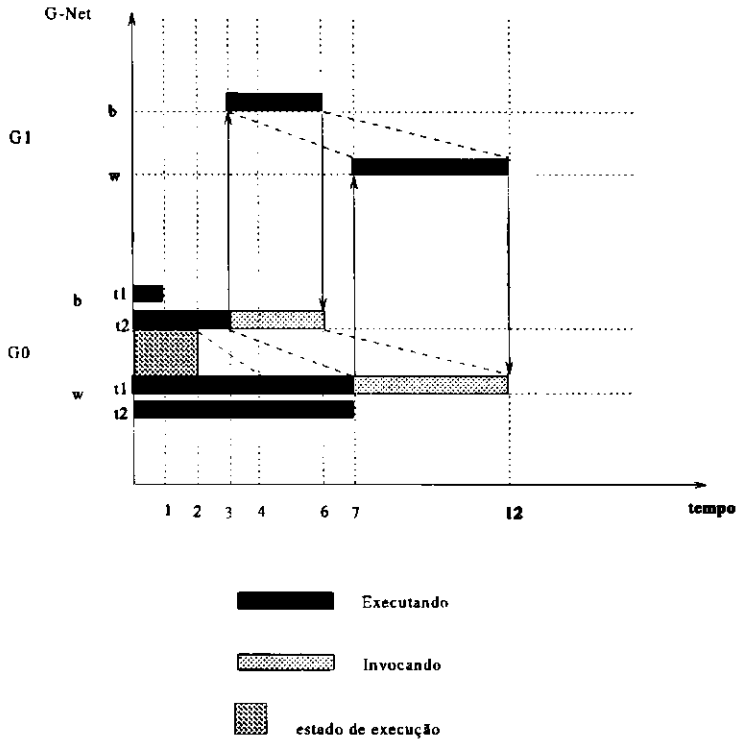

Figura 5.16: *TGIG* para a *G-Net* mostrada na Figura 5.15

um único pior caso. Seguindo a definição de *FTPN*, o melhor caso é determinado pelo máximo entre os melhores tempo dos valores concorrentes. O pior caso é determinado pela operação dual, i.e., o mínimo pior tempo entre os valores concorrentes. Em nosso exemplo, o melhor tempo para começar a execução de *t3* é 2 e o pior caso é 4.

A cópia da tela apresentada na Figura 5.17, mostra os resultados analíticos obtidos do algoritmo de análise temporal bem como do correspondente *TGIG*. A partir dessas duas telas, podemos observar que o diagrama de tempo reflete os resultados obtidos através do algoritmo. Por exemplo, o valor da FTF associado com $isp(C.ms)$ pode ser derivado do *TGIG* observando-se os valores máximos nas linhas horizontais correspondentes ao melhor e pior caso para *G-Net/método* $G(C).ms$.

Embora o uso do *TGIG* possa sofrer o problema das transições sensibilizadas concorrentemente, ele pode ser usado, com sucesso, para determinar alguns índices de desempenho relacionados às trilhas específicas de eventos e ações. A fácil computação de alguns parâmetros e a sua representação gráfica fazem do *TGIG* uma ferramenta interessante na análise temporal de sistemas. Através do *TGIG* é possível guardar uma história sobre a interação e aspectos temporais entre *G-Nets* em um sistema de *G-Nets*.

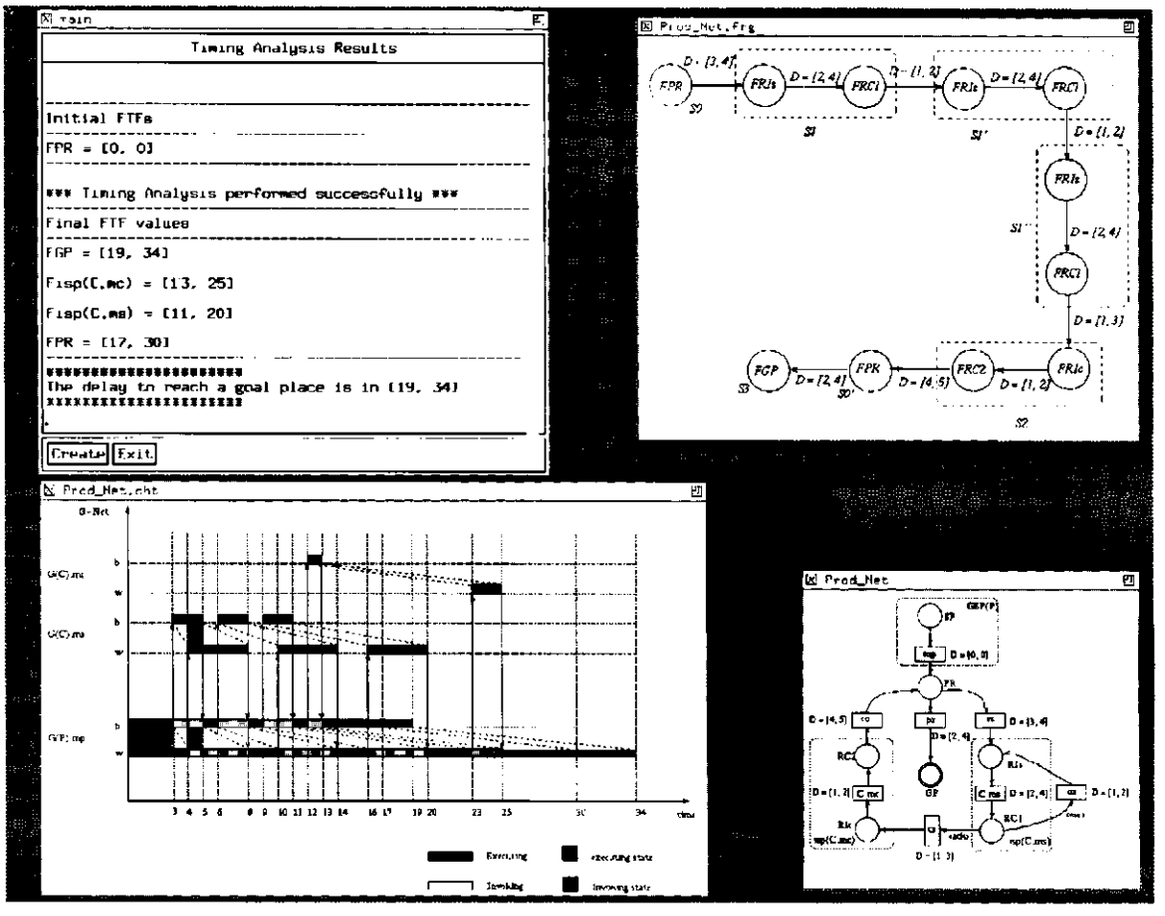

Figura 5.17: Cópia da tela para a análise temporal

Capítulo 6

TOLERÂNCIA A FALHAS E SISTEMAS DISTRIBUíDOS EM TEMPO REAL

6.1 Introdução

Devido ao crescente uso de sistemas de computação em aplicações críticas e complexas como, por exemplo, controle de processos, monitoramento de vôo aéreo, sistemas de cuidados médicos e sistemas de transporte, aspectos relacionados com a confiabilidade, segurança e tolerância a falhas ganharam grande importância na última década. Como todo e qualquer tipo de sistema, os sistemas de computação estão suscetíveis às falhas e, dependendo da natureza do sistema, as conseqüências introduzidas pelas falhas podem variar de uma simples inconveniência até uma catástrofe. Um sistema é tolerante a falhas se ele mantém completos desempenho e capacidade funcional mesmo na presença de falhas.

Para sistemas em tempo real, não apenas as propriedades de corretude lógica tais como *liveness* e *safeness* devem ser verificadas mas também corretude temporal. Portanto, no projeto de sistemas em tempo real, é necessária uma metodologia que inclua mecanismos para verificar as propriedades dependentes do tempo em adição às outras propriedades não dependentes do tempo. No mais, para um grande número de sistemas em tempo real, as propriedades relacionadas com a segurança são da maior importância porque elas interagem com o ambiente que pode produzir entradas não esperadas devido às falhas de hardware, software ou mudanças no ambiente. A propriedade de *timeliness* tem que ser verificada porque efeitos catastróficos podem ocorrer, caso uma ação não seja executada em tempo.

Considerando um sistema distribuído em tempo real (SDTR), requer-se também um alto grau de confiabilidade, disponibilidade e segurança. O sistema deve garantir desempenho como especificado nos domínios de valor e tempo [68, 90]. Em sistemas em tempo real, a maioria das falhas está relacionada com erros de desempenho e sincronização que se manifestam como falhas transitórias do sistema. Levando em conta os requisitos acima, o projetista de um SDTR deve adotar uma abordagem de projeto que suporte a introdução sistemática de propriedades de tolerância a falhas dependentes e

não dependentes do tempo em um componente de software [24, 34, 92].

Neste capítulo, nós discutiremos como as *FTG-Nets* podem ser aplicadas no projeto de SDTR, considerando características tolerantes a falhas. Nós discutiremos ainda os aspectos de tolerância a falhas dependentes do tempo e mostraremos como aplicar as *FTG-Nets* para modelar esquemas tolerantes a falhas dependentes do tempo. Nós também introduziremos uma abordagem para tratar falhas, antecipadamente em um sistema distribuído em tempo real. Por fim, nós apresentaremos um exemplo baseado no problema de controle de tráfego distribuído para mostrar a aplicação das *FTG-Nets* nesta área.

6.2 Conceitos Básicos

Na seqüência, nós definimos os termos *falha*, *erro* e *falta*. Uma definição clássica desses termos é aquela introduzida por Avizienis e Laprie [8]. Uma fonte potencial de um comportamento incorreto do sistema é dito *falta*. A causa de um *erro* é uma *falta*, i.e., um *erro* é portanto a manifestação de uma *falta* no sistema. Uma *falha* é o efeito de um *erro* no serviço.

Por uma questão de coerência com a bibliografia já existente em português, utilizaremos o termo falha para tratar indistintamente falhas e faltas. Existem diferentes tipos de *falhas*. Entre elas, nós podemos incluir as falhas permanentes e transitórias, falhas de software e hardware e, falhas temporais [72]. As falhas permanentes e transitórias estão relacionadas com a presença ou não de uma intervenção externa para repará-las. As falhas de hardware e software indicam se a falha foi manifestada em componentes de hardware ou software. As falhas temporais são aquelas que ocorrem devido a uma violação das restrições de tempo [84].

Concentrando-se nas características de tolerância a falhas, podemos dizer que na maioria dos sistemas controlados por computadores como sistemas de software dedicados (sistemas de controle aéreo e viário), o controle do sistema não pode ser interrompido abruptamente. Portanto, capacidades para responder a falhas de hardware, software, mudanças inesperadas no ambiente e erros humanos devem ser construídas no sistema. Estas respostas podem ter as seguintes formas:

1 - Tolerância a Falhas: é definida como a capacidade de um sistema continuar a prover completas capacidades de desempenho e funcional na presença de falhas de operação.

2 - Falha segura: significa que o sistema tenta limitar o tamanho do dano causado por uma falha. No entanto, nenhuma tentativa é feita para satisfazer as especificações funcionais, com exceção, quando for necessário garantir segurança.

3 - Falha suave: o sistema continua operando mas provê desempenho degradado ou capacidade funcional reduzida até que a falha seja removida.

4 - Vital: O sistema garante que algumas funcionalidades e respectivos desempenhos são garantidos durante um intervalo de tempo pré-definido, após o qual o sistema deve ser totalmente desabilitado em um estado seguro.

Portanto, no projeto de um sistema distribuído confiável, é essencial considerar os aspectos tolerantes a falhas. Tolerância a falhas é necessário para garantir que o sistema possa continuar a funcionar e prover serviços mesmo na presença de falhas em componentes. Diferentes técnicas[1] podem ser adotadas para introduzir tolerância a falhas as quais podem endereçar diferente tipos de falhas. As principais técnicas de tolerância a falhas incluem:

1 - Blocos de Recuperação (Recovery Blocks): é uma técnica tolerante a falhas que encapsula cada função crítica de um processo em um bloco de recuperação [17, 67]. Um bloco de recuperação consiste de três partes: uma rotina primária que executa a função crítica, um teste de aceitação que decide se os resultados providos pelo bloco primário são aceitáveis e, uma ou mais rotinas de *backup* que executam as mesmas funções das rotinas primárias que são gatilhadas se os resultados providos pelas rotinas primárias não são satisfatórios.

2 - Programação N-Versões: é uma técnica tolerante a falhas que é adequada para falhas de software. Neste caso, diferentes versões dos módulos de software são projetadas e implementadas [7]. As diferentes versões são executadas em paralelo e um mecanismo de votação é utilizado para avaliar os resultados e apenas um resultado é selecionado.

3 - Recuperação Regressiva com Pontos de Verificação: é um esquema tolerante a falhas tanto para hardware como para software [116, 118, 120]. A idéia é salvar,

[1] Estas técnicas tolerantes a falhas algumas vezes são chamados de esquemas tolerantes a falhas. Nesta Tese, nós usaremos estes termos indistintamente.

periodicamente, os estados de computação (pontos de verificação) em algum dispositivo confiável. Os pontos de verificação são usados para reiniciar o processo no caso de falhas, i.e., o processo é reiniciado a partir do último ponto de verificação. No caso de ambiente distribuído, é necessária uma técnica de manutenção de consistência em adição ao mecanismo de recuperação para evitar um número incontrolável de retornos. Este efeito pode ocorrer se os pontos de verificação entre os processos comunicantes são inconsistentes e é conhecido como *Efeito Dominó.*

4 - Conversação: é uma forma restrita de recuperação regressiva com pontos de verificação [105]. Neste esquema, um conjunto de processos comunicantes pode ter pontos de verificação independentes um dos outros. No entanto, uma vez começada a conversação, nenhum ponto de verificação é permitido e todos os processos são obrigados a deixarem a conversação em conjunto. Este esquema previne a ocorrência do *Efeito Dominó.*

5 - Troca: é uma forma restrita de conversação [5]. Todos os processos tomam seus pontos de verificação ao mesmo tempo antes de entrarem na troca. Os processos são restringidos a saírem da troca em conjunto. Este tipo de esquema também previne o *Efeito Dominó.*

A recuperação regressiva com pontos de verificação, troca e conversação são classificadas como técnicas de tolerância a falhas para trás. Os blocos de recuperação e programação N-Versões são ditas técnicas de tolerância a falhas para frente.

As técnicas de tolerância a falhas para trás, em muitos casos, não são apropriadas para ambientes em tempo real. A aplicabilidade de técnicas de tolerância a falhas em sistemas em tempo real deve levar em consideração a latência de recuperação que pode causar a violação das restrições de tempo. Por latência de recuperação nós entendemos o tempo decorrido entre a detecção do erro até o fim de sua recuperação. Portanto, as técnicas de tolerância a falhas para frente são mais apropriadas para sistemas em tempo real. As abordagens de tolerância a falhas mais comuns em sistemas em tempo real são: a abordagem *primário/secundário* e a abordagem de *redundância modular* [84].

A abordagem *primário/secundário* é eqüivalente ao esquema de blocos de recuperação apresentado anteriormente. Um módulo primário é executado e, em caso de falhas, o módulo secundário é gatilhado. A abordagem de *redundância modular* é adequada para sistemas em tempo real, devido à baixa latência de detecção e recuperação.

O principal problema com esta abordagem é a sobrecarga e a falta de flexibilidade na especificação de requisitos de tolerância a falhas.

Considerando os métodos formais para analisar e introduzir propriedades de tolerância a falhas em sistemas de computação, as redes de Petri estão entre os mais aplicáveis. Isto porque, usando redes de Petri, é possível unificar a modelagem de aspectos de software, hardware e ambiental através de um único formalismo.

