Formalização de Entidades Gerenciadas Através de Redes de Petri Orientada a Objetos

Aldenor Falcão Martins

COPELE-DEE-UFPB

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

Área de Concentração: Processamento da Informação

Angelo Perkusich

Orientador

Jorge Cesar Abrantes de Figueiredo

Orientador

Campina Grande, Paraíba, Brasil

COPELE-DEE-UFPB , Dezembro de 1997

Formalização de Entidades Gerenciadas Através de Redes de Petri Orientada a Objetos

Aldenor Falcão Martins

COPELE-DEE-UFPB

Dissertação de Mestrado apresentada em Dezembro de 1997

Angelo Perkusich

Orientador

Jorge Cesar Abrantes de Figueiredo

Orientador

Antonio Marcus Nogueira Lima, Doutor.

Componente da Banca

José Augusto Suruagy Monteiro, Ph.D

Componente da Banca

Campina Grande, Paraíba, Brasil, Dezembro de 1997

M379f Martins, Aldenor Falcao.
Formalizacao de entidades gerenciadas atraves de redes de petri orientada a objetos / Aldenor Falcao Martins. - Campina Grande : [S.n.] 1997.
280 f. : il.

Disserrtacao (Mestrado em Engenharia Eletrica) - Universidade Federal de Campina Grande, Centro de Ciencias e Tecnologia.

1. Redes de Petri - Orientada a Objetos. 2. Dissertacao. I. Perkusich, Angelo, Prof. Dr. II. Figueiredo, Jorge Cesar Abrantes de. , Prof. Dr. III. Universidade Federal de Campina Grande - Campina Grande (PB) IV. Título

CDU 621.355(043)

FORMALIZAÇÃO DE ENTIDADES GERENCIADAS UTILIZANDO REDES DE PETRI ORIENTADA À OBJETOS

ALDENOR FALCÃO MARTINS

Dissertação Aprovada em 29.12.1997

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

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

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

JOSÉ AUGUSTO SURUAGY MONTEIRO, Dr., UFPE
Componente da Banca

CAMPINA GRANDE - PB
Dezembro - 1997

Agradecimentos

Aqui vai meu muito obrigado aqueles que de alguma forma contribuiram com este trabalho, agradeco aos meus orientadores, Angelo e Jorge, pela paciência, disposição e amizade, ao professor Marcelo Alencar por ter me ajudado a descobrir essa área, ao CnPq pela oportunidade de realizar este trabalho. Gostaria de agradecer ao Eng. Ido Alexandre R. Alves pelo acesso as normas do ITU-T. Dedico este trabalho, as duas pessoas, que nunca mediram esforços para fornecer carinho, amor e educação, cuja dedicação não conhece limites, a voces, meus pais, (Antonio Martins da Silva e Vera Lúcia Falcão Martins), obrigado.

Deus, obrigado.

Wapi yo

Resumo

À medida que a estrutura de redes corporativas cresce, a dificuldade de gerir os recursos disponíveis cresce na mesma proporção.

No panorama atual de completa heterogeneidade, busca-se padrões que permitam unificar a gerência de redes diversas, usualmente as definições dos recursos que são gerenciados são escritas numa linguagem natural sem nenhum rigor matemático. Tais definições fatalmente incorrem em ambigüidades de interpretação, podendo implicar em erros na própria descrição destes objetos gerenciados.

Este trabalho tenta demonstrar a viabilidade de definir tais entidades gerenciadas de maneira formal, através de um classe de Redes de Petri orientada a objetos, denominada de G-CPN. A fim de demonstrar o poder de G-CPN iremos confrontá-la com uma técnica de descrição formal amplamente utilizada na indústria, SDL (Specification and Description Language), enfatizando as qualidades e deficiências de cada uma dessas técnicas na definição de Objetos Gerenciados.

Abstract

With the growth in corporate networks, the difficulty for resource management increases in the same pace.

Considering the heterogeneous nature of actual network systems, standards to unify the management of these networks are becoming more important. The managed resources are usually described based on a natural language without any mathematical basis. Such definitions will suffer from ambiguities due to misinterpretations, leading to errors in the description of the managed resources.

It is the aim of this work to try to demonstrate the applicability of the formal definition of managed entities, by means of a class of object oriented Petri net, named G-CPN. In order to shown the modeling power of G-CPN we also apply another formal technique, SDL (Specification and Description Language), to the modeling of managed objects and discuss the advantages and disadvantages of each one of the mentioned formal methods.

Sumário

1 Introdução **1**

1.1 Contexto 1

1.2 Definição Semi-Formal de EOGs 4

1.3 Métodos Formais 5

1.4 Descrição Formal de EOGs 5

1.5 Alguns Métodos de Formalização de EOGs 6

1.6 Relevância da Pesquisa 7

1.7 Estrutura do Documento 8

2 O Modelo de Gerenciamento OSI **9**

2.1 O Modelo de Gerenciamento OSI 9

2.1.1 Gerente(ou processo de gerenciamento) 10

2.1.2 Agente 10

2.1.3 MIB 11

2.1.4 Modelo de Gerenciamento de Sistemas 11

2.1.5 Modelo de Gerenciamento da Informação 12

2.1.6 Áreas Funcionais para o Gerenciamento de Sistemas OSI 13

2.2 Linhas Gerais para a Definição de Objetos Gerenciados(GDMO) 14

2.3 Visão de Rede da Interface M4 16

2.4 Arquitetura de Rede e Elemento de Rede 17

3 Técnicas de Descrição Formal **21**

3.1 Sistemas G-CPN 21

3.1.1 Introdução Informal do Modelo Clássico de Redes de Petri . . . 24

3.1.2 Introdução Informal a Redes de Petri Coloridas 25

3.1.3 Módulos G-CPN 27

3.1.4 Introdução Informal aos Sistemas G-CPN 28

3.1.5 Um exemplo de Sistema G-CPN 32

3.1.6 Definições Formais para Módulos G-CPN 36

3.1.7 Comportamento e Semântica de Módulos G-CPN 41

3.1.8 Sintaxe de Sistemas G-CPN 48

3.2 Specification and Description Language - SDL 50

3.2.1 Introdução a SDL 50

3.2.2 Comportamento do Sistema 52

3.2.3 Estrutura de uma instância de um sistema 53

3.3 Representação Gráfica e Textual 55

3.4 Exemplo de Utilização de SDL 55

4 Modelagem Formal dos Objetos **60**

4.1 Introdução 60

4.2 Modelagem das Entidades Gerenciadas 60

4.3 Modelagem Usando G-CPN 63

4.3.1 Entidade Gerenciada vcSubnetwork 63

4.3.2 Entidade Gerenciada vcSubnetworkConnection 68

4.3.3 Entidade Gerenciada vcSubnetworkTP 69

5 Análise das Entidades Modeladas **72**

5.1 Entidades Modeladas 74

5.1.1 atmLinkConnection 74

5.1.2 atmLinkTP 74

5.1.3 atmNetworkAccessProfile 74
5.1.4 atmNetworkTrailTerminationPoint 77
5.1.5 atmNetworkRoutingProfile 77
5.2 Cenário Utilizados 77
5.2.1 Primeiro Cenário 77
5.2.2 Segundo Cenário 82
5.2.3 Terceiro Cenário 82
5.2.4 Quarto Cenário 84
5.2.5 Quinto Cenário 86
5.2.6 Sexto Cenário 86
5.2.7 Sétimo Cenário 86
5.3 Modelagem em SDL 87

6 Conclusões **105**

Lista de Figuras

2.1 Relação Agente/Gerente 11
2.2 Gerência da camada N 12
2.3 Definição de classes de objetos gerenciados 14
2.4 Arquitetura de referência para a interface de gerência do ATM Forum . 16
2.5 Visão de rede 17
2.6 Relação entre as arquiteturas de rede e elemento de rede 18
2.7 Diagrama de herança 19
2.8 Diagrama de contenção 20

3.1 Sistema G-CPN e eventos na interação 30
3.2 Notação para módulo G-CPN 32
3.3 Módulo produtor para o sistema G-CPN produtor/consumidor 33
3.4 Módulo consumidor para o sistema G-CPN produtor/consumidor . . . 35
3.5 Interação da máquina SDL com o ambiente 52
3.6 Comportamento do sistema 53
3.7 Bloco 54
3.8 Sistema 54
3.9 Representação gráfica e textual 55
3.10 Sistema DaemonGame 56
3.11 Bloco Game 57
3.12 Processo Monitor 58
3.13 Processo Game 58

4.1 Descrição Informal das entidades gerenciadas 62
4.2 Representação gráfica para vcSubNetwork 65
4.3 Módulo G-CPN para a entidade sub-rede 66
4.4 Declarações para os tipos e variáveis 67
4.5 Representação gráfica de uma conexão vcSubnetwork 68
4.6 Módulo G-CPN para a entidade vcSubnetwork 70
4.7 Representação gráfica para o ponto de terminação de uma sub-rede . . 71

5.1 G-CPN para a entidade gerenciada de ponto de terminação sub-rede . . 73
5.2 Modelo G-CPN do atmLinkConnection 75
5.3 Modelo G-CPN do Link Termination Point 76
5.4 Modelo G-CPN do Network Access Profile 78
5.5 Modelo G-CPN do Network Trail Termination Point 79
5.6 Modelo G-CPN do atmNetwork Routing Profile 80
5.7 Modelo CPN equivalente para o módulo G-CPN da entidade SubNetwork 83
5.8 Rede CPN equivalente para o modelo G-CPN do SNTP 91
5.9 Modelo CPN equivalente para a conexão de sub-rede 92
5.10 Sistema AtmNetwork 93
5.11 Bloco atmSubNetwork 94
5.12 Bloco atmSubNetworkConnection 95
5.13 Sistema subSNTP 96
5.14 Bloco atmSubNetworkTP 97
5.15 Processo SNTP 98
5.16 Processo Born_SNTP 99
5.17 Processo SNTP_notify 100
5.18 Processo Delete_SNTP 101
5.19 Processo SetupPP 102
5.20 Processo SetupMP 103
5.21 MSC do sistema subSNTP, para a criação de um SNTP 104

Capítulo 1

Introdução

1.1 Contexto

Uma das questões básicas da administração é a gerência da empresa. Numa rede de computadores não é diferente, a medida que a estrutura cresce, mais difícil torna-se gerenciá-la [Bla95]. Como se não bastasse, hoje tem se observado a fusão da redes de dados e de telecomunicações, que anteriormente ocupavam posições definidas [Mes96], implicando assim numa maior complexidade nos sistema de gerência de rede.

A definição mais adequada do que vem a ser gerência pode ser encontrada nos livros de administração e envolve entre outros aspectos: planejamento, organização, monitoramento, *accounting* e controle das atividades e dos recursos. Os aspectos mais focalizados no ambiente OSI (*Open Systems Interconnection*) e Internet são monitoramento, *accounting* e controle das atividades e recursos. Planejamento e organização não são contemplados diretamente pela estrutura OSI e Internet, mas constituem questões necessárias ao desenvolvimento de um sistema de gerência, pois sem esses dois aspectos, qualquer esforço na viabilização de monitoramento, *accounting* e controle torna-se inútil.

Os Sistemas de gerência de redes, são sistemas que se ocupam da gerência do sistema segundo determinados aspectos:

Alta disponibilidade da rede :

Envolve o aumento da eficiência operacional, implicando numa redução dos períodos de indisponibilidade do sistema e da rede.

Redução do custo operacional da rede :

Devido a mudança rápida de tecnologia, torna-se desejável o gerenciamento de sistemas heterogêneos e protocolos múltiplos.

Aumentar flexibilidade de operação e integração :

Deve considerar a possibilidade de absorção de novas tecnologias e aquisição de equipamentos sem dificuldades, os *softwares* de gerenciamento de redes não devem ser extremamentes dependentes das plataformas utilizadas.

Maior eficiência :

Ao reduzir o tempo de indisponibilidade e o custo operacional da rede, a eficiência aumenta como um todo.

Facilidade de utilização :

A utilização de aplicações de gerência não deve envolver uma longa curva de aprendizagem.

Segurança :

Algumas das funções de gerência devem ser seguras. Aqui há uma distinção entre a segurança da rede, do sistema e da funções de gerência da rede.

Segundo a visão de gerenciamento OSI, os recursos disponíveis, sejam eles, lógicos (entidade de uma camada ou uma conexão) ou físicos (componente de um equipamento de comunicações) são abstraídos no que denomina-se Objetos Gerenciados(entidades de objetos gerenciados) [IT92a]. Essa visão é descrita de forma detalhada nas normas da

série X do ITU-T [1], essas normas estabelecem padrões a serem utilizados pela indústria na concepção e desenvolvimento de seus produtos.

Os padrões de gerenciamento de redes OSI usam muitos dos conceitos de Projetos Orientados a Objetos (POO). Os modelos de gerência desenvolvidos pelo IEEE [2] e o IETF [3], utilizam também alguns conceitos de POO, embora o IETF não empregue os conceitos de herança, criação e destruição de objetos.

Na perspectiva OSI de gerenciamento de redes, os objetos gerenciados são descritos e definidos por quatro aspectos:

1. Seus *Atributos*, conhecido como sua interface(fronteira visível).
2. As *Operações* que podem ser executadas no mesmo.
3. As *Notificações* que lhe são permitidas emitir.
4. Seu *Comportamento* que é exibido em função das operações executadas pelo mesmo.

Os elementos de redes (conjunto de parte lógicas e físicas que constituem uma rede) que serão gerenciados (Entidades Objetos Gerenciados-EOG) são definidos no que denomina-se de Base de Gerenciamento da Informação (*Management Information Base* - MIB). A MIB contém seus nomes, seus comportamentos admissíveis e as operações que podem ser executadas sobre eles. Ou seja, a maneira como o sistema enxerga e define esse recurso é abstraída por meio deste objeto. Atualmente as EOGs são definidas em uma linguagem dita semi-formal [Udu96]. A natureza do recursos a serem gerenciados e o comportamento que são esperados exibir são expressos em uma linguagem natural, organizada e estruturada usando um simples conjunto de técnicas de especificação.

[1] International Telecommunication Union - Telecommunication Sector

[2] Institute of Electronics and Electrical Engineers

[3] Internet Engineering Task Force

1.2 Definição Semi-Formal de EOGs

A recomendação [IT92b] define as EOGs numa linguagem natural, escrita em Notação de Sintaxe Abstrata Um (*Abstract Syntax Notation One*- ASN.1), comumente utilizada para prover a comunicação entre dois sistemas, na camada de aplicação do modelo de gerência OSI, a abstração de linguagem é utilizada para providenciar o formato como os dados serão organizados e que significados terão. A definição de EOGs se dá através das Linhas para Definição de Objetos Gerenciados (*Guidelines for the Definition of Managed Objects*-GDMO) [IT92b],

As GDMO são definidas como uma ASN.1 *formal*. Este formato padrão é conhecido como um *arcabouço*, as GDMO disponibilizam os seguintes *arcabouços*:

- Classe de objetos gerenciados
- Pacotes aplicativos
- Atributos
- Grupo de atributo
- Ação
- Comportamento
- Notificação
- Parâmetro
- Atribuição de nome

Estes arcabouços asseguram uma estrutura para as declarações das propriedades (baseado no modelo de gerenciamento do objeto) e estruturação das atribuições informais de comportamento. A especificação unicamente através de GDMO provê apenas indicações de como os objetos são gerenciados, não tendo meios de testar os objetos nas suas especificações e ainda permitindo que sejam mal interpretados quando da sua implementação.

1.3 Métodos Formais

A engenharia tradicional é fortemente alicerçada em modelos matemáticos e cálculos a fim de prover julgamento de um projeto. O termo *Métodos Formais* refere-se a uma variedade de técnicas de modelagem matemática que se aplicam a projetos de sistemas computacionais(*Hardware & Software*).

Modelagem Formal de um sistema geralmente implica em transcrever uma descrição de um sistema de um modelo não matemático (diagrama de fluxo de dados, diagrama de objetos, cenários, texto em inglês, etc.) para uma especificação formal, utilizando-se linguagem formais, isto resulta em um sistema com uma precisão lógica [oSA95]. Ferramentas de modelagem baseadas em Métodos Formais podem ser empregadas para logicamente desenvolver a especifação a fim de prover a completude e consistência das metas estabelecidas.

A aplicação de métodos formais pode tornar as especificações mais precisas, eliminando a ambigüidade inerente da linguagem natural e possibilitando a automatização do processo de implementação e teste.

1.4 Descrição Formal de EOGs

Segundo Derrick [DLT95], uma técnica de descrição formal que pode ser usada para a especificação de EOGs e sua manipulação devem dar suporte a:

- Mecanismos de nomeação e atribuição de nomes as EOGs;
- Relacionamento de contenção e herança entre os objetos e a habilidade de criar estruturas que representem essas relações;
- A definição de um conjunto de ações e i notificações, com sua respectiva parametrização;
- Definição de tipos de atributos, incluindo valores padrões e iniciais, além de faixa

de restrição, regras de alcançabilidade e ligações que ofereçam suporte a definições de sintaxe de abstração;

- comportamento em termos de regras, para a ocorrência de ações ou notificações e relações de seus parâmetros com os atributos dos objetos;
- Leis para criação e destruição de objetos;
- Leis para restrição de concorrência, implicíta no uso de CMIS [4].

Sendo objetivo futuro que os estados de interação entre as EOGs sejam independentemente definidas.

1.5 Alguns Métodos de Formalização de EOGs

Existem, tanto nos meios acadêmicos, quanto na indústria, um desenvolvimento permanente da teoria e prática de Métodos Formais. Os estudos para formalização de EOGs tem boas referências em Festor [Fes95] e Derrick [DLT95]. Em Festor há o desenvolvimento de uma ferramenta especialmente voltada para a formalização de EOGs(MODE), valendo-se de uma lógica de primeira ordem. No entanto, não é discutida a possibilidade de concorrência no modelo. A ferramenta é apresentada como sendo flexível e capaz de integrar outras técnicas formais.

Em Derrick é desenvolvido um estudo sobre a aplicação de *Object-Z* e *RAISE* na formalização de EOGs. *Object-Z* não dispõe da possibilidade de testar código concorrente, a não ser por meio de um artifício, no entanto tais técnicas não apresentam um método gráfico de estudo, o que dificulta a visão do sistema modelado.

A despeito das demais as Redes de Petri são uma das linguagens mais utilizadas para a modelagem de sistemas concorrentes, em particular as Redes de Petri Coloridas(*Coloured Petri Nets* - CPN) [Jen92]. Neste ponto introduzimos G-CPN

[4] Common Management Information Protocol

1.7 Estrutura do Documento

No Capítulo 2 introduz-se os conceitos básicos sobre gerência de redes, enfatizando os aspectos necessários a compreensão do assunto. No Capítulo 3 introduz-se os conceitos básicos de sistemas G-CPN e SDL.

No capítulo 4 apresenta-se a modelagem de alguns aspectos da interface M4 segundo a visão de rede. No Capítulo 6 apresentam-se as conclusões deste trabalho. O apêndice C contém definições extraídas da norma G.805 [IT93b], necessárias ao compreensão do documento.

Capítulo 2

O Modelo de Gerenciamento OSI

Nos meios acadêmicos e na indústria têm se discutido a a viabilidade da adoção em larga escala dos padrões de gerenciamento OSI. Neste trabalho foi adotado este padrão devido a completude do mesmo, não argüindo-se sobre a maior dificuldade de implementação destas técnicas em relação a perspectiva adotada pelo IETF.

2.1 O Modelo de Gerenciamento OSI

A estrutura de gerenciamento OSI é definida na norma X.721 do ITU-T, esta norma basicamente cobre as seguintes áreas:

- Termos e conceitos gerais;
- Modelo de gerenciamento de sistemas;
- Modelo de informação;
- Áreas funcionais no gerenciamento de sistemas;
- Estrutura de padronização no gerenciamento de sistemas.

A motivação do gerenciamento OSI deve-se a necessidade de controlar, coordenar, e monitorar os recursos de sistemas abertos [1].

2.1.1 Gerente(ou processo de gerenciamento)

Um gerente é uma entidade, ou seja aquele que provê serviços, que é responsável por atividades, tais como: gerenciamento de faltas. A fim de resolver esses faltas o gerente tem que enviar comandos e executar operações nestes objetos (Figura 2.1). Como resposta, o gerente recebe confirmações aos seus comandos, se algo de errado ocorre nessa transação, o gerente recebe notificações, que o pôem a par da situação.

2.1.2 Agente

Reporta ao processo de gerenciamento o estado atual do objeto gerenciado e recebe, deste mesmo processo (gerente), as diretrizes a serem aplicadas às EOGs. Cada agente dentro do sistema tem sua própria área de atuação, tendo as seguintes atribuições:

- Um agente deve gerenciar seu próprio ambiente, no qual interage com objetos gerenciados.
- Um agente recebe comandos de um gerente e desse modo gerencia os objetos contidos no seu campo de atuação.
- Um agente recebe notificações dos objetos gerenciados que encontram-se no seu campo de atuação, daí envia as notificações pertinentes ao gerente. A Figura 2.1 descreve a relação existente entre agente e gerente.

[1] Em Udupa [Udu96], um sistema é dito aberto quando oferece suporte a um conjunto vasto e bem aceito de padronizações. A abertura de um sistema não diz respeito a tecnologia do sistema, implementações ou como os sistemas estão conectados entre si.

2.1.3 MIB

Utilizada pelo agente e pelo gerente a fim de determinar a estrutura e conteúdo da informação de gerenciamento. O termo MIB é primeiramente usado quando se descreve a base de gerenciamento segundo a perspectiva do IETF. Na visão OSI, a base de gerenciamento é mais extensa, sendo denominada de Modelo de Gerenciamento da Informação - MIM (*Management Information Model*).

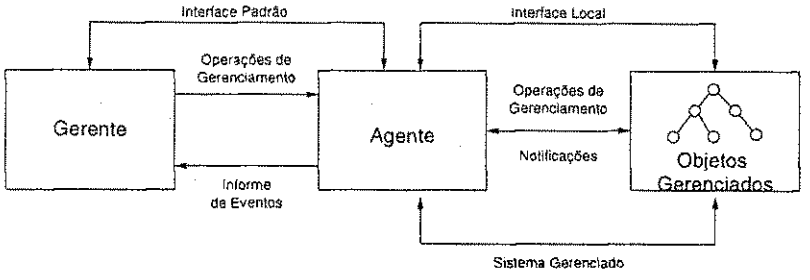


Figura 2.1: Relação Agente/Gerente

2.1.4 Modelo de Gerenciamento de Sistemas

O Modelo OSI prevê três categorias de Gerenciamento que se delimitam através do escopo de utilização. São elas:

- Gerenciamento de Sistemas:

  Cobre os objetos gerenciados em todas as sete camadas do modelo OSI, incluindo a camada de aplicação. Assume uma transferência confiável que atua numa conexão orientada fim-a-fim.

- Gerenciamento de Camada:

  A gerência dos Objetos Gerenciados (OG) está limitada a uma camada particular (Figura 2.2). Os protocolos de gerenciamento fazem uso dos protocolos de comunicação das camadas inferiores. É usado quando não se é possível ir a todas as sete camadas, em tais casos usa-se o gerenciamento de camadas, porque este

é restrito a uma camada em particular. O gerenciamento de camada não deve duplicar o gerenciamento de sistemas.

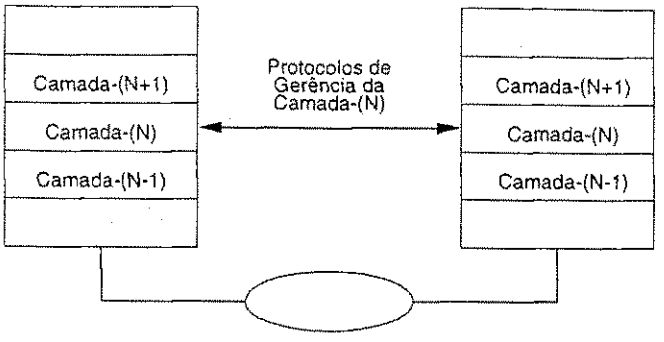


Figura 2.2: Gerência da camada N

- Operação na camada-N:

  Está limitada a monitorar e controlar uma instância de comunicação da informação de gerenciamento dentro de uma única camada.

2.1.5 Modelo de Gerenciamento da Informação

Um OG é a maneira que modelo OSI descreve os recursos com o propósito da gerência. Um recurso pode ser descrito como um objeto gerenciado. Um objeto gerenciado é uma instanciação de uma classe de objeto gerenciados(*Managed Object Class* - MOC), o MOC é constituído de uma coleção de pacotes aplicativos(*packages*), que podem ser obrigatórios ou condicionais. O que difere os objetos instaciados são os valores das características desses objetos. Com o propósito de controlar as atividades dos recursos, o gerente e os agentes devem ter conhecimento dos detalhes destes recursos, que estão armazenados na Base de Gerenciamento da Informação (MIB).

2.1.6 Áreas Funcionais para o Gerenciamento de Sistemas OSI

Gerência de Faltas

O gerênciade faltas é usada para detectar, isolar e reparar problemas. Envolve atividades tais como a habilidade de rastrear faltas através do sistemas, apresentar diagnóstico e atuar sobre as detecções de erros de modo a corrigir as faltas. Este tipo de gerência também é responsável pelo uso e gerência dos registros de ocorrências (*logs*).

Gerência de Contas(*Accounting*)

Esta facilidade é necessária em qualquer ambiente compartilhado, pois define como o uso da rede, taxas e custos devem ser identificados no ambiente OSI. Permite que usuários e gerentes estabeleçam limites ao uso e solicitação de recursos adicionais.

Gerência de Configuração

É usada para identificar e controlar os objetos gerenciados. Define os procedimentos para inicialização, operação e liberação das EOGs além dos procedimentos para reconfiguração destes. Também é usado para associar nomes a EOGs e atribuir valores aos parâmetros destes objetos. Por último, está área de gerência coleta dados sobre as operações num sistema aberto, a fim de reconhecer uma mudança de estado do sistema.

Gerência de Segurança

Esta facilidade é responsável pela segurança dos objetos gerenciados, provendo as regras para os procedimentos de autenticação, das rotinas de controle da manutenção do acesso, oferecem suporte na gerência das chaves de criptografia e manutenção dos registros de ocorrências de segurança.

Gerência de Desempenho

Prestam suporte a todas as atividades que evolvem a coleta de dados estatísticos sobre o desempenho do sistema e aplica esses dados a rotinas de análise de desempenho

do sistema. A mesma permite o uso de modelos para estabelecer se um sistema está fornecendo a vazão devida, se provê um adequado tempo de resposta ou se o sistema está sobrecarregado e, finalmente, se o sistema está sendo usado de maneira eficiente.

2.2 Linhas Gerais para a Definição de Objetos Gerenciados(GDMO)

As Linhas Gerais para a Definição de Objetos Gerenciados estabelecem critérios a serem adotados, quando da definição destes objetos (OG). Esses critérios(arcabouços) estão definidos da seguinte forma:

- Classe de Objetos Gerenciados(MOC):

  Neste esqueleto, relações de herança, necessárias ao reuso de características de outros OGs são definidos. Os MOCs contêm pacotes aplicativos, atributos, grupos de atributos, ações e notificações. A Figura 2.3 detalha o conteúdo utilizado na definição de uma MOC.

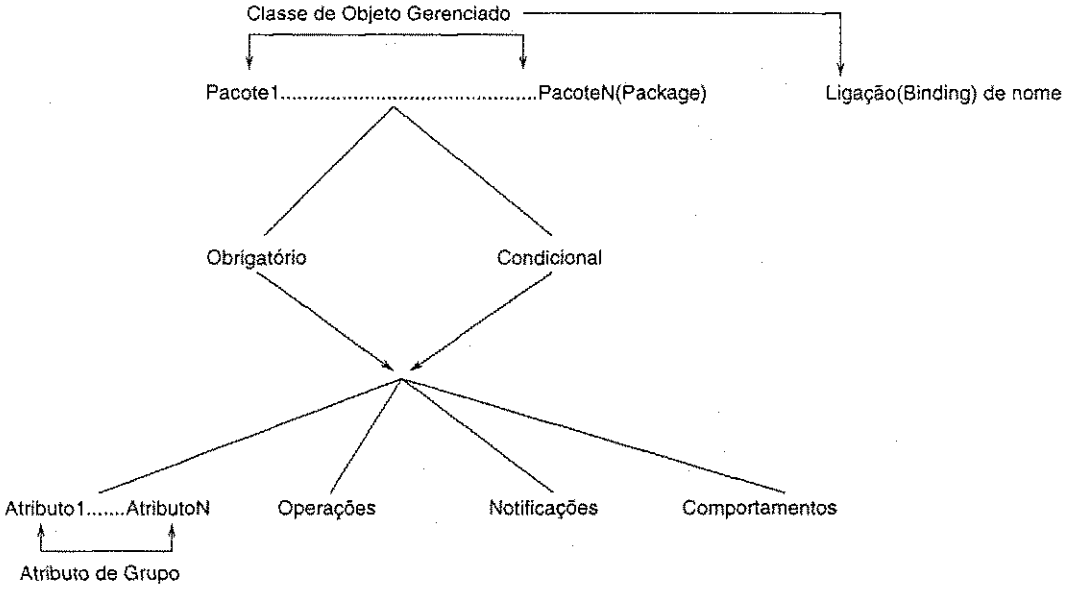


Figura 2.3: Definição de classes de objetos gerenciados

- Pacotes Aplicativos(packages):

  Atributos, Grupos de Atributos, Operações, Notificações, Definições de Comportamentos e Parâmetros são agrupados com o fim de formar um esqueleto identificável. Um pacote aplicativo pode ser inserido dentro de arcabouços MOC.

- Atributos:

  Este esqueleto é usado para incluir e fornecer uma sintaxe ao atributo, regras de teste para valores do atributo, comportamentos, identificadores e parâmetros.

- Grupo de Atributos:

  Os atributos são agrupados por uma questão de conveniência. Estes grupos indicam um conjunto de atributos para o grupo e um valor identificador para cada um destes atributos de grupo.

- Ações:

  Este esqueleto é usado para definir o comportamento e sintaxe de tipos de ações, que são utilizadas no CMIS (*Common Management Information Service*) M-ACTION.

- Comportamento:

  Este esqueleto é utilizado com o fim de estender a semântica dos arcabouços previamente definidos. É útil para explanação mais elaborada de MOCs, ligações de nomes (*binding*), atributos, parâmetros, notificações e ações, definidas em outro lugar.

- Notificação:

  Notificações que são levadas pelo serviço CMIS M-EVENT-REPORT são definidas neste esqueleto.

- Parâmetro:

Parâmetros utilizados na definição de atributos, notificações e operações são definidos neste esqueleto.

- Ligação de nomes (name binding):

  Este esqueleto é utilizado para nomear de maneira única um objeto gerenciado, ele especifica o atributo de nomeação usado para nomear e identificar um objeto superior.

Nas seções seguintes introduziremos os conceitos básicos relacionados à Interface M4 e seu relacionamento com a Interface TMN Q3. Apresentamos ainda a análise dos modelos da entidades gerenciadas formalizadas através de G-CPN e simuladas na ferramenta CPN-Design.

2.3 Visão de Rede da Interface M4

O documento AF-NM-0058.000 do ATM Forum[Com96, Com94] especifica os requisitos funcionais para uma interface capaz de gerenciar Elementos de Rede ATM (ERs, ATM Network Elements (NEs)), denominada Interface M4. A função desta interface pode melhor ser compreendida observando o documento de referência de arquitetura do ATM Forum (The ATM Forum Management Interface Reference Architecture), como mostrado na Figura 2.4.

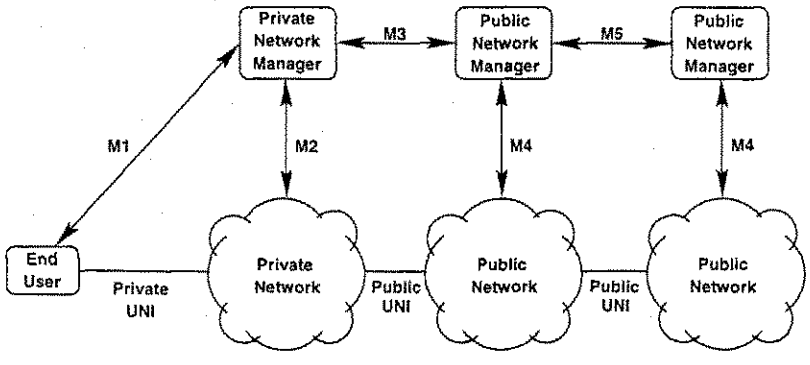


Figura 2.4: Arquitetura de referência para a interface de gerência do ATM Forum

A Interface M4 define duas visões: a Visão de Elemento de Rede (NE) e a Visão de Rede (VR). A Visão de Elemento de Rede engloba o gerenciamento de ERs ATM. A Visão de Rede engloba o gerenciamento de ERs agregados como uma ou mais sub-redes. A arquitetura lógica é mostrada na Figura 2.5.

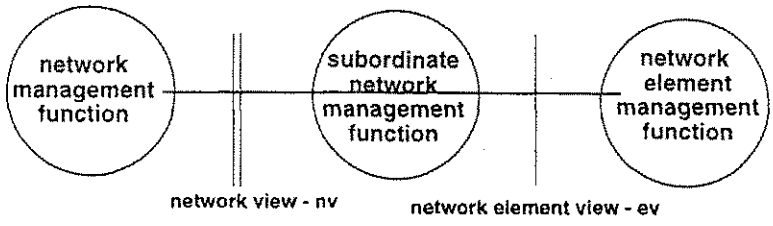


Figura 2.5: Visão de rede

Na Figura 2.5 é apresentado como o gerenciamento de rede utiliza a visão de rede para comunicar-se com a função subordinada de gerência de rede. Para efeito de completude esta visão incorpora uma visão de elemento de rede entre uma função de gerenciamento de sub-rede e função de gerenciamento do elemento. Um Sistema de Gerenciamento de Rede pode utilizar entidades gerenciadas na visão de rede, na visão de elemento de rede, ou ambas. Neste documento discutimos a visão de rede M4. Além disso, é possível relacionar a visão de rede com a visão de elemento de rede. Do ponto de vista de arquitetura é possível oferecer somente uma visão de elemento de rede ou uma visão dos serviços de gerenciamento de rede. O documento AF-NM-0058.000 [Com96] aborda somente a funcionalidade da interface para gerenciar a rede, não provendo requisitos dos sistemas de gerenciamento.

2.4 Arquitetura de Rede e Elemento de Rede

No nível de arquitetura de rede o SGR possui a habilidade para gerenciar entidades de um único elemento de rede (ER) para o Sistema de Gerenciamento de Sub-Rede (SubSRG) (Subnetwork Management System(SubNMS)), neste caso a interface M4 entre o ambiente do SGR e o SubSGR apresenta também uma visão de elemento de rede ATM. Nesta arquitetura, o SGR possui a habilidade para ver e gerenciar

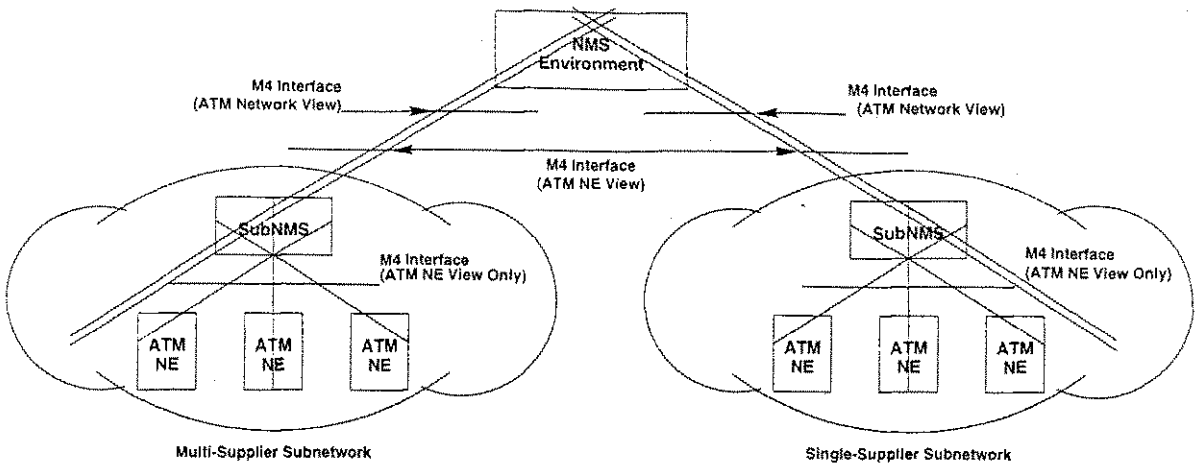


Figura 2.6: Relação entre as arquiteturas de rede e elemento de rede

a rede ATM executando operações na sub-rede como um todo ou desempenhando operações em elementos de rede ATM selecionados. Na Figura 2.6 ilustra-se estas relações. Utilizando a terminologia TMN (Telecommunication Managent Network) a interface M4 é uma Interface TMN Q3[2]. Esta recomendação refere-se a Funções de Operação de Sistema (Operations System Functions (OSF)) a qual mapeia os níveis de Gerenciamento de Rede e Gerenciamento de Elemento de Rede entre um SGR público (ou Funções de Sistema de Operações TMN, e um Elemento de Rede TMN, um Sistema de Gerenciamento de Elemento, ou outros SGRs).