Leveson [70, 71] utiliza *TPNs* [79] para a modelagem e análise de sistemas críticos de segurança em tempo real. A análise no tempo é utilizada para a definição de restrições de tempo nas ações para evitar que o sistema alcance um estado perigoso. Embora esta abordagem tenha sido aplicada com sucesso, ela é baseada em redes de Lugar/Transição [85, 93, 99] que apresenta limitações quando aplicada em sistemas de software complexos [47, 48].

Belli [11] apresenta uma metodologia para integrar, de uma forma sistemática, propriedades tolerantes a falhas dentro do projeto de sistemas de software complexos. O projeto é baseado nas redes de Predicado/Transição (Pr/T nets). Para introduzir propriedades tolerantes a falhas, partes seqüenciais do comportamento das Pr/T nets são isoladas e modeladas por expressões regulares, a partir das quais um código é gerado. Sempre que um erro ocorre, o mecanismo de código pode detectar e, talvez, corrigir o erro. A limitação mais séria para esta abordagem é que ela só pode ser aplicada em partes seqüenciais no comportamento do sistema/módulo.

Shieh [105] apresenta modelos analíticos, utilizando redes de Petri para esquemas tolerantes a falhas usados em sistemas distribuídos. Diferentes esquemas tolerantes a falhas que são modelados usando redes de Petri são apresentados bem como algoritmos para construí-los automaticamente. No entanto, não se discute como usar a abordagem proposta no caso de sistemas em tempo real.

6.2.1 FTG-Nets e Técnicas Tolerantes a Falhas

Para introduzir características tolerantes a falhas no projeto de um sistema, é necessário considerar as falhas que podem ocorrer em um sistema. Como definido anteriormente, uma falha sempre resulta em uma falta. Diferentes tipos de falhas podem ocorrer e podem ser classificadas como dependentes ou não do tempo. De um lado, entre as falhas que não dependem do tempo nós podemos citar:

- Um determinado evento não ocorre.
- A ocorrência de um evento indesejado.

- Dois eventos incompatíveis ocorrendo simultaneamente.

Por outro lado, existem três tipos de falhas dependentes do tempo em um sistema que pode levar a um comportamento faltoso. Estas falhas incluem:

- Exceder as restrições de tempo máximo entre eventos.
- Falha em garantir as restrições de tempo mínimo entre eventos.
- Falhas de duração (por exemplo, uma condição ou um conjunto de condições falham em ser verdadeiras por um certo período de tempo).

Nos sistemas em tempo real em que o desempenho deve ser garantido como especificado nos domínios do tempo e valor, é necessário analisar o comportamento temporal do sistema para verificar se as restrições de tempo e índices de desempenho são cumpridos. Analisando o comportamento temporal é possível evitar a ocorrência de falhas temporais.

A integração entre *FTPN* e o modelo de *G-Nets* pode ser usada no sentido de tolerar falhas dependentes do tempo. Na seqüência, nós apresentamos um esquema baseado na técnica de recuperação regressiva para tratar falhas temporais.

Definição 6.1 *Recuperação regressiva temporizada é um esquema baseado na idéia de watchdog timers. Considerando uma comunicação entre componentes, se um processo que está esperando não recebe resposta em tempo, assume-se que um erro ocorreu no objeto invocado. Portanto, o processo que estava esperando recomeça a execução a partir do ponto de verificação mais recente.*

No que se segue, nós mostramos como embutir *recuperação regressiva temporizada* no projeto de um SDTR baseado em sistemas de *G-Nets*. Para alcançar isso, é necessário considerar o *isp* decomposto, uma vez que o *isp* é o único elemento denotacional em uma *G-Net* que inicia uma comunicação entre redes. A idéia é adicionar, para cada *isp*, restrições temporais correspondente ao tempo encapsulado relacionado com a *G-Net* invocada, neste caso o *pior tempo de execução*. Uma vez que nós temos que especificar restrições de tempo, nós usamos *FTG-Nets* em que as restrições de tempo são representadas por intervalos de tempo. Além do mais, a análise de desempenho pode ser efetuada utilizando a abordagem proposta. A análise de desempenho pode ser efetuada em tempo de execução baseado no tempo em que a invocação de um objeto necessita

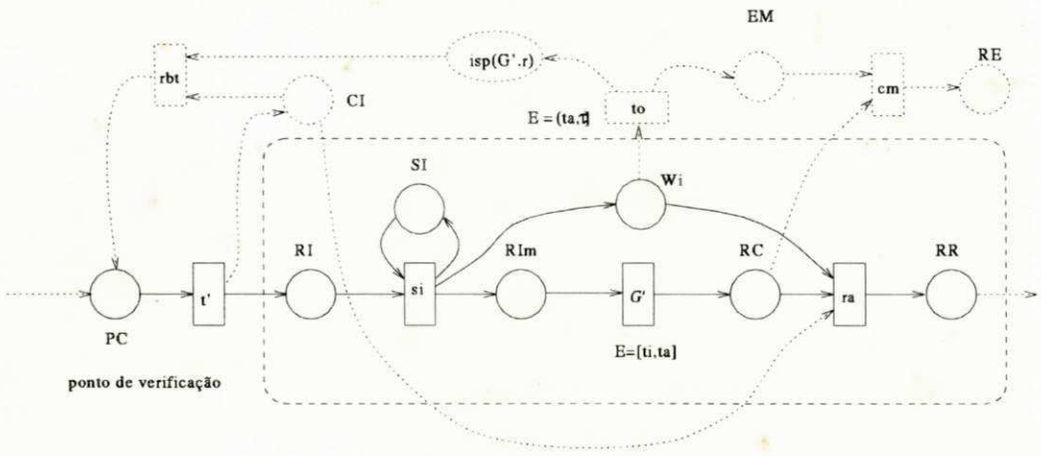

Figura 6.1: Esquema de recuperação regressiva temporizada

para terminar. Portanto, é necessário prover meios para medir o tempo decorrido entre uma invocação e o seu término.

As linhas sólidas na Figura 6.1 mostram um *isp* decomposto com restrições de tempo. A invocação da *FTG-Net* G', utilizando um método m, é representada pela transição rotulada *G'*. Uma restrição de tempo de disparo definida pelo intervalo $[ti, ta]$ é associado com a transição *G'* em que, ti é o tempo mínimo de execução e ta é o tempo máximo de execução associado com a invocação da rede G', utilizando o método m.

Para embutir o esquema de recuperação regressiva temporizada dentro de um *isp* e, conseqüentemente, em uma *FTG-Net*, a informação de estado do lugar *Wi* será usada da seguinte forma. Se a ficha no lugar *Wi*, indicando que a rede *G'* não terminou sua execução, permanece por um tempo maior do que o definido, assume-se que a rede *G'* falhou devido a uma falha de software ou hardware. Portanto, referindo-se a parte pontilhada da Figura 6.1 que corresponde ao esquema de recuperação regressiva temporizada, uma transição externa *to*, representando *time-out* disparará depois do tempo definido por t_a (pior caso). A transição *to* gatilhará duas diferentes ações: a primeira ação é modelada por *rbt* e corresponde a regressão da execução para o mais recente ponto de verificação consistente, neste caso correspondendo ao lugar *PC*. Para recomeçar a execução, a informação sobre o status no ponto de verificação é armazenada no lugar *CI*. Antes de regredir para o ponto de verificação indicado pelo lugar *PC*, é necessário recuperar a rede *G'* através de sua execução, utilizando um método de recuperação que é representado pelo *isp isp(G'.r)*. A segunda ação tem um duplo propósito. Primeiro, para manter uma memória de falhas no sistema fichas faltosas são depositadas no lugar *EM*. O lugar *EM* também tem a função de remover as possíveis

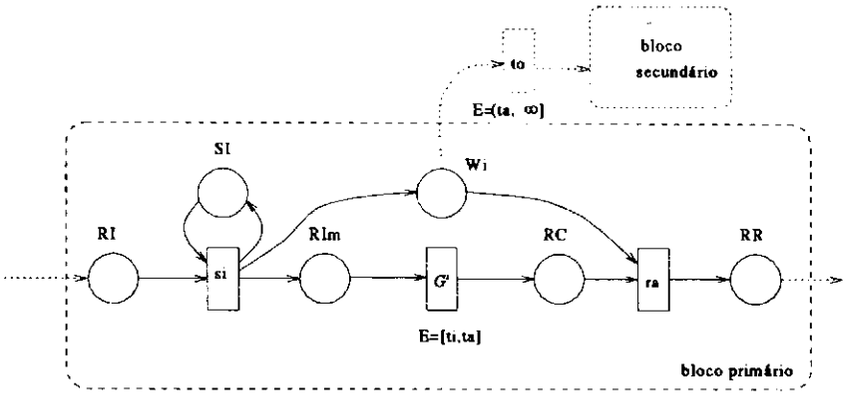

Figura 6.2: *isp* primário/secundário

fichas (mensagens) espúrias que poderiam ser retornadas pela rede faltosa *G'* e que seriam armazenadas no lugar *RE*. Isto é realizado pelo disparo da transição *cm*. Nós não detalhamos o que acontece com as fichas que não foram utilizadas nos lugares *EM* e *RE*, mas elas poderiam ser utilizadas para propósito estatístico, mantendo, portanto, uma história de falhas do sistema. Para obter esta informação, alguém poderia implementar um método para recuperá-la.

Em muitos casos, como apresentado na última seção, as técnicas de tolerância a falhas para trás não são adequadas para sistemas em tempo real. Entretanto, seguindo um procedimento semelhante, pode-se definir um esquema tolerante a falhas para frente baseado nas *FTG-Nets*. O *isp* decomposto de uma *FTG-Net*, modelando um esquema primário/secundário é mostrado na Figura 6.2. O funcionamento do lugar *Wi* é semelhante ao seu funcionamento no esquema de regressão temporizada. Neste caso, depois de decorrido um certo período de tempo (ta), um bloco secundário é acionado. O bloco secundário não é detalhado na figura.

6.3 Sistemas Distribuídos em Tempo Real

Um sistema distribuído é um sistema concorrente caracterizado pela descentralização de sua execução, i.e., um sistema distribuído é executado em um ambiente, consistindo de múltiplos nodos que estão em localizações geográficas diferentes [53]. Os sistemas distribuídos apresentam diversas vantagens: melhor disponibilidade, gerenciamento e controle mais localizados, expansão incremental do sistema, balanceamento de carga e tempo de resposta melhorado.

No projeto de um sistema distribuído em tempo real, este deve suportar diversas características. Entre essas características nós podemos incluir as seguintes:

1 - Distribuição: Esta é uma característica primordial para todo sistema distribuído. Isto significa que, em um sistema distribuído, os componentes podem ser executados cooperativamente em máquinas fisicamente diferentes.

2 - Concorrência: Tipicamente, um sistema concorrente apresenta muitas atividades ocorrendo em paralelo. Concorrência é importante porque reflete o paralelismo natural que existe no domínio do problema, em que diversas atividades acontecem simultaneamente. Esta característica é geralmente aplicável tanto para sistemas em tempo real como para aplicações distribuídas.

3 - Sincronização: É um mecanismo pelo qual o sistema pode modelar e tratar a necessidade de existência de pré-condições para a ocorrência de um evento.

4 - Restrições de Tempo Real: Isto é a base dos sistemas em tempo real. As restrições de tempo real devem ser consideradas no projeto e a metodologia de projeto deve ter mecanismos explícitos para verificá-las.

5 - Tolerância a falhas: Os aspectos de tolerância a falhas são da maior importância em sistemas críticos em tempo real. Portanto, no caso de SDTR, nós temos que prover mecanismos para tratar a ocorrência de falhas.

Devido às suas características, a computação orientada a objetos tem sido amplamente utilizada nos últimos anos. Muitos pesquisadores se concentraram nos modelos de objetos concorrentes que introduzem mecanismos de processamento concorrente dentro do paradigma orientado a objetos [1, 119]. Na verdade, a concorrência surge naturalmente no projeto orientado a objetos porque a análise é executada em termos de objetos autonômos. Além do mais, a metodologia orientada a objetos tem sido utilizada em sistemas em tempo real [15, 62].

Existem diferentes abordagens, utilizando o projeto orientado a objetos que podem ser aplicadas no projeto de SDTRs [24]. Entre estas abordagens nós citamos:

DRO (Distributed Real-Time Object) [117] é um modelo baseado em objetos que provê a facilidade de *melhor esforço* e *sofrimento mínimo*. *Melhor esforço* é definido como um comportamento de escalonamento no qual o servidor e o cliente tentam fazer o melhor para alcançar a operação requerida dentro de um tempo limitado. *Sofrimento mínimo* significa que o objeto cliente tenta fazer o melhor para sobreviver e minimizar suas perdas. O cliente tenta também prevenir a propagação de falhas temporais. *DRO* encapsula em si mesmo as restrições de tempo necessárias. As duas propriedades são

alcançadas pela definição de *invocação polimórfica no tempo* e *método restringido no tempo*. Na *invocação polimórfica no tempo*, restrições de tempo são especificadas para cada operação, i.e., a operação é executada se as restrições a elas associadas são satisfeitas. No *método restringido no tempo*, cada método declarado em um objeto distribuído em tempo real tem o pior tempo de execução como uma informação de tempo, e especifica uma rotina de tratamento de exceção para o caso que ela não seja satisfeita. A maior desvantagem de *DRO* é a falta de uma metodologia formal para analisar os objetos distribuídos em tempo real.

O modelo POT/POP [43] é um modelo de comportamento de um sistema paralelo baseado em objetos. Cada componente do sistema (objeto) é representado por muitos lugares na POT. A linguagem POP imita o comportamento da POT. Embora este modelo possa representar objetos comunicantes, ele é basicamente um modelo teórico e não pode ser facilmente usado nas reais especificações e prototipagens de sistemas distribuídos em tempo real. O modelo pode ser usado para analisar os sistemas distribuídos em tempo real mas características de tolerância a falhas não são consideradas.

PROTOB [9] é um sistema *CASE* com um conjunto de ferramentas integradas, para a especificação, modelagem, prototipagem e implementação de sistemas distribuídos usando um paradigma operacional de ciclo de vida de software. PROTOB suporta uma metodologia orientada a objetos semelhante a HOOD - Projeto Orientado a Objetos Hierárquicos [101] - que divide o sistema em uma hierarquia de objetos para melhorar sua compreensibilidade e para simplificar a modificação e reutilização de seus componentes. Cada objeto é representado por redes PROT que são redes de Petri de alto nível com a adição de um novo elemento, o *GATE* que é usado para enviar fichas para outras redes PROT. Este mecanismo para conexão de redes PROT, relativo à fusão de lugares, permite a preservação do paradigma orientado a objetos. Aspectos de tolerância a falhas não são considerados no modelo bem como as restrições de tempo real.

Outras abordagens que podem ser consideradas são definidas em [10, 15, 16, 52, 60, 102].

6.3.1 FTG-Nets e Invocação Polimórfica no Tempo

Como discutido na Seção 6.2, a aplicação das bem conhecidas técnicas de tolerância a falhas nos sistemas em tempo real apresentam uma série de limitações. Uma tendência natural na área de tolerância a falhas em sistemas em tempo real é a consideração antecipada de falhas e condições operacionais. Isto significa que, quando o sistema sai de seus limites, ele deve ser capaz de continuar funcionando dentro de suas espe-

cificações. Considerando os sistemas em tempo real, existem diferentes técnicas para garantir os requisitos tolerantes a falhas de sistemas e aplicações. Tais técnicas incluem *reconfiguração*, *degradação suave* e *load shedding*. A idéia básica da *reconfiguração* é a realocação em componentes que estão livres de falhas das aplicações que estão rodando em um nodo faltoso. Na técnica de *load shedding*, um conjunto de prioridades estáticas são definidas no sentido de determinar que aplicações serão perdidas no caso de falhas [97]. *Degradação suave* é definida quando os resultados obtidos são degradados e algumas vezes aceitos pelos usuários. A degradação dos resultados pode ser determinada pela redução dos tempos de execução ou pela execução de algumas alternativas de computação com requisitos menores [84].

Seguindo a tendência natural de antecipação das falhas, nós apresentamos uma técnica que emprega alguns conceitos relacionados com as técnicas de *load shedding* e *degradação suave*. A idéia foi adotada a partir da definição da invocação polimórfica no tempo definida em [117]. De acordo com [117], em um sistema distribuído em tempo real, o objeto invocado não pode estaticamente decidir suas restrições temporais, i.e., os objetos comunicantes devem tentar o melhor para alcançar as operações requeridas dentro de um tempo limitado. Para atingir isso, nós introduzimos o conceito de invocação polimórfica no tempo de *G-Net*.

Como apresentado na Seção 4, a invocação de uma *G-Net G'* é efetuada através do *isp*, utilizando um método definido. No caso da invocação polimórfica no tempo de *G'*, o *isp* utiliza um conjunto de métodos para invocá-la e restrições de tempo distintas são especificadas para cada método. Estas restrições de tempo serão usadas na rede invocada para determinar que método deverá ser executado. Os métodos têm uma ordem de prioridade que é determinada pelas restrições de tempo a eles associadas. Estas prioridades são importantes para garantir que o sistema funcione corretamente e que as restrições de tempo sejam cumpridas. Em muitos casos, os resultados, embora degradados, são aceitáveis.

Definição 6.2 *Em uma invocação polimórfica no tempo de G-Net, um isp invoca uma G-Net com um método composto* CM. $CM = \{< m1, Tc1 >, < m2, Tc2 >, ..., < mn, Tcn >\}$ *em que* $m1$, $m2$, ..., mn *são métodos e* $Tc1$, $Tc2$, ..., Tcn $(Tc1 < Tc2 < ... < Tcn)$ *são restrições de tempo associadas com o método correspondente.*

Em *G'*, o pior caso de tempo de execução para cada método é conhecido uma vez que ele pode ser determinado através da análise temporal como discutida nos capítulos anteriores. De acordo com as restrições de tempo associadas aos métodos e o pior caso de

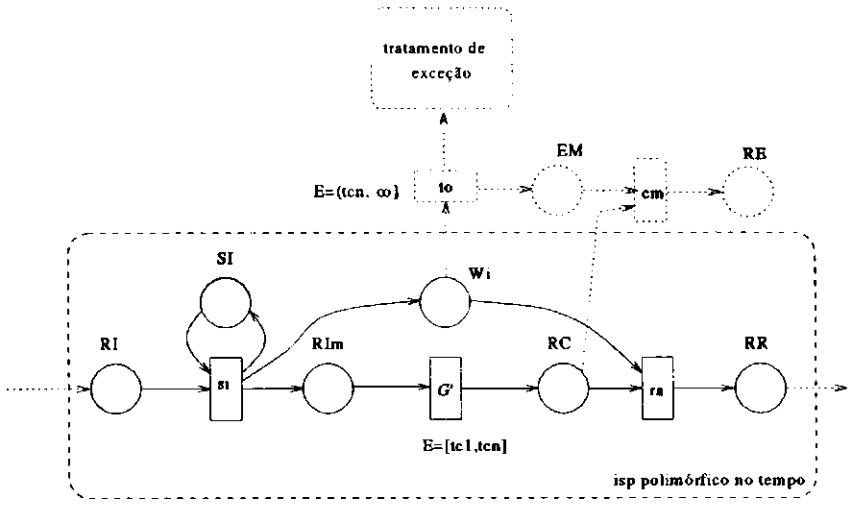

Figura 6.3: *isp* polimórfico no tempo

tempo de execução dos métodos, apenas um método é selecionado para ser executado. Nós não estamos considerando a carga do nó que está rodando *G'*. Neste caso, seria necessário ter mecanismos para especificá-lo. Por exemplo, nós poderíamos utilizar um falso lugar juntamente com uma ficha que carregaria uma FTF, especificando a carga do nodo. Quando a suposta ficha fosse combinada com as fichas iniciais, diferentes resultados (tempo de execução) seriam produzidos refletindo os resultados considerando a carga do nodo.

De acordo com a definição da invocação polimórfica de *G-Nets*, tanto o *isp* como o GSP invocado são responsáveis por manter o comportamento tolerante a falhas do sistema. Na Figura 6.3, nós mostramos o *isp* decomposto, considerando a invocação polimórfica no tempo de *G-Nets*. O *isp* polimórfico no tempo é semelhante ao *isp* decomposto tolerante a falhas. A diferença é que os intervalos de sensibilização associados com as transições são conhecidos a priori (isto segue a tendência de antecipação de falhas). *Tcn* é a restrição de tempo associada ao n-ésimo método, i.e., o método com mais baixa prioridade. Se esta restrição de tempo não pode ser cumprida, uma rotina de tratamento de exceção é gatilhada através do disparo da transição *to*. Os lugares *EM* e *RE*, e a transição *cm* tratam as fichas retornadas sem sucesso da invocação polimórfica no tempo de *G-Net*.

Uma vez que a invocação polimórfica no tempo de *G-Nets* é executada, utilizando diversos métodos, é necessário mecanismo para priorizá-los. Considerando o GSP decomposto da *G-Net* invocada *G'*, nós temos que estender a decomposição do GSP de modo que seja possível esta priorização. Uma vez que as restrições de tempo associadas

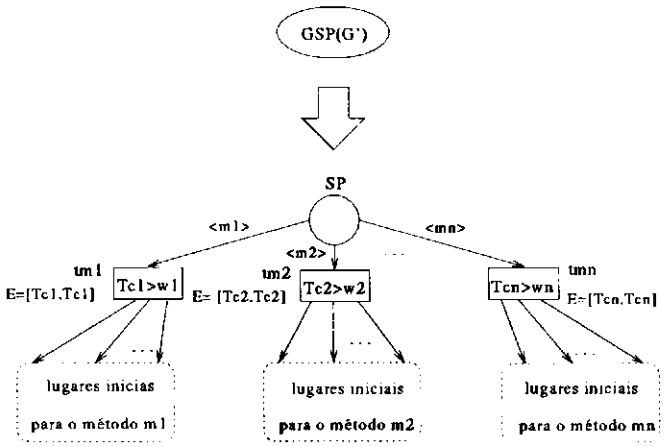

Figura 6.4: GSP polimórfico no tempo

com os métodos na invocação polimórfica são distintas, é possível usá-las para priorizar a execução dos métodos. Para tanto, nós utilizamos os intervalos de sensibilização como descrito no Capítulo 3, em que nós definimos como priorizar o disparo de transições sensibilizadas concorrentemente. A Figura 6.4 mostra o GSP decomposto polimórfico no tempo. Cada arco de saída do lugar inicial SP tem uma expressão que corresponde ao método especificado. No mais, as transições são aumentadas com inscrições para comparar as restrições de tempo com o pior caso de tempo de execução. Os intervalos de sensibilização associados com as transições propiciam um mecanismo de priorização para selecionar o método correto a ser executado. Por exemplo, vamos supor que a *G-Net* G' é polimorficamente invocada com métodos $< m1, Tc1 >$ e $< m2, Tc2 >$, em que $Tc1 < Tc2$. Então, se as inscrições das transições $m1$ e $m2$ são satisfeitas, ambas as transições são sensibilizadas. Uma vez que $Tc1 < Tc2$, o método $m1$ tem prioridade sobre o método $m2$ então, a transição $tm1$ é selecionada para disparar. Disparando a transição $tm1$, as fichas são depositadas nos lugares iniciais para o método $m1$ (representado pelas caixas pontilhadas na Figura 6.4) e o método $m2$ é desabilitado.

6.4 Exemplo: Controle Distribuído de Trens

Antes de apresentarmos claramente a descrição do problema, vamos introduzir alguns conceitos importantes relacionados com o controle de veículos de linha tais como o metrô. Neste tipo de sistema, uma linha é dividida em blocos os quais podem ser compostos de diferentes seções. A Figura 6.5, mostra um pedaço de um sistema de tráfego de linha que é caracterizado pela descentralização ou distribuição do controle dos blocos, i.e., cada bloco tem seu próprio controlador (CB) que pode se comunicar com

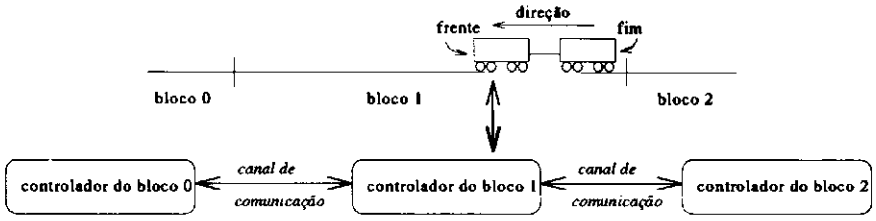

Figura 6.5: Controle Descentralizado de Blocos

os trens que estão trafegando nele. Portanto, *Controlador de Bloco 0 (CB0)* controla os trens no *bloco 0*, *Controlador de Bloco 1 (CB1)* controla os trens no *bloco 1* e *Controlador de Bloco 2 (CB2)* controla os trens no *bloco 3*. Além do mais, cada *CB* pode se comunicar com os *CBs* adjacentes. Considerando a Figura 6.5, *CB1* pode comunicar-se tanto com *CB0* como com o *CB2*.

Os blocos são projetados para garantir os requisitos de segurança e isto é conseguido garantindo que cada seção em um bloco não contém mais do que um trem trafegando no mesmo instante de tempo [23, 35, 46]. Nesta abordagem tradicional, cada trem tem um ponto de referência que é informado para o *CB* de diferentes formas, como por exemplo, através de sensores localizados no começo e fim das seções. Dependendo da direção do trem, a posição da frente (cabeça) e da cauda(fim) do trem são determinadas. Com o objetivo de garantir segurança, sinais são distribuídos em pontos estratégicos da linha. Cada trem tem um delimitador chamado de ponto de parada os quais, neste caso, são estáticos.

Com o objetivo de melhorar o desempenho, especialmente quando as seções são grandes, os trens deveriam ser capazes de rodar em diferentes velocidades no sentido de minimizar a distância entre eles. Obviamente, os trens devem rodar guardando uma posição de segurança entre sua posição e a posição do próximo trem para garantir que não ocorra colisão entre eles quando da ocorrência de alguma falha. Para tanto, esforços têm sido feito para melhorar a comunicação entre os trens e os *CBs* bem como para desenvolver mecanismos que garantam um comportamento seguro quando da detecção de uma falha [54].

Bloco em movimento – moving block (MB) é um bloco dinâmico definido com o objetivo de representar uma seção grande composta por seções de diferentes blocos adjacentes os quais permitem a presença de mais de um trem rodando nele em um mesmo instante de tempo. Neste caso, cada trem tem um ponto de parada que pode ser a posição do próximo trem, se não existir nenhum sinal entre eles, ou o próximo sinal. É interessante notar que os trens estão em movimento e a posição dos sinais

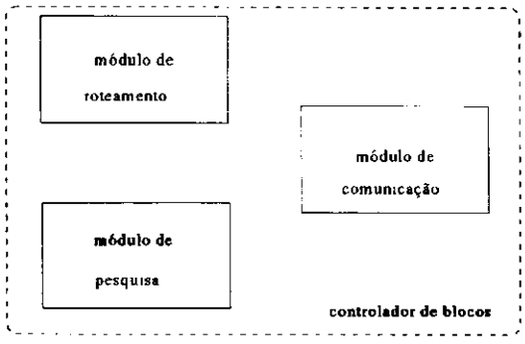

Figura 6.6: Estrutura do controlador de blocos

de parada mudam dinamicamente. No mais, os trens devem comunicar-se com seus respectivos controladores os quais podem mudar periodicamente. Logo, o primeiro requisito para um sistema de *MB* é um vital valor de posição provido por cada trem que está circulando na rede. O sistema deve apresentar características tolerantes a falhas para garantir um comportamento tolerante a falhas do sistema. Na seqüência, nós apresentamos os conceitos básicos de um *MB* e nós detalhamos a comunicação entre trens e *CBs*. Nós também usamos as *FTG-Nets* para introduzir tolerância a falhas em um sistema de *MBs*.

A vital posição do trem é determinada por sua interação com o correspondente *CB*. O trem, de acordo com seu ponto de referência, calcula a localização de sua cabeça e cauda para, periodicamente, informar ao *CB*. O trem ainda tem controle sobre sua velocidade, direção e outras informações como aceleração. Estas informações são importantes porque é fundamental saber se o trem pode parar com segurança, evitando uma colisão ou desastre.

Portanto, o funcionamento vital de um *MB* é determinado pela vital comunicação entre o *CB* e o trem. De um lado, cada *CB* tem seus próprios canais de comunicação. Além do mais, um *CB*, periodicamente, pesquisa todos os trens que estão sob o seu controle. Por outro lado, os trens interagem com o *CB* para saber informações importantes como a posição do sinal de parada.

Embora existam muitas propriedades relacionadas com o *CB* e o controlador do trem, neste exemplo nós nos restringimos à modelagem da comunicação entre o controlador do trem e o *CB*. De uma forma geral, o *CB* pode ser dividido em três módulos como mostrado na Figura 6.6. O módulo de roteamento é responsável por funções relacionadas com a composição de rotas, *interlocking* e controle dos alarmes de segurança. O módulo de pesquisa questiona, periodicamente, os trens para saber o real status de suas posições. Finalmente, o módulo de comunicação representa a comunicação entre *CBs* ou entre o

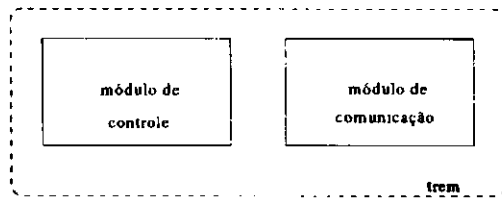

Figura 6.7: Estrutura do trem

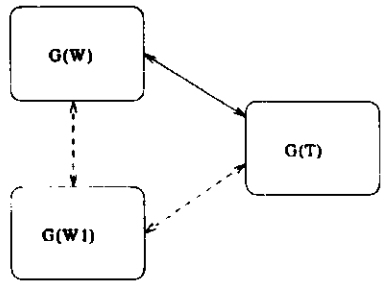

Figura 6.8: Relacionamento entre *G-Nets* no exemplo do controle de tráfego

controlador de blocos e o trem. O *CB* pode ser representado por um sistema de *G-Nets* em que, cada módulo é representado por uma *G-Net* que se comunica com outras *G-Nets* de uma forma bem definida.

Semelhantemente, nós podemos dividir o trem em dois módulos como delineado na Figura 6.7. O bloco de controle do trem é responsável por calcular os parâmetros de velocidade, funções de aceleração e parada, bem como indagar o módulo de comunicação para questionar o *CB* por novos dados. O módulo de comunicação coordena a comunicação entre o trem e o *CB*. O trem pode também ser representado por um sistema de *G-Nets* mas nós só detalharemos a *G-Net* que modela o módulo de comunicação.