A recomendação M3.100 [IT92c] da ITU-T, define um modelo genérico de informação constituído de três visões: A visão de elemento de rede considera a informação requerida para diferenciar um elemento de rede. Inclui informações necessárias para gerenciar os elementos de rede e seus aspectos físicos. A visão de rede considera a informação representando a rede física e logicamente. Considera como as entidades de elementos de rede são relacionadas, topologicamente interconectadas, e configuradas para prover e manter a conectividade fim-a-fim. A visão de serviço considera como aspectos de visão de rede (tais como, um caminho fim-a-fim) são utilizados para prover

[2]Esta terminologia pode ser encontrada na Recomendação M.3010 da ITU-T, International Telecommunication Union - Telecommunication Standardization Sector

um serviço de rede (por exemplo, disponibilidade, custo, etc), e como estes requisitos são alcançados através do uso da rede, e todas as informações relacionadas ao usuário.

O diagrama de herança (Figura 2.7) demonstra a relação existente entre as entidades descritas no documento AF-NM0073 [Man97], este documento contém a MIB escrita em CMIP da interface M4 segundo a visão de rede. Implementações que utilizam-se apenas da visão de rede devem ser definidas usando os objetos definidos em [Man97].

As relações descritas em negrito na Figura 2.8, representam entidades que são utilizadas na visão de elemento de rede, na modelagem realizada no Capítulo 4, são contemplados apenas aspectos relativoas a visão de rede. Portanto as entidades descritas serão desconsideradas no nosso contexto. A Figura 2.8 demonstra que entidades petecem a que camadas e que entidades estão contidas na descrição de uma determinada entidade. Considerando-se uma visão de Elemento de Rede, apenas as entidades contidas em *Managed Element R1* são vistas pelo SGR.

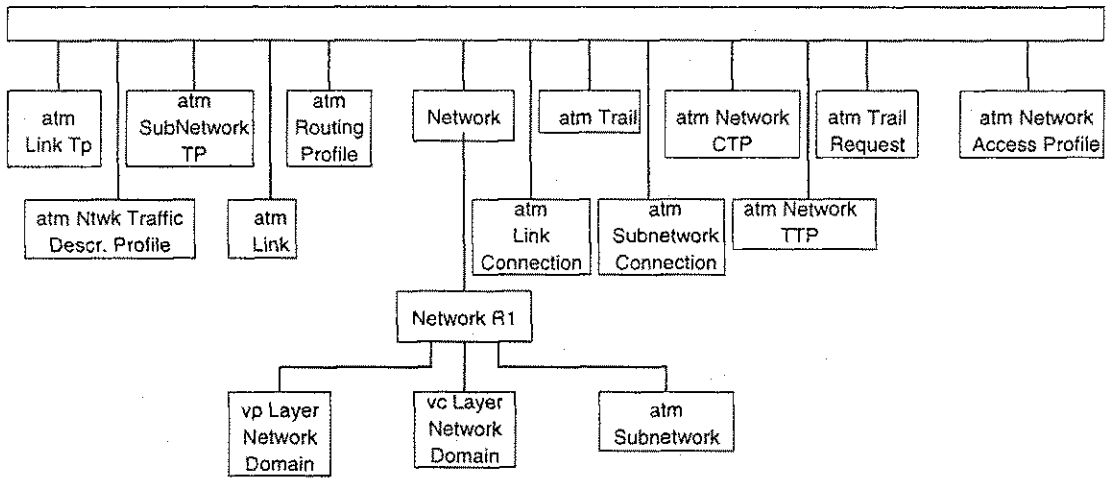


Figura 2.7: Diagrama de herança

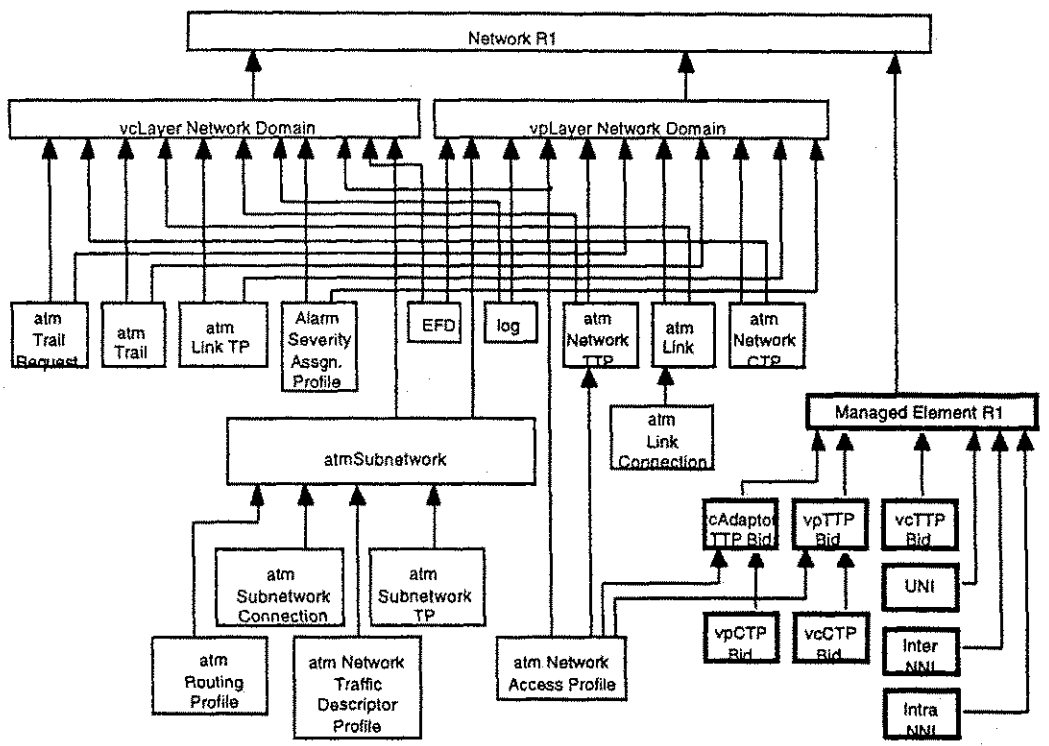


Figura 2.8: Diagrama de contenção

Capítulo 3

Técnicas de Descrição Formal

Aqui são descritas as técnicas formais adotadas na modelagem. Estas técnicas foram escolhidas dentre um vasto conjunto, por apresentarem características que as tornam únicas: oferecerem suporte para descrições Orientadas a Objetos e são(serão) reconhecidas como padrões pelo ITU-T. O tipo de Rede de Petri de Alto nível no qual G-CPN é baseado encontra-se em vias de ser padronizado pelo ITU-T. SDL, no entanto, é um padrão de fato, tanto na indústria, como no ITU-T, dessa maneira, a pesquisa restringiu-se a esses duas técnicas. No entanto, é reconhecida a utilização de outros padrões como Lotos, Estelle e Z, na formalização de processos.

3.1 Sistemas G-CPN

Redes de Petri são uma ferramenta com embasamento matemático rigoroso que permitem a modelagem de sistemas cuja natureza é essencialmente concorrente, assíncrona, distribuída, paralela, não-determinística e/ou estocástica [Mur89]. Além de serem um formalismo matemático, as redes de Petri são uma ferramenta de expressão gráfica e simulação. São gráficas por apresentarem uma notação que permite a completa associação entre elementos de natureza léxica e seus correspondentes sintáticos. A faceta gráfica de redes de Petri permite a melhor interação entre diferentes indivíduos, reduzindo problemas de comunicação. Como as definições semânticas também são ma-

temáticas, torna-se possível a simulação das atividades concorrentes e dinâmicas dos sistemas modelados através da utilização de fichas para demarcar estados dos sistemas. As redes de Petri podem e vêm sendo utilizadas para a descrição e especificação de sistemas de software. Como ferramenta matemática, permitem estabelecer equações de estados, e/ou expressões que compreendam a natureza comportamental dos sistemas, permitindo o rigorismo formal e a precisão na descrição de diagramas de fluxo e de estados, bem como a extração e verificação de características subjacentes. Daí sua larga aplicação na descrição e estudo de sistemas de informação distribuídos e paralelos e em protocolos de comunicação. A simulação do comportamento de sistemas modelados por redes de Petri, permite a visualização da dinâmica do sistema por parte dos desenvolvedores e usuários do sistema, de modo a reforçar a confiabilidade nos requisitos e na especificação. Em geral, a descrição de sistemas complexos reais através de redes de Petri clássicas tornam-se extremamente extensas em termos da estrutura de rede. Percebe-se também que diversos sub-módulos do modelo se repetem inúmeras vezes, caracterizando certa redundância de expressão. Além disso, o modelo clássico não favorece a modelagem de aspectos temporais quantitativos dos sistemas, permitindo apenas estabelecer relações de dependência de eventos em termos de expressões temporais qualitativas. Diversas extensões foram propostas ao modelo clássico com o intuito de sanar as limitações citadas. Uma classe de extensões de redes de Petri agrega a possibilidade de tratamento quantitativo de aspectos temporais dos sistemas. Em geral, as ferramentas permitem a associação de atributos de tempo à semântica do modelo. Assim, torna-se possível especificar além da possibilidade da ocorrência de eventos num sistema, as limitações temporais às quais os eventos são restritos. A classe de redes de Petri temporais é de especial interesse nas atividades de verificação e avaliação de desempenho de sistemas de software [PdFC94]. Uma outra classe de extensões de redes de Petri, denominadas redes de Petri de Alto Nível, propõe diversos mecanismos para a simplificação dos modelos obtidos, através do enriquecimento do poder de expressão. A principal contribuição deste enfoque é a integração das redes

de Petri à Teoria dos Tipos de Dados. Em redes de Petri de Alto Nível, as fichas que denotam o estado do sistema podem representar tipos estruturados de dados, ao invés da simples indicação binária que a presença da ficha representa nas redes Clássicas. Isto permite que os modelos sejam simplificados através da fatoração de subestruturas semelhantes, que agora poderão reconhecer as fichas sobre as quais atuam. As abordagens de redes de Petri de Alto Nível que se destacam são as redes Predicado/Transição (PrT-Nets) [Gen91, Gen89] e as redes de Petri Coloridas (CP-Nets) [Jen87, Jen92]. Entretanto, apesar da introdução das Redes de Petri de Alto nível, a especificação e a modelagem de sistemas complexos pode ainda ser uma tarefa difícil. A principal limitação remanescente é a falta de mecanismos que permitam especificar a estrutura dos sistemas. É natural que projetistas concebam os sistemas através da focalização de determinados aspectos do problema de cada vez, e que a certo nível de abstração desejem uma visão global das diversas partes do modelo, desta forma, estabelecendo como se dá a relação entre as partes, sem se preocupar com detalhes internos de cada uma. Esta forma de abordar a modelagem de sistemas não é suportada pelas redes de Petri de Alto nível convencionais. Como resultado, especificações e modelos de sistemas complexos elaborados através de redes de Petri de Alto Nível têm aspecto essencialmente *plano* onde toda a estrutura faz parte de um único módulo , dado que a natureza dos paradigmas não permite identificar subestruturas do sistema. Para tratar o problema, novas extensões de redes de Petri foram propostas. A idéia inicial era permitir a introdução do conceito de hierarquia, como nos casos das redes de Petri Predicado/Transição Hierárquicas e as redes de Petri Coloridas Hierárquicas. Embora nem todos os mecanismos de abstração importantes e desejáveis sejam hierárquicos, a facilidade de modelagem introduzida permitia controlar melhor a complexidade dos sistemas em estudo. Nos últimos anos, um novo paradigma emergiu como alternativa às atividades de desenvolvimento de sistemas de software: a orientação a objetos. O paradigma é baseado em conceitos simples como objetos, atributos, classes, membros e relações todo-parte. A principal razão do seu sucesso e crescente uso é que esses con-

ceitos básicos associados ao princípio da informação escondida (information hidding) e da abstração, provêem um sólido esquema de estruturação e controle de complexidade para sistemas de software. Como argumento adicional de caráter econômico, a orientação a objetos permite amplamente a reutilização de elementos anteriormente descritos, através dos princípios da herança e da agregação.

No que segue apresenta-se as definições informais e formais mais importantes relacionadas ao modelo de redes de Petri bem como do modelo G-CPN, utilizado nesta dissertação. Detalhes bem como outros exemplos de aplicação do modelo G-CPN podem ser encontrados em [Gue97, GPdF97, GdFP97, GFPdF97, GFaPdF97].

3.1.1 Introdução Informal do Modelo Clássico de Redes de Petri

Redes de Petri são construções matemáticas amplamente utilizadas para a descrição e o estudo de sistemas de processamento de informação que se caracterizem como concorrentes, assíncronos, distribuídos, paralelos, não-determinísticos e/ou estocásticos [Mur89]. A estrutura de uma rede de Petri é um grafo bipartido, direcionado e com pesos. O termo bipartido implica que os nós do grafo pertencem a duas classes disjuntas, denominadas lugares e transições, e que os arcos ligam sempre elementos de classes distintas. O termo direcionado é usado para denotar que os arcos têm origem e destino especificados. Finalmente, é preciso dizer que os arcos também podem ter pesos associados, o que justifica a qualificação *com pesos*. Os pesos dos arcos são valores inteiros que podem ser interpretados como replicações dos arcos, assim um arco com peso w denota a existência de w arcos entre os mesmos nós. A descrição anterior, no entanto, estabelece apenas a natureza sintática de uma rede de Petri. Como o principal objetivo das redes de Petri é a formalização de aspectos dinâmicos de sistemas, é preciso então que seja possível expressar estados de um sistema. Para definir matematicamente o comportamento de uma rede de Petri novos conceitos precisam ser adicionados. Com esse objetivo, associam-se marcas à estrutura da rede de Petri que denotam o estado

em que se encontra o sistema. As marcas são denominadas fichas e podem ser associadas apenas aos lugares da rede. Cada lugar pode ser associado a qualquer número inteiro de fichas. Desta forma, o conjunto total de fichas distribuídas nos lugares da rede denotam um estado, denominado marcação. Deste modo a definição de uma rede de Petri pode ser dada pela união de uma estrutura de rede de Petri (um grafo) a uma marcação inicial (um estado). Está fora do escopo desta dissertação uma discussão mais detalhada sobre o modelo clássico de redes de Petri, o leitor interessado pode consultar [Mur89], para maiores detalhes.

3.1.2 Introdução Informal a Redes de Petri Coloridas

Nesta seção apresenta-se os conceitos básicos das redes de Petri Coloridas (*CPN - Coloured Petri Nets*). O modelo CPN é uma classe de redes de Petri de Alto Nível aplicável à especificação, projeto, simulação, validação e implementação de sistemas complexos de software [Jen92]. Sua ampla utilização deve-se essencialmente à clareza do sólido embasamento matemático, ao enorme leque de métodos formais de análise, verificação e validação e à existência de um grande número de ferramentas de software já plenamente desenvolvidas. Além disso, redes de Petri Coloridas vêm sendo aplicadas aos mais diversos domínios com bastante sucesso. Uma rede de Petri Colorida (CP-Net) é composta essencialmente por uma estrutura, um conjunto de inscrições e um conjunto de declarações. A estrutura de uma CP-Net é semelhante à estrutura das redes Lugar/Transição. Portanto, é também um grafo dirigido e bipartido. Entretanto, ao invés de pesos inteiros, os arcos são associados a inscrições que determinam dinamicamente quantas e quais fichas devem ser removidas ou adicionadas aos lugares associados, na ocorrência de uma transição. Inscrições também podem ser associadas a transições, e permitem restringir ocorrências de eventos a determinadas condições. O estado inicial de uma CP-Net também é determinado por inscrições associadas aos lugares. Cada inscrição é, em geral, uma expressão construída a partir de constantes, variáveis e operadores previamente definidos. O conjunto de declarações de uma CP-

Net serve primordialmente para declarar a natureza dos elementos citados nas diversas inscrições, e por conseguinte, dos objetos manipulados pelo modelo. As inscrições e declarações de uma CP-Net podem, a priori, ser escritas em praticamente qualquer linguagem que tenha sintaxe e semântica bem definidas. Pode-se, por exemplo, usar a notação matemática padrão, embora isso seja de certa forma inconveniente, devido a problemas gerados pelo uso de símbolos. Em geral, CP-Nets vêm sendo utilizadas em associação com uma linguagem denominada CPN-ML, derivada da linguagem funcional Standard ML, cuja sintaxe é bastante semelhante à usada por linguagens de programação convencionais.

No modelo CP-Nets podem ser ainda utilizado um outro tipo de inscrição denominada de guarda. As guardas são associadas às transições, e têm a função de restringir as condições de sua ocorrência. Devido à sua função, as guardas devem ser expressões de natureza booleana, isto é, ao serem avaliadas devem resultar exclusivamente em valores do conjunto verdade. As guardas são representadas graficamente por inscrições entre colchetes ao lado das transições. Por se tratar de uma rede de Petri de Alto Nível, as fichas presentes em CP-Nets sempre transportam algum valor. Cada valor deve pertencer ao domínio de algum tipo de dado bem definido nas declarações. Por razões históricas, em CPN usa-se a expressão conjunto de cores (*colour set*) em substituição a tipo de dados e, por conseqüência, cada valor é denominado de cor (*colour*). Desta forma, cada lugar na estrutura interna é associado a um conjunto de cores, que indica o tipo de fichas que o lugar pode conter. A linguagem CPN-ML dispõe de mecanismos que permitem a definição de conjuntos de cores relativamente complexos, além de oferecer diversos conjuntos previamente definidos.

Um conceito fundamental para o estabelecimento das CP-Nets é a idéia de multi-conjuntos (multi-sets). Multi-conjuntos são construtos semelhantes a conjuntos, exceto pelo fato de que podem conter múltiplas aparições de um mesmo elemento. Todo multi-conjunto é definido sobre um conjunto a partir do qual os elementos são tomados. A título de exemplo, os multi- conjuntos a seguir foram criados sobre o conjunto dos

números naturais: $\{0, 2, 3, 7\}$, $\{0, 1, 1, 3\}$, $\{0, 1, 1, 2, 3, 3, 7\}$, $\{\}$. Uma marcação para uma CP-Net é uma distribuição de fichas nos seus lugares. A marcação de cada lugar é definida como um multi-conjunto sobre o conjunto de cores associado ao lugar.

É comum usar a expressão variáveis da transição para nos referirmos ao conjunto de variáveis presentes nas inscrições dos arcos e na guarda da referida transição. Outro conceito de extrema importância em redes de Petri Coloridas é o de ligação (*binding*) de uma transição. Uma ligação de uma transição pode ser vista como a substituição de cada variável da transição por uma cor (valor). É requerido, entretanto, que as cores pertençam aos conjuntos de cores apropriados e que impliquem na avaliação da guarda como verdadeira. Estabelecidos os conceitos de marcação e ligação podemos definir como se comportam as redes de Petri Coloridas. Em cada marcação, a ocorrência de uma transição sob uma determinada ligação é dita habilitada se todos os seus lugares de entrada tiverem fichas suficientes para satisfazer às expressões dos arcos. Cada expressão deve ser devidamente avaliada segundo as substituições determinadas pela ligação, a fim de determinar quantas e quais fichas são requeridas nos lugares de entrada. Caso a transição ocorra, então são retiradas fichas dos lugares de entrada e depositadas novas fichas nos lugares de saída. A quantidade de fichas é determinada também pela avaliação das expressões dos arcos segundo as substituições implicadas pela ligação.

3.1.3 Módulos G-CPN

O objetivo principal desta seção é introduzir os conceitos e as definições relativas à estrutura G-CPN. Inicialmente apresenta-se uma descrição informal de Sistemas G-CPN acompanhada de um exemplo simples e em seguida uma seqüência de definições formais que evoluem de forma incremental a partir da definição consolidada de redes CPN [Jen92] em direção à estrutura para Módulos G-CPN. É importante destacar que as definições foram elaboradas objetivando abstrair ao máximo a natureza do modelo, sem deixar-nos influenciar por possíveis questões de implementação de um

Sistema G-CPN. G-CPN é uma estrutura de redes de Petri baseada em objetos para a especificação e modelagem de sistemas distribuídos de software. O Modelo G-CPN foi elaborado partindo dos princípios que originaram a estrutura G-Net, introduzida em [Den92, DCdFP93b] e nas redes de Petri Coloridas [Jen92].

O modelo G-CPN permite o desenvolvimento incremental e não-monotônico da especificação de um sistema de software, favorecendo as condições de manutenção e de reusabilidade. Isto ocorre pela utilização dos conceitos de módulos fracamente acoplados, interfaces bem-definidas e informação escondida, que resultam no fato de que um módulo só pode ter acesso a informações e métodos de outros módulos no sistema através de um mecanismo de abstração. Dado que é altamente improvável que as primeiras tentativas de estabelecer soluções para sistemas complexos resultem imediatamente corretas [Boo94], a possibilidade de efetuar alterações, de permitir o ir e vir na especificação, aproveitando ao máximo sub-partes consideradas corretas, é essencial em ferramentas de análise e síntese. A formalização de G-CPN foi construída de forma a preservar aspectos gerais da natureza sintática das redes CPN [Jen92, Jen87], por exemplo, a mesma linguagem funcional é utilizada para as inscrições das redes: CPN-ML [Uni96]. Isto significa, que a certo nível, a modelagem em G-CPN se assemelha à modelagem em CPN. Muito embora, o comportamento das redes G-CPN seja diferente, implicando em definições semânticas diferentes. Do ponto de vista intuitivo, o comportamento associado a um módulo G-CPN é perfeitamente dedutível para desenvolvedores habituados às redes CPN.

3.1.4 Introdução Informal aos Sistemas G-CPN

Sistemas G-CPN são conjuntos de módulos G-CPN concorrentes, cooperantes e fracamente acoplados cujo objetivo comum é a modelagem/especificação formal e executável de sistemas distribuídos de software. É importante estabelecer a diferença entre um Sistema G-CPN e um Módulo G-CPN ou simplesmente uma G-CPN. Estruturalmente um Sistema G-CPN é a integração de um conjunto de módulos e um ambiente de

interação. Cada Módulo G-CPN, ou cada G-CPN, define um objeto instanciável do sistema. O ambiente de interação é o elemento responsável pela semântica associada ao sistema. Em se tratando da modelagem de sistemas distribuídos, o ambiente pode ser visto como a abstração da rede de comunicação, sendo o responsável pela semântica de transferência de mensagens entre os módulos. Quando um módulo requisita um serviço de outro módulo, o ambiente de interação é responsável pelo transporte das mensagens de requisição e de resposta. Desta forma, do ponto de vista do módulo, a transferência pode ser abstraída, bastando indicar qual o serviço desejado e o identificador do módulo que o oferece. Uma outra característica relevante de módulos G-CPN, é o fato de poderem efetuar chamadas recursivas. O modelo de interação entre módulos G-CPN é tipicamente cliente-servidor [Sim96]. Quando a requisição da execução de um serviço é atendida pelo ambiente, diz-se que uma invocação ocorreu. A partir daí, o módulo apenas espera que o ambiente faça efetivamente a ativação do serviço remoto, e passa a esperar a chegada de resposta. O ambiente, de posse das informações necessárias, efetua uma ativação do método invocado no módulo servidor através da passagem dos dados necessários ao método. O método servidor, agora ativo, atende ao pedido. A estrutura interna responsável por atender o pedido, é sintaticamente uma rede CPN. Quando for obtido algum resultado, o ambiente detecta a disponibilidade desses resultados (fichas em algum lugar pré-determinado) e os retoma a fim de retransmití-los ao chamador, este evento é denominado terminação. Após a ocorrência da terminação no módulo servidor, o ambiente detecta o módulo cliente correspondente e devolve corretamente os dados, forçando uma ocorrência do último evento associado à interação, denominado retorno.

Os quatro eventos associados à interação entre dois módulos, mostrados na Figura 3.1, são: invocação, ativação, terminação e retorno. É importante ressaltar que todos os eventos ocorrem a nível de ambiente ou Sistema G-CPN, entretanto apenas dois deles ocorrem em cada um dos módulos envolvidos. Uma invocação e um retorno ocorrem no módulo cliente, enquanto que uma ativação e uma terminação ocorrem

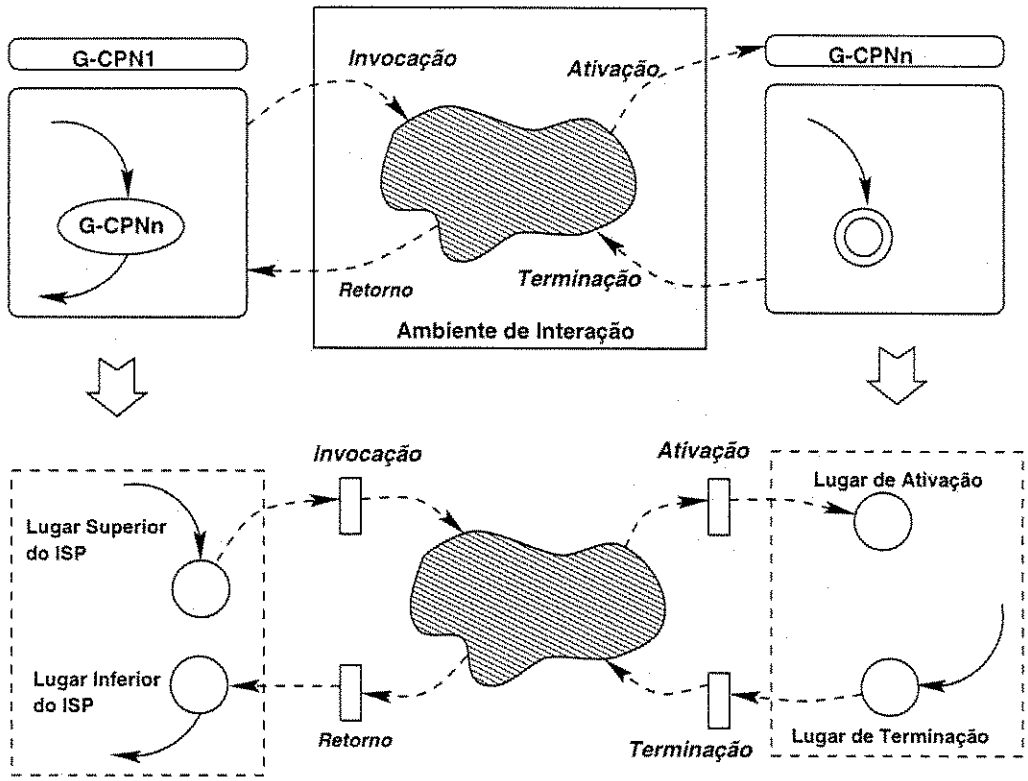


Figura 3.1: Sistema G-CPN e eventos na interação

no módulo servidor. Cada módulo G-CPN é estruturalmente representado por duas subestruturas: a primeira representa a abstração do módulo e é denominada GSP (Generic Switching Place) ou simplesmente interface; a segunda é a estrutura interna do módulo - IS (*Internal Structure*). A interface determina a visão do módulo para o resto do sistema. Nela estão declarados os atributos, e métodos encapsulados. Para cada método declarado na interface, existem dois lugares diferenciados na IS: o primeiro determina onde serão colocadas as fichas de ativação do método; e o segundo especifica qual o lugar de terminação. A estrutura interna é (sintaticamente) uma rede de Petri Colorida.

Para representar invocações de métodos remotos na IS, utiliza-se um lugar especial denominado ISP (*Instantiating Switching Place*). Seu funcionamento é intuitivo: quando uma ficha é depositada, uma invocação pode ocorrer, implicando no desaparecimento da ficha; em seguida, dependendo do funcionamento do método remoto, deverá ocorrer um retorno, e uma outra ficha com novos dados será depositada no ISP, permitindo a habilitação das transições de saída.

Os lugares normais, bem como atributos e lugares que indicam ativação de métodos são representados graficamente por círculos, veja a Figura 3.2. Os lugares de terminação são representados por círculos de borda dupla. ISPs são denotados por elipses e uma inscrição interna indicando o método e a G-CPN a invocar. Os demais elementos (transições e inscrições) seguem a mesma notação usada em CPN. Uma exceção deve ser notada: dois conjuntos de cores são associados a cada ISP, um para fichas de invocação e outro para fichas de terminação. Um único módulo G-CPN pode atender a qualquer número de pedidos de serviços concorrentemente. Isso é possível, porque ao ocorrer uma ativação, uma nova instância do módulo, criada automaticamente, será encarregada de atender o pedido. Cada instância é tratada pelo que denomina-se de contexto. Um contexto pode ser visto como um novo objeto cuja estrutura funcional é uma cópia fiel da estrutura interna do módulo invocado. A única exceção é quanto aos lugares que representam atributos, que não serão copiados, mas sim compartilhados.

O contexto é automaticamente destruído quando não restarem mais fichas relativas àquela invocação na rede.

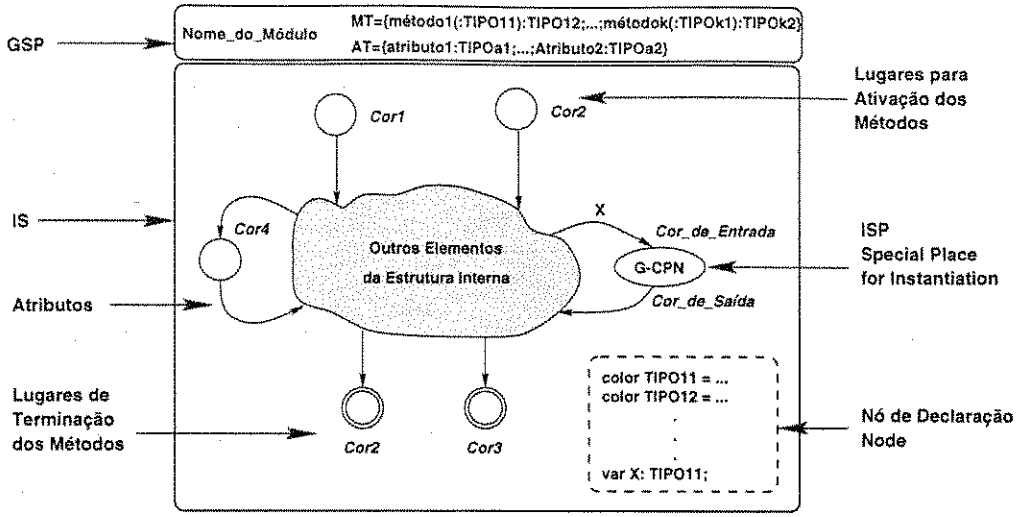


Figura 3.2: Notação para módulo G-CPN

Os atributos em módulos G-CPN são associados a lugares na estrutura interna. Como atributos são considerados parte do estado global do objeto, o seu estado (marcação) é único e portanto, compartilhado por todos os contextos do mesmo módulo G-CPN. Conseqüentemente, os diversos contextos ativos concorrem pela utilização dos atributos (fichas nos lugares que os representam). Esta forma de definir o modelo G-CPN permite que durante a modelagem, um módulo G-CPN represente ou um objeto propriamente dito, com características intrinsecamente concorrentes, ou uma classe, que a cada invocação constrói objetos idênticos. Neste caso, os atributos declarados devem representar atributos de classe.

3.1.5 Um exemplo de Sistema G-CPN

Nesta seção aborda-se o clássico problema produtor-consumidor como exemplo da aplicação de Sistemas G-CPN. A nossa abordagem é baseada na solução proposta em [DCdFP93b, PdFC94]. O problema produtor-consumidor, bastante difundido na literatura sobre comunicação inter-processos e sobre sistemas distribuídos, consiste na

[illegible] conhecido como o problema do buffer

sincronização entre processos produtores e processos consumidores. Cada processo produtor gera itens que devem ser enviados a um consumidor através do uso de mensagens ou espaço compartilhado, daí porque é também conhecido como o problema do buffer limitado (bounded buffer).

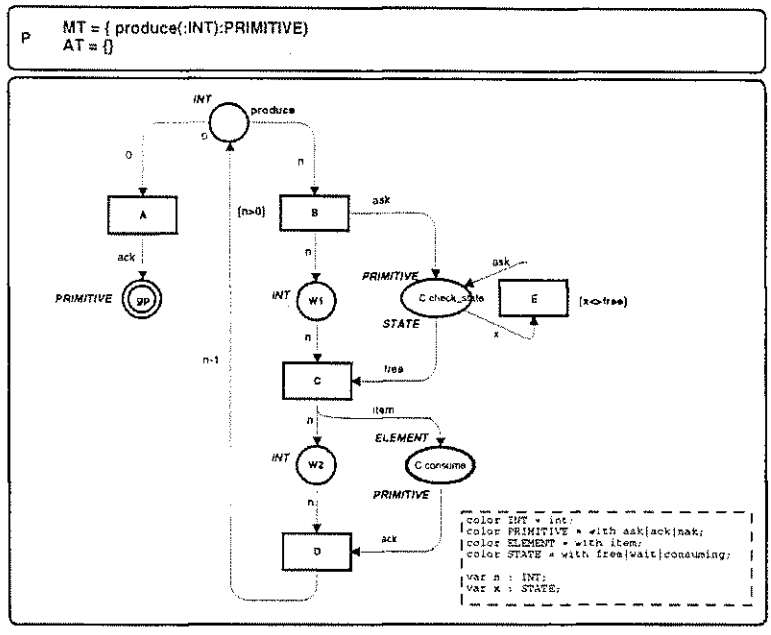


Figura 3.3: Módulo produtor para o sistema G-CPN produtor/consumidor

Através do modelo G-CPN, são mostrados os módulos que modelam o comportamento de processos produtores na Figura 3.3 e consumidores (Figura 3.4). Observando a interface do módulo produtor pode-se verificar que apenas um método existe. O lugar de mesmo nome na estrutura interna determina o lugar de ativação, onde a ficha de inicialização será depositada. O lugar com borda dupla indica a terminação do método. O método deve ser acionado, a fim de que o objeto produza um certo número de itens e os envie para consumo. O módulo produtor não possui nenhum atributo. O funcionamento do método é simples: ao ser ativado, a ficha de inicialização que transporta um número inteiro indica quantos itens devem ser produzidos. Se o valor dessa ficha for 0, apenas a transição A pode ocorrer, o que finaliza a atividade do método. Caso o valor

da ficha seja positivo, a transição B dispara e coloca fichas nos lugares de saída com valor n no lugar W1 e valor ask (ver definições de cores no nó de declaração) no ISP, invocando o método C.check_state. O método check_state do objeto C deve retornar uma indicação do estado do consumidor. Dado que a cor de retorno do ISP é STATE, apenas três respostas são possíveis: *free* indicando que o consumidor estava livre, e que agora espera pelo envio de um item para consumo; *wait* indicando que o consumidor está em espera pelo envio de algum ítem de outro produtor; e *consuming* indicando que o consumidor está em processo de consumo e portanto não pode atender a novos pedidos. Caso a resposta seja diferente de *free* o produtor entra em loop através do disparo da transição E que repetidamente envia *asks* para o método C.check_state. Quando a resposta obtida for free, a transição C ocorre, implicando na produção de um item, que será enviado para o método *C.consume* do consumidor, e no depósito de uma nova ficha de valor n no lugar W2. Espera-se que o método *consume* retorne apenas um valor: *ack* indicando o consumo e a liberação do consumidor. Após isso, a transição D ocorre, colocando uma nova ficha de valor n-1 no lugar *produce*, repetindo o ciclo até que não haja mais itens por produzir. O módulo consumidor por sua vez tem dois métodos declarados na interface: *check_state* e *consume*. Apenas um atributo é utilizado no módulo consumidor e indica o estado do único consumidor no sistema. Esta forma de definir o sistema implica que o módulo produtor G-CPN modela uma classe de produtores, enquanto o módulo consumidor modela um objeto específico. Isto caracteriza alternativas de modelagem disponíveis para o desenvolvedor de sistemas.