Como descrito acima, a interação entre o controlador de blocos e o trem é efetuada através dos seus respectivos módulos de comunicação. O objetivo deste exemplo é mostrar como aplicar o conceito de *FTG-Nets* para introduzir tolerância a falhas no projeto de sistemas. Considerando o exemplo de controle ferroviário, nosso foco é na comunicação entre os trens e os controladores de blocos. Basicamente, nós consideramos um trem que está trafegando em uma direção pré-definida o qual requisita, do seu correspondente *CB*, informação sobre a posição dos seus sinais de parada. Cada componente é modelado por uma *G-Net* que interagem com outros componentes como mostrado na Figura 6.8. Primeiro, o trem modelado pela *G-Net* $G(T)$ requisita algumas informações ao seu correspondente *CB* $(G(W))$. Os dados são então retornados para o trem. Entretanto, em casos especiais, quando o trem está deixando o bloco e tem que ser controlado por um diferente *CB*, o *CB* primário deve requisitar ao próximo *CB* (re-

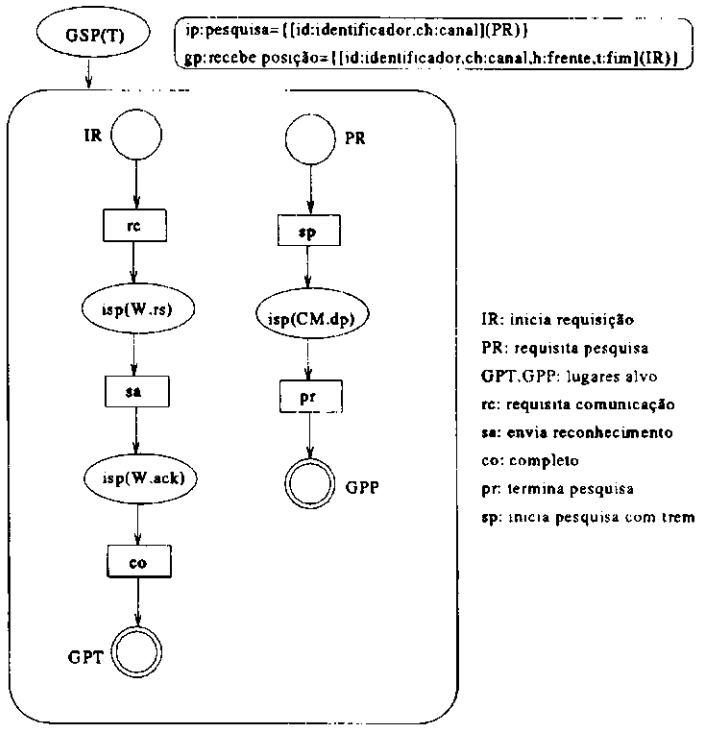

Figura 6.9: *G-Net* representando o módulo de comunicação do trem

presentado na Figura pela interação entre $G(W)$ e $G(W1)$) algumas informações para permitir a comunicação entre $G(T)$ e $G(W1)$. Após receber a informação sobre o novo canal de comunicação da *G-Net* $G(W1)$, $G(W)$ informa o novo canal para o trem $G(T)$. Então, o trem envia um reconhecimento para o seu novo *CB* e o processo continua, i.e., $G(T)$ se comunica com $G(W1)$. Na seqüência, nós apresentamos o modelo de *G-Net* para cada módulo de comunicação. Nós não detalhamos os outros módulos e eles são apenas referenciados quando usados.

A *G-Net*, $G(T)$, representando o módulo de comunicação de um trem é representado na Figura 6.9. $G(T)$ tem dois métodos: *gp* e *ip*. O método *gp* significa obter a posição do sinal de parada enquanto o método *ip* indica a solicitação de pesquisa. Se $G(T)$ é invocada com método *gp*, significa que o módulo controlador do trem, não discutido aqui, necessita atualizar a posição de seus sinais de parada e, conseqüentemente, ajustar adequadamente suas restrições. A invocação começa quando uma ficha é depositada no lugar *IR*, o que significa dizer que o módulo de controle necessita da informação sobre o sinal de parada. Após o disparo de *rc*, uma ficha alcança *isp(W.rs)* e a interação com o *CB* começa. Após receber a informação do *CB*, o trem envia um reconhecimento para o *CB*. Isto é representado por uma ficha em *isp(W.ack)*. A invocação termina quando uma ficha é depositada no lugar alvo *GPT*. Os resultados são enviados de volta ao módulo

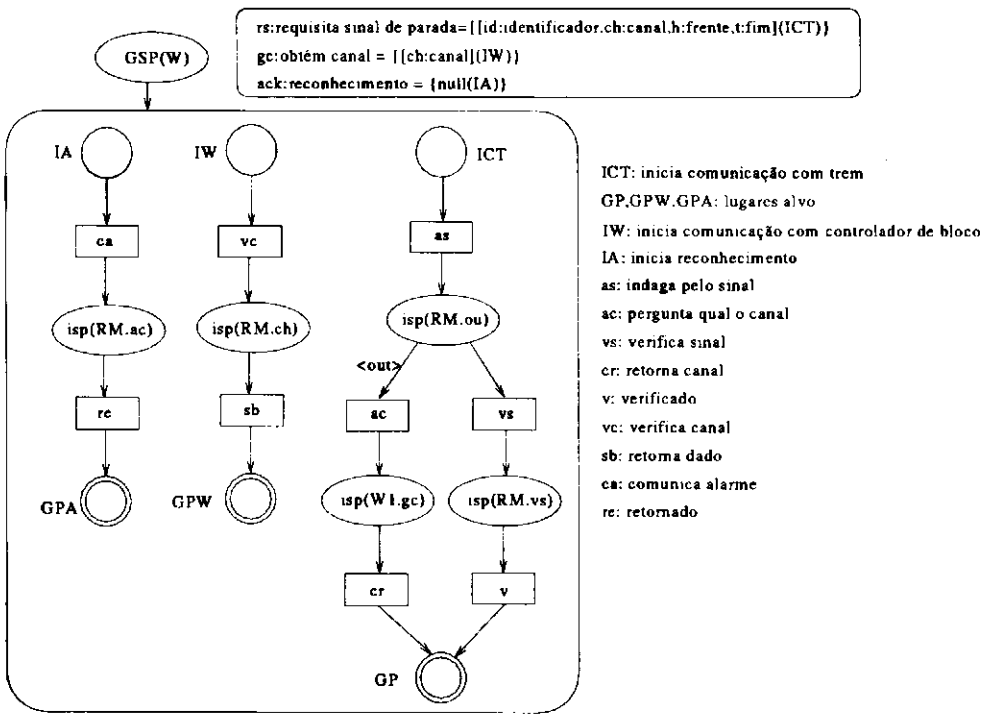

Figura 6.10: *G-Net* representando o módulo de comunicação do controlador de bloco

de controle do trem para computar novos dados ou executar algumas ações relevantes. Do contrário, se $G(T)$ é invocado com método *ip* pelo módulo de pesquisa do *CB* para se conhecer a posição corrente, uma ficha é depositada no lugar *PR* indicando que o *CB* está perguntando pela posição do trem. Após disparar *sp*, uma ficha é depositada em *isp(CM.dp)* representando a invocação do módulo de controle do trem com o método *define posição dp*. A posição corrente do trem é computada pelo módulo de controle. A posição do trem é retornada ao *CB* quando uma ficha alcança o lugar alvo *GPP*.

A *G-Net*, $G(W)$, modelando o módulo de comunicação do *CB* é mostrada na Figura 6.10. Existem três métodos para a rede: *gc*, *ac* e *rs*.

O método requisita sinal, *rs*, representa a operação de obter a nova posição do sinal de parada do trem que fez a requisição. Quando o trem invoca o *CB*, uma ficha é depositada no lugar *ICT*, carregando algumas informações tais como a posição da cabeça e da cauda. Após disparar a transição *as*, o módulo de roteamento do *CB* é invocado (ficha em lugar *isp(RM.ou)*) para informar se o trem permanece ou não sob sua supervisão, i.e., se a posição de cabeça do trem pertence ao bloco controlado por ele. Dependendo do resultado, duas diferentes ações podem ser gatilhadas. Se o trem permanece no mesmo bloco, o bloco de roteamento é invocado novamente para determinar os novos dados (disparo de *vs*). No outro caso, o trem está fora do bloco e

um novo canal de comunicação é requisitado (disparo da transição *ac*) para o próximo *CB* (veja explanação para o método *gc* abaixo). Esta requisição é para permitir a comunicação entre o trem e seu novo *CB*.

Em nosso modelo, nós estamos considerando que a seção é unidirecional e o próximo *CB* é pré-definido (representado no exemplo por *W*1). Para generalizar o modelo para uma seção bidirecional, nós temos que usar dois diferentes *isp*s, representando os próximos *CBs* à esquerda e à direita. A decisão é tomada de acordo com algumas informações sobre a direção do trem que pode ser carregada pelas fichas. Então, os dados providos pelo novo *CB* são retornados para o *CB* original que, por sua vez, retorna esses valores para o trem (ficha em lugar alvo *GP*). O módulo de comunicação do trem recebe como resultado, os novos valores (objetivo, sinal de parada, etc) e, em alguns casos, o número do novo canal de comunicação.

O método obtém canal, *gc*, representa a operação para determinar o novo canal de comunicação entre o trem e *CB*. Inicialmente, uma ficha é depositada no lugar *IW*. Então, o módulo de roteamento é invocado para saber o novo canal de comunicação, representado por uma ficha em *isp(RM.ch)*. Além do novo canal de comunicação, são também informados os novos valores dos dados para o trem. Quando o módulo de roteamento é invocado para requisitar o novo canal para comunicação com o trem que está começando a entrar no bloco, um alarme de interrupção é informado. Esta operação não é detalhada aqui porque ela é definida no bloco de roteamento. O objetivo desta mensagem de alarme é evitar acidentes devido a um trem perdido. Portanto, depois de decorrido um certo período de tempo sem reconhecimento do trem, o sistema é interrompido pelo *CB*.

O terceiro método, *ac*, representa o reconhecimento do trem. Neste caso, o módulo de roteamento é invocado novamente (ficha no lugar *isp(RM.ac)* para reiniciar o alarme. A operação termina quando uma ficha alcança o lugar alvo *GPA*.

Considerando que o bom funcionamento do sistema é determinado por uma vital comunicação entre o trem e o *CB*, é importante embutir propriedades de tolerância a falhas em alguns pontos do sistema. Um desses pontos está relacionado com a requisição dos novos objetivos e sinais de parada pelo trem. Portanto, se os dados requisitados não chegam em um certo período de tempo, um bloco tolerante a falhas deve ser acionado. O bloco tolerante a falhas se comunicará com o módulo de controle do trem e, uma decisão correta tem que ser executada para evitar um desastre. O bloco tolerante a falhas pode ser adicionado como discutido na Seção 6.2.1.

Para introduzir tolerância a falhas na comunicação entre o trem e o *CB*, nós temos

que primeiro efetuar a análise temporal do módulo de interação do *CB*, considerando a requisição do trem. Para fazer isso, nós decompomos as *G-Nets* como explicado no Capítulo 5 e nós consideramos o *isp* decomposto simplificado introduzido no último anterior.

Seguindo a abordagem apresentada no Capítulo 5, a Figura 6.11 mostra a *G-Net* decomposta, representando o módulo de comunicação do *CB*. A decomposição é representada pelos elementos dentro da parte pontilhada. A *G-Net*, representando o módulo de comunicação do *CB*, foi modificada introduzindo-se os correspondentes intervalos de tempo de atraso que, associados com as transições relacionadas à decomposição do *isp*, representam o tempo necessário para executar a rede invocada com um específico método. Então, o intervalo de tempo de atraso *DRM.ac* associado com a transição *RM.ac* especifica o melhor e pior tempo para executar a *G-Net* que representa o módulo de roteamento do *CB* com o método *ac*. *DRM.ch* associado com a transição *RM.ch* representa a execução da mesma *G-Net* com o método *ch*, e assim por diante. Nós nos referimos à *G-Net* decomposta como *G(Wd)*.

A análise temporal para o módulo de comunicação do *CB* pode ser efetuada para computar seu tempo de execução. Uma vez que nós não modelamos os módulos de rotina e pesquisa, vamos supor que o tempo de execução daqueles módulos são conhecidos. Esses valores são importantes para computar o tempo de execução do módulo de comunicação porque ele invoca o módulo de roteamento para executar algumas funções. Portanto, aplicando a análise de alcançabilidade nebulosa nós podemos construir o grafo de alcançabilidade nebulosa para a *G-Net* decomposta *G(Wd)*.

Na Figura 6.12 é mostrado o grafo de alcançabilidade para a *G-Net G(Wd)* considerando os métodos *gc*, *ms* e *ack*. A partir do grafo de alcançabilidade nebuloso da Figura 6.12 e, considerando que o símbolo $\oslash$ representa a função definida na Seção 3 que combina a função de tempo nebulosa das fichas e os intervalos de tempo, as seguintes equações podem ser computadas[2] para o método *gc*:

$$FIW = FSP \oslash Dmgc. \tag{6.1}$$

$$FCH1 = FIW \oslash Dvc. \tag{6.2}$$

[2] Nós estamos considerando a mesma notação do último capítulo, i.e., *FIW* representa a função nebulosa de tempo da ficha no lugar *IW*, *FSP* a *FTF* do lugar *SP*, e assim por diante.

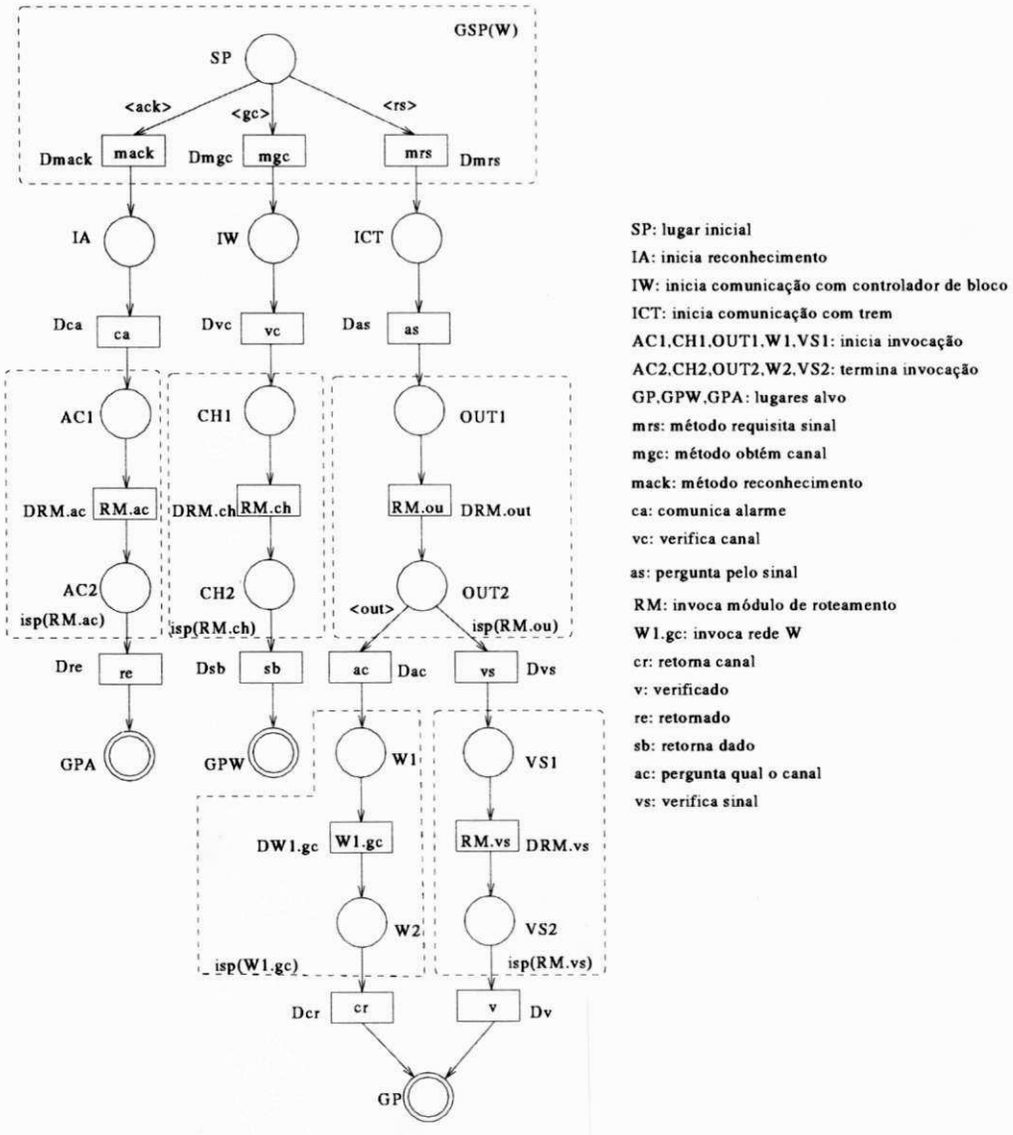

Figura 6.11: *G-Net* G(Wd) para a análise temporal do módulo de comunicação do controlador de bloco

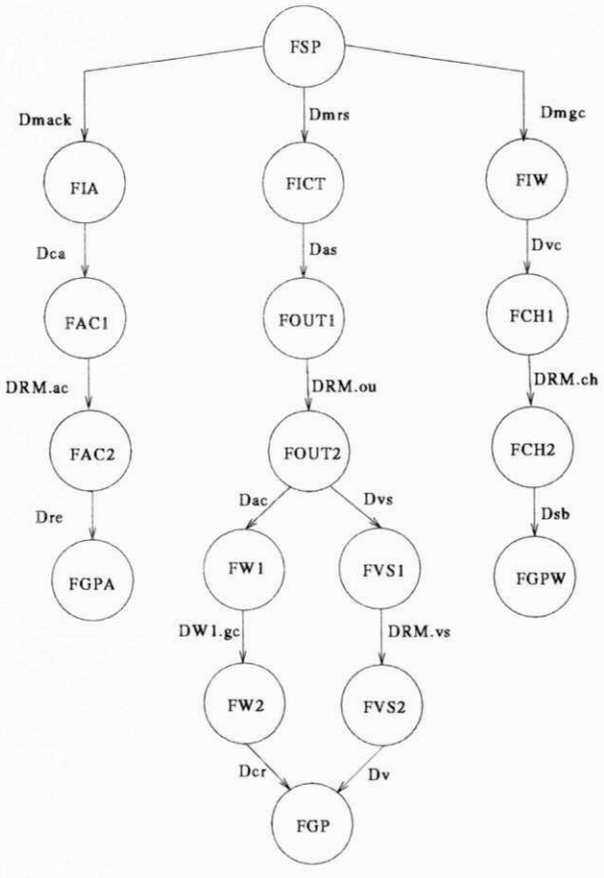

Figura 6.12: *FRG* para a *G-Net GP(W)*

$$FCH2 = FCH1 \oslash DRM.ch. \tag{6.3}$$

$$FGPW = FCH2 \oslash Dsb. \tag{6.4}$$

De forma semelhante, nós temos as seguintes equações para o método rs:

$$FICT = FSP \oslash Dmrs. \tag{6.5}$$

$$FOUT1 = FICT \oslash Das. \tag{6.6}$$

$$FOUT2 = FOUT1 \oslash DRM.ou. \tag{6.7}$$

$$FW1 = FOUT2 \oslash Dac. \tag{6.8}$$

$$FVS1 = FOUT2 \oslash Dvs. \tag{6.9}$$

$$FW2 = FW1 \oslash DW1.gc. \tag{6.10}$$

$$FVS2 = FVS1 \oslash DRM.vs. \tag{6.11}$$

$$FGP = FW2 \oslash Dcr = FVS2 \oslash Dv. \tag{6.12}$$

A análise da execução da *G-Net* $G(W)$, considerando o método rs, depende da análise da *G-Net* modelando o *CB* quando executando o método *gc*. Isto ocorre porque quando $G(W)$ executa o método rs, ela pode ou não invocar $G(W1)$ com o método *gc*. Uma vez que nós estamos interessados no pior caso de tempo de execução, nós consideraremos a execução de $G(W)$ quando ela invoca $G(W1)$, i.e., quando a posição da cabeça do trem está fora do bloco. Para fazer isso, nós computamos a função nebulosa de tempo associada à ficha no lugar alvo, supondo que as FTFs iniciais são dadas pelo intervalo $[0, 0]$. Primeiro nós temos que analisar a *G-Net*, representando o *CB* com o método *gc*, e usar os resultados para computar a solução final. A partir da Equação 6.1 e supondo que as fichas no lugar *SP* carregam uma FTF definida como $FSP = [0, 0]$, FIW é computada como:

$$FIW = Dmgc. \tag{6.13}$$

Aplicando o valor computado FIW e utilizando o mesmo procedimento, a partir das Equações 6.2, 6.3 e 6.4, o tempo de execução para este método é dado por:

$$FGPW = Dmgc \oslash Dvc \oslash DRM.ch \oslash Dsb. \tag{6.14}$$

Para analisar $G(W)$ com o método rs, o resultado obtido na Equação 6.14 substitui-se $DW1.gc$ na Equação 6.10. Aplicando-se a mesma abordagem, o tempo de execução final para a invocação do *CB* é a função nebulosa de tempo no lugar alvo *FGP*:

$$FGP = Dmrs \oslash Das \oslash DRM.ou \oslash Dac \oslash FGPW. \tag{6.15}$$

Vamos considerar que os limites inferior e superior para FGP sejam $d1$ e $d2$, respectivamente. Isto é,

$$FGP = Dmrs \oslash Das \oslash DRM.ou \oslash Dac \oslash FGPW = [d1, d2]. \quad (6.16)$$

A Figura 6.13 mostra a *G-Net* modelando o módulo de comunicação para o trem aumentado com capacidade de tolerância a falhas. A expansão é representada por um bloco tolerante a falhas representado na figura pelas linhas pontilhadas. O bloco tolerante a falhas é gatilhado quando a *G-Net* $G(W)$ é invocada e o resultado não é retornado, no pior caso, em um período de tempo igual ao limite superior $d2$ do pior caso de tempo de execução computado acima. Isto é representado pelos intervalos de tempo de sensibilização associados com as transições *W.rs* e *ift*. O bloco tolerante a falhas pode ser composto por um *isp* que chamará o módulo de controle de trem com um método *ad*. Este método define algumas operações especiais para serem executadas no sentido de evitar qualquer tipo de desastre.

Uma abordagem semelhante pode ser usada para inserir capacidade de tolerância a flahas em diferentes pontos da comunicação. Um exemplo típico seria relacionado à interrupção do sistema que é gatilhado quando decorre um certo período de tempo após o alarme ter comunicado a presença de um novo trem no bloco. A falta de comunicação pode caracterizar a presença de um trem perdido.

6.4.1 Resultados Numéricos

Nesta seção, o controle distribuído de veículos é estendido incorporando valores numéricos para os intervalos de atraso. Novamente, por uma questão de simplicidade, nós não consideraremos os intervalos de sensibilização. O algoritmo implementado é usado para automaticamente gerar os valores das FTFs. A Figura 6.14 mostra a *G-Net* $G(W)$ modelando o *CB* com valores numéricos de intervalos de atraso associados com as transições. Sem perda de generalidade, nós estamos considerando que não existe atraso na invocação da rede. Isto é representado pelo intervalo de atraso $D = [0, 0]$ associado com as transições no GSP decomposto, i.e., as transições *mack*, *mgc* e *mrs*. Uma vez que nós estamos abstraindo alguns módulos do *CB* (por exemplo o módulo de roteamento RM), vamos supor que o tempo de execução para estes módulosl são conhecidos. Logo, na Figura 6.14, o tempo de execução para a rede representando o módulo de roteamento do *CB*, usando método *ch* é dado por um valor entre 4 e 8. Isto é representado na figura pelo intervalo de tempo de atraso $D = [4, 8]$ associado com as transições *RM.ch* do *isp* decomposto *isp(RM.ch)*. É importante notar que os valores numéricos do intervalo de atraso associado com *isp(W1.gc)* não são expressados pois

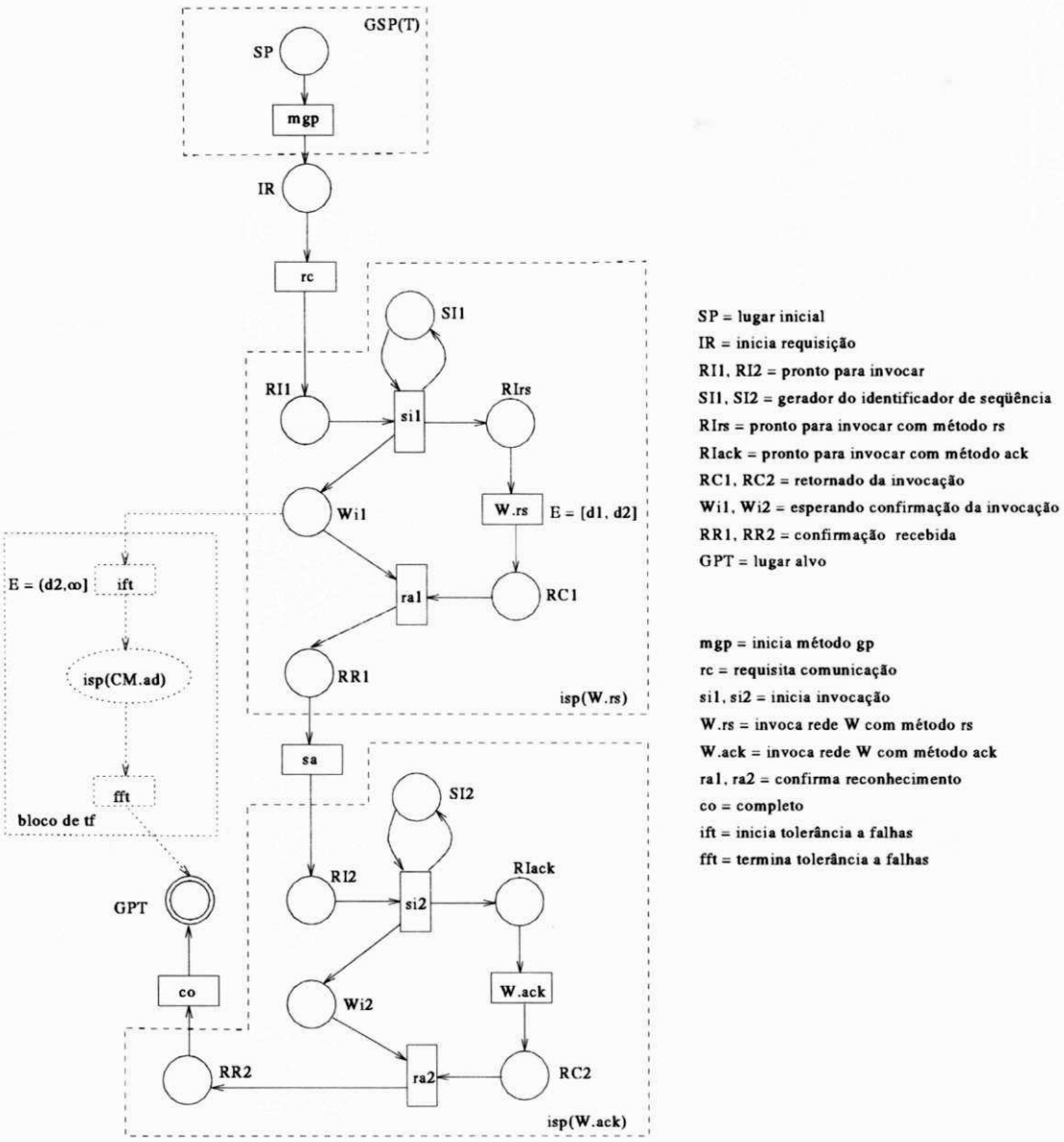

Figura 6.13: Módulo de comunicação do trem considerando tolerância a falhas

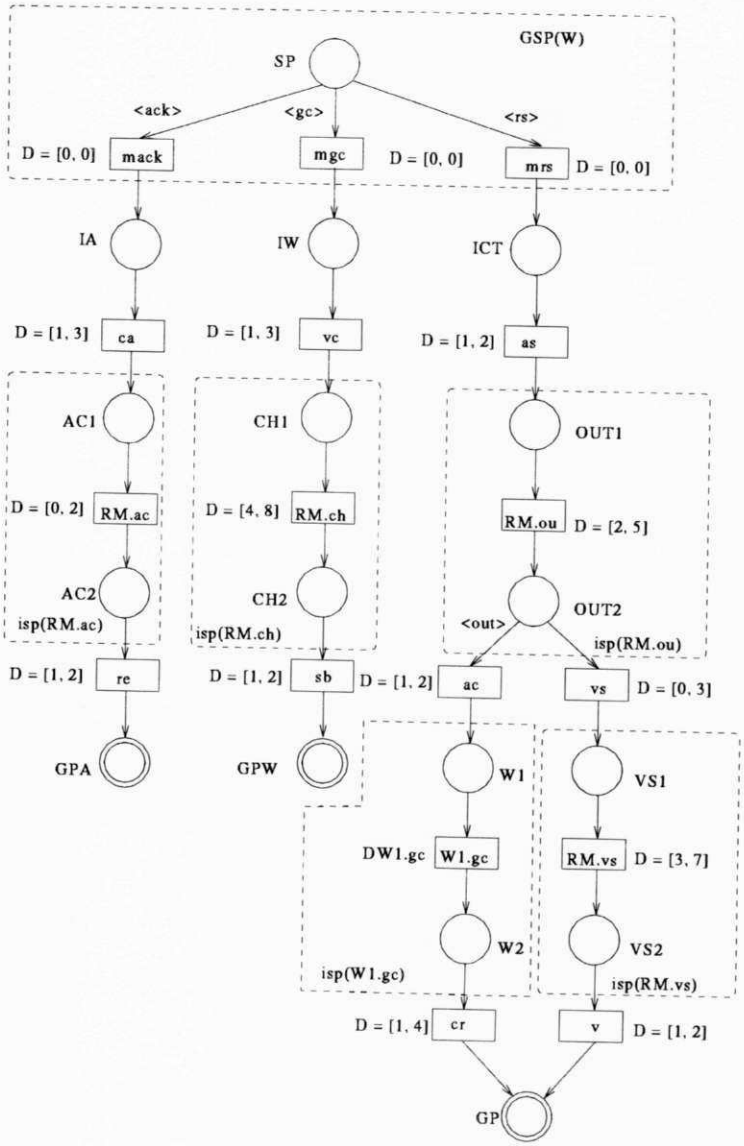

Figura 6.14: *G-Net* $G(W)$ com valores numéricos

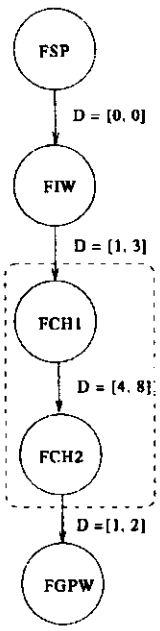

Figura 6.15: *FRG* para o método *gc* incluindo valores numéricos

seus valores serão determinados através da análise da *G-Net* $G(W)$ utilizando o método *gc*.