O método check_state funciona como esperado pelo objeto produtor: ao chegar uma ficha no lugar de ativação, que deve conter um ask, habilita- se a transição F. Quando F ocorre, o estado do consumidor é ligado à variável x. Caso x seja igual a free, o novo estado do consumidor passa a ser wait. Caso x seja diferente de free, o estado será mantido inalterado. Em qualquer caso, o estado retornado (ficha depositada no lugar de terminação) será x, igual ao estado anterior do consumidor. A presença de uma ficha de cor ELEMENT no lugar de ativação do método consume habilita a ocorrência da

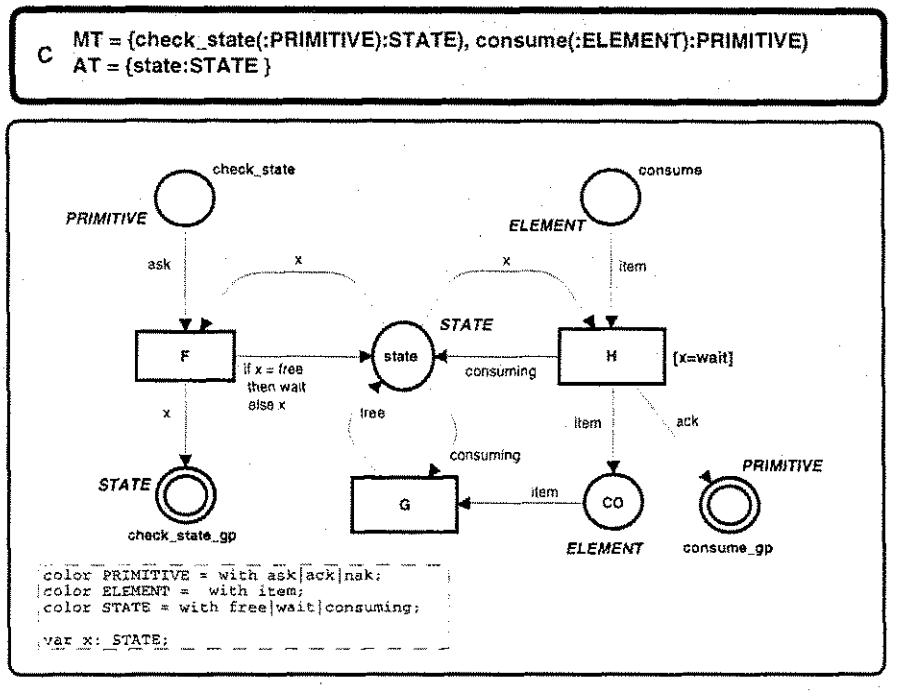


Figura 3.4: Módulo consumidor para o sistema G-CPN produtor/consumidor

transição H, desde que o estado do consumidor seja wait o que é assegurado pela guarda da transição. Se a transição H ocorre, então três fichas são colocadas nos lugares de saída: o estado do consumidor é estabelecido como consuming, uma confirmação ack liberando o produtor é depositada no lugar de terminação, e o processo de consumo propriamente dito é iniciado através do depósito de item no lugar CO. Ao término do consumo, a transição G libera novamente o estado do consumidor, mudando seu estado de consuming para free. O depósito de uma ficha no lugar de terminação antes da finalização da ação do método permite obter maior proveito da natureza concorrente do sistema, tornando possível ao produtor voltar a produzir enquanto o consumidor exerce seu papel. Embora trate-se de um exemplo simples, a especificação e a modelagem formais deste sistema através de G-CPN e a sua transformação em um sistema CPN puro, através da técnica abordada no Capítulo 4 desta dissertação, permitiram-nos a verificação de diversas características desejadas, assim como a extração de novos conhecimentos sobre o sistema. Além disso, diversos erros semânticos foram corrigidos no decorrer do desenvolvimento da modelagem através da interação com a ferramenta CPN. Como exemplo, pode-se citar o fato de que a necessidade de estabelecer uma marcação inicial para o atributo STATE no módulo consumidor (o que não foi deixado explícito) só foi percebida na primeira tentativa de simulação, quando nada funcionou como esperado! Além disso, avalia-se o grafo de estados do sistema para diversas situações com vários produtores concorrentes. O que se observou foi a inexistência de bloqueios e o fato de que a marcação final é sempre a mesma: o estado do consumidor volta sempre a free e nenhuma ficha é acumulada noutros lugares!

3.1.6 Definições Formais para Módulos G-CPN

A fim de completar o modelo G-CPN como proposto informalmente na Seção 3.1.4, faz-se necessário acrescentar algum mecanismo de interação entre módulos. Isto será feito através da introdução de um novo elemento sintático e novos eventos que permitem que um módulo utilize serviços externos. O funcionamento do elemento sintático

necessário, denominado ISP, bem como os eventos a ele associados (invocações e retornos) foram informalmente expostos na Seção 3.1.4. Esta seção apresenta a versão final da formalização para um Módulo G- CPN. A fim de tornar esta seção independente das demais, todas as definições necessárias foram aqui dispostas, apesar de eventuais semelhanças entre estas e as de módulos anteriormente descritos. Finalmente, esta formalização e a nomenclatura associada será usada como base para a definição formal de Sistemas G-CPN na Seção 3.1.8.

Sintaxe

Com o objetivo de preservar a definição sintática da estrutura interna inalterada, será necessário incluir os novos elementos sintáticos na interface do módulo, e associá-los a elementos normais da estrutura interna. Embora, nenhum novo elemento seja adicionado à IS, algumas restrições serão impostas aos elementos que os representem, a fim de que se possa inferir o comportamento adequado às invocações de serviços remotos. Por se tratar de um elemento de comunicação, o ISP deve ser declarado no GSP do módulo. Na estrutura interna, cada ISP será devidamente decomposto em dois lugares. O primeiro lugar representa a parte superior do ISP na qual são depositadas as fichas indicando intencionalidade de invocação de serviços remotos. O segundo lugar representa a parte inferior do ISP na qual são depositadas as fichas de retorno do serviço invocado. Uma função presente no GSP permite interligar a declaração do elemento invocador aos seus respectivos lugares na IS. Outra função na interface mapeia cada ISP a um serviço prestado externamente.

No que segue introduz-se os conceitos formais básicos para um módulo G-CPN. Detalhes são omitidos mas podem ser encontrados em [Gue97, GPdF97, GdFP97].

Definição 3.1 *Sintaxe de um Módulo G-CPN*

Um Módulo G-CPN é uma dupla $\langle GSP, IS \rangle$ onde GSP, também denominado de *interface* do módulo é a tupla $\langle MT, AT, ISP, AP, GP, UP, LP, OBJ \rangle$ e IS, a *estrutura interna*, é a tupla $\langle \Sigma, P, T, A, N, C, G, E, I \rangle$, que satisfazem os seguintes critérios:

Relativo à interface - GSP:

1. MT é um conjunto finito e não vazio de *métodos*
2. $AT \subseteq P$ é um conjunto finito de *atributos*
3. ISP é um conjunto finito de *invocadores* ou *isp's*
4. $AP : MT \rightarrow P$ e $GP : MT \rightarrow P$ são as funções de *inicialização* e *termicação*, respectivamente, onde $\forall m \in MT : [(AP(m) = p1 \Rightarrow I(p1) = 0) \wedge (GP(m) = p2 \Rightarrow I(p2) = 0)]$
5. $UP : ISP \rightarrow P$ e $LP : ISP \rightarrow P$ as funções dos lugares *superiores* e *inferiores* para os *invocadores*, e $\forall isp \in ISP : [UP(isp) = p1 \wedge LP(isp) = p2 \Rightarrow p1\bullet = \bullet p2 = \emptyset \wedge I(p1) = I(p2) = 0]$.
6. $OBJ : ISP \rightarrow S$ é a função *objetivo* para um *invocador*, onde S é o conjunto de todos os métodos invocáveis.

   Relativo a estrutura interna - IS:

7. Σ é um conjunto finito e não vazio de tipos denominado de *conjunto de cores*
8. P é um conjunto finito de *lugares*
9. T é um conjunto finito de *transições*
10. A é um conjunto finito de *arcos* tal que $P \cap T = P \cap A = T \cap A = \emptyset$
11. $N : A \rightarrow P \times T \cup T \times P$ é uma *função de nós*
12. $C : P \rightarrow \Sigma$ é uma *função de cor*
13. $G : T \rightarrow EXPRESSIONS$ é uma *função de guarda*, tal que $\forall t \in T : [Type(G(t)) = BOOLEAN \wedge Type(Var(G(t))) \subseteq \Sigma]$

14. $E : A \rightarrow EXPRESSIONS$ é uma *função de expressão de arcos*, tal que $\forall a \in A : [Type(E(a)) = C(p(a))_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma]$ onde $p(a)$ é o lugar de $N(a)$.

15. $I : P \rightarrow EXPRESSIONS$ é uma *função de inicialização de expressão*, tal que $\forall p \in P : [Type(I(p)) = C(p)_{MS} \wedge I(p)]$ é fechada.

Observações:

1. A definição da estrutura interna de um módulo G-CPN é idêntica à definição sintática de redes coloridas não-hierárquicas (vide [Jen92]).

2. O conjunto de atributos define os tipos de dados sobre os quais o módulo G-CPN atua. O conjunto de métodos declara e identifica os serviços permitidos através da interface.

3. O conjunto de invocadores contém um elemento identificador para cada ponto na estrutura interna que deva ser considerado sob a semântica de um ISP.

4. As funções de ativação e terminação de métodos associam a cada método dois lugares: o primeiro indica onde devem ser colocadas as fichas de ativação e o segundo indica de onde devem ser retiradas as fichas de terminação. Além disso, os conjunto de cores associados aos lugares de ativação e terminação definem indiretamente os tipos de dados recebidos e retornados pelos métodos. Exige-se que estes lugares não tenham marcação inicial nenhuma.

5. As funções de lugar superior e lugar inferior de invocadores associam dois lugares a cada invocador declarado no GSP. A presença de fichas no lugar superior caracteriza as habilitações de invocação. O lugar inferior é onde devem ser colocadas as fichas relativas a retornos. Lugares superiores não podem ter transições de saída, e lugares inferiores não podem ter transições de entrada. De forma análoga aos lugares de ativação e terminação, também determinam as cores dos dados enviados e recebidos. Exige-se que tenham marcação inicial nula.

6. Finalmente a função objetivo leva cada elemento de invocação - ISP da estrutura interna no identificador do serviço que deve ser invocado. O conjunto S é descrito apenas informalmente a fim de que represente, a nível de módulo, um conjunto de serviços. Na definição de Sistemas G-CPN, o conjunto S será adequadamente formalizado.

A título de exemplo apresenta-se a seguir a versão formal da sintaxe do módulo G-CPN produtor (PROD) do Sistema G-CPN Produtor-Consumidor apresentado na Seção 3.1.5

1. $PROD=\langle GSP, IS \rangle$
2. $GSP=\langle MT, AT, ISPAP, GP, UP, LP, OBJ \rangle$
3. $IS=\langle S, P, T, A, N, C, G, E, I \rangle$
4. $S=\{INT,\ PRIMITIVE,\ ELEMENT,\ STATE\}$
5. $P=\{PRODUCE, W1, W2, UP1, LP1, UP2, LP2, GP\}$
6. $T=\{A, B, C, D, E\}$
7. $A=\{produce\_a,\ produce\_b,\ a\_gp,\ b\_w1,\ b\_up1,\ w1\_c,\ lp1\_c,\ lp1\_e,\ e\_up1,\ c\_w2,\ c\_up2,\ w2\_d,\ lp2\_d,\ d\_produce\}$
8. $N=\{(produce\_a,\ (PRODUCE,\ A)),\ (produce\_b,\ (PRODUCE,\ B)),\ (a\_gp,\ (A,\ GP)),\ (b\_w1,\ (B,\ W1)),\ (b\_up1,\ (B,\ UP1)),\ (w1\_c,\ (W1,\ C)),\ (lp1\_c,\ (LP1,\ C)),\ (lp1\_e,\ (LP1,\ E)),\ (e\_up1,\ (E,\ UP1)),\ (c\_w2,\ (C,\ W2)),\ (c\_up2,\ (C,\ UP2)),\ (w2\_d,\ (W2,\ D)),\ (lp2\_d,\ (LP2,\ D)),\ (d\_produce,\ (D,\ PRODUCE))\}$
9. $C=\{(PRODUCE,\ INT),\ (W1,\ INT),\ (W2,\ INT),\ (UP1,\ PRIMITIVE),\ (LP1,\ STATE),\ (UP2,\ ELEMENT),\ (LP2,\ PRIMITIVE),\ (GP,\ PRIMITIVE)\}$
10. $G=\{(A,\ ''),\ (B,\ 'n > 0'),\ (C,\ ''),\ (D,\ ''),\ (E,\ 'x \neq free')\}$

11. *E={(produce_a, '0'), (produce_b, 'n'), (a_gp, 'ack'), (b_w1, 'n'), (b_up1, 'ask'), (w1_c, 'n'), (lp1_c, 'free'), (lp1_e,' x'), (e_up1, 'ask'), (c_w2, 'n'), (c_up2, 'item'), (w2_d, 'n'), (lp2_d, 'ack'), (d_produce, 'n-1')}*

12. *I={(PRODUCE, ''), (W1, ''), (W2, ''), (UP1, ''), (LP1, ''), (UP2, ''), (LP2, ''), (GP, '')}*

13. *MT={produce}*

14. *AT={}*

15. *ISP={isp1, isp2}*

16. *AP={(produce, PRODUCE)}*

17. *GP={(produce, GP)}*

18. *UP={(isp1, UP1), (isp2, UP2)}*

19. *LP={(isp1, LP1), (isp2, LP2)}*

20. *OBJ={(isp1, 'C.check_state'), (isp2, 'C.consume')}*

3.1.7 Comportamento e Semântica de Módulos G-CPN

As definições a seguir referem-se explicitamente a módulos G-CPN apresentados formalmente pela Definição 3.1. No que segue, as seguintes convenções serão utilizadas:

1. $G = \langle GSP, ISP \rangle$ é qualquer módulo G-CPN

2. $GSP = \langle MT, AT, ISP, AP, GP, UP, LP, OBJ \rangle$ é a interface de G

3. $IS = \langle S, P, T, A, N, C, G, E, I \rangle$ é a estrutura interna de G

4. $ISP = \{isp_1, isp_2, \cdots, isp_i, \cdots, isp_n\}$ é o conjunto de serviços de G

5. M, M_1 e M_2 são marcações alcançáveis de G

6. $Var(t)$ é o conjunto de variáveis associadas à transição t

7. $Type(v)$ é o conjunto de cores da variável v

8. O é o conjunto de contextos.

Definição 3.2 *Elemento de Ficha e Conjunto de Todos os Elementos de Ficha*
Seja O um conjunto infinito de contextos. Um t *elemento* é a tupla $\langle p, c, o \rangle$ onde:

1. $p \in P$,

2. $c \in C(p)$, e

3. $o \in O$, onde O é um conjunto infinito de contextos.

O conjunto de todos os elementos de ficha é denominado TE.

A Definição 3.2 estabelece que um elemento de ficha é determinado por um lugar na estrutura interna do módulo, uma cor pertencente ao conjunto de cores do lugar, e um contexto ao qual a ficha pertence.

Definição 3.3 *Marcação e Sub-marcação de um Contexto*
Uma marcação é um multi-conjunto sobre TE. Se M é a marcação de G, então o multi-conjunto M_o definido como segue é a sub-marcação relativa ao contexto o:

1. $M_o \subseteq M$ e

2. $\forall \langle p, c, o \rangle \in M : \langle p, c, o \rangle \in M_o$

A Definição 3.3 estabelece uma marcação como sendo um multi-conjunto sobre o conjunto de elementos de ficha. A definição de sub-marcação é construída usando a notação de função para multi-conjuntos. A sub-marcação relativa a um dado contexto é dada pela marcação do módulo menos os elementos de ficha cujo contexto não é o referido. Em termos intuitivos, a sub-marcação de um contexto *ignora* fichas dos demais contextos.

Definição 3.4 *Ligação*

Uma ligação de uma transição t é uma função b definida sobre $Var(t)$, tal que:

1. $\forall v \in Var(t) : b(v) \in Type(v)$ e

2. $G(t)\langle b \rangle$, o conjunto de todas as ligações (*bindings*) para t denotado por $B(t)$.

Devido à sua independência da definição de elemento de ficha, a definição de ligação permanece inalterada em relação à definição de mesmo nome para redes CPN. O conjunto de todas as ligações para a transição t será denotado por $B(t)$.

Definição 3.5 *Elemento de Ligação e Conjunto dos Elementos de Ligação*

Seja O um conjunto infinito de contextos. Um elemento de ligação é uma tripla $\langle t, b, o \rangle$, na qual se verifica que:

1. $t \in T$ é uma transição;

2. $b \in B(t)$ é uma ligação pertencente ao conjunto de ligações da transição t;

3. $o \in O$ é um contexto.

A Definição 3.5 estabelece um elemento de ligação como sendo a conjunção entre uma transição e uma ligação específica daquela transição. Dado um elemento de ligação, pode-se identificar de forma única, que substituições ocorrerão, numa determinada transição. O conjunto de todos os elementos de ligação será denotado por BE.

Definição 3.6 *Passo*

Dado um passo $Y \in BE$ de um módulo G. Um multi-conjunto Y_o é o sub-passo relativo ao contexto o ou simplesmente o passo do contexto o se e somente se:

1. $Y_o \subseteq Y$ e

2. $\forall \langle t, b, o \rangle \in Y : \langle t, b, o \rangle \in Y_o$i.

As definições de elemento de ligação e passo são bastante semelhantes às de elemento de ficha e marcação. A tupla de um elemento de ligação também transporta o valor do contexto ao qual é associado, e um sub-passo é definido de forma semelhante à usada para definir uma sub-marcação. Um sub-passo relativo a um dado contexto é dado pelo passo em questão menos os elementos de ligação de outros contextos. Intuitivamente, um sub-passo de um contexto *ignora* os elementos de ligação dos demais contextos.

Definição 3.7 *Sub-passo habilitado*

Um sub-passo Y_o é dito habilitado em uma marcação M_o se e somente se:

$$\forall p \in P : \sum_{(t,b,o) \in Y_o} E(p,t)\langle b \rangle M_o(p)$$

O conceito de sub-passo habilitado tem sua utilidade no sentido de simplificar a definição seguinte. Intuitivamente, um sub-passo está habilitado se os lugares de entrada relativos a todos os elementos de ligação têm fichas suficientes, do contexto em questão, para satisfazer as avaliações das expressões dos arcos.

Definição 3.8 *Passo habilitado*

Um passo Y é dito habilitado na marcação M se e somente se:

1. $\forall o \in O : \forall M_o \subseteq M : Y_o$ está habilitada em M_o
2. $\forall p \in AT : \sum_{(t,b,o) \in Y} E(p,t)\langle b \rangle \leq M(p)$.

Esta definição permite detectar as pré-condições para uma ocorrência de passo num módulo G-CPN. Um passo pode ocorrer apenas se (i) todos os sub-passos estão habilitados e (ii) os lugares que representam atributos têm fichas suficientes para satisfazer simultaneamente as expressões relativas a todos os contextos simultaneamente. A partir da Definição 3.14 até a Definição 3.18 apresenta-se a formalização dos eventos propriamente ditos em módulos G-CPN.

Definição 3.9 *Ocorrência de um Passo*

Um passo Y habilitado na marcação M_1 pode ocorrer. Se ocorre, então a marcação do módulo muda de M_1 para M_2 dada por:

$$\forall p \in P : M_2(p) = (M_1(p) - \sum_{(t,b,o)\in Y} E(p,t)\langle b\rangle) + \sum_{(t,b,o)\in Y} E(t,p)\langle b\rangle$$

Um passo está habilitado se todos seus sub-passos de contextos estiverem habilitados e se existem fichas suficientes nos lugares de atributos para todas as elementos de ligação em Y. Portanto, nunca dois elementos de ligação compartilham ficha alguma em um passo habilitado.

Em um módulo G-CPN, a ocorrência de um passo não é o único evento capaz de modificar as marcações. Cinco tipos de eventos são definidos para um sistema G-CPN: ocorrências de passos ou simplesmente ocorrências, invocações, ativações, terminações e retornos. Invocações, são eventos que removem fichas dos lugares superiores dos *isp's*. Ativações, adicionam fichas com novos contextos nos lugares iniciais dos métodos invocados. Terminações, são eventos que removem fichas dos lugares alvo. E retornos são eventos que adicionam fichas aos lugares inferiores dos *isp's*. A seguir define-se formalmente estes eventos do ponto de vista de um módulo.

Definição 3.10 *Invocação*

Dado um $isp \in ISP$. A invocação de um método $OBJ(isp)$ está habilitada em uma marcação M se e somente se:

$$\exists \langle p, c, o\rangle \in M : p = UP(isp)$$

Definição 3.11 *Ocorrência da Ativação*

Dado um $isp \in ISP$. A invocação de um método $OBJ(isp)$ habilitado pelo elemento de ficha $\langle p, c, o\rangle$ em uma marcação M_1 pode ocorrer. Se ele ocorre, então a marcação do módulo muda de M_1 para M_2 definida por:

$$M_2 = M_1 - \langle p, c, o\rangle,$$

Onde $p \in P$, $c \in C(p)$, e $o \in O$.

Uma invocação pode ocorrer se e somente se uma ficha é depositada no lugar superior de um *isp*. Se ela ocorre, então a ficha será removida.

Definição 3.12 *Habilitação da Ativação*
Uma ativação (habilitada) em uma marcação M_1, pode ocorrer. Se ela ocorre a marcação do módulo é modificada de M_1 para M_2, definida por:

$$M_2 = M_1 + \langle p_1, c_1, o_1 \rangle$$

Onde $p_1 \in P$, $c_1 \in C(p_1)$, $o_1 \in O$ e $\forall \langle p, c, o \rangle \in M_1 : o_1 \neq o$.

Definição 3.13 *Ativação*
Dado um método $m \in MT$. A ativação do método m pode ocorrer num módulo cuja marcação é M_1. Se ocorre, então a marcação do módulo muda de M_1 para M_2 dada por:

$$M_2 = M_1 - \langle p, c, o \rangle.$$

Onde $p \in P$, $c \in C(p)$, e $o \in O$.

A ocorrência da ativação de um método provoca uma única alteração no estado do objeto: acresce uma ficha, denominada ficha de inicialização, no lugar de ativação do método ativado. É interessante perceber que do ponto de vista do módulo, a ativação de um de seus métodos é um evento sempre habilitado.

Definição 3.14 *Terminação*
Dado um método $m \in MT$. A terminação do método m é habilitada na marcação M_1 pelo elemento de ficha $\langle p, c, o \rangle$ se e somente se $p = GP(m)$. Neste caso, a terminação pode ocorrer, se ocorre, a marcação do módulo muda de M_1 para M_2 definida por:

$$M_2 = M_1 - \langle p, c, o \rangle.$$

Definição 3.15 *Invocação*
A invocação de $isp_i \in ISP$ é habilitada na marcação M_1 pelo elemento de ficha $\langle p, c, o \rangle$ se e somente se $p = UP(isp_i)$. Neste caso, a invocação pode ocorrer, se ocorre, a marcação do módulo muda de M_1 para M_2 definida por:

$M_2 = M_1 - \langle p, c, o \rangle$.

Onde $p \in P$, c $\in C(p)$, e $o \in O$

A única conseqüência da ocorrência de uma terminação ou uma invocação num módulo G-CPN, é a eliminação da ficha que habilita o evento.

Definição 3.16 *Retorno*
Um retorno de $isp_i \in ISP$ pode ocorrer num módulo cuja marcação é M_1. Se ocorre, então a marcação do módulo muda de M_1 para M_2 dada por:

$M_2 = M_1 + \langle p_1, c_1, o_1 \rangle$

Onde $p_1 \in LP(isp)$, $c_1 \in C(p_1)$, e $o_1 \in O$

Perceba que a ocorrência de um retorno, em termos formais, é um evento sempre habilitado dentro de um módulo G-CPN. Portanto, do ponto de vista formal, retornos podem ocorrer, mesmo sem ocorrências prévias de invocações. Embora isto não pareça desejável do ponto de vista intuitivo, é interessante deixar a formalização da relação entre os dois eventos para uma instância hierárquica superior. Assim, qualquer que seja o sistema usuário do módulo G-CPN, ele será o responsável por garantir a seqüência adequada dos eventos externos.

Apresenta-se nesta seção as definições formais relativas à estrutura e ao comportamento de um Módulo G-CPN. O módulo assim definido permite a descrição executável de objetos de software em termos de atributos e métodos encapsulados. O acesso às informações se dá exclusivamente através de uma interface denominada GSP que permite a invocação dos métodos. Dentro da estrutura interna, denominda IS, o formalismo permite descrever métodos invocáveis concorrentemente sem que o desenvolvedor ou o projetista precise dedicar esforços especiais para o controle. Além disso, a própria natureza das Redes de Petri permite a descrição de métodos que desempenhem suas funções através de procedimentos intrinsecamente concorrentes. A fim de oferecer maior facilidade em termos de descrição de controle de concorrência, os atributos são

elementos formalizados como lugares de acesso comum na estrutura interna, garantindo que apenas uma linha de execução terá acesso instantâneo. Além disso, através de um elemento de invocação, denominado ISP é possível que descrições internas dos métodos invoquem serviços de elementos externos ao módulo. Tudo isto é possível sem fazer nenhuma suposição sobre o sistema usuário do módulo, exceto que ele manterá a seqüência apropriada de eventos de comunicação. Na seção seguinte define-se a sintaxe para sistemas G-CPN.

3.1.8 Sintaxe de Sistemas G-CPN

De forma intuitiva, um Sistema G-CPN poderia ser formalizado simplesmente como um conjunto de Módulos G-CPN. Entretanto, isso pode levar a diversos problemas de integração entre os módulos e conseqüentemente dificuldades bem maiores no estabelecimento das definições semânticas. Por exemplo, considere o fato de que cada módulo precisa *visualizar* os serviços providos pelos outros módulos. Isto é conseguido através da externalização dos elementos de acesso aos métodos de cada módulo a fim de formar um novo conjunto de serviços, visível para todos os módulos igualmente. Ainda assim, um Sistema G-CPN será definido como uma estrutura que integra um conjunto de Módulos G-CPN num sistema com um objetivo em comum. Entretanto, é preciso esclarecer precisamente como se forma essa estrutura que integra além de módulos G-CPN, alguns componentes adicionais. Além disso, não é qualquer conjunto de módulos G-CPN que pode formar um Sistema G-CPN. É preciso estabelecer critérios que permitam restringir a natureza dos módulos e suas relações para que determinem um Sistema G-CPN. Assim, por exemplo, dois módulos que não usem nenhum de seus serviços mutuamente não apresentarão nenhum problema de integração. Entretanto, se quisermos integrar os módulos A e B nos quais A invoca algum método do módulo B, então é preciso garantir que a passagem de parâmetros seja possível. Além disso, embora tenha ficado implícito, é necessário que cada método seja identificável de forma única em todo o sistema. A definição sintática dada a seguir foi elaborada conside-

rando as observações anteriores. O Sistema deve ser, portanto, um ambiente comum aos módulos, onde cada objeto (módulo G-CPN) pode *ver* os demais objetos e seus serviços. A estrutura formal de um Sistema G-CPN, portanto, deve prover além do conjunto de módulos, um conjunto de identificadores de serviços que cada módulo pode invocar na sua estrutura interna. Além disso a estrutura inclui o conjunto total de domínios sobre o qual o sistema opera. Nas definições que se seguem, utilizamos o operador ponto . já consolidado pelas linguagens orientadas a objetos, para fazer referências a elementos pertencentes aos módulos G-CPN do sistema. Assim, o conjunto de conjuntos de cores dos módulos G_j e G_i, por exemplo, serão indicados por $G_j.S$ e $G_i.S$ ao invés de S_j e S_i.

Definição 3.17 *Sintaxe de um Sistema G-CPN*
Um Sistema G-CPN é a tripla$(\Sigma, S, GCPN)$, onde:

1. Σ é um conjunto finito e não vazio de domínios denominado *conjunto de cores*

2. $GCPN = \{G_1, G_2, \cdots G_n\}$ é um conjunto finito de *módulos* tal que:

   $G_1.P \cap G_2.P \cap ... \cap G_n.P = \emptyset$,

   $G_1.T \cap G_2.T \cap ... \cap G_n.T = \emptyset$, and

   $G_1.A \cap G_2.A \cap ... \cap G_n.A = \emptyset$

   $\forall k, 1 \leq k \leq: G_k.\Sigma \subseteq \Sigma$

3. S é um conjunto de *serviços* do sistema tal que:

   $S = G_1.S \cup G_2.S \cup \cdots \cup G_i.S \cup \cdots \cup G_n.s$

   $\forall isp \in G_i : \forall s \in G_j.S :$

   $Gi.OBJ(isp) = G_j.s \Rightarrow G_iC(UP(isp)) = G_j.C(AP(s)) \wedge G_iC(LP(isp)) = G_jC(GP(s))$

A definição acima estabelece que um sistema G-CPN é formado por um conjunto de módulos G-CPN cujos métodos são unidos para formar o conjunto de serviços do

sistema. É imposto, ainda, que cada módulo defina um subconjunto do conjunto de domínios do sistema como seu próprio conjunto de conjuntos de cores, e finalmente é requerido que todo par invocador-serviço tenha em comum os conjuntos de cores de seus lugares correspondentes. A formalização do sistema mostrado na Figura 3.4 é dada a seguir como exemplo de um Sistema G-CPN descrito usando a definição formal: Seja o sistema Produtor-Consumidor, denotado por SPC, a tripla $\langle S, S, GCPN \rangle$, onde

1. S=$\{INT, PRIMITIVE, ELEMENT, STATE\}$
2. $GCPN$=$\{P, C\}$ onde:

   (* Definições formais dos módulos G-CPN propriamente ditos *)

   $P = \langle P.GSP, P.IS \rangle$

   ...

   $C = \langle C.GSP, C.IS \rangle$

   ...

   (* *)

3. $S = P.S \vee C.S = \{P.produce, C.check\_state, C.consume\}$

Neste documento não detalha-se a semântica de sistemas G-CPN bem como a sua equivalência com o modelo CPN, o leitor interessado pode consultar Guerrero [Gue97, GPdF97].

3.2 Specification and Description Language - SDL

3.2.1 Introdução a SDL

SDL é uma linguagem formal para a especificação e descrição de sistemas. O que entende-se por descrição de um sistema é a descrição do seu comportamento real. Por especificação de um sistema entende-se a descrição do comportamento necessário. SDL

é particularmente voltada para a especificação de aspectos comportamentais de um sistema [IT93c]. SDL tem sido adotada amplamente na indústria, e a razão disso, deve-se basicamente ao poder da linguagem e o fato de ser um padrão, a linguagem nasceu com o objetivo de especificar de maneira formal as especificações do protocolos do CCITT. Atualmente a linguagem tem sido largamente utilizada para a especificação de software em Tempo-Real e de sistemas distribuídos.

A linguagem está totalmente padronizada na recomendação Z.100 do ITU-T [IT93c]. A primeira versão da recomendação foi baseada na revisão aplicada em 1987 e chamada de SDL88. A linguagem sofreu mais uma revisão em 92, chamada SDL92, onde foram adicionadas algumas caracteristícas de Orientação a Objetos. Em 1996 houve mais algumas moficações que tratam basicamente de inconsistências existentes nas especificações anteriores, chamada SDL96 [EHS97]. Uma completa descrição da linguagem encontra-se na recomendação Z.100 [IT93c] e nos livros textos[EHS97, BHS91]. Nas recomendações apêndices I e II [IT93e], são descritos alguns modelos quando da especificação em SDL. Os anexos [IT93d] tratam da descrição formal de SDL, na linguagem Meta-IV.

Uma especificação em SDL define o comportamento de um sistema, em modo estímulo/resposta, assumindo que ambos (estímulo e resposta) são discretos e transmitem informações. De maneira particular um sistema é visto como uma seqüência de respostas a uma dada sequência de estímulos. Tal conceito está baseado em máquinas de Estado Finito, particularmente, Máquinas de Estados Finitos Comunicantes Estendidas, que são representadas por processos. A comunicação é representada através de sinais, que podem ocorrer entre processos, ou entre processos e o ambiente do sistema que está sendo modelado. Alguns aspectos de comunicação entre processos estão relacionados à descrição da estrutura do sistema. Uma EFSM [Liu85] consiste num número de estados e um número de transições interligando estes estados. Um dos estados é designado estado inicial.

SDL provê conceitos de estruturação que facilitam a especificação de sistemas con-

siderados grandes ou complexos. Ao contrário de G-CPN, SDL contém um número grande de conceitos a se dominar, que fornecem ao analista do sistema um poder maior na definição, mas que implica numa maior tempo para aprender a lidar com um vocabulário bem maior do que o existente em G-CPN.

Todo ambiente de Simulação em SDL contém o que convenciona-se chamar de sistema, constituído por um conjunto de blocos interligados. Cada bloco por sua vez é constuído por um ou mais processos, que são basicamente máquinas de estado finito comunicantes estendidas.

A principal propriedade definida por uma especificação em SDL é a definição do comportamento do sistema. Isto é facilmente verificado através da utilização maciça de SDL na especificação de protocolos.

A especificação define como o sistema reage frente aos eventos que ocorrem no ambiente, que por sua vez, são transmitidos através de *sinais* enviados para o sistema. A única forma de interação que um sistema SDL tem com seu ambiente ocorre via sinais (Figura 3.5).

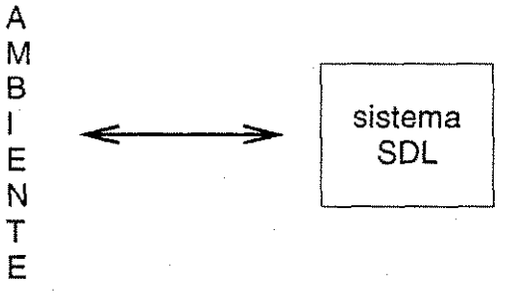


Figura 3.5: Interação da máquina SDL com o ambiente

3.2.2 Comportamento do Sistema

Os processos (Figura 3.6) comunicam-se entre si e com o ambiente via sinais. Não há portanto um hierarquia entre as instâncias de processos, elas coexistem em paralelo, tendo os mesmos direitos. Instâncias de processos podem criar outros processos, mas uma vez criados, os mesmos terão os mesmos direitos.

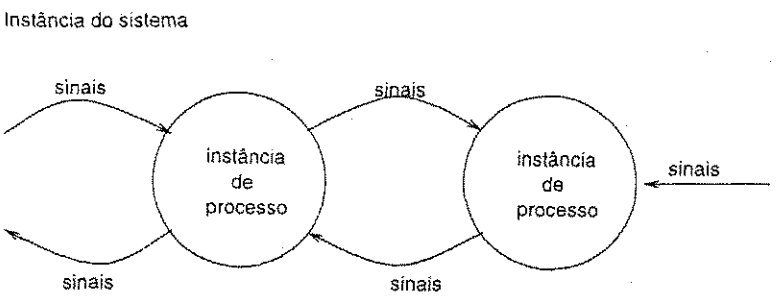


Figura 3.6: Comportamento do sistema

3.2.3 Estrutura de uma instância de um sistema

Como uma maneira de organizar uma grande quantidade de processos de maneira estruturada, SDL provê graus de abstração que permitem lidar de maneira organizada com a complexidade.

Um alto nível de abstração apresenta apenas o modo como os componentes principais de um sistema interagem, enquanto que um baixo nível de abstração apresenta todos os detalhes inerentes àquela parte do sistema. Esta separação entre níveis de abstração permite a gerência da troca de sinais por meio de restrições no modo como componentes do sistemas interagem.

O bloco de construção básico, utilizado para a estruturação de um sistema (Figura 3.7), é conhecido como *block* (bloco), num sistema simples o bloco contém um ou mais conjuntos de processos que se comunicam entre si e com o ambiente do bloco via rota de sinais.

Os blocos, ou são componentes de um sistema ou estão incluídos num bloco que está num nível de abstração mais elevado. Os blocos se comunicam através de canais, estes estabelecem a comunicação da mesma maneira que as rotas de sinais dentro do bloco(Figura 3.8). Os canais conectados a um bloco, estão conectados às rotas de sinais existentes dentro deste bloco. Enquanto as rotas de sinais transferem os sinais de maneira imediata, os canais podem ser especificados como canais com atraso ou canais sem atraso. Canais com atraso experimentam um atraso não determinístico no envio dos sinais. Contudo a ordem dos sinais na fila é sempre preservada.