Na Figura 6.15, é apresentado o grafo de alcançabilidade nebuloso para a *G-Net* $G(W)$ considerando os valores numéricos. Então, aplicando-se o algoritmo implementado descrito no capítulo anterior, o tempo de execução é computado. A cópia da tela para a computação do tempo de execução da *G-Net* $G(W)$, considerando o método *ms* é desenhada na Figura 6.16. A computação começa chamando a rotina de análise temporal, considerando a rede representando o *CB* e respectivo método, i.e., *G-Net* $G(W)$ com o método *gc*. Isto é representado pela tela superior do lado esquerdo (tela *console*) na Figura 6.16, onde *Wayside.mgc* indica a rede e o método a ser analisado. A rede completa e o *FRG* nebuloso também são mostrados, telas *Wayside_Net* e *Wayside.mgc*, respectivamente. Por fim, os resultados computados são mostrados na tela de resultados de análise de tempo (tela *main*). De acordo com os valores de atraso, o tempo de execução da *G-Net* $G(W)$ com método *gc* está entre 6 e 13. Estes valores correspondem à FTF carregada pela ficha no lugar alvo GPW.

Depois de computar o tempo de execução, os resultados são embutidos no *isp* decomposto. Por brevidade, nós não mostraremos novamente a rede completa com os valores embutidos. A Figura 6.17 mostra o *FRG* da *G-Net* $G(W)$, considerando o tempo de execução computado acima. Então, aplicando-se o algoritmo implementado, o resultado final é computado.

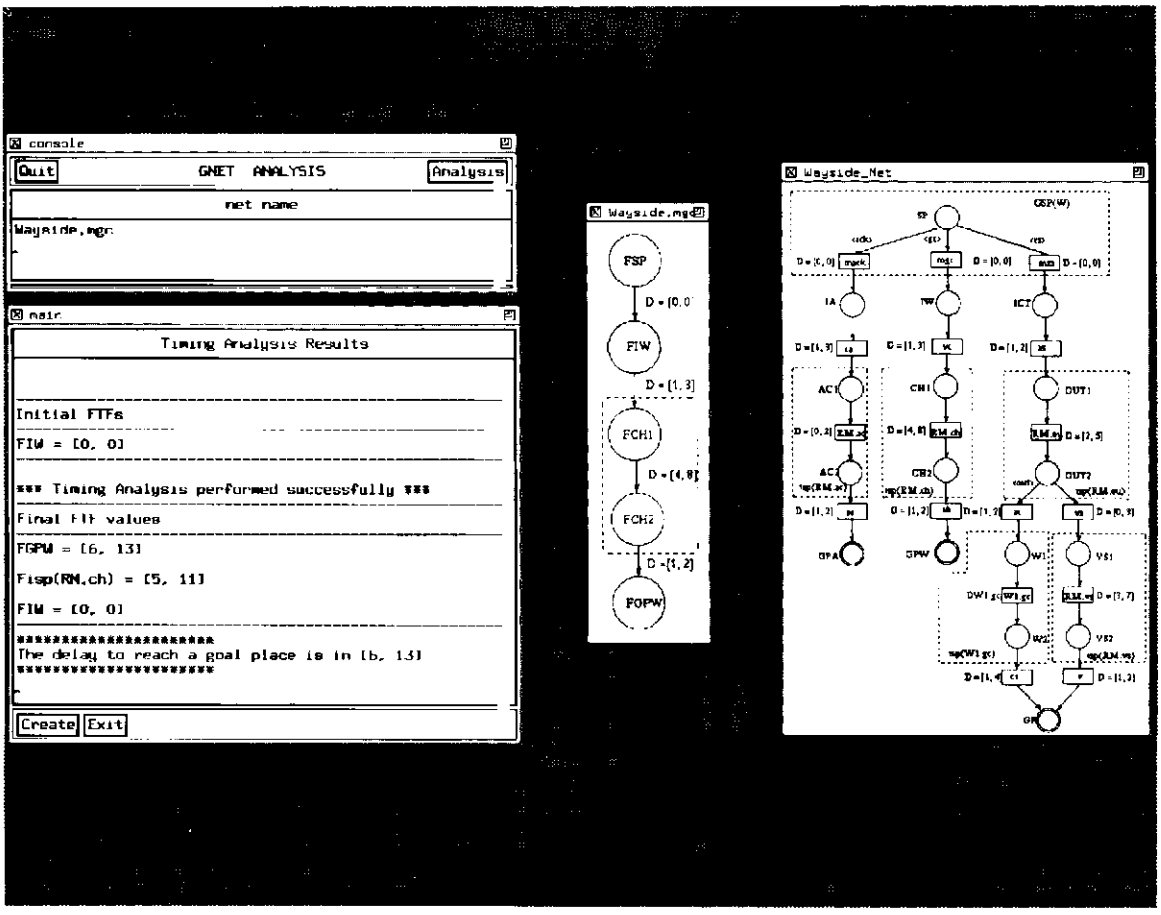

Figura 6.16: Cópia da tela para a análise temporal de $G(W)$ com o método gc

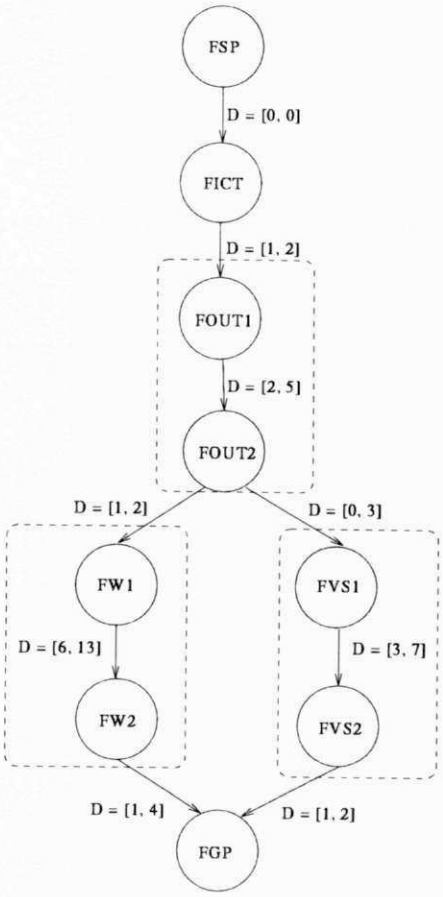

Figura 6.17: *FRG* para o método rs incluindo valores numéricos

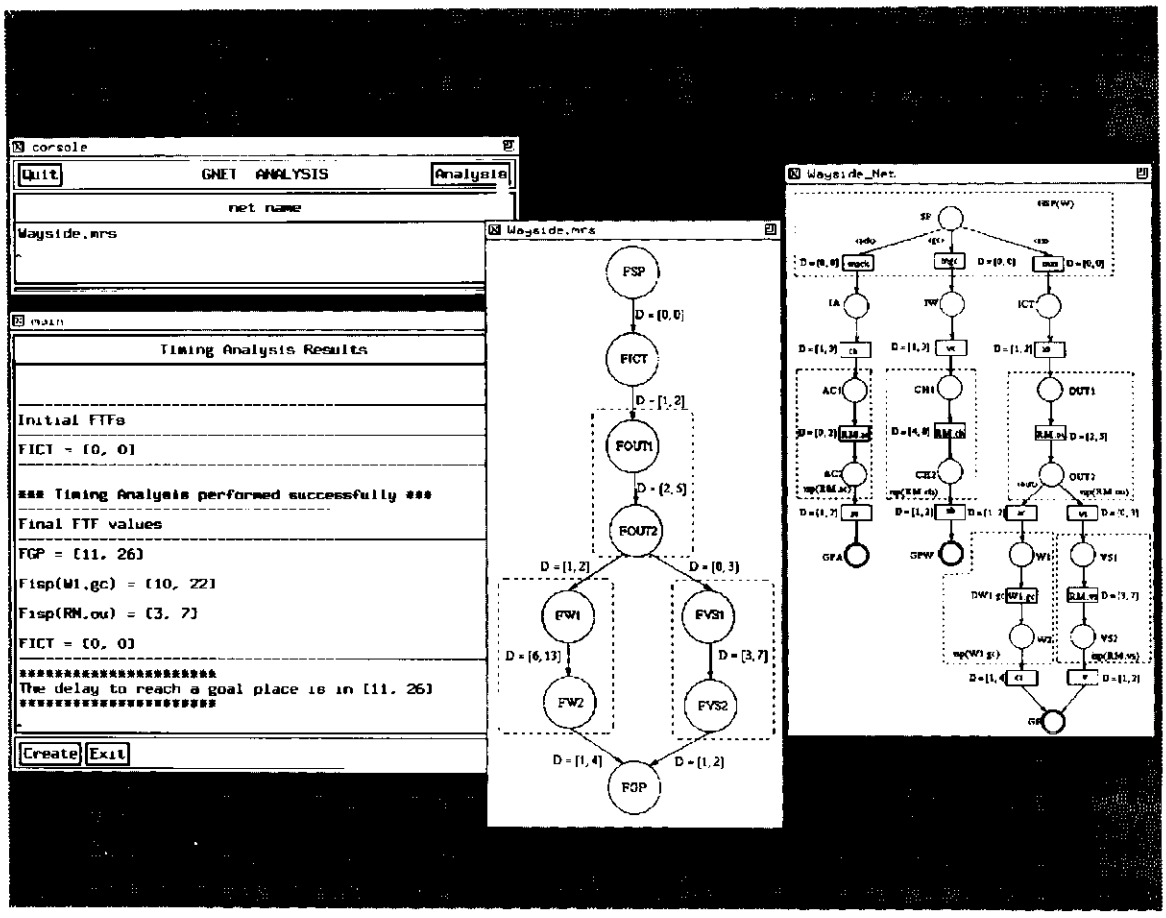

Figura 6.18: Cópia da tela para a análise de $G(W)$ com o método rs

Semelhantemente, a cópia da tela desta computação é apresentada na Figura 6.18. Primeiro, a rotina de análise de tempo é aplicada para a *G-Net* $G(W)$ com método rs que é representado pelo comando *Wayside.mrs* na tela *console*. A tela *Wayside_Net* e *Wayside.mrs* apresentam a *G-Net* com valores numéricos e seu correspondente *FRG*, respectivamente. Os resultados da análise temporal são, então, apresentados indicando que o tempo de execução final está entre 11 e 26.

Na Figura 6.19, a cópia da tela para a análise gráfica da rede $G(W)$ é mostrada. A partir do *TGIG* mostrado na tela *Wayside.cht*, nós podemos observar que existem dois estados de execução e dois estados de invocação. Entre tempos 0 e 1 a *G-Net* $G(W)$ está em estado de execução com método rs. A *G-Net* $G(RM)$ está em estado de execução com método *ou* entre os tempos 2 e 3. Por outro lado, a *G-Net* $G(W)$ está em estado de invocação entre os tempos 2 e 3, e entre 9 e 10.

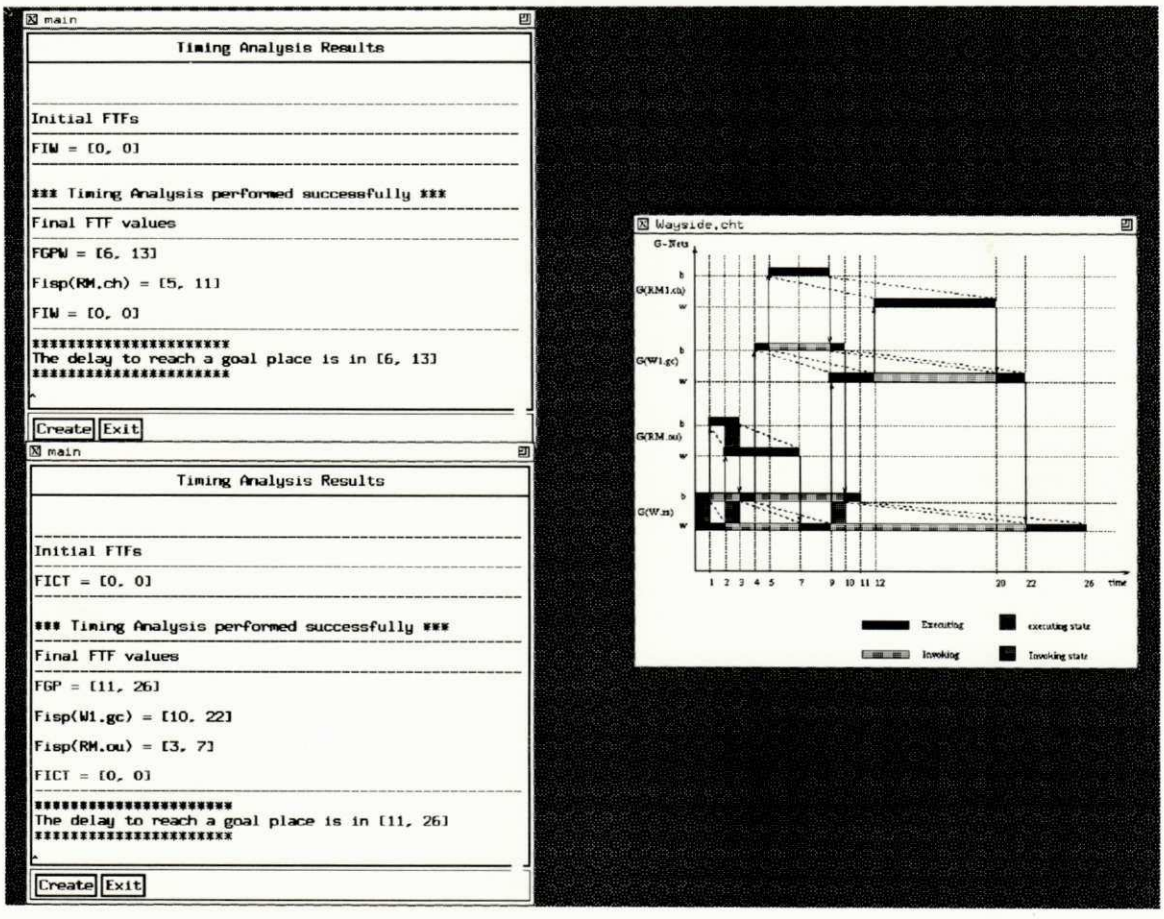

Figura 6.19: Cópia da tela mostrando o *TGIG* para a *G-Net* $G(W)$

Capítulo 7
CONCLUSÃO

7.1 Sumário

Nos capítulos anteriores, nós introduzimos uma abordagem nebulosa para incorporar tempo em redes de Petri. A principal motivação desta abordagem é a falta de generalidade das extensões temporais de redes de Petri. A introdução de características de tempo no modelo de redes de Petri foi alvo de muitas pesquisas e pode ser dividida em duas categorias: determinística e estocástica. Nós revisamos as mais importantes extensões determinísticas e estocásticas antes de apresentarmos nossa extensão, chamada de *Rede de Petri com Temporização Nebulosa (FTPN).*

O objetivo do modelo *FTPN* é integrar, em uma forma complementar, os aspectos positivos das extensões determinísticas e estocásticas. As *FTPN*s são caracterizadas por dois tipos de intervalos nebulosos de tempo (de sensibilização e de disparo) associados às transições e pela função nebulosa de tempo, FTF, carregada pelas fichas. O intervalo de sensibilização representa os tempos mínimo e máximo que devem decorrer entre a sensibilização da transição e o seu disparo. O intervalo de disparo representa o atraso necessário para executar a ação representada pela correspondente transição. Estes dois tipos de intervalos nebulosos são importantes para modelar sistemas em tempo real em que nós temos que representar tempos mínimos, tempos máximos e restrições de duração. A caracterização nebulosa dos intervalos permite a computação de índices de desempenho agregado. A função nebulosa de tempo carregada pelas fichas representa a possibilidade de existir uma ficha em um determinado lugar em um dado instante de tempo. A função nebulosa de tempo das fichas e os intervalos nebulosos de tempo associados às transições determinam o comportamento dinâmico de uma *FTPN.* Nós também apresentamos as regras para a computação das FTFs associadas com as fichas criadas durante a execução da rede.

Do ponto de vista de análise, o modelo *FTPN* é também interessante. Nós estendemos o conceito de grafo de alcançabilidade, incorporando aspectos nebulosos de tempo do modelo *FTPN* com o objetivo de efetuar análise temporal. Nós também definimos uma técnica de análise baseada na técnica de análise do grafo de alcançabilidade. O grafo

de alcançabilidade modificado, chamado de grafo nebuloso de alcançabilidade, permite a computação de diferentes tipos de índices de desempenho. Entre eles, nós podemos citar alguns índices individuais[1], índices agregados[2] e condições de alcançabilidade[3]. Nós também definimos e implementamos um algoritmo de análise temporal que computa todas as $FTFs$ automaticamente. A complexidade deste algoritmo foi determinada como sendo $O(p \times S^2)$, em que p é o número de lugares e S o número de estados.

Para tratar sistemas grandes complexos, nós integramos o modelo *FTPN* com *G-Nets*. A integração permite uma análise temporal modular de sistemas complexos. Isso é conseguido, considerando a estrutura interna de uma *G-Net* como sendo uma *FTPN* e utilizando uma abordagem de decomposição. Logo, nós podemos dividir um sistema em subsistemas, estudá-los em separado e integrar os resultados para a obtenção da solução global. Nós também definimos um gráfico de visualização temporal que mostra os aspectos temporais e de interação das *FTG-Nets* em um sistema de *G-Nets*. Este gráfico também permite a computação de alguns índices de desempenho.

As *FTG-Nets* podem ser usadas em diferentes tipos de aplicações. Nós mostramos a aplicação da ferramenta integrada nas áreas de tolerância a falhas e sistemas distribuídos em tempo real. Considerando a área de tolerância a falhas, nós representamos alguns esquemas tolerantes a falhas através das *FTG-Nets*, e nós definimos uma abordagem para representar a tendência natural de considerar as falhas por antecipação. Por outro lado, nós mostramos a adequabilidade das *FTG-nets* no projeto de sistemas distribuídos em tempo real, uma vez que, características como distribuição, concorrência, sincronização, restrições em tempo real e tolerância a falhas podem ser facilmente representadas.

Comparando o modelo *FTPN* com as extensões determinísticas e estocásticas, nós podemos fazer algumas conclusões. As extensões determinísticas que utilizam um atraso fixo, como por exemplo *TdPN*, não são muito expressivas porque não é possível representar durações variáveis. *TPN*, *TB nets* e *ITCPN* utilizam um intervalo ao invés de atrasos fixos mas elas são restritas quando utilizadas para efetuar análise de desempenho. No modelo *FTPN*, é possivel avaliar desempenho de sistemas uma vez que a imprecisão caracterizada pelos intervalos nebulosos de tempo é quantificada através de graus de possibilidade. Com as *FTPNs* também é possível modelar, de forma elegante, *time-outs* e representar atrasos de processamento. Além do mais, as *FTPNs* diferem das extensões

[1] Por exemplo, o tempo de atraso mínimo, máximo e mais provável para atingir um determinado estado.

[2] Por exemplo, a possibilidade de um determinado estado.

[3] De acordo com os intervalos nebulosos de tempo, determinar se um determinado estado é alcançável.

determinísticas porque neste último caso, as características temporais são modeladas explicitamente associando restrições de tempo aos lugares ou transições. Modelando as restrições de tempo explicitamente, perde-se alguma habilidade de expressar eventos e condições no futuro usando apenas o estado inicial. Também, no caso das extensões determinísticas, é muito difícil modelar casos em que o tempo de disparo depende da característica temporal de duas ou mais condições distintas. Logo, a função associada às fichas permite a análise do sistema, considerando diferentes circunstâncias sem ter que mudar a descrição da rede, i.e., a análise pode representar diferentes situações as quais podem ser determinadas pelas FTFs iniciais.

As redes de Petri estocásticas são adequadas para fazer avaliação de desempenho de sistemas mas apresentam três principais limitações: elas não são apropriadas para a modelagem e computação de características de desempenho de sistemas em tempo real, dificuldade de derivar uma exata distribuição de probabilidade e como avaliar desempenho de sistemas complexos, i.e., dividir o modelo em submodelos para serem estudados isoladamente e depois combinar os resultados. No caso da *FTPN*, a avaliação de desempenho de sistemas pode ser computada a partir das funções nebulosas de tempo que são carregadas pelas fichas e os sistemas em tempo real podem ser modelados utilizando os intervalos de tempo associados com as transições. O sistema é dividido em subsistemas, estudados isoladamente, os resultados então combinados para chegar a uma solução final para o modelo.

Considerando as extensões nebulosas para redes de Petri, o modelo *FTPN* proposto difere dos tipos de redes de Petri nebulosas que já foram apresentados. Os modelos nebulosos propostos por Looney, Chen e Cao não levam em consideração o tempo. Eles descrevem como fazer um raciocínio nebuloso. O único trabalho que usa aspectos nebulosos e tempo é o trabalho proposto por Cardoso e Valette no qual eles tratam marcações imprecisas. Tempo neste caso é considerado quando a existência de uma ficha é conhecida mas sua localização não é precisa. Em nosso trabalho, o tempo nebuloso é considerado para fazer avaliação de desempenho de sistemas, i.e., a localização da ficha é precisa mas faz-se uma estimativa da presença da ficha no lugar. Baseado nesses valores, alguns aspectos temporais são considerados.

7.2 Trabalhos Futuros

O trabalho que foi apresentado nesta Tese pode ser estendido nos aspectos práticos e teóricos.

Embora as *FTPNs* possam ser utilizadas com sucesso para resolver diversos proble-

mas, existem alguns criticismos que nós discutiremos nessa Seção. Direções para futuras pesquisas teóricas estão concentradas nas soluções para resolver estes aspectos críticos com que nos deparamos quando usamos *FTPN*.

Uma das principais limitações da abordagem estocástica é como caracterizar as variáveis aleatórias para representar a imprecisão e incerteza dos dados de tempo quando não existem dados históricos sobre o sistema. Embora, no interminável debate contra os *probabilistas*, os *fuzicistas* defenderem a idéia de que é mais fácil definir variáveis nebulosas do que variáveis aleatórias na modelagem de incerteza quando do projeto de um novo sistema, o primeiro aspecto crítico a ser considerado é como definir a caracterização nebulosa dos intervalos de tempo e das funções de tempo carregadas pelas fichas da marcação inicial.

O segundo aspecto é como tratar com a dependência entre fichas. Isso acontece quando o disparo de uma transição gera fichas que, em um futuro, serão fichas de entrada para o disparo de uma outra transição. Dependendo dos intervalos nebulosos de tempo associados com as transições intermediárias, resultados errôneos podem ser obtidos.

Por fim, uma vez que a computação das funções nebulosas de tempo é baseada na aritmética de números nebulosos, alguns problemas podem surgir. Isso ocorre porque não podemos garantir que a operação entre dois números nebulosos resultará em um número nebuloso, i.e., um subconjunto nebuloso que seja normal e convexo. Nesse caso, a condição de normalidade não é satisfeita e, para se usar o resultado da operação, nós temos que normalizá-lo. A normalização do resultado pode aumentar a imprecisão nos resultados finais. Uma possível solução para o problema da normalização seria a utilização de quantidades nebulosas ao invés de números nebulosos para servir como base para a computação das funções nebulosas de tempo. O conceito de quantidade nebulosa é mais geral do que o conceito de números nebulosos uma vez que são definidos como subconjuntos nebulosos e, não existe nenhuma restrição relacionada com os aspectos de normalidade e convexidade.

Considerando os aspectos práticos, um melhoramento seria a extensão do algoritmo de análise temporal para suportar valores nebulosos ao invés apenas de valores *crisp*. Esta extensão permite a computação de índices de desempenho, considerando-se diferentes níveis de presunção. Uma outra direção é a geração automática do $TGIG$. O desenvolvimento de um ambiente para análise temporal constituído por dois métodos de análise (FRG e $TGIG$) resultaria em uma poderosa ferramenta para computar índices de desempenho.

BIBLIOGRAFIA

[1] G.A. Agha. Actors: A Model of Concurrent Computation in Distributed Systems. Relatório Técnico 844, M.I.T., Cambridge, MA, 1985.

[2] A.V. Aho, J.E. Hopcroft, e J.D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[3] M. Ajmone Marsan. Stochastic Petri Nets: An Elementary Introduction. Em *Advances in Petri Nets 1989*, volume 424 of *Lecture Notes in Computer Science*. Springer-Verlag, 1989.

[4] M. Ajmone Marsan, D. Balbo, e 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, Maio 1984.

[5] T. Anderson e J.C. Knight. A Framework for Software Fault Tolerance in Real-Time Systems. *IEEE Transactions on Software Engineering*, SE-9(3):355–364, Maio 1983.

[6] K. Atanassov e G. Gargov. Interval Valued Intuitionistic Fuzzy Sets. *Fuzzy Sets and Systems*, (31):343 - 349, 1989.

[7] A. Avizienis. The N-Version Approach to Fault-Tolerant Software. *IEEE Transactions on Software Engineering*, SE-11(12):1491–1506, Dezembro 1985.

[8] A. Avizienis e J.C. Laprie. Dependability Computing: From Concepts to Design Diversity. *Proceeding of IEEE*, 74(5):67–80, Maio 1986.

[9] M. Baldassari, G. Bruno, V. Russi, e R. Zompi. PROTOB a Hierarchical Object-Oriented CASE Tool for Distributed Systems. Em C. Ghezzi e J.A. McDermid, editores, *ESEC'89*, volume 387 of *Lecture Notes in Computer Science*, páginas 425–445. Springer-Verlag, 1989.

[10] E. Battiston, F. De Cindio, e G. Mauri. OBJSA Nets: a Class of High-level Nets Having Objects as Domains. Em G. Rozenberg, editor, *Advances on Petri Nets 1988*, volume 340 of *Lecture Notes in Computer Science*, páginas 20–43. Springer-Verlag, 1988.

[11] F. Belli e K.-E. Grosspeitsch. Specification of Fault-Tolerant Systems Issues by Predicate/Transitions Nets and Regular Expressions-Approach and Case Study. *IEEE Transactions on Software Engineering*, 17(6):513–526, Junho 1991.

[12] G. Berthelot. Checking Properties of Nets Using Transformations. Em *Lecture Notes in Computer Science*, volume 222. Springer-Verlag, 1986.

[13] B. Berthomieu e M. Diaz. Modeling and Verification of Time Dependent Systems Using Time Petri Nets. *IEEE Transactions on Software Engineering*, 17(3):259–273, Março 1991.

[14] I.I. Bestuzheva e V.V. Rudnev. Timed Petri Nets: Classification and Comparative Analysis. *Automation and Remote Control*, 51(10):1303 – 1318, Outubro 1990.

[15] T. Bihari e P. Gopinath. Object Oriented Real-Time Systems: Concepts and Examples. *IEEE Computer*, páginas 25–32, Dezembro 1992.

[16] T. Bihari, P. Gopinath, e K. Schwan. Object-Oriented Design of Real-Time Systems. Em IEEE CS Press, editor, *Proc. 10th Real-Time Systems Symposium*, páginas 194–201, 1989.

[17] K.P. Birmam, T.A. Joseph, T. Raeuchle, R. Abadi, A.E.Koo, e S. Toueg. Checkpointing and Rollback–Recovery for Distributed System. *IEEE Transactions on Software Engineering*, SE-13(1):23–31, Janeiro 1987.

[18] G. Booch. *Object Oriented Design: With Application.* Benjamin/Cummings, 1991.

[19] T. Cao e A.C. Sanderson. Task Sequence Planning Using Fuzzy Petri Nets. Em *International Conference on Systems, Man, and Cybernetics*, páginas 349 – 354, 1991.

[20] T. Cao e A.C. Sanderson. Variable Reasoning and Analysis About Uncertainty with Fuzzy Petri Nets. Em M. Ajmone Marsan, editor, *Application and Theory of Petri Nets 1993*, volume 691 of *Lecture Notes in Computer Science*, páginas 126 – 145. Springer-Verlag, Junho 1993.

[21] J. Cardoso, R. Valette, e D. Dubois. Petri Nets with Uncertain Markings. Em G. Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*, páginas 64–78. Springer-Verlag, June 1990.

[22] S. Chanas e J. Kamburowski. The Use of Fuzzy Variables in Pert. *Fuzzy Sets and Systems*, (5):11 – 19, 1981.

[23] V. Chandra e M.R. Verma. A Fail-Safe Interlocking System for Raylways. *IEEE Design & Test of Computers*, 8(1):58–66, Março 1991.

[24] S.K. Chang, A. Perkusich, J.C.A. de Figueiredo, B. Yu, e M.J. Ehrenberger. The Design of Real-Time Distributed Information Systems with Object-Oriented and Fault-Tolerant Characteristics. Em *Proc. of The Fifth International Conference on Software Engineering and Knowledge Engineering*, San Francisco, California, Junho 1993.

[25] S.M. Chen, J.S. Ke, e J.F. Chang. Knowledge Representation Using Fuzzy Petri Nets. *IEEE Transactions on Knowledge and Data Engineering*, 2(3):311–319, 1990.

[26] Z. Chen. Uncertain Temporal Knowledge Management. Em *Proc. of the 4th Int. Conference on Software Engineering and Knowledge Engineering*, Capri, Italy, Junho 1992.

[27] Z. Chen e F. Terrier. About Temporal Uncertainty. Em *Proceedings of IEEE/ACM International Conference on Developing and Managing Expert System Programs*, páginas 223 – 230, 1991.

[28] G. Ciardo. *Analysis of Large Stochastic Petri Net Models.* Tese de Doutorado, Duke University, 1989.

[29] B. Dasarathy. Timing Constraints of Real-Time Systems: Constructs for Expressing Them, Methods of Validating Them. *IEEE Transactions on Software Engineering*, 11(1):80 – 86, Janeiro 1985.

[30] J.C.A. de Figueiredo. Fuzzy Time Petri Nets. Relatório técnico, Department of Computer Science, University of Pittsburgh, 1992.

[31] J.C.A. de Figueiredo e A. Perkusich. Análise Temporal Baseada em Redes de Petri para Sistemas de Software. Em *Anais do XX Seminário Integrado de Software e Hardware, SEMISH 94*, Caxambu, MG, Agôsto 1994.

[32] J.C.A. de Figueiredo, A. Perkusich, e S.K. Chang. Fuzzy Time Petri Nets for Timing Analysis of Real-Time Software Systems. Submitted to IEEE Transaction on Systems, Man and Cybernetics, Dezembro 1993u.

[33] J.C.A. de Figueiredo, A. Perkusich, e S.K. Chang. Timing Analysis of Real-Time Software Systems Using Fuzzy Time Petri Nets. Em *Proc. of The sixth International Conference on Software Engineering and Knowledge Engineering*, páginas 257–266, Riga, Latvia, Junho 1994.