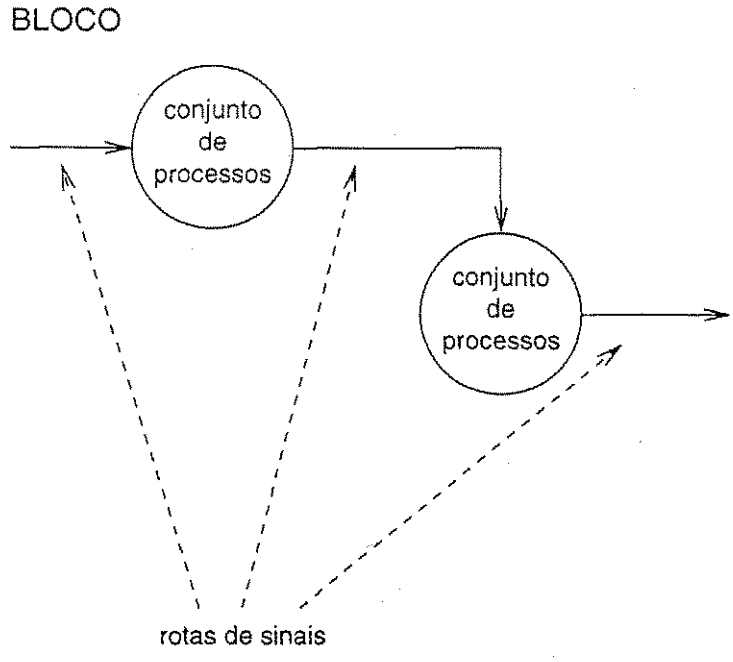


Figura 3.7: Bloco

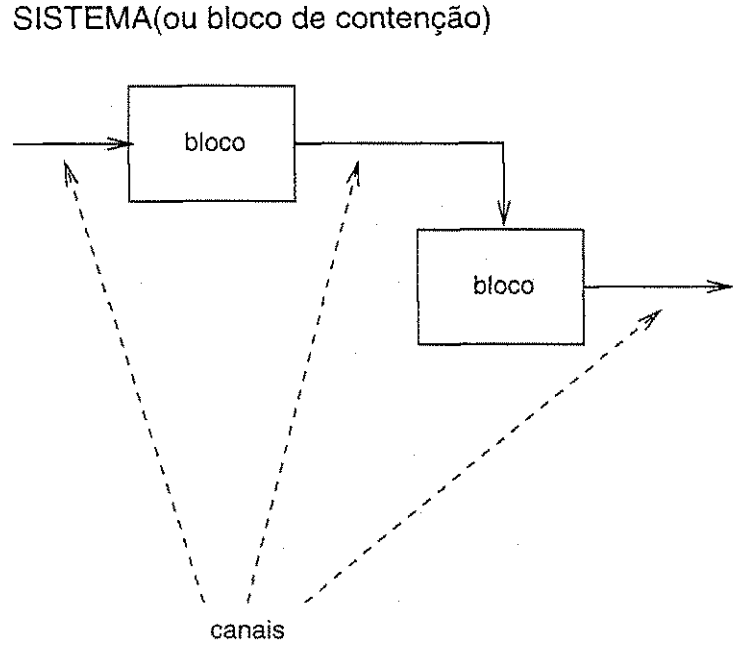


Figura 3.8: Sistema

3.3 Representação Gráfica e Textual

A despeito da maioria da linguagens formais (certamente excluindo-se Redes de Petri) SDL, além da representação textual, apresenta um representação gráfica (Figura 3.9) que permite uma interação mais amigável com o usuário. Portanto, os sistema podem ser definidos de maneira familiar, através de diagramas. Isto é, um conjunto de diagramas interrelacionados que descrevem diferentes níveis de detalhamento, de uma visão geral a uma visão detalhada.

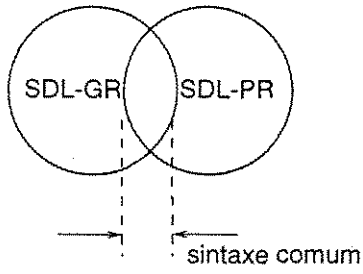


Figura 3.9: Representação gráfica e textual

A representação SDL-PR utiliza apenas a sintaxe textual. No entanto, a representação gráfica utiliza também alguns componentes da representação textual, pois algumas especificações (dados e sinais) são melhor representadas na maneira textual. Assim, essas representações tem formas idênticas nestas representações.

3.4 Exemplo de Utilização de SDL

Devido a restrição de espaço não há possibilidade e sequer faz sentido descrever todas as características da linguagem.

A referência completa está contida nas normas [IT93c, IT93e, IT93d], portanto será demonstrada a utilização da mesma na modelagem do sistema **DaemonGame** (Figura 3.10). O sistema contém apenas um bloco chamado **gamei** (Figura 3.11) que recebe do ambiente as mensagens **Newgame**, **Probe**, **Result**, **Endgame** e envia para o ambiente as mensagens **gameid**,**Win**, **Lose**, **Score**, o bloco **Game** (Figura 3.11) é

constituído de dois tipos de processos **Monitor(1,1)** e **Game(0)**.

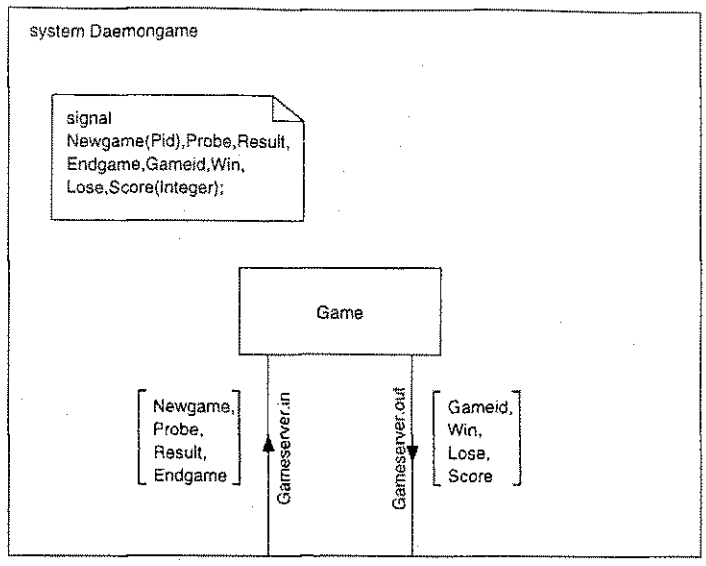


Figura 3.10: Sistema DaemonGame

Neste bloco verifica-se que, quando da inicialização do sistema, é criado automaticamente uma instância de **Monitor**, isto é indicado através dos parentêses que seguem **Monitor** indicando o número máximo e mínimo de processos do tipo **Monitor**. É permitido (min,max) onde $min = 1$ e $max = 1$, no caso processo **Game(0,)**, o número mínimo de processos é 0, na inicialização do bloco há 0 instâncias do processo **Game** e o número máximo de processos **Game** é ilimitado. A linha tracejada que vai do processo **Monitor** para o processo **Game**, indica que **Monitor** pode criar processos do tipo **Game**.

O processo **Game** (Figura 3.13) envia para o ambiente os sinais **Gameid**, **Win**, **Lose**, **Score** e para o processo **Monitor**, o sinal **Gameover**.

Em SDL cada processo ao ser criado, pelo ambiente ou por outro processo, exibe de maneira padrão 4 identificadores de outros processos.

São eles **Self**, **Sender**,**Offspring**, **Parent** que indicam respectivamente o seu próprio **PId**(*Process Identification*), o **PId** de quem enviou um determinado sinal, o **PId** do processo mais recentemente criado e o **PId** daquele que criou este determinado processo. Neste caso, **Game** apresenta como **Parent** o processo **Monitor**.

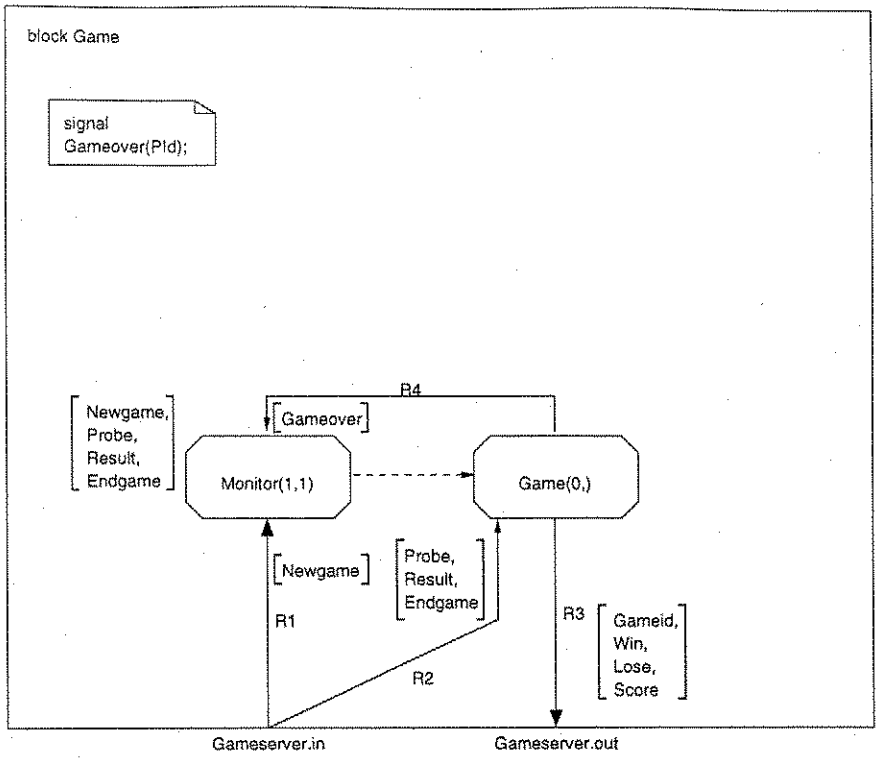
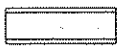


Figura 3.11: Bloco Game

Na descrição do processo **Monitor** (Figura 3.12) temos que este processo ao receber um sinal do tipo **NewGame** cria um processo do tipo **Game**, denotando assim o símbolo de criação de processos  e retorna a máquina de estados para o estado **idle**.

Após a inicialização do processo **Game**, é enviado o sinal **Gameid** para o ambiente e a máquina entra no estado **even**. Se a máquina receber o sinal **Probe**, a mesma envia o sinal **Lose** para o **player**, decrementa a variável **count** e volta ao estado **even**.

Se a máquina não contiver nenhum sinal na fila de entrada (*input queue*), a mesma é levada ao estado **odd** e se receber o sinal **Probe**, a mesma envia o sinal **Win** para o processo indicado por **player**, incrementa o contador e volta ao estado **odd**. Se não houver nenhum sinal na fila de entrada o processo volta ao estado **even**.

Se a máquina estiver em qualquer um dos dois estados e receber o sinal **Result**, o sinal **Score** é enviado para **player** e retorna ao estado em que se encontrava. Se receber

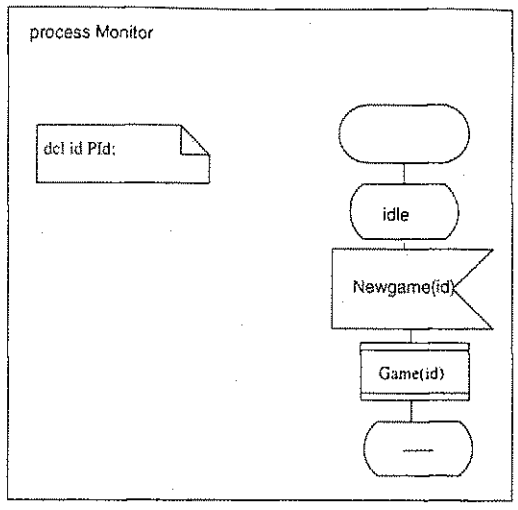


Figura 3.12: Processo Monitor

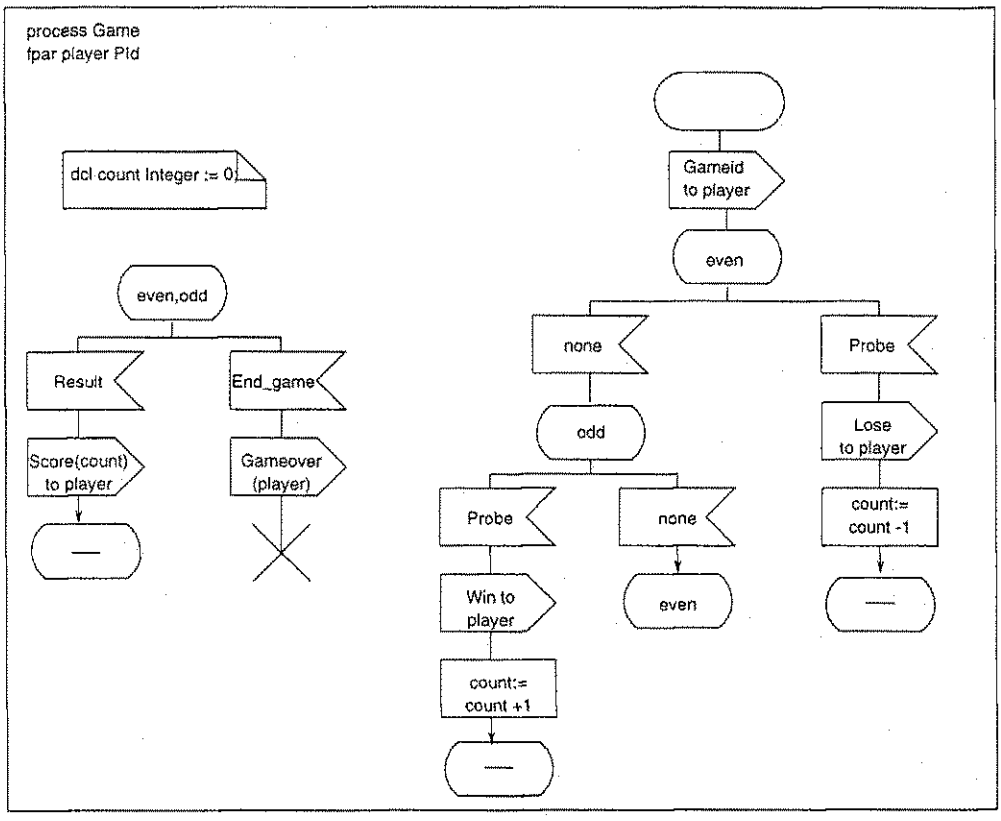


Figura 3.13: Processo Game

o sinal **End_game**, então o sinal **Gameover** com o identificador do processo **player** é enviado para o processo **Monitor**, e esta instância do processo **Game** é finalizada, através do símbolo $\times$.

Capítulo 4

Modelagem Formal dos Objetos

Neste capítulo são apresentadas algumas modelagens realizadas com as técnicas anteriormente descritas, enfatizando-se as vantagens e desvantagens de cada uma delas quanda das modelagens dessas entidades.

4.1 Introdução

Neste capítulo apresentamos uma proposta de formalização da Interface M4 para Entidades de Objetos Gerenciáveis (EOG, Object Managed Entities (OME)) informalmente introduzidas em [Com96], através do modelo G-CPN apresentado no Capítulo 4. Estas EOGs constituem parte de uma MIB em um Sistema de gerenciamento de Rede (SGR). A Interface M4 trata com o Gerenciamento da Informação (Management Information(MI)), relacionada com o SGR, considerando diferentes requisitos da visão de rede, detalhes podem ser encontrados na documentação do ATM Forum [Com96].

4.2 Modelagem das Entidades Gerenciadas

No documento AF-NM-0058.000 [Com96], as definições de todas as entidades gerenciadas são apresentadas de maneira informal. Um pequeno parágrafo descreve cada entidade e declara seu propósito no sistema, e apresenta-se uma figura para mostrar o

relacionamento de entidade em questão com as outras. Quatro aspectos são então considerados: atributos, notificações, relacionamentos e operações. Atributos são certas propriedades ou características que distinguem objetos gerenciados. Eles são descritos por uma lista simples que declara o tipo (domínio) e qualificadores leitura/escrita ou somente leitura. Além disso, uma curta explicação de cada atributo é apresentada. Notificações são mensagens (primitivas assíncronas de comunicação) enviadas pelas entidades gerenciadas ao processo gerente. Notificações são necessárias para permitir que o SGR seja informado sobre eventos que ocorrem no sistema. Duas classes de eventos são definidas em [Bla95]. Uma relacionada aos eventos gerados por estímulos internos (internal stimuli) e outro pelos eventos externos (external stimuli). Estímulos internos são eventos ou modificações de estado causadas por condições locais à entidade gerenciada. São exemplos de estímulos internos todas as falhas ocorridas nos equipamentos e nos recursos. Estímulos externos são eventos oriundos da recepção de uma mensagem indicando uma nova situação em uma outra entidade ou a solicitação de um determinado serviço. Em alguns casos, estímulos externos são gerados pelo próprio SGR de forma direta ou indireta. Como um exemplo, o gerente pode decidir que em um determinado instante algum enlace deve ser desabilitado, e envia uma mensagem ao agente correspondente. Quando do recebimento da mensagem, a entidade gerenciada altera seu estado administrativo e envia uma notificação ao sistema de gerenciamento. Entretanto, mensagens da mesma natureza poderiam ser enviadas à entidade gerenciada por uma outra entidade gerenciada do sistema. Neste caso, a notificação é ainda mais importante, pois ela informa o processo gerente sobre o novo estado, pelo qual ela não requisitou informação. Cada entidade gerenciada pode ser relacionada a outras entidades de diversas formas. Uma sub-rede, por exemplo, pode ser particionada em diversas outras sub-redes; um enlace topológico é determinado por pontos de terminação de dois enlaces topológicos. Então, é necessário determinar com precisão a natureza, a cardinalidade e a semântica de todos os relacionamentos entre entidades gerenciadas. No documento AF- NM-0058.000 [Com96] relacionamentos são

expressos no que é denominado de uma linguagem semi-formal baseada em álgebra relacional. De fato, tem-se uma versão escrita da abordagem entidade/relacionamento. A forma como cada implementação de um protocolo específico da MIB tratará este relacionamento é deixada em aberto. Existe uma lista de operações para cada entidade gerenciada. Cada operação é descrita por meio de parâmetros de entrada, parâmetros de saída, condições de erro e seu comportamento. Os parâmetros de entrada e saída são descritos de forma similar à ASN.1. Uma lista de condições de erro é declarada informalmente com sua semântica. Finalmente, o comportamento é descrito de maneira completamente informal.

Apesar de que diversas operações são listadas para cada entidade gerenciada, muitas operações auxiliares (e necessárias) não são descritas.

Então, muitas outras operações devem ser implementadas durante o desenvolvimento de um protocolo de entidade gerenciada da MIB sem nenhuma recomendação. Não é nem sequer necessário manter uma correspondência um-a-um entre cada elemento descrito na MIB lógica e na implementação. Certamente, isto leva a interpretações errôneas e consequentemente a uma falta de compatibilidade entre diferentes fabricantes, desta forma desconsiderando o principal objetivo do padrão. A especificação formal de um subconjunto de entidades gerenciadas definidas na visão de rede da interface M4 [Com96] é desenvolvida com o objetivo de aplicar e experimentar o modelo G-CPN. Devido a isso, a especificação aqui apresentada não é completa e nem pretende ser. Ainda mais, muitos detalhes dos modelos desenvolvidos foram omitidos de modo a simplificar os exemplos e as explanações.

4.3 Modelagem Usando G-CPN

Nossa abordagem de modelagem provê uma breve descrição informal de cada entidade gerenciada (similar àquelas encontrada em [Com96, Man97], sendo que o documento [Man97], incorpora as correções feitas em [Com96], além de descrever os modelos em MIB CMIP), e imediatamente depois , uma explanação de como o modelo G-CPN

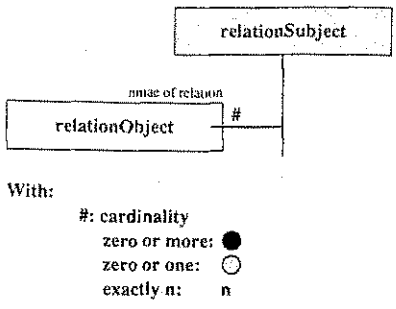


Figura 4.1: Descrição Informal das entidades gerenciadas

pode ser obtido. Representações gráficas de partes do modelo (veja a Figura 4.1) são alternadas com explanações. Por fim, o módulo completo é apresentado.

Esta seção mostra exemplos modelagem entidades gerenciadas utilizando cenários simples de simulação. Nosso objetivo é modelar a entidade gerenciada vcSubnetwork com todos os seus atributos e relacionamentos e com algumas de suas notificações e operações. A idéia é representar cada classe de entidade gerenciada no sistema por meio de um módulo G-CPN, e cada instância de uma entidade gerenciada por meio de um contexto de G-CPN no módulo apropriado. Desta forma, para criar uma nova instância de uma entidade gerenciada, tudo que o sistema deve fazer é invocar o método correto. Atributos são modelados pela linguagem CPN-ML. Também, os relacionamentos são modelado como atributos. Definimos um conjunto de cores para cada entidade gerenciada do sistema. Diversos outros conjuntos de cores são definidos de modo a permitir a criação de variáveis utilizadas nas expressões dos arcos. O envio de uma notificação é modelado pela invocação de um método. Então, o processo gerente, o qual não é modelado aqui, deve oferecer em sua interface uma operação de notificação para permitir a recepção de notificações. Finalmente, operações são expressas diretamente na estrutura interna de um módulo. Cada operação sobre atributos ou sobre os relacionamentos é modelada especificamente por um método.

4.3.1 Entidade Gerenciada vcSubnetwork

A entidade gerenciada vcSubnetwork representa, de acordo com a norma G.805[1] [IT93b], o elemento topológico conceitual utilizado para dar suporte a informações características (células ATM em uma sub-rede ATM). O nome vcSubnetwork significa canal bidirecional de sub-rede, isto ocorre pois esta entidade gerenciada é especializada por nível. Esta entidade gerenciada é definida para ser criada automaticamente pela entidade gerenciada de rede. Entretanto, um método é necessário para possibilitar a recepção de uma mensagem de criação de uma rede.

Os atributos da entidade vcSubnetwork são:

- Identificador de sub-rede (Subnetwork ID): este é um atributo somente de leitura. Todas as identificações são modeladas em G-CPN por um conjunto de cor denominado ID.

- Identificação de Sinal (Signal Identification): este atributo representa o formato específico que o recurso transporta. No caso desta entidade o atributo é fixado para o nível VC no momento da criação da entidade. Em G-CPN, especificaremos um conjunto de cores específico que contém duas cores: VC e VP.

- Rótulo de Usuário (User Label): este é um atributo de leitura/escrita. Permite a identificação da organização do gerente. Existem somente três eventos que causam a geração de notificações nesta entidade gerenciada. A notificação UserLabelChange é utilizada para informar ao gerente sobre as modificações no atributo referente ao rótulo do usuário. A notificação vcSubnetworkCreated é utilizada para informar sobre a criação de uma instância desta entidade gerenciada.

Relacionamentos são modelados de um modo semelhante aos atributos. Todos os conjuntos de cores para relacionamentos são colocados juntos em um único produto de conjunto de cores. Entretanto, nenhum lugar especial na estrutura interna é utilizado

[1] O documento [Com96] na página 45 descreve a relação

para manter estes relacionamentos. O mesmo lugar (lugar de atributo no contexto de G-CPN) mantém todos os atributos e relacionamentos de cada instância de entidade gerenciada. Então, este lugar possui uma entrada (ficha) para cada vcSubnetwork no sistema. Os relacionamentos da entidade vcSubnetwork são com as entidades gerenciadas vcSubnetworkTPs, vcSubnetworkConnection, vcSubnetwork, vcTopologicalLink, e vcTopologicalLinkTP. Todos os relacionamentos são explicados por diagramas entidade-relacionamento, como mostrado na Figura 4.2.

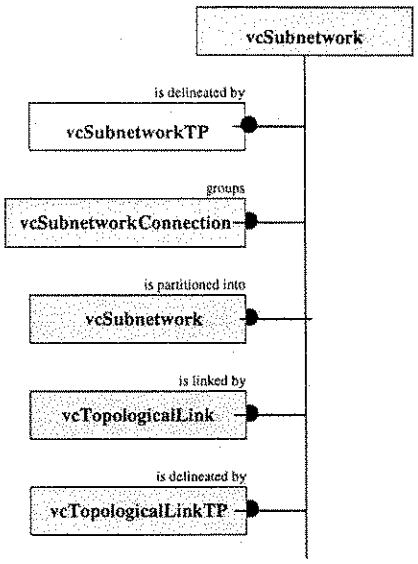


Figura 4.2: Representação gráfica para vcSubNetwork

Existem diversas operações descritas para a entidade vcSubnetwork. Muitas delas são operações muito simples de consulta, as quais podem ser facilmente descritas por um método simples com uma única transição, que lê a estrutura vcSubnetwork e seleciona a informação apropriada. Então, por simplicidade, estas não serão detalhadas.

A operação setupSubnetworkConnection é mais complexa pois ela modifica o estado de uma dada sub-rede e possivelmente interage com entidades gerenciadas relacionadas. Em alguns casos, é também possível que novas entidades gerenciadas sejam criadas.

O módulo G-CPN apresentado na Figura 4.3 descreve algumas características da classe entidade gerenciada ATM vcSubnetwork. Na Figura 4.4 são apresentadas as declarações para os tipos e variáveis. As declarações seguem as informações fornecidas

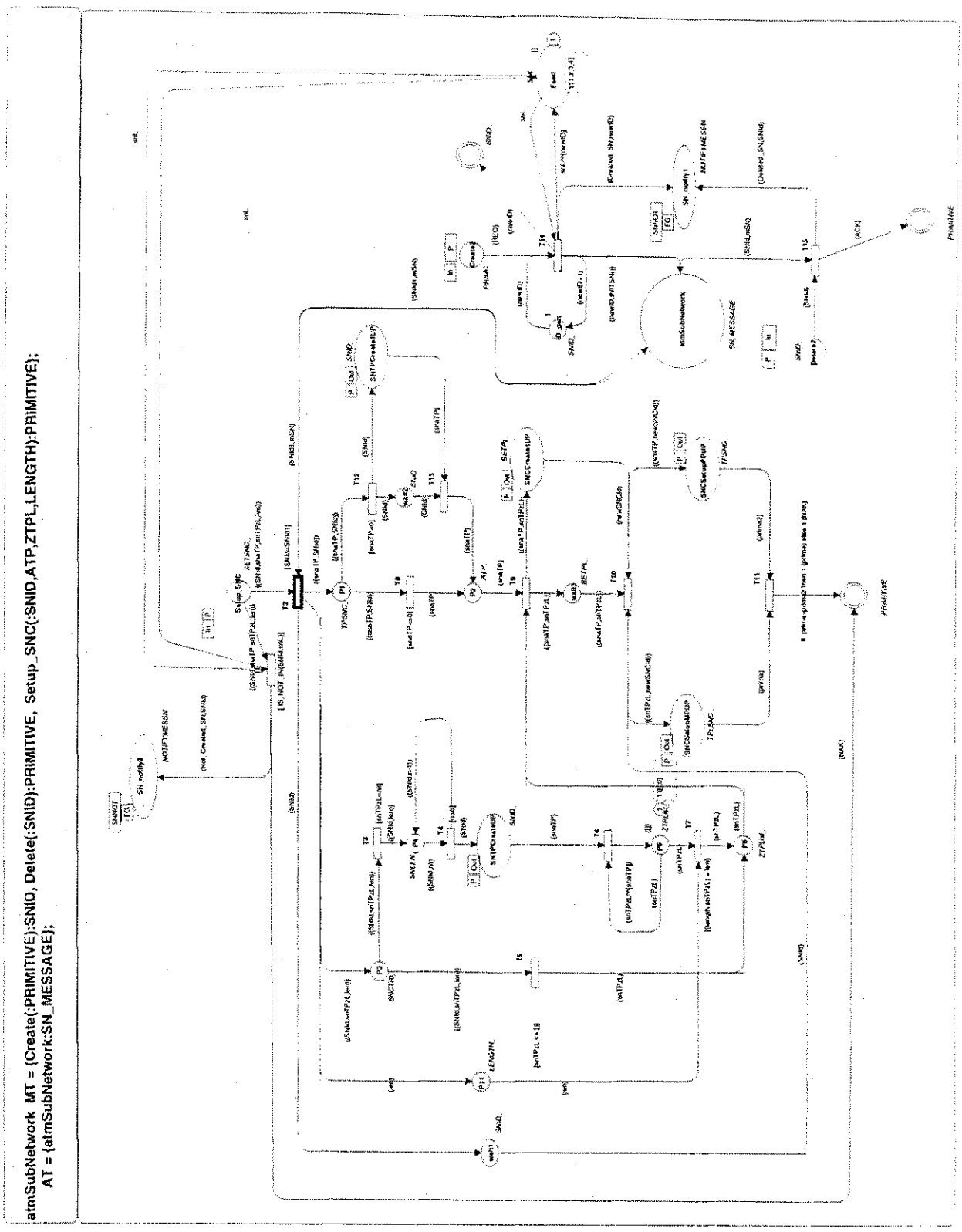


Figura 4.3: Módulo G-CPN para a entidade sub-rede

na MIB M4, descrevendo cada cor para os atributos da entidade, bem como, cores definindo cada entidade que se relaciona com a mesma.

Os três métodos mostrados na Figura 4.3 expressam o comportamento do processo agente no caso de criação/destruição de um vcSubnetwork e no caso do estabelecimento de uma conexão de rede. O método *create* opera de um modo muito simples. Quando uma ficha de requisição (REQ) é colocada no lugar inicial do método, a ocorrência da transição remove a ficha de requisição e pega um novo identificador válido do lugar ID_Gen. A ocorrência também põe uma nova ficha no lugar atmSubnetwork, o qual é um atributo no sentido de G-CPN. O valor para cor *characteristicInformation* de atmSubnetwork é definido como VC. Também, uma ficha é colocada no ISP que invoca o método de notificação no sistema de gerenciamento. O identificador da nova entidade vcSubnetwork é enviado da transição *T14* para o elemento invocador através do lugar alvo.

```
color SNID = int;
color PRIMITIVE = with REQ | RES | ACK | NAK;
color NOTIFICATION = with CreatedSN | DeletedSN | ModifiedSN | ERROR_SETUP;
color NOTIFYMES = product NOTIFICATION*SNID;
color ATP = int;
color ZTPList = list ATP;
color BETPL = product ATP*ZTPList;
color TPSNC = product ATP*SNID;
color TPLSNC = product ZTPList*SNID;
color SNCID = int;
color LID = int;
color LList = list LID;
color LTP = int;
color LENGTH = int;
color SNCTRI = product SNID*ZTPList*LENGTH;

color UserLabel = string;
color CharacteristicInformation = with VP | VC;
color ContainedSNL = list SNID;
color SupportedLTPL = list LTP;
color ContainedLL = list LID;
color SN_TOKEN = record ulabel:UserLabel*charinf:CharacteristicInformation*
cLL:ContainedLL*cSNL:ContainedSNL*sLTPL:SupportedLTPL;
color SN_MESSAGE = product SNID*SN_TOKEN;

fun INITTP(ztpL:ZTPList):SN_TOKEN={ulabel="",charinf=VC,cLL=[],cSNL=[],sLTPL=[]};

var SNId,newID : SNID;
var SNCId,newSNCId : SNCID;
var sneTP : ATP;
var snTPzL : ZTPList;
var len : LENGTH;
var prima , prima2 : PRIMITIVE;
```

Figura 4.4: Declarações para os tipos e variáveis

O método Delete recebe uma ficha do tipo SNId e se houver uma ficha com o mesmo identificador no atributo SubNetwork, ocorrerá a habilitação da transição T16

que removerá a ficha do e enviará uma notificação de destruição desta sub-rede.

O método setupSubnetworkConnection recebe ficha do tipo SETSNC. O primeiro campo representa a sub-rede para qual a mensagem foi enviada. Os outros três são, respectivamente, o ponto de terminação, uma lista de pontos de terminação e o comprimento desejado da lista.

4.3.2 Entidade Gerenciada vcSubnetworkConnection

Esta entidade gerenciada descreve uma entidade de transporte que transfere informação através de uma sub-rede, como definido na Recomendação G.805 [IT93b] da ITU-T. Os atributos desta entidade gerenciada são:

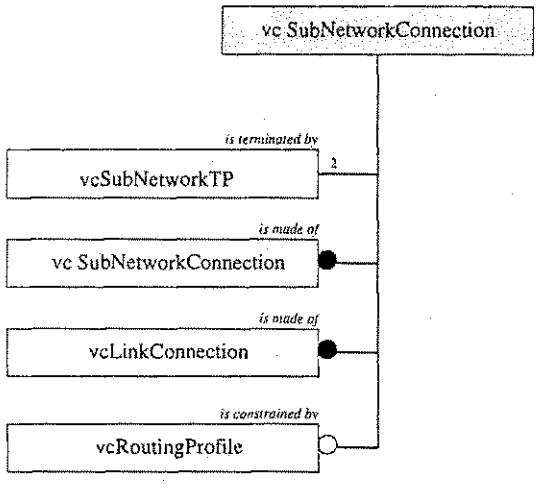


Figura 4.5: Representação gráfica de uma conexão vcSubnetwork

- vcSubnetworkConnection: um atributo somente de leitura similar ao ID da entidade gerenciada vcSubnetwork.
- Direcionalidade (Directionality): um atributo somente de leitura sempre definido como bi-direcional.
- Estado Operacional (Operational State): um atributo somente de leitura que identifica se uma instância da entidade gerenciada de conexão de sub-rede é

capaz de executar suas funções normais. Somente é modificada por um estímulo externo.

- Estado Administrativo (Administrative State): um atributo de leitura e escrita utilizado para bloquear e desbloquear células através da conexão de sub-rede.
- Rótulo de Usuário (User Label): um atributo de leitura e escrita utilizado para permitir a identificação do usuário para quem o serviço está sendo disponibilizado.

Esta entidade gerenciada é capaz de enviar as mesmas notificações que a entidade vcSubnetwork e notificações adicionais de mudança de estado para informar tanto estados operacionais como administrativos. Esta entidade gerenciada mantém relacionamentos com: vcSubnetworkTPs, vcSubnetworkConnections, vcLinkConnections e com os descritores de qualidade de transporte, os quais não estão definidos aqui. A Figura 4.2 mostra estes relacionamentos. A Figura 4.6 representa o módulo G-CPN para esta classe.

4.3.3 Entidade Gerenciada vcSubnetworkTP

Uma vcSubnetworkTP representa o objeto conceitual que ternina uma uma conexão de sub-rede (subnetworkConnection) ou um enlace de conexão (linkConnection) em uma sub-rede. O único atributo para vcSubnetworkTP é seu identificador (ID). Somente são oferecidos suporte, por esta entidade, a notificações de criação, destruição e modificação. Esta entidade gerenciada mantém relações com as seguintes entidades: vcTTP, vcSubnetwork, vcLinkConnection e vcSubnetworkConnection. Na Figura 4.7 estas relações são apresentadas. A Figura 5.1 apresenta o módulo G-CPN para esta classe. O propósito de modelar as entidades gerenciadas vcSubnetworkConnection e vcSubnetworkTP é essencialmente prover os serviços requisitados pela operação setupSubnetworkConnection da entidade gerenciada vcSubnetworkTP.