[34] J.C.A. de Figueiredo, A. Perkusich, e M.E. Morais. Tolerância a Falhas em Sistemas de Software Utilizando uma Abordagem por Redes de Petri. Em *Anais do V Simpósio de Computadores Tolerantes a Falhas*, Outubro 1993.

[35] R. de Lemos, A. Saeed, e T. Anderson. Analysis of Timeliness in Safety-Critical Systems. Em J. Vytopil, editor, *Formal Techniques in Real-Time Fault-Tolerant Systems*, volume 571 of *Lecture Notes in Computer Science*, páginas 571–592. Springer-Verlag, 1992.

[36] Y. Deng. *A Unified Framework for the Modeling, Prototyping and Design of Distributed Information Systems.* Tese de Doutorado, Department of Computer Science, University of Pittsburgh, 1992.

[37] Y. Deng e S.K. Chang. A Framework for the Modeling and Prototyping of Distributed Information Systems. *International Journal of Software Engineering and Knowledge Engineering*, 2(3):203–226, Setembro 1991.

[38] Y. Deng e S.K. Chang. Unifying Multi-Paradigms in Software System Design. Em *Proc. of the 4th Int. Conf. on Software Engineering and Knowledge Engineering*, Capri, Italy, Junho 1992.

[39] Y. Deng, S.K. Chang, J.C.A. de Figueiredo, e A. Perkusich. Integrating Software Engineering Methods and Petri Nets for the Specification and Prototyping of Complex Software Systems. Em M. Ajmone Marsan, editor, *Application and Theory of Petri Nets 1993*, volume 691 of *Lecture Notes in Computer Science*, páginas 206 – 223. Springer-Verlag, Chicago, USA, Junho 1993.

[40] D. Dubois e H. Prade. *Fuzzy Sets and Systems: Theory and Applications.* Academic Press, 1980.

[41] D. Dubois e H. Prade. Processing Fuzzy Temporal Knowledge. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(4):729–744, Julho 1989.

[42] J.B. Dugan, K.S. Trivedi, R.M. Geist, e V.F. Nicola. Extended Stochastic Petri Nets: Applications and Analysis. Em *Performance '84*, Dezembro 1984.

[43] J. Engelfriet, G. Leih, D. Rozenberg, G.Janssens, e G. Rozenberg. Net-Based Description of Parallel Object-Based Systems, or POTs and POPs. Em J.W. de Bakker, W.P. de Roever, J.W. Rozenberg, G.de Bakker, W.P. de Roever, e G. Rozenberg, editores, *Foundations of Object-Oriented Languages*, Lecture Notes in Computer Science, páginas 229–244. Springer-Verlag.

[44] R. Fairley. *Software Engineering Concepts.* MacGraw-Hill, New York, NJ, 1985.

[45] D. Ferrari. *Computer Systems Performance Evaluation.* Prentice-Hall, 1978.

[46] S. Fisher, A. Scholtz, e D. Taubner. Verification in Process Algebra of the Distributed Control of Track Vehicles – A Case Study. Em G.V. Bochman e D.K. Probst, editores, *CAV'92, Computer Aided Verification, 4th Int. Workshop*, volume 663 of *Lecture Notes in Computer Science.* Springer-Verlag, Julho 1992.

[47] H.J. Genrich. Predicate/Transition Nets. Em W. Brauer, W. Reisig, e G. Rozemberg, editores, *Petri Nets: Central Models and Their Properties*, volume 254 of *Lecture Notes in Computer Science*, páginas 207–247. Springer-Verlag, 1987.

[48] H.J. Genrich e K. Lautenbach. System Modeling with High Level-Petri Nets. *Theoretical Computer Science*, 13:109–136, 1981.

[49] C. Ghezzi, D. Mandrioli, S. Morasca, e M. Pezze. A General Way to Put Time in Petri Net. Em *5th International Workshop on Software Specifications and Design*, páginas 60–66, Maio 1989.

[50] C. Ghezzi, D. Mandrioli, S. Morasca, e M. Pezze. A Unified High-Level Petri Net Formalism for Time-Critical Systems. *IEEE Transactions on Software Engineering*, 17(2):160–171, Fevereiro 1991.

[51] A. Giordana e L. Saitta. Modeling Production Rules by Means of Predicate Transition Networks. Em *Information Sciences*, volume 35, páginas 1–41. Elsevier Science Publishing, 1985.

[52] H. Gomaa. Structuring Criteria for Real Time Systems Design. Em *Proc. of 11th. Int Conference on Software Engineering*, páginas 290–301, Pittsburgh, PA, USA, 15-18, Maio 1989.

[53] H. Gomaa. *Software Design Methods for Concurrent and Real-Time Systems.* Addison-Wesley, 1993.

[54] B. Guilleux. Signalling on New Lines: the Transition to the TVM 430 System. *Revue Générale des Chemins de Fer*, páginas 59 – 65, 1992.

[55] P.J. Haas e G.S. Shedler. Regenerative Stochastic Petri Nets. *Performance Evaluation*, 6(3):189–204, Setembro 1986.

[56] H.A. Hansson. *Time and Probability in Formal Design of Distributed Systems.* Tese de Doutorado, Uppsala University, 1991.

[57] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.

[58] I. Jacobson, M. Christerson, P. Jonsson, e G. Övergaard. *Object Oriented Software Engineering: A Use Case Driven Approach.* Addison-Wesley, 1992.

[59] R. Jain. *The Art of Computer Systems Performance Analysis.* John Wiley & Sons, 1991.

[60] H.-M. Jarvinen, R. Kurki-Suonio, M. Sakkinen, e K. Systa. Object-Oriented Specification of Reactive Systems. Em *Proc. of 13th Conference on Software Engineering*, páginas 63–71, Austin, Texas, USA, 13-16, Maio 1989.

[61] K. Jensen. Coloured Petri Nets: A High Level Language for System Design and Analysis. Em *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.

[62] Y.K. J.M. GwinnLee e S.J. Park. OPNets: An Object-Oriented High-Level Petri Net Model for Real-Time System Modeling. *ACM SIGPLAN Notices, month = feb, pages = 47-56, title = "Object-Oriented Programs in Realtime", volume = 27, year = 1992, anannote = an analysis of the object-oriented approach and its consequences in realtime applications. Journal of System and Software*, 20(1):69–86, Janeiro 1993.

[63] C.B. Jones. *Development using VDM.* Prentice-Hall, 1986.

[64] A. Kaufmann e M.M. Gupta. *Introduction to Fuzzy Arithmetic - Theory and Applications.* Van Nostrand Reinhold Company, 1985.

[65] A. Kaufmann e M.M. Gupta. *Fuzzy Mathematical Models in Engineering and Management Science.* North-Holland, 1988.

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

[67] R. Koo e S. Toueg. Checkpointing and Rollback-Recovery for Distributed Systems. *IEEE Transactions on Software Engineering*, SE-13(3):23–31, Janeiro 1987.

[68] H. Kopetz, A. Damm, C. Koza, M. Mulazzani, W. Schwabl, C. Senft, e R. Zailinger. Distributed Fault-Tolerant Real-Time Systems: The Mars Approach. *IEEE Micro*, 9(1):25–40, Fevereiro 1989.

[69] L. Lamport. Sometimes is sometimes not never. Em *7th ACM Symposium of Principles Programming Languages*, páginas 174– 185, Las Vegas, NV, Janeiro 1980.

[70] N.G. Levenson. Software Safety: Why, What, and How. *ACM Computing Surveys*, 18(2):125–163, Junho 1986.

[71] N.G. Leveson e J.L. Stolzy. Safety Using Time Petri Nets. *IEEE Transactions on Software Engineering*, SE-13(3):386–397, Março 1987.

[72] S.-T. Levi e A.K. Agrawala. *Fault Tolerant System Design.* McGraw Hill, 1992.

[73] C.G. Looney. Fuzzy Petri Nets for Rule-Based Decisionmaking. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):178 – 183, Janeiro 1988.

[74] F.A. Lootsma. Stochastic and Fuzzy PERT. *European Journal of Operational Research*, (43):174 – 183, 1989.

[75] K. Majmudar e M.A. Jafari. Functional and Performance Analysis of Time Petri Nets. Em *International Conference on Systems, Man, and Cybernetics*, volume 2, páginas 980 – 985, Outubro 1992.

[76] D. Marinescu e C. Lin. On Stochastic High-Level Petri Nets. Em *International Workshop on Petri Nets and Performance Models*, Agôsto 1987.

[77] M. Menasche e B. Berthomieu. Time Petri Nets for Analyzing and Verifying Time Dependent Protocols. Em *3rd International Workshop on Protocol Specification, Testing and Verification*, páginas 161–172, Junho 1983.

[78] P.M. Merlin. Specification and Validation of Protocols. *IEEE Transactions on Communications*, COM-27(11):1671–1680, Novembro 1979.

[79] P.M. Merlin e D.J. Farber. Recoverability of Communication Protocols - Implications of a Theoretical Study. *IEEE Transactions on Communication*, COM-24(9):1036–1043, Setembro 1976.

[80] R. Milner. A Calculus of Communicating Systems. volume 92 of *Lecture Notes in Computer Science.* Springer-Verlag, 1980.

[81] M.K. Molloy. *On the Integration of Dealy and Throughput Measures in Distributed Processing Models.* Tese de Doutorado, UCLA, 1981.

[82] M.K. Molloy. Performance Analysis Using Stochastic Petri Nets. *IEEE Transactions on Computers*, c-31(9):913–917, Setembro 1982.

[83] M.F. Morris e P.F. Roth. *Computer Performance Evaluation.* Van Nostrand Reinhold Company, 1982.

[84] D. Mossé. *Design, Development, and Deployment of Fault-Tolerant Applications for Distributed Real-Time Systems.* Tese de Doutorado, University of Maryland, College Park, 1993.

[85] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proc. of the IEEE*, 77(4):541–580, Abril 1989.

[86] S.H. Nasution. Fuzzy Durations in Critical Path Method. Em *Second IEEE Conference on Fuzzy Systems*, volume 2, páginas 1069 – 1073, Março 1993.

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

[88] J.S. Ostroff. *Temporal Logic for Real-Time Systems.* John Wiley and Sons, 1989.

[89] A. Perkusich. Analysis of Complex System Based Upon G-Net Decomposition. Relatório técnico, Department of Computer Science, University of Pittsburgh, 1992.

[90] A. Perkusich, J.C.A. de Figueiredo, e S.K Chang. Embedding Fault-Tolerant Properties in the Design of Complex Systems. *Journal of Systems and Software*, 25(2):23–37, Fevereiro 1994.

[91] A. Perkusich, J.C.A. de Figueiredo, e M.E. Morais. Análise e Verificação de Sistemas Baseados em Objetos Utilizando uma Abordagem por Redes de Petri. Em *Anais do XIX Seminário Integrado de Software e Hardware, SEMISH 93*, Setembro 1993.

[92] A. Perkusich, J.C.A. de Figueiredo, e M.E. Morais. Projeto de Sistemas em Tempo Real Distribuídos com Característica Baseada em Objetos e Tolerância a Falhas. Em *Anais do XIX Seminário Integrado de Software e Hardware, SEMISH 93*, Setembro 1993.

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

[94] A. Pnueli. The Temporal Logic of Programs. Em *18th IEEE Annual Symposium on the Foundations of Computer Science*, páginas 46 – 57, Providence, 1977.

[95] C.V. Ramamoorthy e G.S. Ho. Performance Evaluation of Asynchronous Concurrent Systems Using Petri Nets. *IEEE Transactions on Software Engineering*, SE-6(5):440–449, 1980.

[96] C. Ramchandani. Analysis of Asynchronous Concurrent Systems by Petri Nets. Relatório Técnico Project MAC-TR120, M.I.T., Cambridge, MA, 1974.

[97] J.K. Ramos-Thuel, S. Strosnider. The Transient Server Approach to Scheduling Time-Critical Recovery Operations. Em *IEEE Real-Time Systems Symposium*, páginas 286 – 295, San Antonio, Texas, Dezembro 1991.

[98] R.R. Razouk. The Derivation of Performance Expressions for Communication Protocols from Timed Petri Net Models. Em *ACM - SIGCOMM '84 Tutorials & Symposium: Communications, Architectures & Protocols*, páginas 210–217, Junho 1984.

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

[100] W. Reisig, G. Rozenberg, K. Jensen, e M. Ajmone Marsan. Introductory Tutorial I. Em *PN93*, 1993.

[101] P. Robinson, editor. *Object-oriented Design.* Chapman & Hall, 1992.

[102] Y. Sami e G. Vidal-Naquet. Formalization of the Behaviour of Actors by Colored Petri Nets and Some Applications. Em J. van Aarts, E. H. L. Leeuwen e M. Rem, editores, *PARLE '91 Conference on Parallel Architectures and Languages Europe*, volume 555 of *Lecture Notes in Computer Science*, páginas 110–127. Springer-Verlag, 1991.

[103] M. Schneider e A. Kandel. Properties of the Fuzzy Expected Value and the Fuzzy Expected Interval in Fuzzy Environment. *Fuzzy Sets and Systems*, (28):55 – 68, 1988.

[104] S.M. Shatz. *Development of Distributed Software.* Macmillan Publishing Company, New York, 1993.

[115] I. Suzuki e 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, Maio 1989.

[116] L. Svobodova. Resilient Distributed Computing. *IEEE Tans. on Software Engineering*, SE-10(3):257–268, Maio 1984.

[117] K. Takashio e M. Tokoro. DROL: An Object-Oriented Programming Language for Distributed Real-Time Systems. Em *Proc. of OOPSLA'92*, páginas 276–294, Vancouver, British Columbia, Canada, 18-22, Outubro 1992.

[118] Y. Tamir e C. S'equin. Error Recovery in Multicomputer Using Global Checkpoints. Em *Proceedings of Int'l Conf. on Parallel Processing*, páginas 32–41, 1984.

[119] M. Tokoro. Towards Computing Systems in 2000. volume 563 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992.

[120] A.M. Tyrrel e D.J. Holding. Design of Reliable Software in Distributed Systems Using Conversation Scheme. *IEEE Transactions on Software Engineering*, SE-12(9):921–928, Setembro 1986.

[121] R. Valette, J. Cardoso, e D. Dubois. Monitoring Manufacturing Systems by Means of Petri Nets with Imprecise Markings. Em *IEEE International Symposium on Intelligent Control*, páginas 233 - 238, Albany, NY, Setembro 1989.

[122] W.M.P. van der Aalst. *Timed Coloured Petri Nets and Their Application to Logistic Systems*. Tese de Doutorado, Eindhoven University of Technology, 1992.

[123] W.M.P. van der Aalst. Interval Timed Coloured Petri Nets and Their Analysis. Em M. Ajmone Marsan, editor, *Application and Theory of Petri Nets 1993*, volume 691 of *Lecture Notes in Computer Science*, páginas 453 - 472. Springer-Verlag, Junho 1993.

[124] P.T. Ward e S.J. Mellor. *Structured Development for Real-Time Systems*. Yourdon, 1985.

[125] L.A. Zadeh. Fuzzy Sets. *Information and Control*, 8:338–353, 1965.

[126] A. Zenie. Colored Stochastic Petri Nets. Em *1st Workshop on Timed Petri Nets*, Julho 1985.

[127] H.-J. Zimmermann. *Fuzzy Set Theory and Its Applications.* Kluwer Academic Publishers, 1991.

[128] W.M. Zuberek. Timed Petri Nets and Preliminary Performance Evaluation. Em *7th Annual Symposium on Computer Architecture*, páginas 88–96, Maio 1980.

[129] W.M. Zuberek. Timed Petri Nets: Definitions, Properties, and Applications. *Microelectronics and Reliabilitu.* 31(4):627 – 644. 1991.