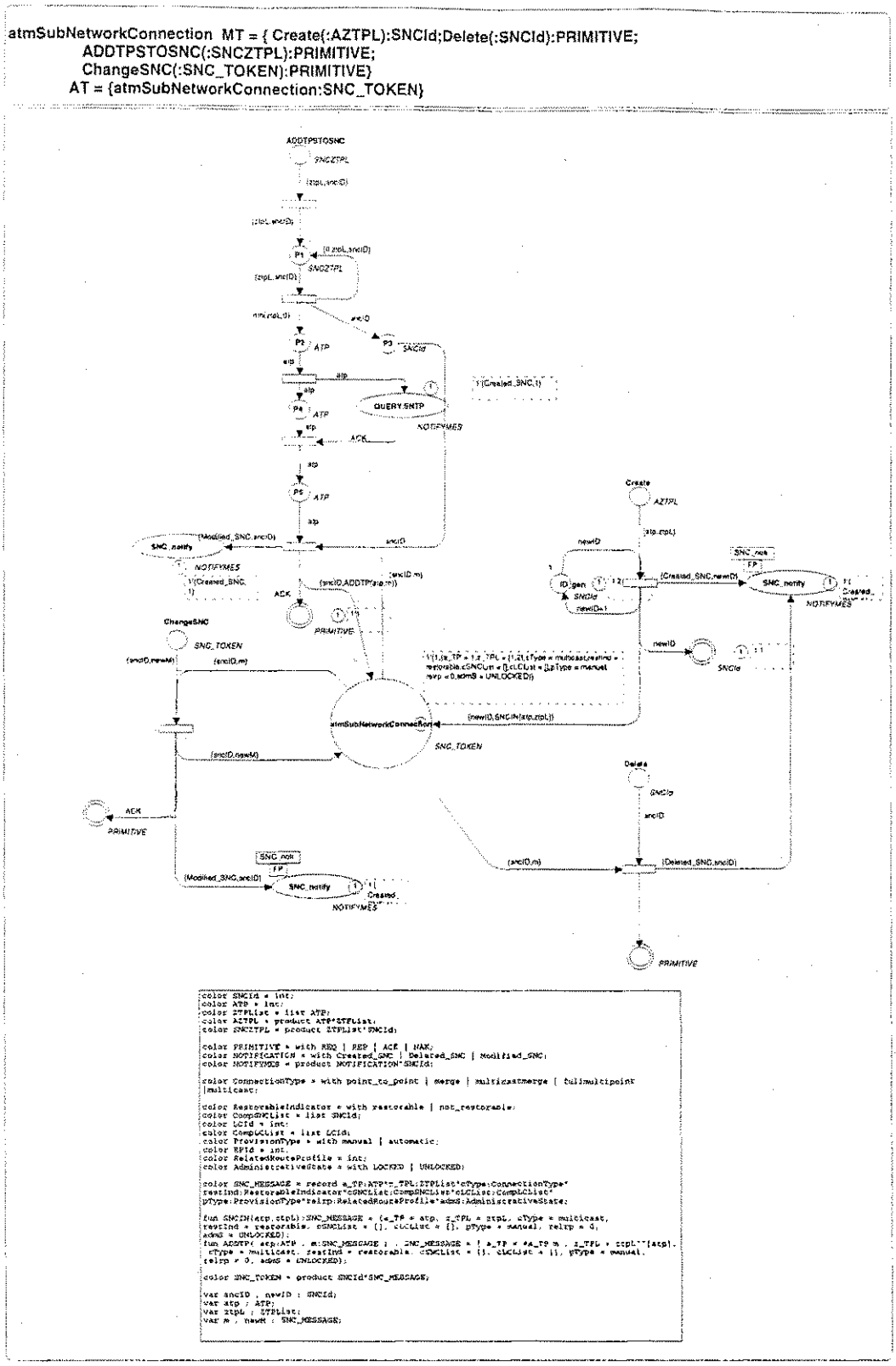


Figura 4.6: Módulo G-CPN para a entidade vcSubnetwork

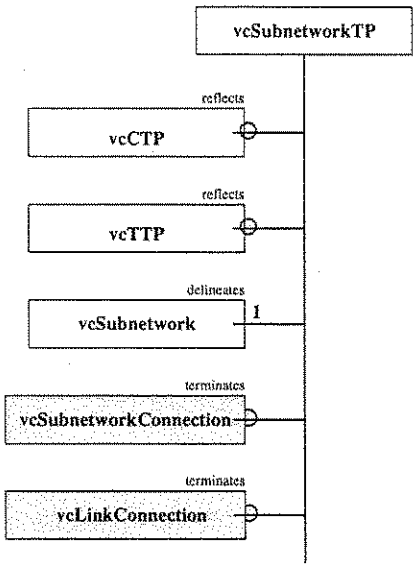


Figura 4.7: Representação gráfica para o ponto de terminação de uma sub-rede

Capítulo 5

Análise das Entidades Modeladas

Utilizamos a ferramenta Design/CPN[1] para analisar os modelos G-CPN apresentados neste capítulo. Para tanto, primeiramente obtivemos os modelos equivalentes em rede colorida hierárquica. Basicamente, considerou-se um cenário e o resultado foi analisado utilizando a análise do grafo de ocorrência gerado pelo Design/CPN, permitindo a detecção de estados proibidos, *deadlocks* e *livelocks* no modelo. O cenário considerado representando uma operação de estabelecimento de uma conexão de sub-rede para a entidade gerenciada vcSubnetwork. Então, definimos uma marcação inicial com uma ficha nos lugares setupSubnetworkConnection e create da G-CPN vcSubnetwork, representando que a entidade vcSubnetwork foi automaticamente criada.

O restante das entidades gerenciadas que compõem a interface M4 não foram simuladas no cenário utilizado, pois na sua maioria realizam apenas operações de *query* ou *create*. A exceção fica por conta da entidade *Link*, que exibe métodos semelhantes aos da entidade *SubNetwork*. Cada uma das entidades restantes foram simuladas apenas no cenário que contempla a criação e destruição destas entidades, além de mudanças no atributo de cada uma delas.

[1]Maiores referências podem ser encontradas no endereço http://www.daimi.aau.dk/designCPN

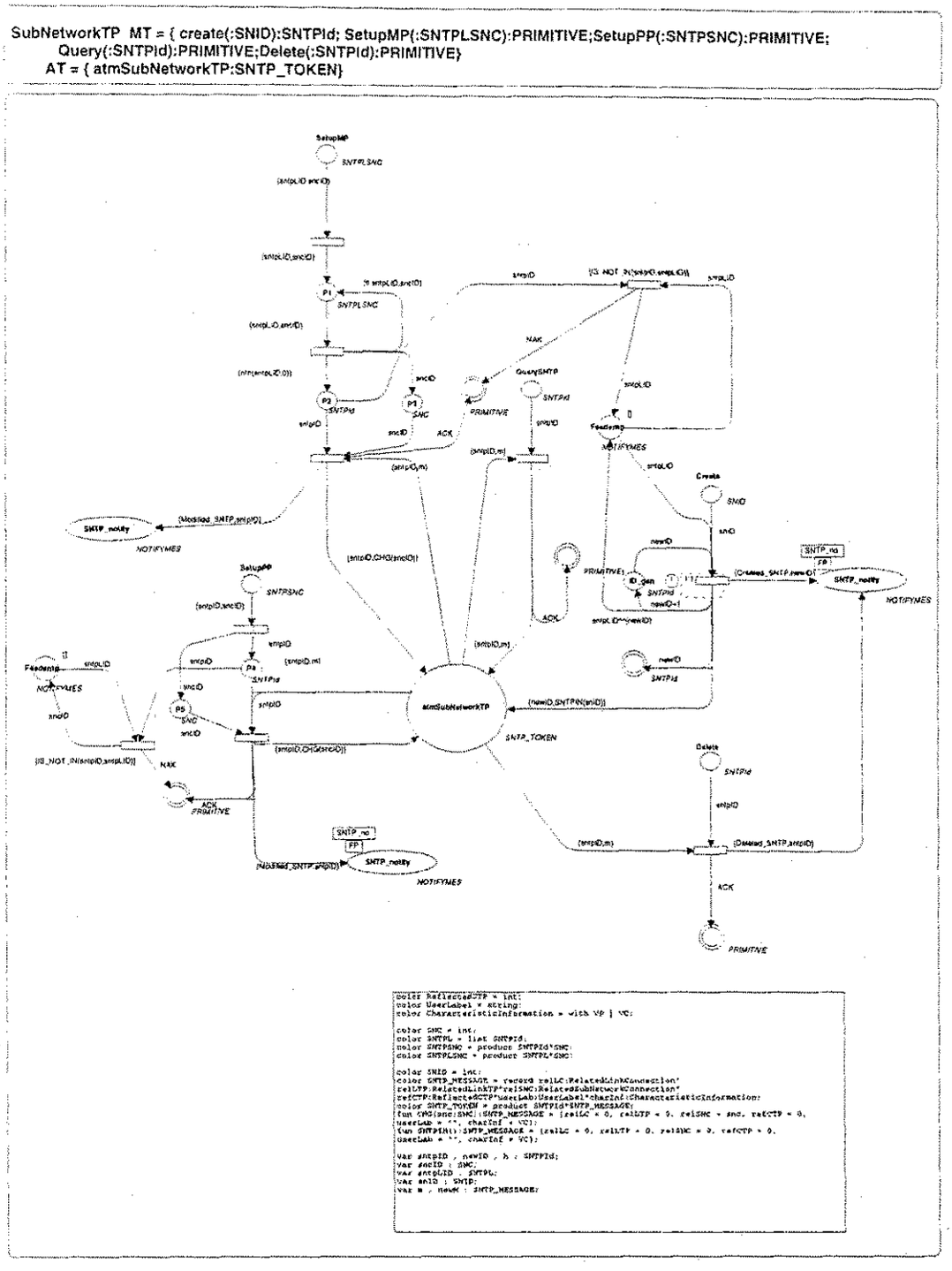


Figura 5.1: G-CPN para a entidade gerenciada de ponto de terminação sub-rede

5.1 Entidades Modeladas

As informações relativas a descrição dos objetos e suas definições são fornecidas no documento AF-NM0073 do ATM-FORUM[2]. Este documento contempla as revisões efetuadas no documento [Com96]. As especificações aqui estão escritas em CMIP MIB, ou seja, seguindo as regras estabelecidas por esqueletos *GDMO*. No Apêndice A foi transcrito a parte desta MIB indispensável ao entendimento dos modelos G-CPN, de onde foi extraída a informação necessária a construção dos mesmos(modelos). Não temos a pretensão de explicar as definições necessárias ao entedimento da mesma. Informações adicionais podem ser encontradas nos livros textos [Bla95, Udu96] e na recomendação [IT92b] do ITU-T.

5.1.1 atmLinkConnection

O objeto atmLinkConnection (Figura 5.2) representa uma conexão de enlace I.326. Esta entidade é responsável pelo transporte de informação característica entre sub-redes, e pertence a um enlace(atmLink).

5.1.2 atmLinkTP

É definido (Figura 5.3) como o componente topológico usado para representar a terminação de um *atmLink* e provê a capacidade de armazenar informação de configuração ao nível do *Link*.

5.1.3 atmNetworkAccessProfile

Esta entidade (Figura 5.4) contém a informação que descreve a máxima banda de ingresso e egresso, bem como a faixa de valores VPI e VCI que são aplicáveis ao

[2]Este documento pode ser recuperado via web no endereço: http://www.atmforum.com/atmforum/specs/approved.html ou via ftp em: ftp://ftp.atmforum.com/pub/approved-specs/af-nm-0073.000.txt

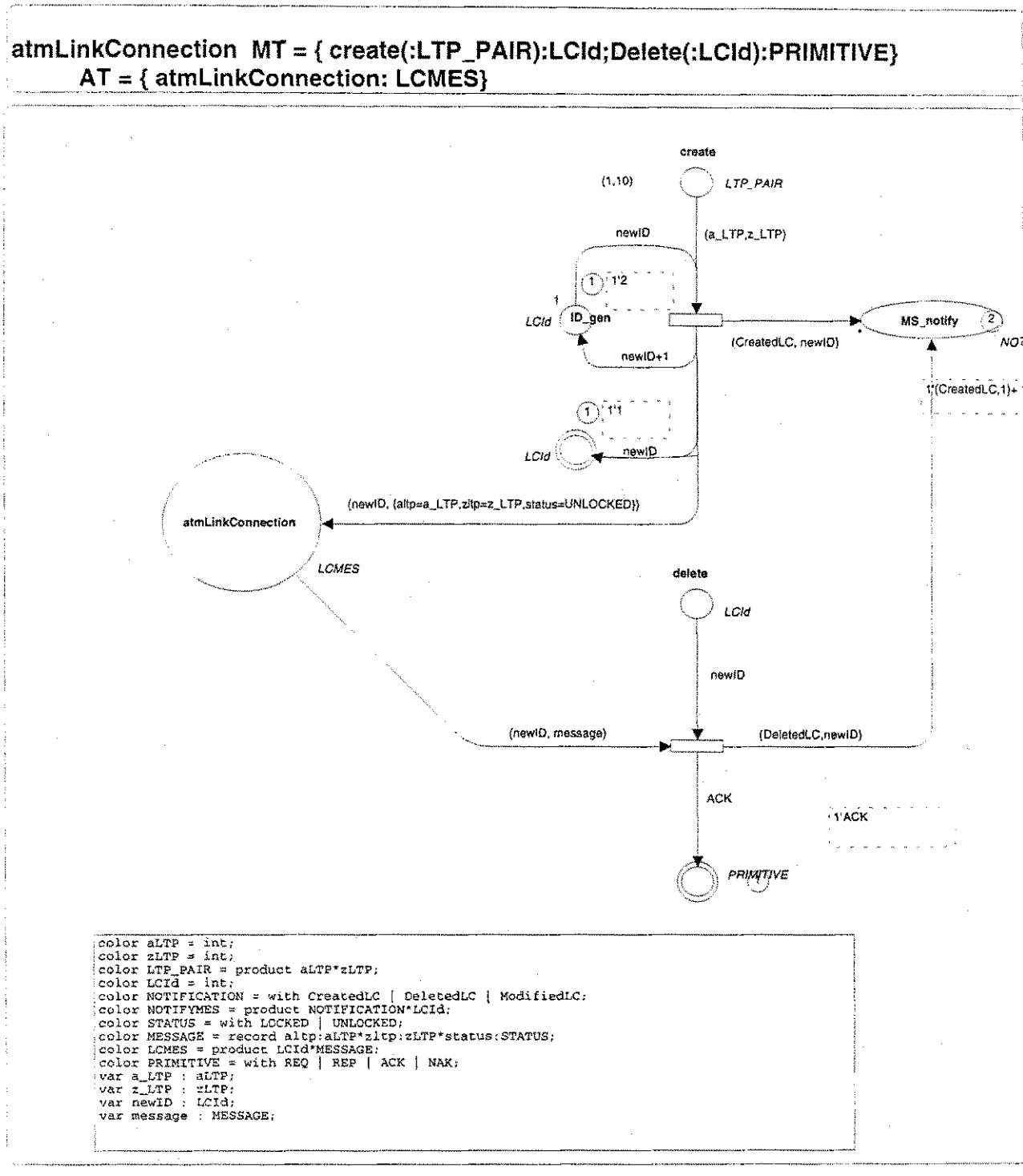


Figura 5.2: Modelo G-CPN do atmLinkConnection

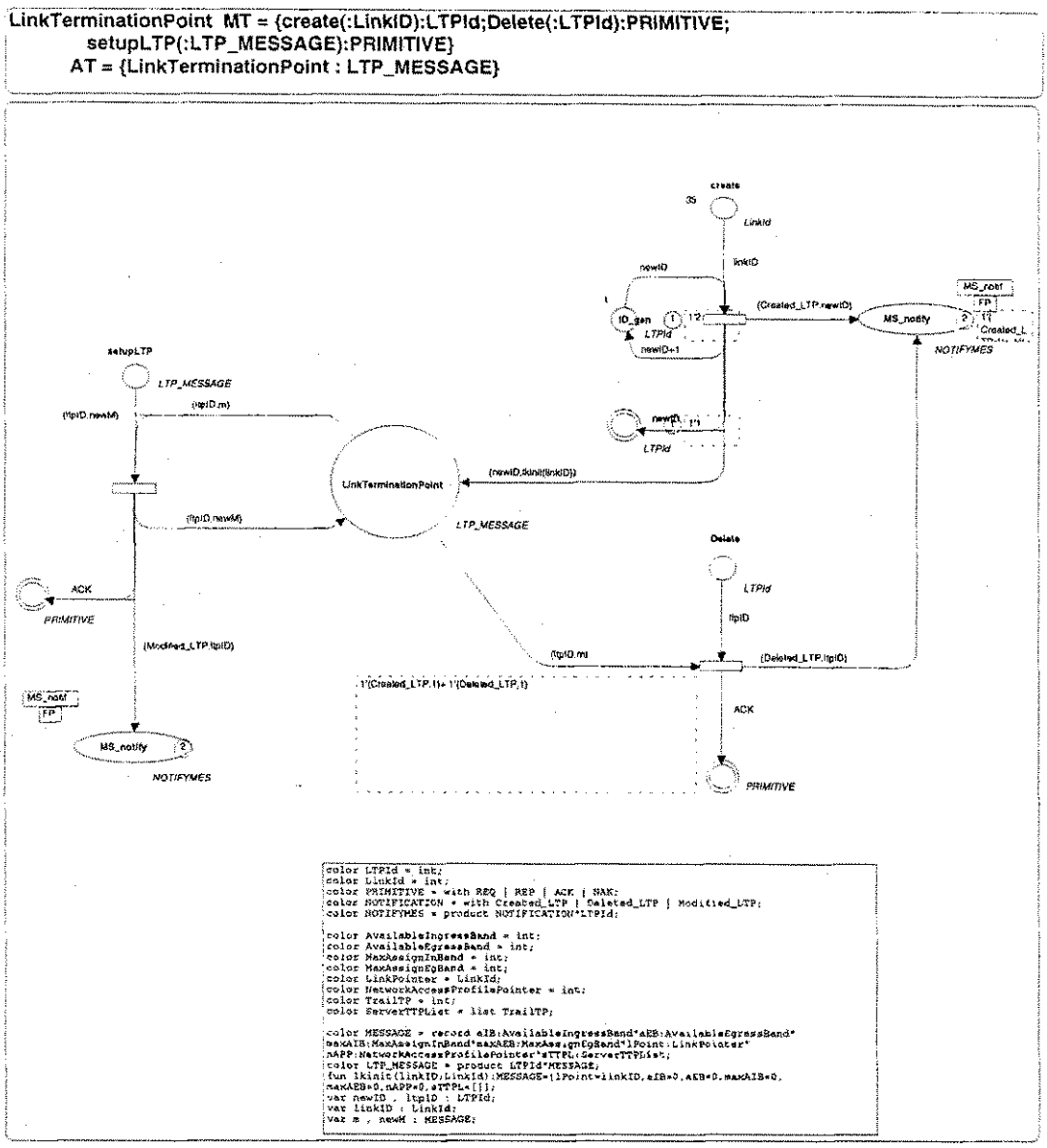


Figura 5.3: Modelo G-CPN do Link Termination Point

atmLink ou atmLinkTP que o referenciam.

5.1.4 atmNetworkTrailTerminationPoint

Esta entidade é utilizada quando a visão de rede é disponibilizada, o atributo relatedAtmCTP é usado para associar o CTP final de uma VCC ou VPC com o TTP (Figura 5.5).

5.1.5 atmNetworkRoutingProfile

Esta classe de representa um conjunto de restrições topológicas de roteamento que podem ser aplicados a uma nova conexão ou trilha durante a configuração (Figura 5.6).

5.2 Cenário Utilizados

Foram utilizados casos com potencial possibilidade de ocorrência, que exemplificam a utilização das entidades e permitem uma melhor compreensão do objeto modelado. As saídas do gráfico de ocorrência estão descritas nop Apêndice C, elas fornecem basicamente os limites superiores e inferiores das marcações dos lugares pertencentes a cada entidade, bem os lugares de marcações mortas (marcações sem possibilidade de habilitação de nenhuma outra transição, a partir deste lugar), alem do número de nós, arcos e em quanto tempo a simulação foi concluída. Bem como, o estado da completude da simulação, se completo ou parcial. Maiores referências , quanto a leitura e utilização da ferramenta podem ser encontradas no manual do Gráfico de Ocorrência [OG96].

5.2.1 Primeiro Cenário

No primeiro caso, utilizou-se o seguinte cenário:

1. Criação de duas sub-redes.

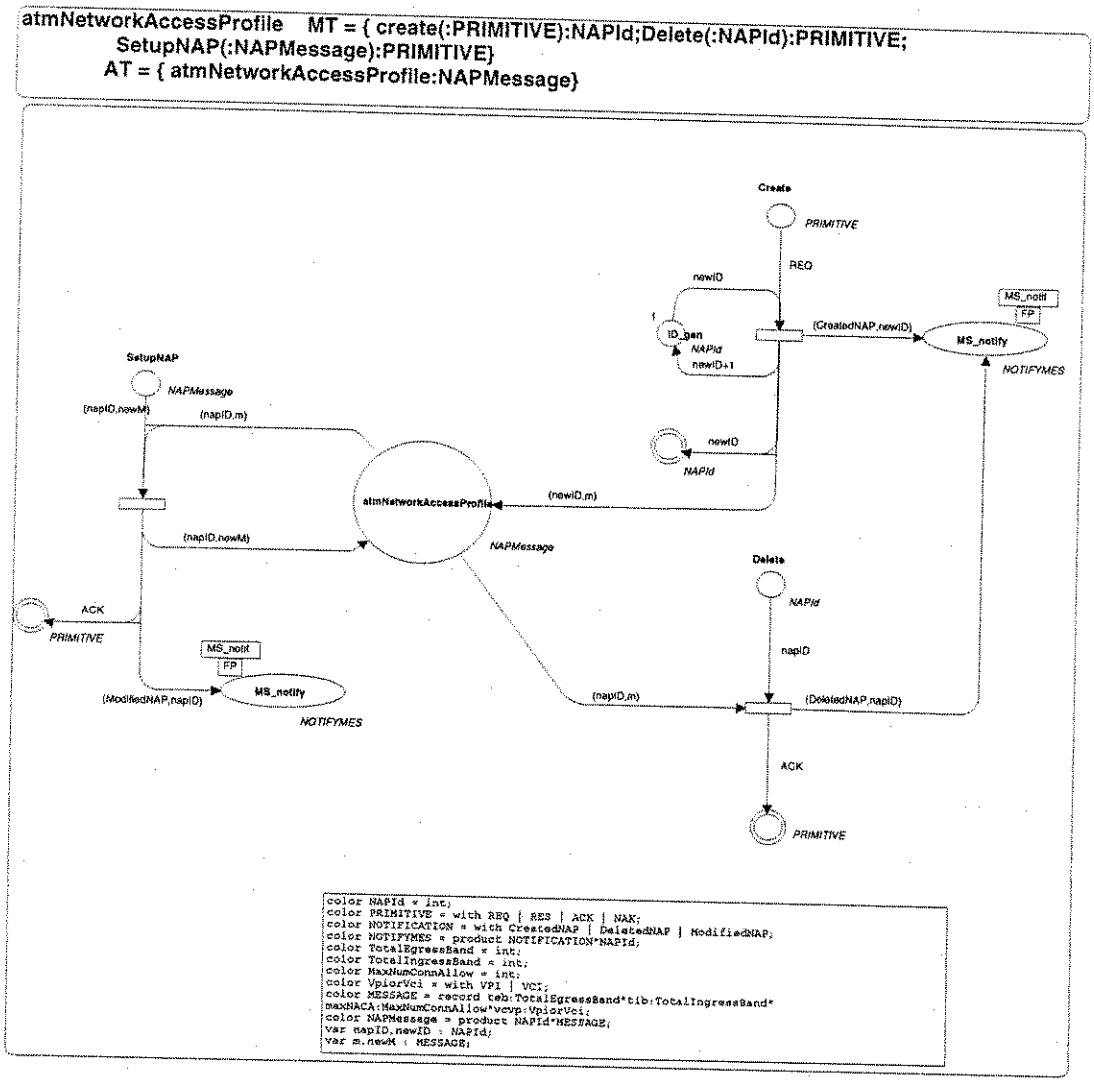


Figura 5.4: Modelo G-CPN do Network Access Profile

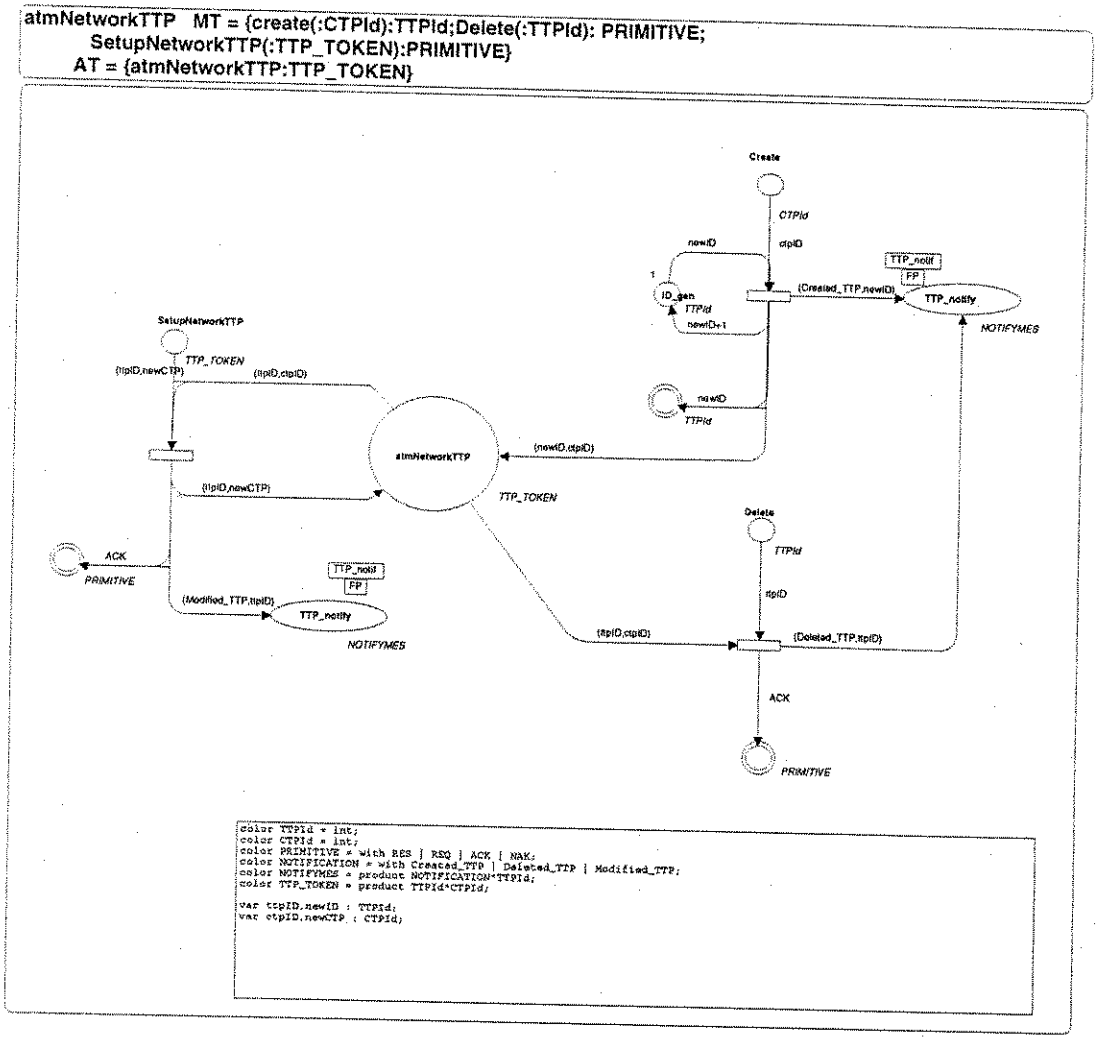


Figura 5.5: Modelo G-CPN do Network Trail Termination Point

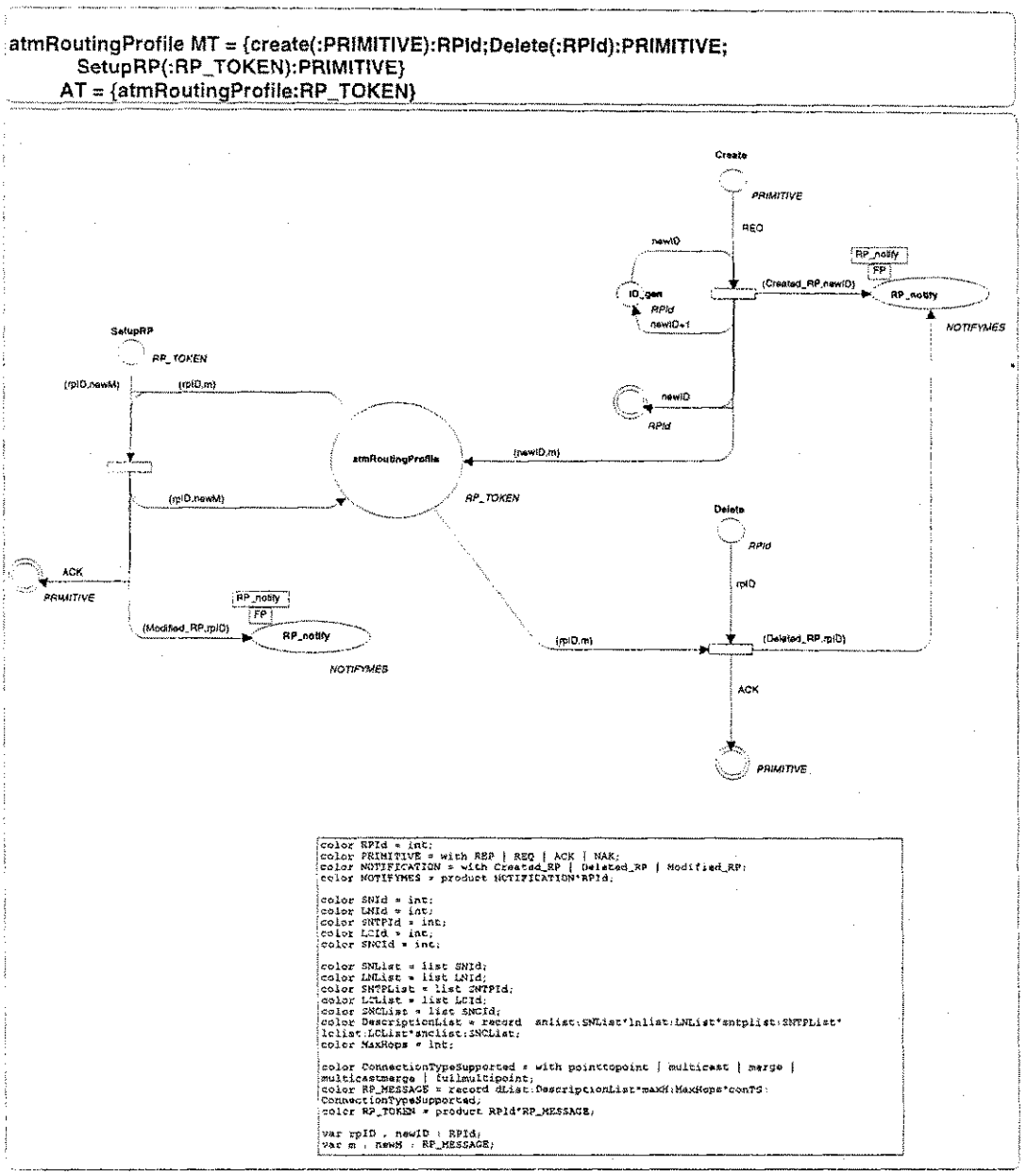


Figura 5.6: Modelo G-CPN do atmNetwork Routing Profile

2. Configuração de uma conexão de sub-rede.

Na criação coloca-se duas fichas 2'*REQ* no lugar Create2 no módulo G-CPN (Figura 4.3). Os identificadores da sub-rede são gerados através da habilitação da transição T14. O valor do identificador é então enviado para o lugar de terminação do método Create2 e para o lugar Feed. Uma notificação *Created_SN* é enviada para o lugar *SN_notify* e uma ficha com os atributos do identificador criado é colocada no lugar de atributo atmSubNetwork, finalizando o método.

O método *Setup_SNC* tem a capacidade de configurar uma conexão de sub-rede ponto a ponto ou *multicast* [Com96]. Como parâmetros para invocação do método temos: O identificador de sub-rede, um dos pontos de terminação, a lista do pontos de terminação e comprimento esperado da lista.

Temos a seguinte situação:

Se o identificador de sub-rede for existente, o método prossegue na configuração. Se o identificador for inexistente, então uma mensagem de erro é enviada para o *SN_notify* através da habilitação da transição *T1*.

Se os identificadores de sub-rede forem existentes o método prossegue através da habilitação das transições *T5* e *T8* que, consequentemente, habilitam a transição *T9*. A partir deste ponto uma ficha da cor BETPL é colocada no ISP SNCCreate, que retorna uma ficha contendo um identificador da conexão de sub-rede criada, habilitando assim a transição *T10*.

A habilitação da transição *T10* implica nas invocações dos métodos SNCSetupPP e SNCSetupMP, que correspondem à configuração dos pontos de terminação relativos a um único TP e a uma lista de TPs, com a informação sobre a conexão de sub-rede. No retorno destes ISP temos fichas da cor PRIMITIVE, indicando se houve ou não sucesso na configuração do TP. Por fim temos a habilitação da transição *T11* que retorna um ACK ou NAK dependendo do sucesso ou fracasso na configuração da informação nos pontos de terminação. O resultado é então colocado no lugar de terminação do método *Setup_SNC*.

No primeiro caso houve mudança apenas na ficha que é colocada no lugar de ativação *Setup_SNC.*

A ficha colocada em *Setup_SNC* assume que foram criados anteriormente três pontos de terminação: um para o atp e dois para a lista de ztpL. Desse modo a configuração será de uma conexão de sub-rede do tipo *multicast.* Observou-se que o espaço de estados, para esse caso, tem 20736 estados e que as fichas encontradas na única marcação morta existente, corresponde ao esperado.

O laço representado na transição T6 será executado três vezes. No restante a rede se comporta de maneira semelhante ao primeiro caso.

Os relatórios de cada cenário de simulação estão contidos no Apêndice B.

Esta rede foi simulada na ferramenta Design/CPN através da conversão do modelo G-CPN para CPN. Obtivemos resultados consistentes tanto na simulação quanto na geração do gráfico de ocorrência. O grafo de ocorrência para este cenário conta com 20736 estados. De acordo com a análise obtida do Design/CPN apenas uma marcação morta foi encontrada, correspondendo ao lugar de terminação do método *Setup_SNC.*

O modelo CPN equivalente dessa rede encontra-se na Figura 5.7.

5.2.2 Segundo Cenário

Este caso contempla, além do mesmo cenário para a criação de sub-redes, uma mudança na ficha colocada no lugar de ativação *Setup_SNC.* É fornecido para snTPzl uma lista nula, que será criada dinamicamente durante a simulação.

O gráfico de ocorrência para esse cenário apresentou 28561 estados, com apenas uma marcação morta.

5.2.3 Terceiro Cenário

As simulações relativas à criação e destruição de pontos de terminação e conexões de sub-rede, também, apresentaram resultado consistente com esperado. O que implica que a consistência da MIB será mantida através dos métodos invocados.

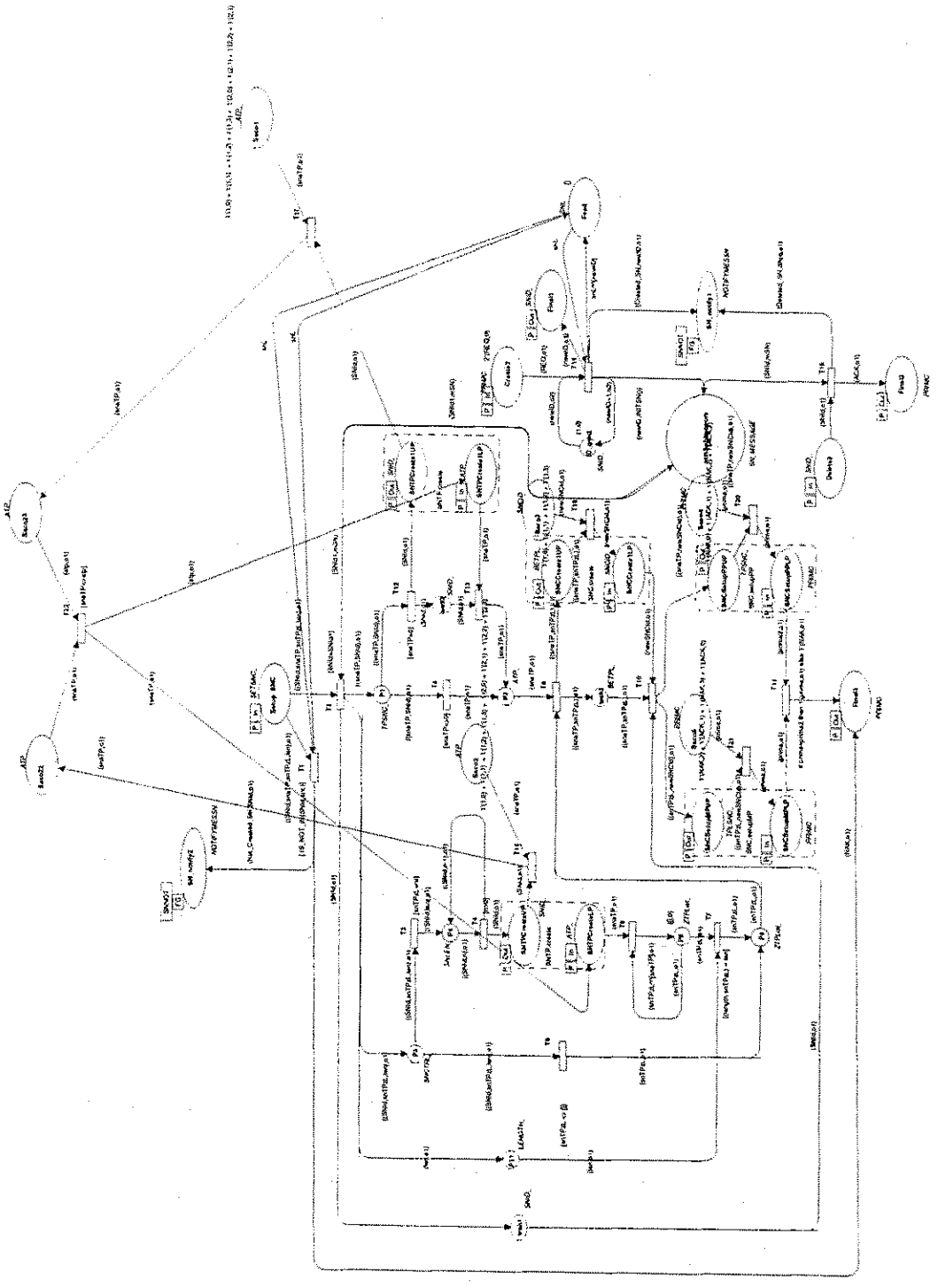


Figura 5.7: Modelo CPN equivalente para o módulo G-CPN da entidade SubNetwork

Na rede CPN equivalente ao módulo G-CPN para o SNTP (Figura 5.8), num primeiro passo, foram criados dois pontos de terminação de sub-rede e verificado a consistência do modelo, através da indicação de uma única marcação morta.

5.2.4 Quarto Cenário

Foi executada também a destruição de um objeto do tipo ponto terminação, apresentando resultados consistentes tanto na simulação, quanto na geração do gráfico de ocorrência.

Além disso houve a ativação dos métodos SetupPP e SetuMP, relativos à configuração dos pontos de terminação, existentes e inexistente, respectivamente.

Os métodos contidos na módulo G-CPN (Figura 4.6), para a conexão de sub-rede são semelhantes e de complexidade menor que os contidos no modelo G-CPN para o ponto de terminação de sub-rede (Figura 5.1).

Os modelos CPN equivalentes para a conexão de sub-rede encontram-se, respectivamente, nas Figuras 5.8 e 5.9.

Também aqui foi verificada a consistência dos modelos criados, para esses dois métodos.

Bem como o nó de declaração destas redes 5.2.4.

Nó de Declaração para a rede CPN equivalente da sub-rede

```
color INT = int with 0..500;

color CONTEXT = int with 0..500;

color NOTIFICATION = with Created_SN | Deleted_SN | Modified_SN |
ERROR_SETUP | Created_SNC | Deleted_SNC | Modified_SNCcolor |
Created_SNTP | Deleted_SNTP | Modified_SNTP | Not_Created_SN;


color SNID = INT;
color SNID_ = product SNID*CONTEXT;
color PRIMITIVE = with REQ | REP | ACK | NAK;
color PRIMC = product PRIMITIVE*CONTEXT;

color NOTIFYMESSN = product NOTIFICATION*SNID*CONTEXT;
color ATP = INT;
color ATP_ = product ATP*CONTEXT;
color ZTPList = list ATP;
color ZTPList_ = product ZTPList*CONTEXT;
color BETPL = product ATP*ZTPList;
color BETPL_ = product BETPL*CONTEXT;
color TPSNC = product ATP*SNID;
color TPSNC_ = product TPSNC*CONTEXT;
color TPLSNC = product ZTPList*SNID;
color TPLSNC_ = product TPLSNC*CONTEXT;
color SNCID = INT;
color SNCID_ = product SNCID*CONTEXT;
color LID = INT;
color LID_ = product LID*CONTEXT;
color LList = list LID;
color SNTPID = INT;
```

```
color SNTPID_ = product SNTPID*CONTEXT;

color SNC = INT;
color SNC_ = product SNC*CONTEXT;
color SNTPL = list SNTPID;
color SNTPL_ = product SNTPL*CONTEXT;

color SNTPSNC = product SNTPID*SNC;
color SNTPSNC_ = product SNTPSNC*CONTEXT;
color SNTPLSNC = product SNTPL*SNC;
color SNTPLSNC_ = product SNTPLSNC*CONTEXT;

color LTP = INT;
color LTP_ = product LTP*CONTEXT;
color LENGTH = INT;
color LENGTH_ = product LENGTH*CONTEXT;
color SNCTRI = product SNID*ZTPList*LENGTH;
color SNCTRI_ = product SNCTRI*CONTEXT;
color SNLEN = product SNID*LENGTH;
color SNLEN_ = product SNLEN*CONTEXT;
color SETSNC = product SNID*ATP*ZTPList*LENGTH;
color SETSNC_ = product SETSNC*CONTEXT;
color SBETPL = product SNCID*BETPL;
color SBETPL_ = product SBETPL*CONTEXT;

color AZTPL = product ATP*ZTPList;
color AZTPL_ = product AZTPL*CONTEXT;

color ConnectionType = with point_to_point | merge | multicastmerge |
fullmultipoint | multicast;
color RestorableIndicator = with restorable | not_restorable;
color CompSNCList = list SNCID;
color LCId = INT;
color CompLCList = list LCId;
color ProvisionType = with manual | automatic;
color RPId = INT;
color RelatedRouteProfile = INT;
color AdministrativeState = with LOCKED | UNLOCKED;

color SNC_MESSAGE = record a_TP:ATP*z_TPL:ZTPList*cType:ConnectionType*
restInd:RestorableIndicator*cSNCList:CompSNCList*cLCList:CompLCList*
pType:ProvisionType*relrp:RelatedRouteProfile*admS:AdministrativeState;

fun SNCIN(atp,ztpL):SNC_MESSAGE = {a_TP = atp, z_TPL = ztpL, cType = multicast,
restInd = restorable, cSNCList = [], cLCList = [], pType =
        manual, relrp = 0, admS = UNLOCKED};

color SNC_TOKEN = product SNCID*SNC_MESSAGE;
color SNC_TOKEN_ = product SNC_TOKEN*CONTEXT;
color NOTIFYMESSNC = product NOTIFICATION*SNCID*CONTEXT;

color INVOKER = with  SN_isp1|SN_isp2|
SN_isp3|SN_isp4|SN_isp5;
color SERVICE = with SN_setup|SN_del|
SN_cr|SNC_cr|SNC_del|SNC_ch|
SNTP_cr|SNTP_pp|SNTP_mp | SNTP_cri | SNTP_del;

color AC = product INVOKER * CONTEXT * CONTEXT;
color SIGMA = union is:INT + bet:BETPL +
tp:TPLSNC + tp1:TPSNC + set:SETSNC + sid:SNID +
ps:PRIMITIVE + az:AZTPL + cid:SNCID +
tk:SNC_TOKEN + tid:SNTPID + ssnc:SNTPSNC +
sslc:SNTPLSNC;
color IP = product SERVICE * SIGMA * CONTEXT;
color TP = union ptp:PRIMC + sn:SNTPID_ +
sn1:SNCID_ + sn2:SNID_;

var o1, on, oc, os, o2, o3 : CONTEXT;


color NOTIFYMESSNTP = product NOTIFICATION*SNTPID*CONTEXT;

color RelatedLinkConnection = INT;
color RelatedLinkTP = INT;
color RelatedSubNetworkConnection = INT;
color ReflectedCTP = INT;
color UserLabel = string;
color CharacteristicInformation = with VP | VC;


color SNTP_MESSAGE = record relLC:RelatedLinkConnection*
relLTP:RelatedLinkTP*relSNC:RelatedSubNetworkConnection*
refCTP:ReflectedCTP*userLab:UserLabel*charInf:
CharacteristicInformation*sn:SNID;
color SNTP_TOKEN = product SNTPID*SNTP_MESSAGE;
color SNTP_TOKEN_ = product SNTPID*SNTP_MESSAGE*CONTEXT;

color LList_ = product LList*CONTEXT;

color ContainedSNL = list SNID;
```

```
color SupportedLTPL = list LTP;
color ContainedLL = list LID;
color SN_TOKEN = record ulabel:UserLabel*charInf:CharacteristicInformation*
cLL:ContainedLL*cSNL:ContainedSNL*sLTPL:SupportedLTPL;
color SN_MESSAGE = product SNID*SN_TOKEN;
color SN_MESSAGE_ = product SNID*SN_TOKEN*CONTEXT;


fun CHG(snc:int):SNTP_MESSAGE = {relLC = 0, relLTP = 0, relSNC = snc,
refCTP = 0, userLab = "", charInf = VC, sn = 0};
fun SNTPIN(snid:int):SNTP_MESSAGE = {relLC = 0, relLTP = 0,
relSNC = 0, refCTP = 0, userLab = "", charInf = VC, sn = snid};
fun INITSN():SN_TOKEN = {ulabel = "", charInf =  VC, cLL = [], cSNL = [],
sLTPL = []};

fun IS_NOT_IN(elem,[]) = true | IS_NOT_IN(elem,x::y) = if elem = x
then false else IS_NOT_IN(elem,y);

color SNL = list SNID;
var snL : SNL;

var sntpID , newIDSNTP , h : SNTPID;
var sntpLID : SNTPL;
var mSNTP , newMSNTP : SNTP_MESSAGE;



var sncID , newIDSNC : SNCID;
var atp : ATP;
var ztpL : ZTPList;
var mSNC , newMSNC : SNC_MESSAGE;


var prima , prima2 , prima3 : PRIMITIVE;
var SNId , SNId1, newIDSN, newID : SNID;
var SNCId , newSNCId : SNCID;
var snaTP : ATP;
var snTPzL : ZTPList;
var len , n : LENGTH;
var mSN : SN_TOKEN;
```

5.2.5 Quinto Cenário

Neste cenário, procede-se à mudança de informação contida nos pontos de terminação. São fornecidos para a rede os pontos de terminação atp e uma lista com dois identificadores para o snTPzL.

5.2.6 Sexto Cenário

Mesmo caso anterior, sendo que aqui um dos TPs contidos na lista snTPzl não existe no atributo, levando a um NAK.

5.2.7 Sétimo Cenário

Um último caso, diz respeito à tentativa de configurar um conexão de sub-rede no modelo Subnetwork, com lista dos pontos de terminação completamente inexistentes,

$aTP = 0$ e $TPzL = []$, a geração do gráfico de ocorrência levou a uma explosão de estados, implicando, numa determinação através de heurística, que toma como o base o tempo de simulação sendo inversamente proporcional a possiblidade de ocorrência do caso, de que casos realmente merecem interesse.

5.3 Modelagem em SDL

Esta seção introduz a modelagem da interface M4 em SDL, esta incluída apenas uma descrição do sistema **atmNetwork**, **atmSubNetworkConnection** e **atmSubNetwork**, apenas o bloco **atmSubNetworkTP** foi efetivamente modelado.

O sistema **atmNetwork** (Figura 5.10) é formado por 3 blocos, onde os sinais que são trocados entre o ambiente e os blocos que constituem o sistema e entre os blocos, são determinados através da recomendações do ATM FORUM[Com96, Man97] aqui temos que o bloco **atmSubNetwork** recebe do ambiente os sinais **primitive**, **snid**, **sntp**, **sntpl**, e uma lista de sinais do **SNSL**, que transporta os sinais ul,ci,cll,csnl,cltpl.

O sinal **primitive** é do tipo **Prim**, que pode transportar as mensagens **ACK,NAK,REQ** e **RES**, **snid** é um sinal do tipo **Snid**, que define uma estrutura de dados que contém os sinais **id**, tipo **PId**, sinais **ul** e **cf**, tipo **Charstring**, e os sinais **cLL,cSNL** e **cLTPL**, tipo **PIdset**, ou seja um conjunto de dados do tipo **PId**.

O bloco **atmSubNetwork** (Figura 5.11) envia para o ambiente os sinais **primitive, notification** e **IDs**. O sinal **notification** é do tipo **Notific** e transporta mensagens do tipo **Created, Deleted, Modified** e **ERRO**. Os sinais trocados entre **atmSubNetwork** e os blocos **atmSubNetworkTP** e **atmSubNetworkConnection** são os mesmos trocados entre o ambiente e **atmSubNetwork** mais o sinal **snc**, tipo **Snc**, uma estrutura do que contém os sinais **id, atp** e **relATMRP**, tipo **PId**, **ztpL, cSNCL** e **cLCL**, tipo PIdset e **cType, rInd, pType** do tipo **Charstring**.

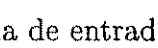
O bloco **atmSubNetworkConnection** (Figura 5.12) trata sinais dos tipos já descritos mais a lista de sinais **SNCSL**, que transporta os sinais **atp, ztpl, ct, ri, csncl, clc, pt, rarp, adms**.

No bloco **atmSubNetworkTP** temos a mais a lista de sinais SNTPSL, descrita na declaração do sistema.

Como apenas o bloco **atmSubNetworkTP** (Figura 5.14) foi efetivamente modelado, foi criado um sistema (Figura 5.13) apenas com esse bloco, tal sistema é constituído dos blocos, **Born_SNTP**, **SNTP_notify**, **Delete_SNTP**, **SetupMP**, **SetupPP** e **SNTP**.

Os processos **Born_SNTP**, **SNTP_notify**, **SetupMP**, **SetupPP** e **Delete_SNTP** são criados quando da inicialização do sistema, com permissão de ter apenas uma instância de cada bloco desses no sistema.

O processo **SNTP** (Figura 5.15) é criado apenas quando o processo **Born_SNTP** (Figura 5.16) recebe uma mensagem **REQ** via um sinal **primitive**.

Quando o processo **Born_SNTP** é iniciado ele entra no estado **idle** e fica aguardando o recebimento do sinais **primitive** e **sntp**, se um sinal **sntp** é enviado ele é salvo  da fila de entrada para ser consumido posteriomente.

Se um sinal do tipo *primitive* é recebida a variável **msg** é avaliada e se contiver uma mensagem **REQ** um processo SNTP é criado, uma notificação é enviada para o processo **SNTP_notify** e a máquina de estados passa para o estado **sntptoenv**, onde se houver um sinal sntp na fila de entrada o mesmo será consumido, esse sinal é enviado para o ambiente e o campo **id** da variável **idsntp**, do tipo **PId** é enviada para o processo **SNTP_notify** e a máquina volta ao estado **idle**.

O processo **SNTP** ao ser criado entra no estado **idle** e aguarda o recebimento dos sinais descritos no processo **SNTP**. Se o mesmo recebe os sinais **rlc,rltp,rsnc,rctp** esses campos são alterados na variável **idsntp** e o sinal **sntp** é enviado para o processo **Born_SNTP** via rota de sinal **sntp_to_born** e a máquina volta ao estado **idle**, se o sinal **sntp** é recebido é verificado se o contém o identificador deste processo, caso tenha, uma mensagem de **ACK** é enviada para o processo **Delete_SNTP**, caso não, o processo retorna ao estado **idle**.

O processo **SNTP_notify** (Figura 5.17) ao ser inicializado entra no estado **idle**

e fica aguardando o recebimento dos sinais **notification** e **IDs**, caso a variável **msg** contenha a mensagem **Created**, ambos, **notification e IDs** são enviados para o ambiente e a máquina volta ao estado **idle**, do contrário, a mensagem é enviada para o ambiente, sem o identificador e a máquina volta ao estado **idle**.

No processo **Delete_SNTP** (Figura 5.18), se a máquina recebe um sinal **sntp** , este sinal é enviado para o processo **SNTP**, se recebe um **primitive**, a mensagem é testada, se contiver um **ACK**, a mensagem é enviada para o ambiente, daí então, uma mensagem **Deleted** e um identificador de **SNTP** são enviados para o processo **SNTP_notify** e a máquina volta ao estado **maybedelete**, se falso um **NAK** é enviado para o ambiente e uma notificação de erro é enviada para o **SNTP_notify** e a máquina volta ao estado **maybedelete**.

Os processos **SetupPP** (Figura 5.19) e **SetupMP** (Figura 5.20) aguardam os sinais descritos e trocam os campos relativos aos pontos de terminação e a conexão de sub-rede atreladas. Particularmente, há apenas uma indicação da troca através de uma mensagem, e uma notificação é enviada para o ambiente, retornando a máquina ao estado **idle**.

A simulação que segue tem como cenário a criação de um **SNTP** neste sistema, a simulação é descrita através de um Gráfico de Sequência de Mensagens (*Message Sequence Charts*(MSC)) (Figura 5.21), tal gráfico demonstra a troca de mensagens entre as diferentes máquinas, o gráfico mostra que o ambiente envia o sinal primitive para a máquina **BORN**, daí a máquina **BORN** cria um processo do tipo **SNTP** e envia um notificação de criação para o processo **SNTP_notify**, a seguir temos as mensagens com sinais dos tipos(rlc, rltp, rsnc, rctp) com *Pids* do tipo *null*.

A máquina **SNTP** envia uma mensagem do tipo **sntp** para **BORN**, a seguir temos as mensagens **sntp** que por não terem sido consumidas são indicadas por setas que nao estão conectadas a nenhum outro processo. A máquina **SNTP_notify** então uma envia notificação de criação e o **PId** para o ambiente. Uma mensagem **sntp** é então enviada de **BORN_SNTP** para o ambiente. Na sequência **BORN_SNTP** envia o **PId** do

processo **SNTP** criado para o **SNTP_notify**.

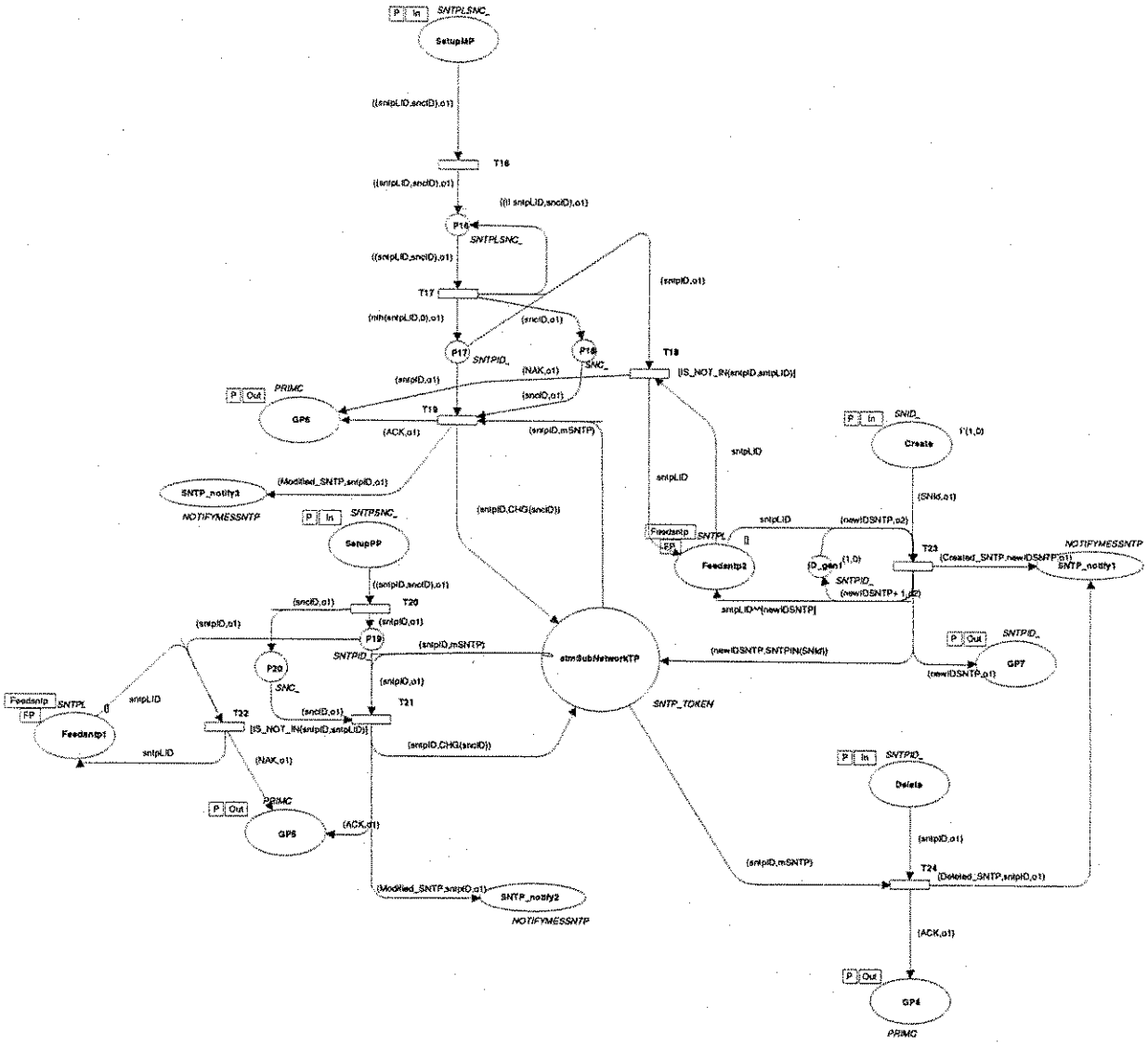


Figura 5.8: Rede CPN equivalente para o modelo G-CPN do SNTP

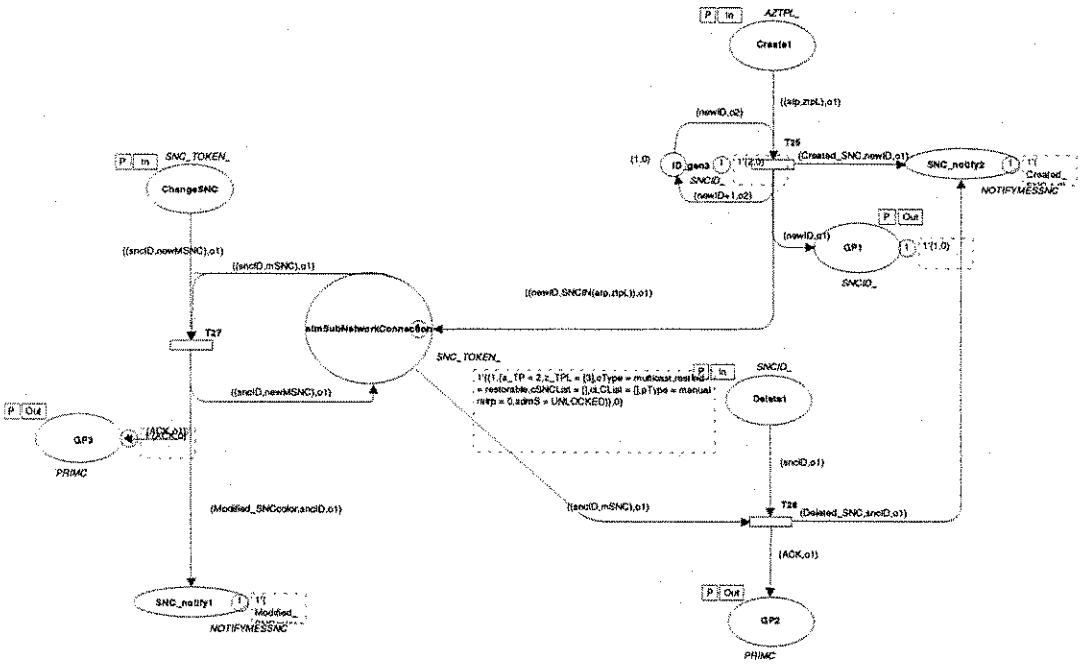


Figura 5.9: Modelo CPN equivalente para a conexão de sub-rede

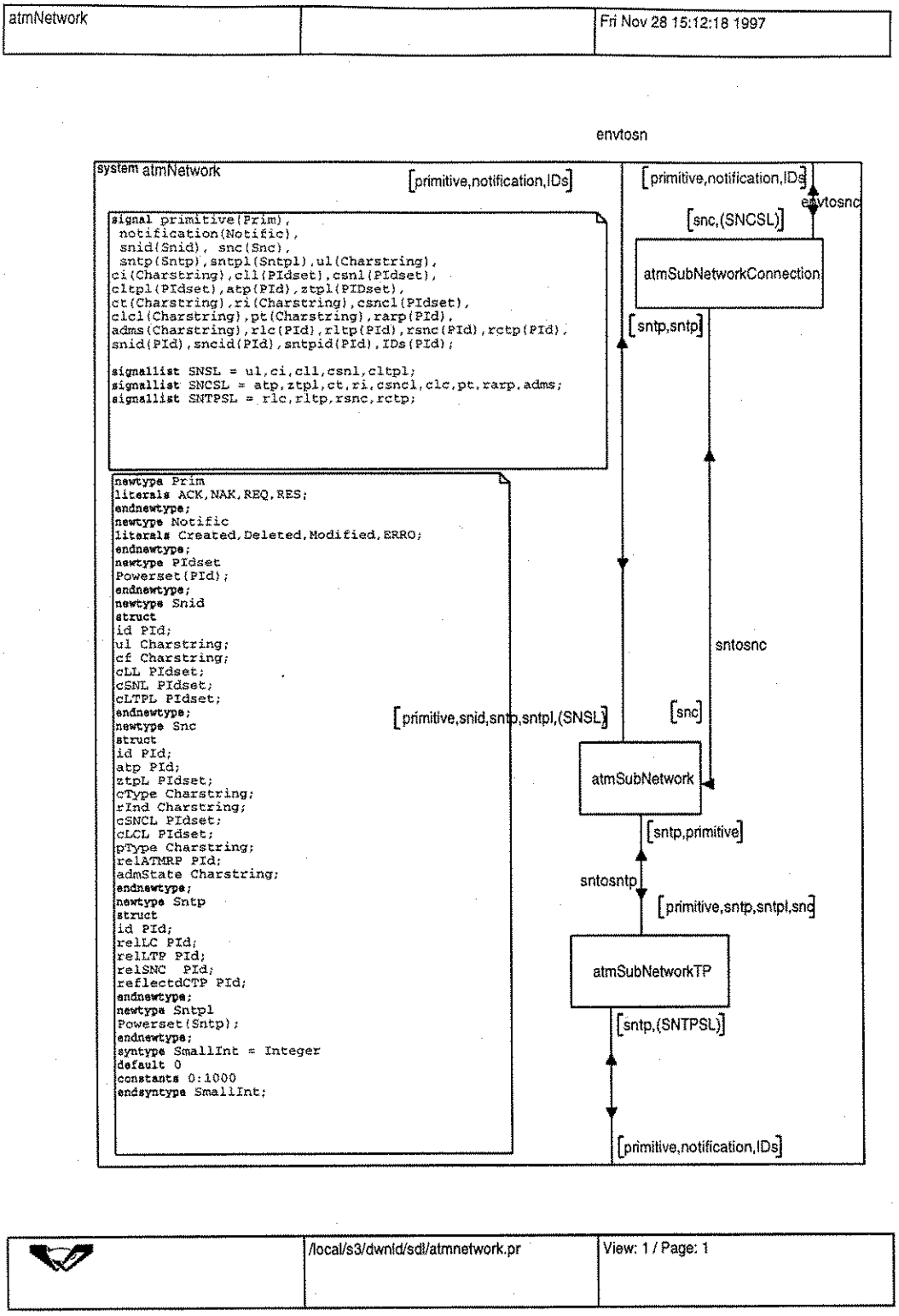


Figura 5.10: Sistema AtmNetwork

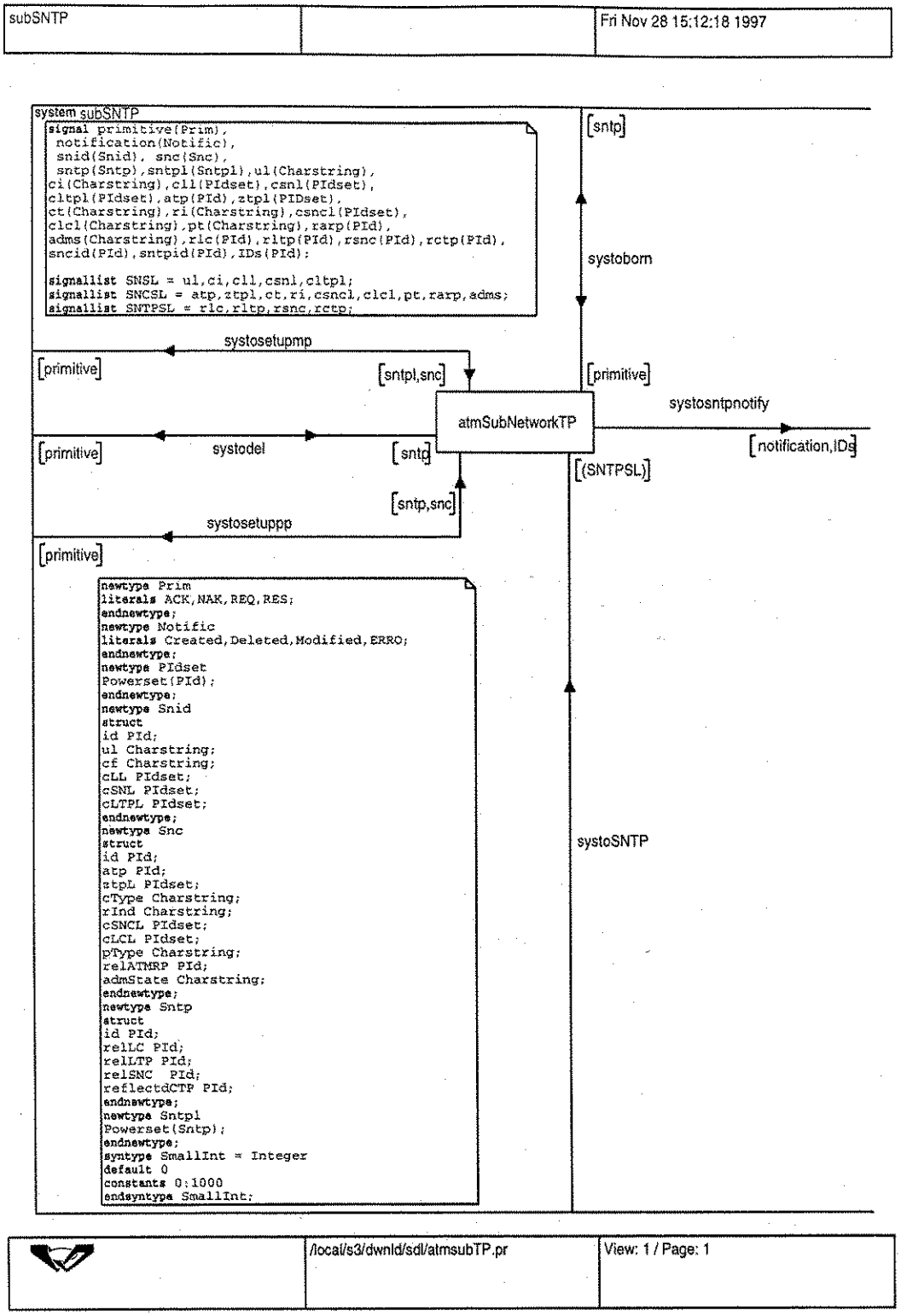


Figura 5.13: Sistema subSNTP

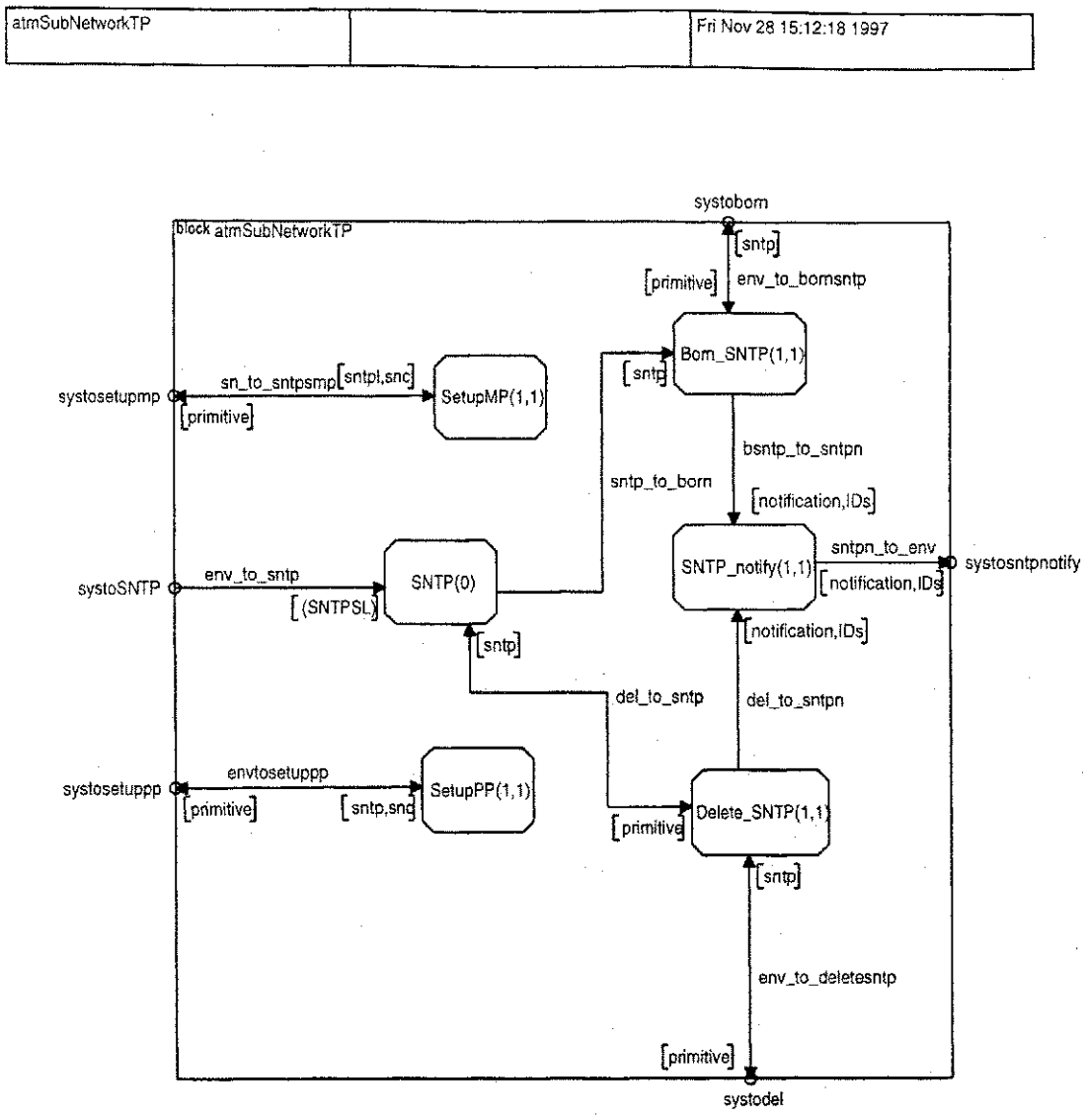


Figura 5.14: Bloco atmSubNetworkTP

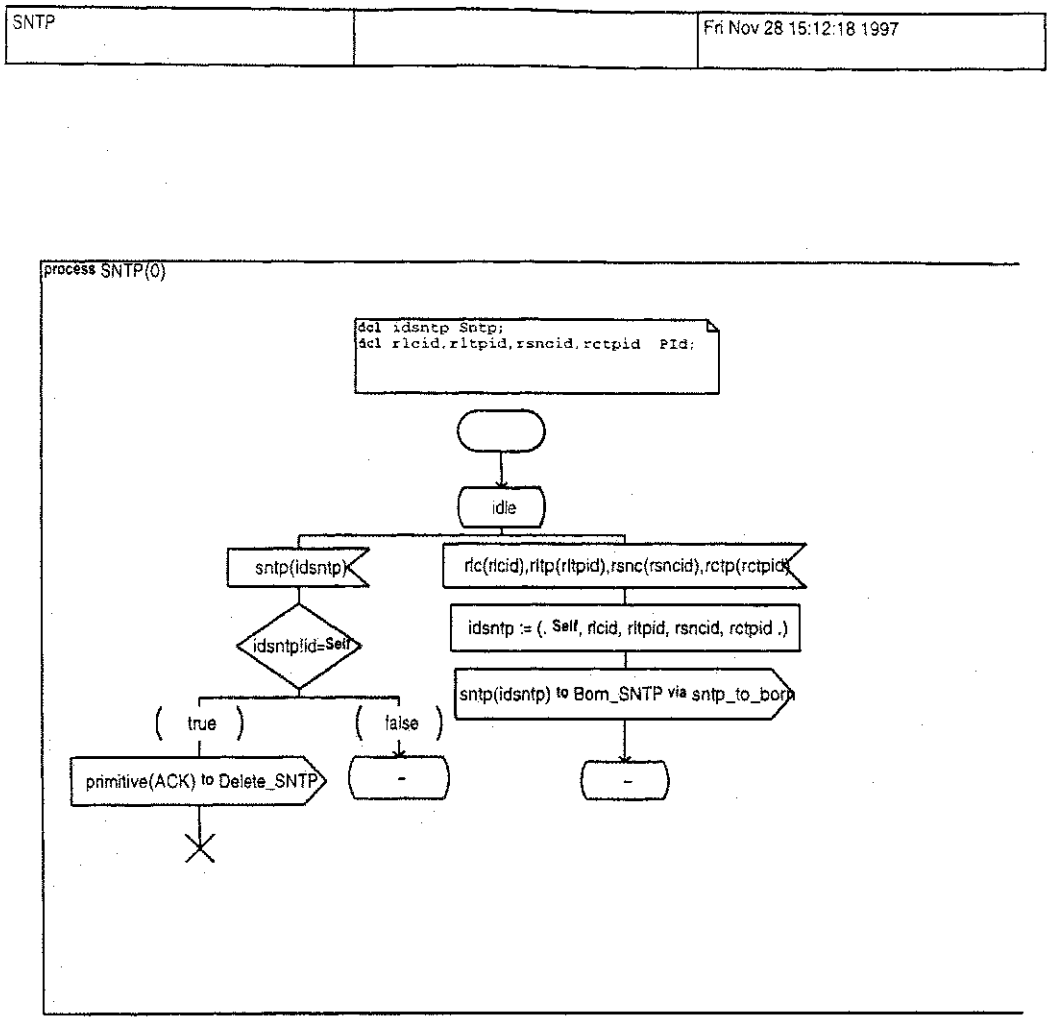


Figura 5.15: Processo SNTP

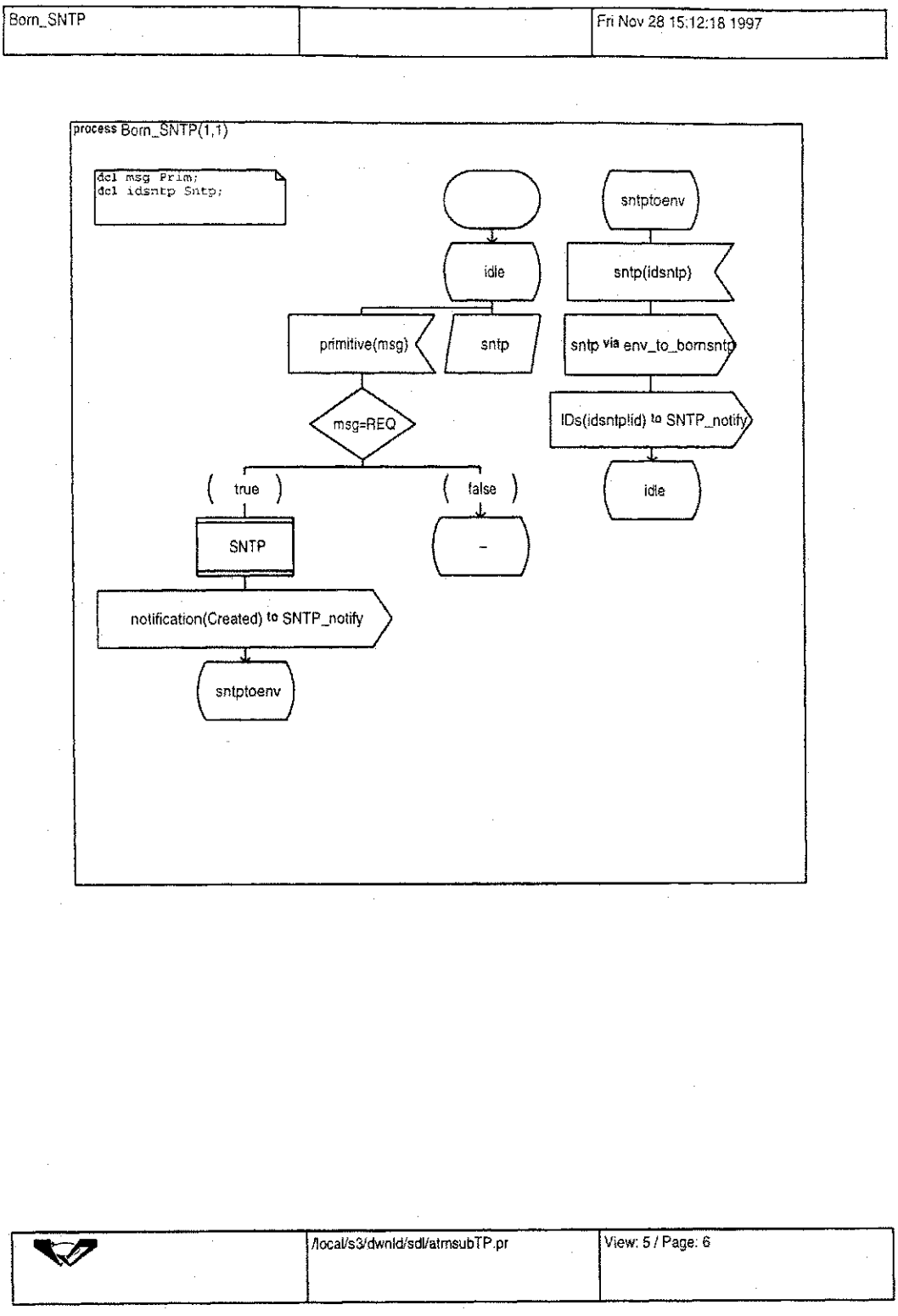


Figura 5.16: Processo Born_SNTP

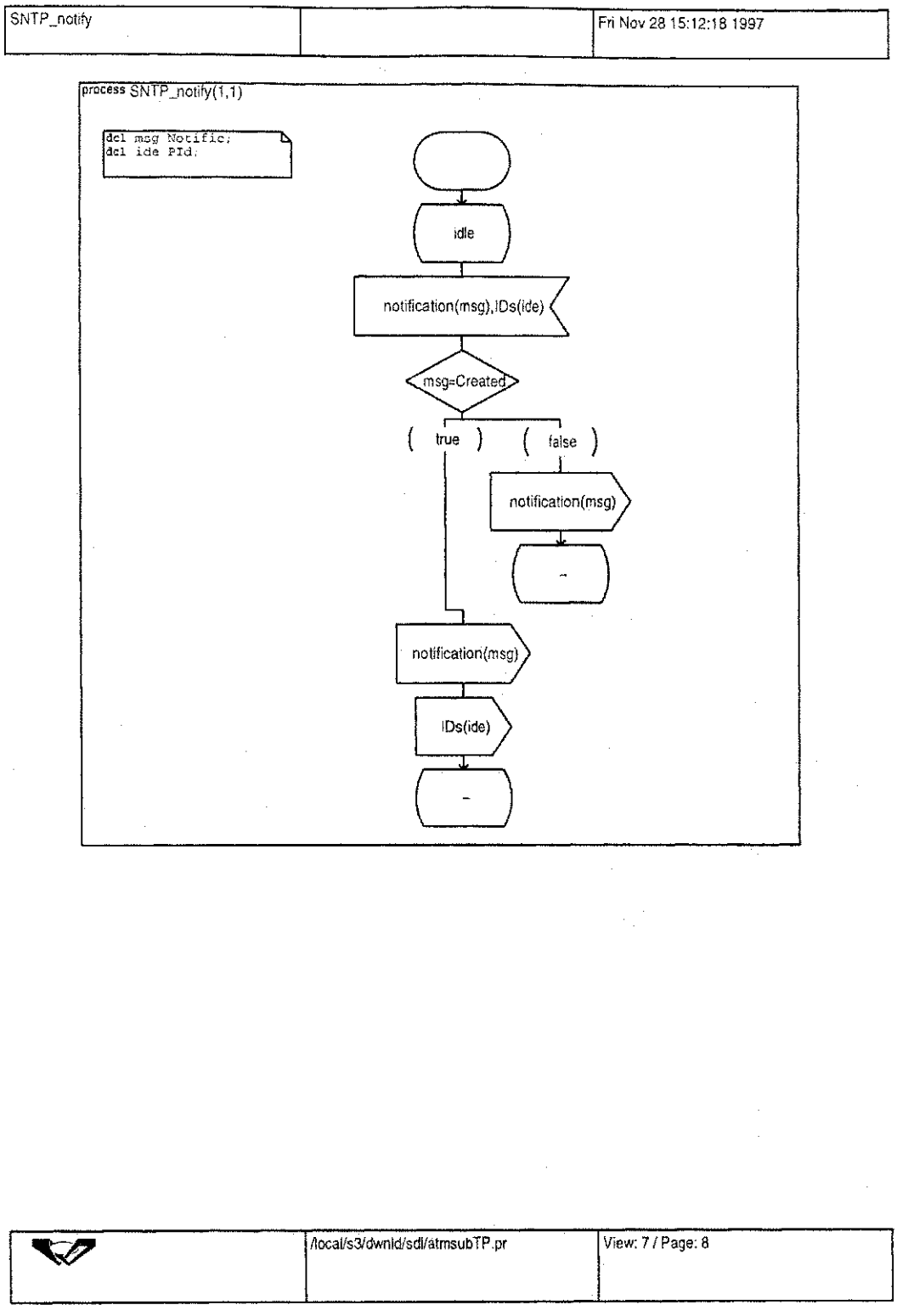


Figura 5.17: Processo SNTP_notify

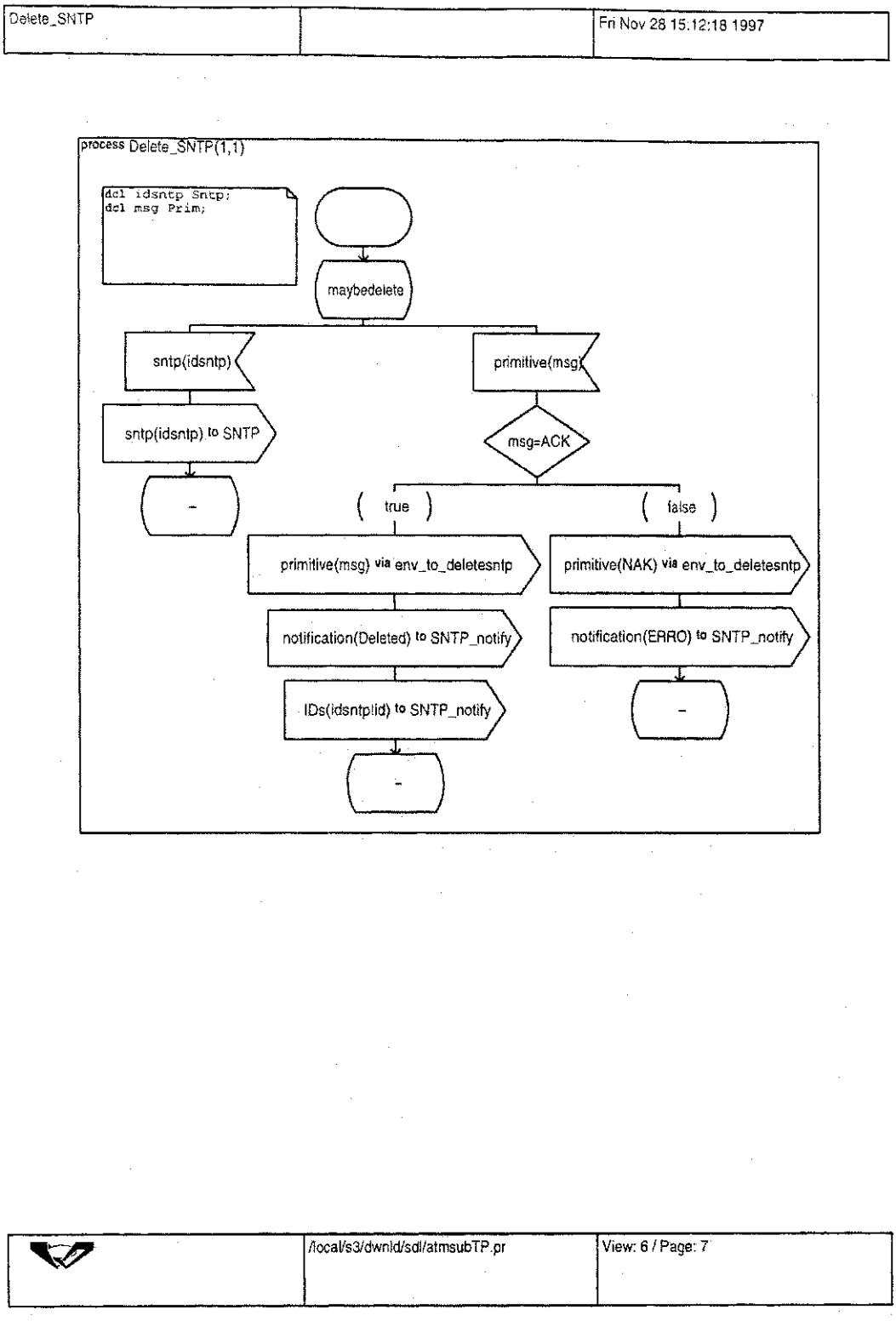


Figura 5.18: Processo Delete_SNTP

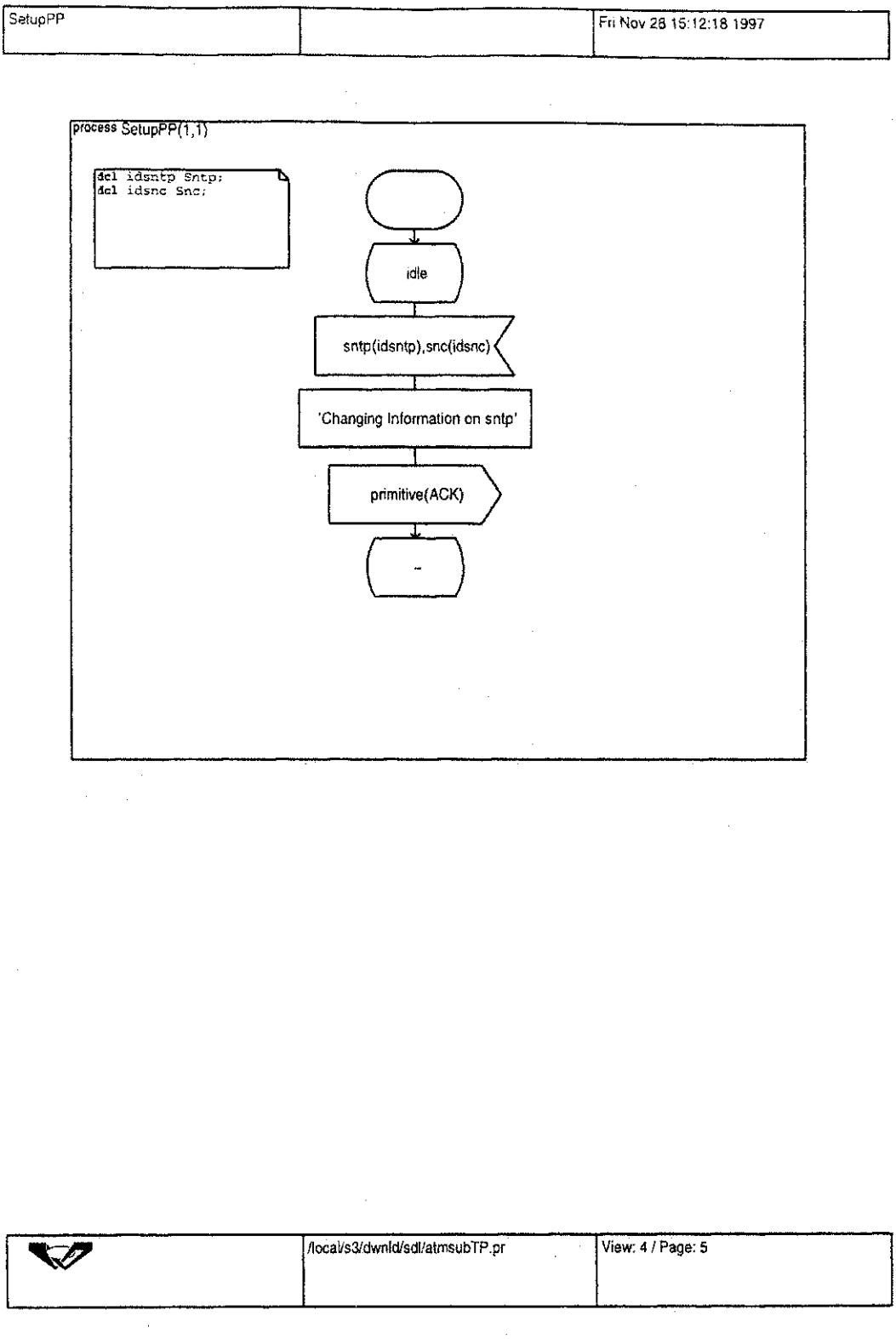


Figura 5.19: Processo SetupPP

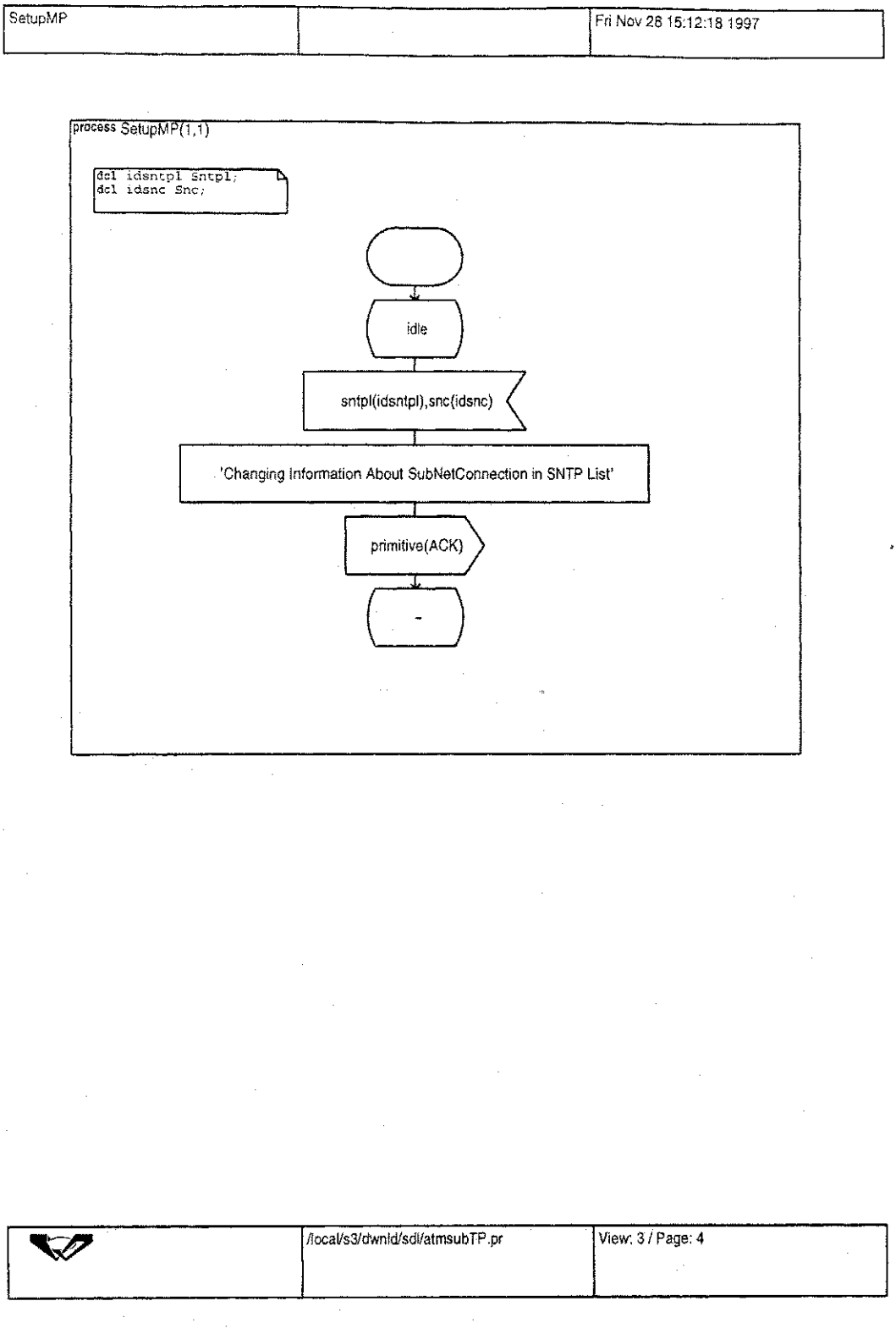


Figura 5.20: Processo SetupMP

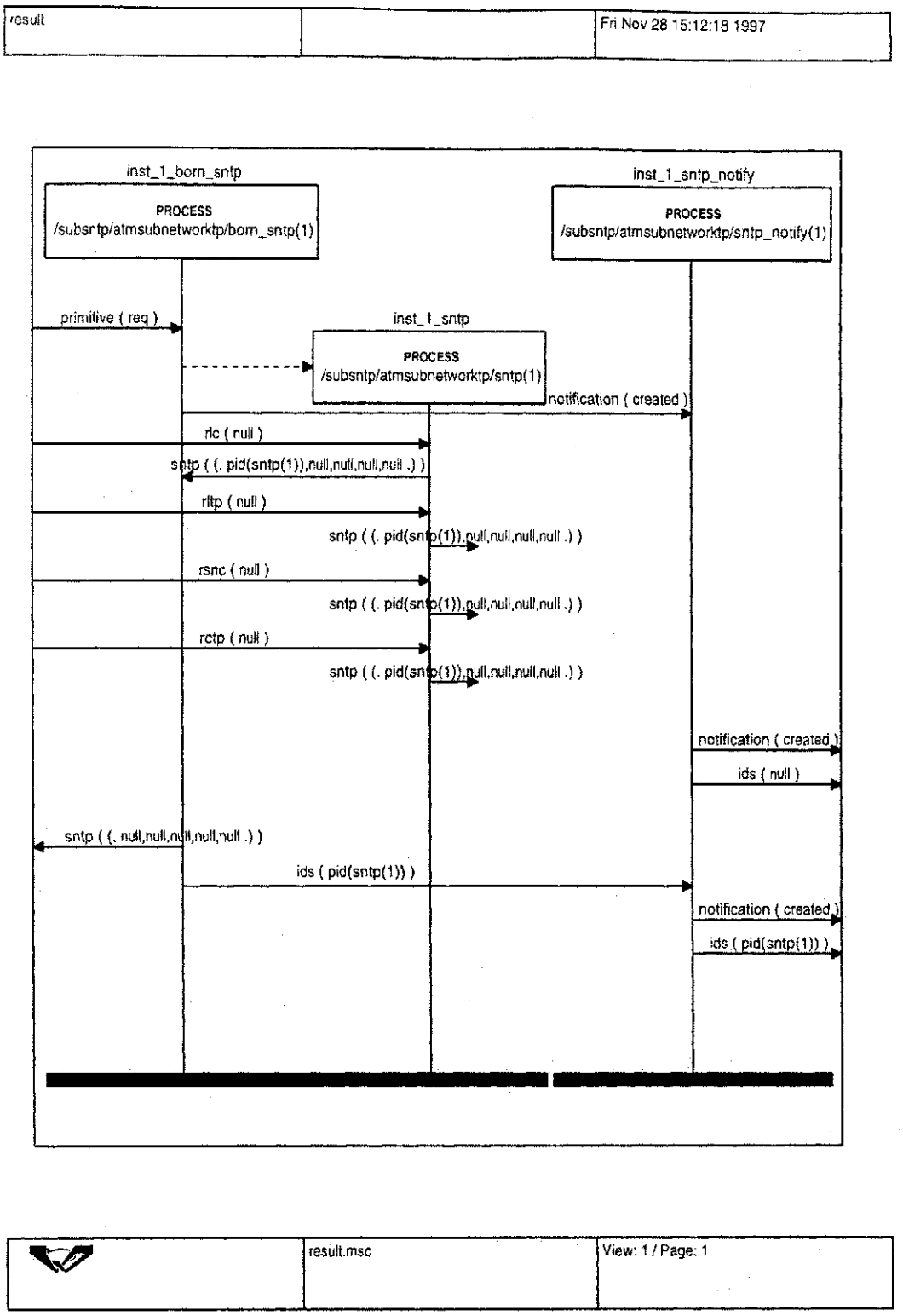


Figura 5.21: MSC do sistema subSNTP, para a criação de um SNTP

Capítulo 6

Conclusões

Em G-CPN não há necessidade de artifícios para a simulação de eventos concorrentes. A pesquisa tomou como base técnicas de descrição formal padronizadas ou em vias de padronização[1] pelo ITU-T e ISO, portanto foi considerada inicialmente uma abordagem de avaliação de G-CPN, SDL'92 e Lotos, como as duas primeiras já dispõem de caracteristícas de orientação a objetos, a linguagem Lotos foi então descartada.

Observa-se que o problema da definição de EOGs é um problema do domínio da Engenharia de Software e que os sistemas desenvolvidos em G-CPN conseguem tratar naturalmente a concorrência e troca de mensagens entre instâncias de objetos.

Verificou-se que G-CPN caracteriza-se por sua simplicidade, por dispor apenas de duas primitivas, lugares e transições. SDL, por outro lado, tem cerca de 20 ou mais primitivas (create, task, start, etc.). Um representação em SDL poder ser expressa tanto textualmente quanto gráficamente. SDL ainda dispõe de um padrão, que cria especificações gráficas através de simples representações textuais, conhecida como CIF (Commom Interchange Format), que facilita o intercâmbio de especificações entre usuários, mesmo que se utilizem de ferramentas de diferentes fabricantes. Devido ao fato de SDL ter se tornado um padrão, também na indústria, as principais ferramentas desenvolvidas (ObjectGEODE e SDT) disponibilizam um suporte razoável, o mesmo

[1] CPN encontra-se em vias de padronização pelo ITU-T e ISO

não acontece com ferramentas para Redes de Petri, pelo fato não terem larga utilização na indústria.

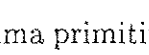
A experiência com a modelagem em G-CPN e fato dos modelos já estarem prontos facilitou a construção do sistema em SDL.

A utilização de G-CPN na modelagem destas entidades resultou em mudanças na própria especificação formal da mesma, como exemplo, podemos citar que sistemas G-CPN não dispunham anteriomente de primitivas de comunicação assíncronas. Foi verificado ainda que, o processo de refinamento dos modelos implicou em mudanças sobre a própria semântica de funcionamento do mesmo. A modelagem serviu também para aprendermos sobre o próprio funcionamento do sistema. Cria-se assim um ciclo de aprimoramento do sistema modelado. As dúvidas relativas ao modelo não existiriam, se as mesmas não fossem apresentadas de maneira informal, implicando assim em ambigüidades de interpretações.

Pode-se observar que a utilização da ferramenta DesignCPN introduz algumas dificuldades para a anaálise de sistemas G-CPN. Desta forma a necessidade de uma ferramenta para a modelagem, análise e simulação de sistemas G-CPN diretamente teria simplicado bastante o processo de análise dos modelos G-CPN apresentados neste trabalho. Por outro lado, deve-se enfatizar que independentemente da existência de ferramentas o modelo G-CPN adequa-se mais fortemente para a modelagem de sistemas onde características do paradigma de orientação a objetos possam contribuir de forma efetiva para a especificação, modelagem, análise e simulação de sistemas complexos.

Dentre os requisitos impostos por Derrick[DLT95] apenas a herança múltipla, ainda não é contemplada por G-CPN. Este trabalho serviu portanto para aperfeiçoar o formalismo de G-CPN e demonstrar que G-CPN serve ao objetivo de modelagem de EOGs e que possibilita uma análise menos superficial, do que SDL, dos modelos. Ao menos, diante das ferramentas utilizadas.

Devido ao vasto conjunto de primitivas, SDL interage de uma melhor forma com o engenheiro de software, imagine a seguinte situação: deseja-se criar um processo, em

SDL dispomos de uma primitiva **create**,  , que se encarrega de criar o processo, em G-CPN temos que criar toda a estrutura necessária ao método criar. Devido ao completo embasamento matemático de G-CPN, pode-se pensar na utilização de G-CPN para representação formal de SDL, a exemplos de trabalhos como o de Hans[FG97], ao invés de máquinas de estado finito. Isso passa pela representação dos conceitos de SDL em redes G-CPN(processo, fila de entrada, criação, etc.), As possibilidades de uma ferramenta que recebe-se especificações em SDL(Padrão largamente utilizado) e as transforma-se em G-CPN(definições coesas), e daí em código, tem mercado certo.

Como contribuição, este trabalho levou a redefinição de alguns aspectos do formalismo de G-CPN:

- Anteriormente G-CPN não continha primitivas de comunicaão assíncronas. Quando uma mensagem era enviada o sistema aguardava o retorno da resposta do ISP, a fim de continuar a simulação.

- G-CPN não continha primitivas de notificação, onde envia-se uma mensagem sem esperar um retorno.

- Foi verificado que o acesso aos contextos é difícil. Tem-se procurado soluções para este problema, de modo que o acesso ao contexto seja facilitado.

- A tarefa de construção de uma ferramenta para modelagem em G-CPN é indispensável. A tarefa de modelagem é ardua, pois é necessária uma transformação em CPN equivalente e daí simular na ferramenta Design/CPN.

- Espera-se que os estados introduzidos pelo ambiente sejam transparentes para efeito da análise do modelos. Quando se desenvolve um modelo CPN equivalente, os estados que fazem parte do ambiente são adicionados a simulação. Implicando num número de estados maior do que existentes, quando o ambiente é desconsiderado.

- Contribuições relativas a metodogia utilizada para a modelagem de EOG incluem basicamente:

    1. Caso haja uma descrição MIB, partir da descrição através da mesma. Do contrário levantar as características da entidade que será gerenciada e desenvolver o modelo formal a partir destas características.
    2. Simular o modelo e verificar se cumpre as especificações do OG, do contrário, é preciso redefinir o modelo formal.
    3. Uma vez que o modelo funcione como esperado, sendo verificado através das análises realizadas, o processo de definição da MIB pode ser automatizado, por meio de um gerador automático de código. Este gerador recebe especificações formais e as converte em código GDMO(CMIP).

Por fim, quando se define formalmente os OG ganha-se através:

- Possibilidade de automatização do processo de verificação;
- Aumento da confiabilidade;
- Anula-se a possibilidade de erro de projeto;
- Diminui-se a redundância do sistema que contém o objeto;
- As duas Técnicas de Descrição Formal prestam-se para isso, sendo que a escolha reside basicamente na familiaridade e grau de padronização.

Referências Bibliográficas

[BHS91] F. Belina, D. Hogrefe, and A. Sarma. *SDL With Applications from Protocol Specification.* Bcs Practioner. Prentice-Hall, 1991.

[Bla95] U. Black. *Network Management Standards.* McGraw-Hill Series on Computer Communications. McGraw-Hill, 2nd edition, 1995.

[Boo94] G. Booch. *Object Oriented Design: With Applications.* Menlo Park, CA: Benjamin/Cummings, 1994.

[Com94] ATM Forum Technical Committee. M4 interface requirements and logical mib - af-nm-0020.000. Technical report, ATM Forum, October 1994.

[Com96] The ATM Forum Technical Committee. M4 network - view interface requirements, and logical mib - af-nm-0058.000. Technical report, The ATM forum, March 1996.

[DCdFP93a] Y. Deng, S.K. Chang, J.C.A. de Figueiredo, and A. Perkusich. Integrating software engineering methods and petri nets for the specification and prototyping of complex software systems. In M. Ajmone Marsan, editor, *Application and Theory of Petri Nets 1993*, volume 691 of *Lecture Notes in Computer Science*, pages 206 – 223. Springer-Verlag, Chicago, USA, June 1993.

[DCdFP93b] Y. Deng, S.K. Chang, J.C.A. de Figueiredo, and A. Perkusich. Integrating software engineering methods and petri nets for the specification and

prototyping of complex software systems. In M. Ajmone Marsan, editor, *Application and Theory of Petri Nets 1993*, volume 691 of *Lecture Notes in Computer Science*, pages 206 - 223. Springer-Verlag, Chicago, USA, June 1993.

[Den92] Y. Deng. *A Unified Framework for the Modeling, Prototyping and Design of Distributed Information Systems.* PhD thesis, Department of Computer Science, University of Pittsburgh, 1992.

[DLT95] J. Derrick, P.F. Linington, and S.J. Thompson. Formal description techniques for object management. *Integrated Network Management*, IV:641–653, January 1995.

[EHS97] J. Ellsberger, D. Hogrefe, and A. Sarma. *SDL - formal Object Oriented Language for Communicating Systems.* Prentice-Hall, 1997.

[Fes95] O. Festor. Mode: a development environment for managed objects based on formal methods. *Integrated Network Management*, IV:616 - 628, January 1995.

[FG97] Hans Fleischhack and Bernd Grahlmann. Towards compositional verification of sdl systems. 1997.

[GdFP97] D.D.S. Guerrero, J.C.A. de Figueiredo, and A. Perkusich. Object-based high-level petri nets as a formal approach to distributed information systems. In *Proc. of IEEE Int. Conf. on Systems Man and Cybernetics*, page To Appear, Orlando, USA, Invited Paper, October 1997.

[Gen89] H.J. Genrich. Equivalence transformations of PrT-Nets. In G. Rozenberg, editor, *Advances in Petri Nets 1989*, volume 424 of *Lecture Notes in Computer Science*, pages 179–208. Springer-Verlag, 1989.

[Gen91] H.J. Genrich. Predicate/Transition nets. In K. Jensen and G. Rozenberg, editors, *High-Level Petri Nets: Theory and Application*. Springer-Verlag, 1991.

[GFaPdF97] D.D.S. Guerrero, A. Falcão, A. Perkusich, and J.C.A. de Figueiredo. A g-cpn approach for m4 network-view interface managed objects entities modeling. In *Anais do XV Simpósio Brasileiro de Teleomunicações*, pages 35–39, Recife, PE, Brasil, September 1997.

[GFPdF97] D.D.S. Guerrero, J. P. Fernandes, A. Perkusich, and de Figueiredo. An object based petri net model: Application to manufacturing systems. In *Proc. of IEEE Int. Conf. on Systems Man and Cybernetics*, page To Appear, Orlando, USA, October 1997.

[GPdF97] D.D.S. Guerrero, A. Perkusich, and J.C.A. de Figueiredo. Modeling a cooperative environment based on an object-based modular petri net. In *Proc. of The 9th International Conference on Software Engineering and Knowledge Engineering*, pages 240–247, Madrid, Spain, June 1997.

[Gue97] D.D.S. Guerrero. Sistemas de redes de petri modulares baseadas em objetos. Dissertacao de mestrado, COPIN - Universidade Federal da Paraiba, 1997.

[IT92a] ITU-T. Information technology - open systems interconnection - systems management overview - recommendation x.701. Technical report, ITU, Geneva, 1992.

[IT92b] ITU-T. Information technology - opens systems interconnection - system management overview - recommendation x.722. Technical report, ITU, Geneva, 1992.

[IT92c] ITU-T. Maintenance: Telecommunications management network - ge-

neric network information model - recommendation m.3100. Technical report, ITU, Geneva, October 1992.

[IT93a] ITU-T. Digital networks - architeture of transport networks based on the synchronous digital hierarchy(sdh) - itu-t recommendation g.803. Technical report, ITU, Geneva, 1993.

[IT93b] ITU-T. Digital networks - generic functional architeture of trasnport networks - itu-t recommendation g.805. Technical report, ITU, Geneva, 1993.

[IT93c] ITU-T. Programming languages - ccitt specification and description language - itu-t recommendation z.100. Technical report, ITU, Geneva, 1993.

[IT93d] ITU-T. Programming languages - sdl formal definition - itu-t z.100 annex f. Technical report, ITU, Geneva, 1993.

[IT93e] ITU-T. Programming languages - sdl methodology guidelines adn bibliography - itu-t z.100 appendix 1 and 2. Technical report, ITU, Geneva, 1993.

[Jen87] K. Jensen. Coloured petri nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Lecture Notes in Computer Science, Advances in Petri Nets: Petri Nets, Central Models and Their Properties*, volume 254, pages 248–299. Springer-Verlag, 1987.

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

[Liu85] C.S. Liu. *Elements of Discrete Mathematics.* Computer Science. McGraw-Hill, 2nd edition, 1985.

[Man97] The ATM Forum Technical Committee Network Management. M4 network - view cmip mib specification version 1.0 - af-nm-0073.000. Technical report, The ATM forum, January 1997.

[Mes96] D.G. Messerschmitt. The converge of telecommunications and computing: What are the implications today? *Departament of Electrical Engineering adn Computer Sciences - University of California, Berkeley, California 94720*, 1996.

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

[OG96] University of Aarhus. *Design/CPN - Occurrence Graph Manual - Version 3.0*, 1996.

[oSA95] Office of Safety and Mission Assurance. Formal methods specification and verification guidebook for software and computer vol i: Planning and technology insertion. Technical report, National Aeronautics and Space Administration, Washington, DC, July 1995.

[PdF97] A. Perkusich and J.C.A. de Figueiredo. G-nets: A petri net based approach for logical and timing analysis of complex software systems. *To Appear in Journal of Systems and Software*, 1997.

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

[PdFM93] A. Perkusich, J.C.A. de Figueiredo, and M.E. Morais. Análise e verificação de sistemas baseados em objetos utilizando uma abordagem por redes de petri. In *Anais do XIX Seminário Integrado de Software e Hardware, SEMISH 93*, September 1993.

[Per94] A. Perkusich. *Análise de Sistemas Complexos Baseada na Decomposição de G-Nets.* PhD thesis, Curso de Doutorado em Engenharia Elétrica, Universidade Federal da Paraíba, Campina Grande, PB, August 1994. Também disponível em inglês com título: Analysis of G-Net Systems Based Upon Decomposition.

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

[Sim96] E. Simon. *Distributed Information Systems.* McGraw-Hill, 1996.

[Udu96] D.K Udupa. *Network Management Systems Essentials.* Computer Communications. McGraw-Hill, 1996.

[Uni96] University of Aarhus, Aarhus, Denmark. *Design CPN - Overview of CPN ML Syntax, version 3.0*, 1996.

Apêndice A - M4-Network View CMIP Managemmnet Information Base

```
atmLinkTP MANAGED OBJECT CLASS
        DERIVED FROM "ITU-T Rec. X.721 | ISO/IEC 10165-2":top;
        CHARACTERIZED BY
                "ITU-T M.3100:1995":createDeleteNotificationsPackage,
                "ITU-T
M.3100:1995":attributeValueChangeNotificationPackage,
                "ITU-T M.3100:1995":characteristicInformationPackage,
                atmLinkTPPackage  PACKAGE
                        BEHAVIOUR atmLinkTPBeh;
                        ATTRIBUTES
                                atmLinkTPId
                                        GET
                                        SET-BY-CREATE,
                                availableIngressBandwidth
                                        GET,
                                availableEgressBandwidth
                                        GET,
```

```
                                maxAssignableIngressBandwidth
                                        GET,
                                maxAssignableEgressBandwidth
                                        GET,
                                linkPointer
                                        GET,
                                atmNetworkAccessProfilePointer
                                        GET-REPLACE,
                                serverTTPList
                                        GET-REPLACE ADD-REMOVE;;;
        CONDITIONAL PACKAGES
                supportingUNIorNNIPackage PRESENT IF "The atmLinkTP
allows a pointer to the supporting UNI or NNI";
REGISTERED AS {atmfM4NwObjectClass 3};

atmLinkTPBeh BEHAVIOUR
        DEFINED AS
" An atmLinkTP is a topological component used to represent the
termination of an atmLink.  Link level configuration information may be
associated with the atmLinkTP object.  An instance of atmLinkTP is
created explicitly by the management system, the managed system, or the
setupLink ACTION.  Access parameters and pointers to the underlying TTPs
are represented in the related atmNetworkAccessProfile objects.

The atmLinkTP terminates atmLink and provides the capability to store
link level configuration information.

The available bandwidth attributes describe the aggregated amount of
unallocated bandwidth in the ingress and egress directions.  The maximum
assignable bandwidth attributes describe for each direction the maximum
```

bandwidth that may actually be assigned to a new connection. In the case where the atmLink is supported by a single server trail, the maximum assignable bandwidth is the same as the available bandwidth. The associated atmNetworkAccessProfile describes the total amount of bandwidth (i.e., allocated and unallocated) for the atmLink. This total bandwidth shall be consistent with the bandwidth allocated to the server trails.

The serverTTPList attribute points to the underlying or server TTPs that support the atmLinkTP. If applicable, at the vcLayer these are instances of the vpTTPBid object class. If applicable, at the vpLayer these are instance of the tcAdaptorTTPBid object class.

Attribute value change notifications are applicable to the linkPointer, atmNetworkAccessProfilePointer, and serverTTPList attributes, but not to the attributes associated with available and assignable bandwidth because of their dynamic nature.

The characteristics described by the atmNetworkAccessProfile associated with an atmLinkTP shall be consistent with the atmNetworkAccessProfile of the related atmLink. Note that the related atmNetworkAccessProfile information is also in the NE-view atmAccessProfile object contained in the tcAdaptorTTPBidirectional or in the vpTTPBidirectional object:

- If the profiling information in the atmNetworkAccessProfile object includes attribute values that are more constraining than those in the Access Profile contained in the server TTP and if the corresponding atmLinkTP points to multiple server TTPs, then the server TTP Access Profile information takes precedence over the one in the atmNetworkAccessProfile. If not, the atmNetworkAccessProfile

```
information applies.

- If the profiling information in the atmNetworkAccessProfile object
includes attribute values that are less constraining than those in the
Access Profile contained in the server TTP and if the multiple atmLinkTP
points to a single server TTP, then the atmNetwork AccessProfile
information takes precedence over the server TTP Access Profile.  If
not, the server TTP Access Profile information applies.";



atmNetworkAccessProfile MANAGED OBJECT CLASS
        DERIVED FROM "ITU-T Rec. X.721 | ISO/IEC 10165-2":top;
        CHARACTERIZED BY
                "ITU-T M.3100:1995":createDeleteNotificationsPackage,
                "ITU-T
M.3100:1995":attributeValueChangeNotificationPackage,
                atmNetworkAccessProfilePackage  PACKAGE
                        BEHAVIOUR atmNetworkAccessProfileBeh;
                        ATTRIBUTES
                                atmNetworkAccessProfileId
                                        GET
                                        SET-BY-CREATE,
                                totalEgressBandwidth
                                        GET-REPLACE,
                                totalIngressBandwidth
                                        GET-REPLACE,
                                maxNumActiveConnectionsAllowed
                                        GET-REPLACE,
                                vpiOrVciRange
```

GET-REPLACE;;;

CONDITIONAL PACKAGES

"ATMF M4 NEView": atmSubscriberAddressPkg PRESENT IF "The atmLinkTP has a subscriber address assigned directly.",

"ATMF M4 NEView": preferredCarrierPkg PRESENT IF "The atmLinkTP has a preferred carrier assigned directly.";

REGISTERED AS {atmfM4NwObjectClass 4};

atmNetworkAccessProfileBeh BEHAVIOUR

DEFINED AS

" An atmNetworkAccessProfile contains information that describe the maximum ingress and egress bandwidth, along with the range of VPI or VCI values that are applied to the atmLink or atmLinkTP object instances that point to it.

Note that NE-view atmAccessProfile object contained in the tcAdaptorTTPBidirectional or in the vpTTPBidirectional object contains information that shall be consistent with the atmNetworkAccessProfile:

- If the profiling information in the atmNetworkAccessProfile object includes attribute values that are more constraining than those in the Access Profile contained in the server TTP and if the corresponding atmLinkTP points to multiple server TTPs, then the server TTP AccessProfile information takes precedence over the one in the atmNetworkAccessProfile. If not, the atmNetworkAccessProfile information may apply.

- If the profiling information in the atmNetworkAccessProfile object includes attribute values that are less constraining than those in the Access Profile contained in the server TTP and if the multiple atmLinkTP

```
points to a single server TTP, then the atmNetworkAccessProfile
information takes precedence over the server TTP Access Profile.  If
not, the server TTP Access Profile information applies.";


atmNetworkTTP MANAGED OBJECT CLASS
        DERIVED FROM "ITU-T Rec. X.721 | ISO/IEC 10165-2":top;
        CHARACTERIZED BY
                "ITU-T M.3100:1995":
attributeValueChangeNotificationPackage,
                "ITU-T M.3100:1995": createDeleteNotificationsPackage,
                "ITU-T M.3100:1995":characteristicInformationPackage,
                atmNetworkTTPPkg PACKAGE
                        BEHAVIOUR atmNetworkTTPBeh;
                        ATTRIBUTES
                                atmNetworkTTPId
                                        GET,
                                relatedAtmCTP
                                        GET-REPLACE;;;
                CONDITIONAL PACKAGES
                "ATMF M4 NEView": oamCellLoopbackPkg
                        PRESENT IF "the termination point supports OAM
cell Loopbacks",
                "ITU-T
M.3100:1995":tmnCommunicationsAlarmInformationPackage
                        PRESENT IF "communication alarms are supported
by this object.",
                "ITU-T
M.3100:1995":alarmSeverityAssignmentPointerPackage
                        PRESENT IF "communication alarms are supported
```

```
by this object.",
                "ITU-T Rec. X.721 | ISO/IEC 10165-
2":availabilityStatusPackage
                        PRESENT IF "the object supports an indication of
a degraded or failure state.";
REGISTERED AS {atmfM4NwObjectClass 6};

atmNetworkTTPBeh BEHAVIOUR
        DEFINED AS
" The atmNetworkTTP object class is used when the Network View only is
provided.

The relatedAtmCTP attribute is used to associate the final CTP of a VCC
or VPC with the Trail Termination Point.

The conditional package oamCellLoopbackPkg provides the M-ACTION used to
request the termination point to insert an OAM cell for downstream
loopback and to report whether or not the cell was returned within the
required time.

The availabilityStatusPackage is a conditional package that may be used
to indicate the availability of the atmNetworkTTP.  Changes in the
availabilityStatus are reported using the
attributeValueChangeNotification.

Supported values for the availabilityStatus are:
- Failed:  The atmLinkConnection cannot function
- Empty SET (none of the availableStatus conditions exist). ";
```

```
atmRoutingProfile MANAGED OBJECT CLASS
        DERIVED FROM "ITU-T Rec. X.721 | ISO/IEC 10165-2":top;
        CHARACTERIZED BY
                "ITU-T M.3100:1995": createDeleteNotificationsPackage,
                atmRoutingProfilePkg PACKAGE
                        BEHAVIOUR atmRoutingProfileBeh;
                        ATTRIBUTES
                                atmRoutingProfileId
                                        GET,
                                routeDescriptionList
                                        GET-REPLACE ADD-REMOVE,
                                maxHops
                                        GET-REPLACE,
                                connectionTypesSupported
                                        GET-REPLACE ADD-REMOVE;;;
REGISTERED AS {atmfM4NwObjectClass 8};

atmRoutingProfileBeh BEHAVIOUR
        DEFINED AS
" The atmRoutingProfile object class represents a set of topological
routing constraints that can be applied to a new connection or trail
during setup.  A routing profile may be created automatically based on
the routing description in the setup action.  The management system may
also create profiles directly.  Each atmSubnetworkConnection or atmTrail
may point to an atmRoutingProfile.  Connections shall not be established
(or re-established) if the routing criteria cannot be met.  If maxHops
is specified, the connection shall not be established (or re-
established) if the maximum number of hops is exceeded (including hops
that may not be visible to the managing systems).
```

The maxHops attribute is the maximum number of hops between nodes that the new connection may traverse. This attribute may be set to NULL to indicate that the maxHops constraint does not apply.

The routeDescriptionList attribute is a list of objects (such as Links, Subnetworks, existing connections) and their use in routing (exclude, mandatory, preferred, same route, diverse route).

The connection types that the routing profile supports are indicated in the connectionTypesSupported attribute. For all types of multipoint connections only the sameRoute condition can be applied. All of the criteria can be applied to point-to-point connections.

Objects (such as atmSubnetwork, atmLink, or managedElement, etc) may be referenced by the routeDescriptionList as being excluded, mandatory, or preferred. If an object is described as mandatory it must be used in setting up a new connection. An attempt must be made during setup to include an object described as preferred. An excluded object must not be used in a connection.

Connection objects (such as atmTrail, atmSubnetworkConnection, etc) may be referenced by the routeDescriptionList as same route or diverse route. A new connection being created shall follow the same route as a sameRoute referenced object. A new connection must follow a different route than a referenced object referred to as diverseRoute.

The routing information in setup actions may be either explicitly stated in the action or the action can point to an existing instance of the atmRoutingProfile object class. ";

```
atmSubnetwork MANAGED OBJECT CLASS
        DERIVED FROM "ITU-T M.3100:1995":networkR1;
        CHARACTERIZED BY
                "ITU-T M.3100:1995":createDeleteNotificationsPackage,
                "ITU-T
M.3100:1995":attributeValueChangeNotificationPackage,
                "ITU-T Rec. X.721 | ISO/IEC 10165-
2":availabilityStatusPackage,
                "ITU-T M.3100:1995":userLabelPackage,
                "ITU-T M.3100:1995":characteristicInformationPackage,
                atmSubnetworkPackage  PACKAGE
                        BEHAVIOUR atmSubnetworkBeh;
                        ATTRIBUTES
                                "ITU-T
M.3100:1995":supportedByObjectList
                                        GET-REPLACE ADD-REMOVE,
                                containedLinkList
                                        GET-REPLACE ADD-REMOVE,
                                containedSubnetworkList
                                        GET-REPLACE ADD-REMOVE,
                                supportedLinkTPList
                                        GET-REPLACE ADD-REMOVE;;;
        CONDITIONAL PACKAGES
                subnetworkMultipointActionsPackage PRESENT IF "the
atmSubnetwork
                        supports multipoint connections.",
                subnetworkConnectionManagementPackage PRESENT IF "the
```

subnetwork

represented by the object instance supports connection management";

REGISTERED AS {atmfM4NwObjectClass 9};

atmSubnetworkBeh BEHAVIOUR

DEFINED AS

"An atmSubnetwork is a topological component used for carrying characteristic information(ATM cells within a layer network). The atmSubnetwork is delineated by ATM Subnetwork Termination Points (atmSubnetworkTP). An atmSubnetwork may: be empty containing no atmSubnetworkTP instances; associated with a single linkTP in which case it referred to as a point subnetwork; or associated with many termination points. Subnetworks are used for making subnetwork connections. An instance of atmSubnetwork is specific to the VC or VP layer and is contained in the appropriate vcLayerNetworkDomain or vpLayerNetworkDomain. A point subnetwork does not contain any visible subnetwork connections.

The atmSubnetwork object provides an abstraction that allows the establishment and removal of connections across the atmSubnetwork.

characteristicInformation describes the format of the characteristic information that the resource carries. This is set to vcCI (I.751) for VC Layer atmSubnetworks and vpCI (I.751) for VP Layer atmSubnetworks. The characteristicInformation, where present, for dependent objects shall match this attribute

The userLabel may be used to describe the managing organization. In cases where the atmSubnetwork is managed by a different system the

inherited systemTitle may be used.

The supportedByObjectList points to managed elements that support the subnetwork. (specific information about these elements is available through the M4 NE view)

Supported values for the availabilityStatus are:
- Degraded: The atmSubnetwork is degraded in some respect. For instance, the atmSubnetwork cannot perform the function of establishing new atmSubnetworkConnections while it can still accept ACTIONS to tear down existing connections.
- Empty SET (none of the availableStatus conditions exist).

The setupSubnetworkConnection ACTION sets up a point-to-point or a multipoint connection between non-connected subnetworkTPs in the atmSubnetwork.

The modifySubnetworkConnection ACTION modifies the QOS and traffic descriptors of a point-to-point or a multipoint connection between subnetworkTPs in the atmSubnetwork.

The addTpsToSubnetworkConnection ACTION adds atmSubnetworkTPs to an existing multipoint subnetwork connection.

The removeTpsFromSubnetworkConnection ACTION releases atmSubnetworkTPs from a multipoint atmSubnetworkConnection.

The releaseSubnetworkConnection ACTION releases a point-to-point or a multipoint connection between subnetworkTPs in the atmSubnetwork.";

```
atmSubnetworkConnection MANAGED OBJECT CLASS
        DERIVED FROM "ITU-T Rec. X.721 | ISO/IEC 10165-2":top;
        CHARACTERIZED BY
                "ITU-T M.3100:1995":createDeleteNotificationsPackage,
                "ITU-T
M.3100:1995":attributeValueChangeNotificationPackage,
                "ITU-T M.3100:1995":characteristicInformationPackage,
                "ITU-T M.3100:1995":userLabelPackage,
                "ITU-T Rec. X.721 | ISO/IEC 10165-
2":availabilityStatusPackage,
                atmSubnetworkConnectionPackage  PACKAGE
                        BEHAVIOUR atmSubnetworkConnectionBeh;
                        ATTRIBUTES
                                atmSubnetworkConnectionId
                                        GET,
                                a-TPInstance
                                        GET,
                                z-TPList
                                        GET,
                                connectionType
                                        GET-REPLACE,
                                restorableIndicator
                                        GET-REPLACE,
                                componentSubnetworkConnectionList
                                        GET-REPLACE ADD-REMOVE,
                                componentLinkConnectionList
                                        GET-REPLACE ADD-REMOVE,
                                provisionType
```

```
                                        GET-REPLACE,
                                relatedAtmRoutingProfile
                                        GET-REPLACE,
                                "ITU-T Rec. X.721 | ISO/IEC 10165-2":
                                                administrativeState
                                        GET-REPLACE;;;
        CONDITIONAL PACKAGES
                retainResourcesPackage PRESENT IF "retention of
supporting resources
                        after atmTrail or containing
atmSubnetworkConnection release is
                        supported.",
                "ITU-T
M.3100:1995":tmnCommunicationsAlarmInformationPackage
                        PRESENT IF "communication alarms are supported
by this object.",
                "ITU-T
M.3100:1995":alarmSeverityAssignmentPointerPackage
                        PRESENT IF "communication alarms are supported
by this object.";
REGISTERED AS {atmfM4NwObjectClass 10};

atmSubnetworkConnectionBeh BEHAVIOUR
        DEFINED AS
" An atmSubnetworkConnection represents a connection across a
subnetwork.  An atmSubnetworkConnection is responsible for transporting
cells across a subnetwork.  It is always bidirectional.  An instance of
atmSubnetworkConnection is terminated by atmSubnetworkTPs.

An instance of this object is created by the managed system or by an
```

action on the atmSubnetwork object. An atmSubnetworkConnection in a composite subnetwork is made up of a series of atmSubnetworkConnections and atmLinkConnections. An atmSubnetworkConnection cannot be created between a composite subnetwork and one of its component subnetworks.

Supported values for the availabilityStatus are:
- Failed: The atmSubnetworkConnection cannot function
- Empty SET (none of the availableStatus conditions exist).

The administrativeState is used for administratively locking and unlocking the atmSubnetworkConnection. When unlocked, the atmSubnetworkConnection functions normally. When in the locked state, the atmSubnetworkConnection is prohibited from the transport of characteristic information.

For point to point Subnetwork Connections the a-TPInstance and a single entry in the z-TPList are used to indicate the endpoints. Multiple entries in the z-TPList and the a-TPInstance are used to represent the end points of broadcast (point-to-multipoint), merge (multipoint-to-point), and composite connections. The a-TPInstance identifies the primary endpoint. Only the z-TPList is used identify all end points in a multipoint-to-multipoint connection (there is no primary end point). In this case the a-TPInstance shall be NULL.

Optionally where both the Network View and NE View are supported and the atmSubnetworkConnection is not broken down into component network level connections, the componentSubnetworkConnectionList of an atmSubnetwork Connection may point directly to the NE View Cross Connection object instances instead of atmSubnetworkConnections, while the componentLinkConnectionList points to atmLinkConnections whose

```
termination points are instances of the NE View vp or vc CTP object
class.  ";



atmSubnetworkTP MANAGED OBJECT CLASS
        DERIVED FROM "ITU-T Rec. X.721 | ISO/IEC 10165-2":top;
        CHARACTERIZED BY
                "ITU-T M.3100:1995":createDeleteNotificationsPackage,
                "ITU-T
M.3100:1995":attributeValueChangeNotificationPackage,
                "ITU-T M.3100:1995":characteristicInformationPackage,
                "ITU-T M.3100:1995":userLabelPackage,
                atmSubnetworkTPPackage  PACKAGE
                        BEHAVIOUR atmSubnetworkTPBeh;
                        ATTRIBUTES
                                atmSubnetworkTPId
                                        GET
                                        SET-BY-CREATE,
                                relatedLinkConnection
                                        GET,
                                relatedLinkTP
                                        GET,
                                relatedSubnetworkConnection
                                        GET,
                                reflectedCTP
                                        GET-REPLACE;;;
REGISTERED AS {atmfM4NwObjectClass 11};

atmSubnetworkTPBeh BEHAVIOUR
```

```
        DEFINED AS
"An atmSubnetworkTP represent the termination of an
atmSubnetworkConnection or an atmLinkConnection on an atmSubnetwork.
Each atmSubnetworkTP instance refers to a CTP object instance allowing a
single representation of the network resources.  That is, through the
layers of subnetwork decomposition, the instances of atmSubnetworkTP at
each layer all point to the same CTP object instance, thus representing
the relationship among endpoints of composite subnetworks supported by
the same network resource.  An instance of atmSubnetworkTP is created by
the managed system during the set-up of link connections and/or
subnetwork connections.  However, this does not preclude the pre-
provisioning of atmSubnetworkTPs.

The atmSubnetworkTP terminates atmSubnetworkConnections and
atmLinkConnections and provides the capability to associate endpoints of
composite subnetworks with the supporting network resource (CTP).

In cases where the M4 NE view is also supported, the reflectedCTP
attribute may point directly to an instance of the vcCTPBidirectional or
vpCTPBidirectional object classes.";



atmTrail MANAGED OBJECT CLASS
        DERIVED FROM "ITU-T Rec. X.721 | ISO/IEC 10165-2":top;
        CHARACTERIZED BY
                "ITU-T M.3100:1995":createDeleteNotificationsPackage,
                "ITU-T
M.3100:1995":attributeValueChangeNotificationPackage,
                "ITU-T M.3100:1995":characteristicInformationPackage,
```

```
"ITU-T M.3100:1995":userLabelPackage,
atmTrailPackage  PACKAGE
        BEHAVIOUR atmTrailBeh;
        ATTRIBUTES
                atmTrailId
                        GET,
                a-TPInstance
                        GET,
                z-TPList
                        GET,
                connectionType
                        GET-REPLACE,
                restorableIndicator
                        GET-REPLACE,
                provisionType
                        GET-REPLACE,
                relatedAtmRoutingProfile
                        GET-REPLACE,
                "ITU-T Rec. X.721 | ISO/IEC 10165-2":
                                administrativeState
                        GET-REPLACE;;;
    CONDITIONAL PACKAGES
            retainConnectionsPackage PRESENT IF "retention of
supporting connections
                    after trail release is supported.",
            "ITU-T
M.3100:1995":tmnCommunicationsAlarmInformationPackage
                    PRESENT IF "communication alarms are supported
by this object.",
            "ITU-T
```

```
M.3100:1995":alarmSeverityAssignmentPointerPackage
                        PRESENT IF "communication alarms are supported
by this object.";
REGISTERED AS {atmfM4NwObjectClass 12};
```

Apêndice B - Relatórios da Análise dos Cenários Utilizados

Relatório da simulação primeiro cenário:

- método: Setup_SNC da rede SN
- ficha: um snaTP e uma lista com dois tps em snTPzL

```
Statistics
------------------------------------------------------------------------

  Occurrence Graph
    Nodes:  20736
    Arcs:   89856
    Secs:   446
    Status: Full


Boundedness Properties
------------------------------------------------------------------------

  Best Integers Bounds     Upper      Lower
  SN'Create2 1             0          0
  SN'Delete2 1             0          0
  SN'Feed 1                1          1
  SN'Final1 1              2          2
  SN'Final2 1              0          0
  SN'Final3 1              4          0
  SN'ID_gen2 1             1          1
  SN'P1 1                  4          0
  SN'P11 1                 4          0
  SN'P2 1                  4          0
  SN'P3 1                  4          0
  SN'P4 1                  0          0
  SN'P6 1                  1          1
  SN'P8 1                  4          0
  SN'SNCCreate1LP 1        4          0
  SN'SNCCreate1UP 1        4          0
```

```
  SN'SNCSetupMPLP 1       4          0
  SN'SNCSetupMPUP 1       4          0
  SN'SNCSetupPPLP 1       4          0
  SN'SNCSetupPPUP 1       4          0
  SN'SNTPCreate1LP 1      0          0
  SN'SNTPCreate1UP 1      0          0
  SN'SNTPCreateLP 1       0          0
  SN'SNTPCreateUP 1       0          0
  SN'SN_notify1 1         2          2
  SN'SN_notify2 1         2          2
  SN'Saco1 1              12         12
  SN'Saco2 1              12         12
  SN'Saco22 1             0          0
  SN'Saco23 1             0          0
  SN'Saco3 1              4          0
  SN'Saco4 1              4          0
  SN'Saco5 1              4          0
  SN'Setup_SNC 1          4          0
  SN'atmSubNetwork 1      2          2
  SN'wait1 1              4          0
  SN'wait2 1              0          0
  SN'wait3 1              4          0

  Best Upper Multi-set Bounds
  SN'Create2 1          empty
  SN'Delete2 1          empty
  SN'Feed 1             1`[1,2]
  SN'Final1 1           1`(1,0)+ 1`(2,0)
  SN'Final2 1           empty
  SN'Final3 1           1`(ACK,1)+ 1`(NAK,0)+ 1`(NAK,2)+ 1`(NAK,3)
  SN'ID_gen2 1          1`(3,0)
  SN'P1 1               1`((1,1),0)+ 1`((1,1),1)+ 1`((1,1),2)+ 1`((1,1),3)
  SN'P11 1              1`(2,0)+ 1`(2,1)+ 1`(2,2)+ 1`(2,3)
  SN'P2 1               1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)
  SN'P3 1               1`((1,[2,3],2),0)+ 1`((1,[2,3],2),1)+
1`((1,[2,3],2),2)+ 1`((1,[2,3],2),3)
  SN'P4 1               empty
  SN'P6 1               1`([],0)
  SN'P8 1               1`([2,3],0)+ 1`([2,3],1)+ 1`([2,3],2)+ 1`([2,3],3)
  SN'SNCCreate1LP 1     1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)
  SN'SNCCreate1UP 1     1`((1,[2,3]),0)+ 1`((1,[2,3]),1)+ 1`((1,[2,3]),2)+
1`((1,[2,3]),3)
  SN'SNCSetupMPLP 1     1`(ACK,0)+ 1`(ACK,1)+ 1`(NAK,2)+ 1`(NAK,3)
  SN'SNCSetupMPUP 1     1`(([2,3],1),0)+ 1`(([2,3],1),1)+ 1`(([2,3],1),2)+
1`(([2,3],1),3)
  SN'SNCSetupPPLP 1     1`(ACK,1)+ 1`(ACK,3)+ 1`(NAK,0)+ 1`(NAK,2)
  SN'SNCSetupPPUP 1     1`((1,1),0)+ 1`((1,1),1)+ 1`((1,1),2)+ 1`((1,1),3)
  SN'SNTPCreate1LP 1    empty
  SN'SNTPCreate1UP 1    empty
  SN'SNTPCreateLP 1     empty
  SN'SNTPCreateUP 1     empty
  SN'SN_notify1 1       1`(Created_SN,1,0)+ 1`(Created_SN,2,0)
  SN'SN_notify2 1       1`(Created_SN,1,0)+ 1`(Created_SN,2,0)
```

```
  SN'Saco1 1          1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)+ 1`(2,0)+
1`(2,1)+
1`(2,2)+ 1`(2,3)+ 1`(3,0)+ 1`(3,1)+ 1`(3,2)+ 1`(3,3)
  SN'Saco2 1          1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)+ 1`(2,0)+
1`(2,1)+
1`(2,2)+ 1`(2,3)+ 1`(3,0)+ 1`(3,1)+ 1`(3,2)+ 1`(3,3)
  SN'Saco22 1         empty
  SN'Saco23 1         empty
  SN'Saco3 1          1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)
  SN'Saco4 1          1`(ACK,1)+ 1`(ACK,3)+ 1`(NAK,0)+ 1`(NAK,2)
  SN'Saco5 1          1`(ACK,0)+ 1`(ACK,1)+ 1`(NAK,2)+ 1`(NAK,3)
  SN'Setup_SNC 1      1`((1,1,[2,3],2),0)+ 1`((1,1,[2,3],2),1)+
1`((1,1,[2,3],2),2)+ 1`((1,1,[2,3],2),3)
  SN'atmSubNetwork 1  1`(1,{ulabel = "",charInf = VC,cLL = [],cSNL = [],
sLTPL = []})+ 1`(2,{ulabel = "",charInf = VC,
cLL = [],cSNL = [],sLTPL = []})
  SN'wait1 1          1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)
  SN'wait2 1          empty
  SN'wait3 1          1`((1,[2,3]),0)+ 1`((1,[2,3]),1)+ 1`((1,[2,3]),2)+
1`((1,[2,3]),3)

  Best Lower Multi-set Bounds
  SN'Create2 1        empty
  SN'Delete2 1        empty
  SN'Feed 1           1`[1,2]
  SN'Final1 1         1`(1,0)+ 1`(2,0)
  SN'Final2 1         empty
  SN'Final3 1         empty
  SN'ID_gen2 1        1`(3,0)
  SN'P1 1             empty
  SN'P11 1            empty
  SN'P2 1             empty
  SN'P3 1             empty
  SN'P4 1             empty
  SN'P6 1             1`([],0)
  SN'P8 1             empty
  SN'SNCCreate1LP 1   empty
  SN'SNCCreate1UP 1   empty
  SN'SNCSetupMPLP 1   empty
  SN'SNCSetupMPUP 1   empty
  SN'SNCSetupPPLP 1   empty
  SN'SNCSetupPPUP 1   empty
  SN'SNTPCreate1LP 1  empty
  SN'SNTPCreate1UP 1  empty
  SN'SNTPCreateLP 1   empty
  SN'SNTPCreateUP 1   empty
  SN'SN_notify1 1     1`(Created_SN,1,0)+ 1`(Created_SN,2,0)
  SN'SN_notify2 1     1`(Created_SN,1,0)+ 1`(Created_SN,2,0)
  SN'Saco1 1          1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)+ 1`(2,0)+
1`(2,1)+
1`(2,2)+ 1`(2,3)+ 1`(3,0)+ 1`(3,1)+ 1`(3,2)+ 1`(3,3)
  SN'Saco2 1          1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)+ 1`(2,0)+
1`(2,1)+
```

```
1`(2,2)+ 1`(2,3)+ 1`(3,0)+ 1`(3,1)+ 1`(3,2)+ 1`(3,3)
  SN'Saco22 1          empty
  SN'Saco23 1          empty
  SN'Saco3 1           empty
  SN'Saco4 1           empty
  SN'Saco5 1           empty
  SN'Setup_SNC 1       empty
  SN'atmSubNetwork 1   1`(1,{ulabel = "",charInf = VC,cLL = [],cSNL =
[],sLTPL = []})+ 1`(2,{ulabel = "",charInf = VC,cLL = [],cSNL = [],sLTPL
= []})
  SN'wait1 1           empty
  SN'wait2 1           empty
  SN'wait3 1           empty


 Liveness Properties
--------------------------------------------------------------------------------
  Dead Markings:  [20736]
  Dead Transitions Instances:

  SN'T1 1
  SN'T12 1
  SN'T13 1
  SN'T14 1
  SN'T15 1
  SN'T17 1
  SN'T18 1
  SN'T22 1
  SN'T3 1
  SN'T4 1
  SN'T6 1
  SN'T7 1
```

Relatório da simulação segundo cenário:

- método: Setup_SNC da rede SN
- ficha: um snaTP e uma lista vazia em snTPzL, comprimento da lista igual a 2.

```
 Statistics
--------------------------------------------------------------------------------
  Occurrence Graph
    Nodes:  28561
    Arcs:   149396
    Secs:   1130
    Status: Full


 Boundedness Properties
```

Best Integers Bounds	Upper	Lower
SN'Create2 1	0	0
SN'Delete2 1	0	0
SN'Feed 1	1	1
SN'Final1 1	2	2
SN'Final2 1	0	0
SN'Final3 1	0	0
SN'ID_gen2 1	1	1
SN'P1 1	4	0
SN'P11 1	4	0
SN'P2 1	4	0
SN'P3 1	4	0
SN'P4 1	4	0
SN'P6 1	1	1
SN'P8 1	0	0
SN'SNCCreate1LP 1	0	0
SN'SNCCreate1UP 1	0	0
SN'SNCSetupMPLP 1	0	0
SN'SNCSetupMPUP 1	0	0
SN'SNCSetupPPLP 1	0	0
SN'SNCSetupPPUP 1	0	0
SN'SNTPCreate1LP 1	0	0
SN'SNTPCreate1UP 1	0	0
SN'SNTPCreateLP 1	0	0
SN'SNTPCreateUP 1	4	0
SN'SN_notify1 1	2	2
SN'SN_notify2 1	2	2
SN'Saco1 1	12	12
SN'Saco2 1	12	8
SN'Saco22 1	4	0
SN'Saco23 1	0	0
SN'Saco3 1	4	4
SN'Saco4 1	4	4
SN'Saco5 1	4	4
SN'Setup_SNC 1	4	0
SN'atmSubNetwork 1	2	2
SN'wait1 1	4	0
SN'wait2 1	0	0
SN'wait3 1	0	0

Best Upper Multi-set Bounds	
SN'Create2 1	empty
SN'Delete2 1	empty
SN'Feed 1	1`[1,2]
SN'Final1 1	1`(1,0)+ 1`(2,0)
SN'Final2 1	empty
SN'Final3 1	empty
SN'ID_gen2 1	1`(3,0)
SN'P1 1	1`((1,1),0)+ 1`((1,1),1)+ 1`((1,1),2)+ 1`((1,1),3)
SN'P11 1	1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)
SN'P2 1	1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)
SN'P3 1	1`((1,[],1),0)+ 1`((1,[],1),1)+ 1`((1,[],1),2)+

```
1‘((1,[],1),3)
  SN’P4 1              1‘((1,0),0)+ 1‘((1,0),1)+ 1‘((1,0),2)+
1‘((1,0),3)+
1‘((1,1),0)+ 1‘((1,1),1)+ 1‘((1,1),2)+ 1‘((1,1),3)
  SN’P6 1              1‘([],0)
  SN’P8 1              empty
  SN’SNCCreate1LP 1    empty
  SN’SNCCreate1UP 1    empty
  SN’SNCSetupMPLP 1    empty
  SN’SNCSetupMPUP 1    empty
  SN’SNCSetupPPLP 1    empty
  SN’SNCSetupPPUP 1    empty
  SN’SNTPCreate1LP 1   empty
  SN’SNTPCreate1UP 1   empty
  SN’SNTPCreateLP 1    empty
  SN’SNTPCreateUP 1    1‘(1,0)+ 1‘(1,1)+ 1‘(1,2)+ 1‘(1,3)
  SN’SN_notify1 1      1‘(Created_SN,1,0)+ 1‘(Created_SN,2,0)
  SN’SN_notify2 1      1‘(Created_SN,1,0)+ 1‘(Created_SN,2,0)
  SN’Saco1 1           1‘(1,0)+ 1‘(1,1)+ 1‘(1,2)+ 1‘(1,3)+ 1‘(2,0)+
1‘(2,1)+
1‘(2,2)+ 1‘(2,3)+ 1‘(3,0)+ 1‘(3,1)+ 1‘(3,2)+ 1‘(3,3)
  SN’Saco2 1           1‘(1,0)+ 1‘(1,1)+ 1‘(1,2)+ 1‘(1,3)+ 1‘(2,0)+
1‘(2,1)+
1‘(2,2)+ 1‘(2,3)+ 1‘(3,0)+ 1‘(3,1)+ 1‘(3,2)+ 1‘(3,3)
  SN’Saco22 1          1‘(1,0)+ 1‘(1,1)+ 1‘(1,2)+ 1‘(1,3)+ 1‘(2,0)+
1‘(2,1)+
1‘(2,2)+ 1‘(2,3)+ 1‘(3,0)+ 1‘(3,1)+ 1‘(3,2)+ 1‘(3,3)
  SN’Saco23 1          empty
  SN’Saco3 1           1‘(1,0)+ 1‘(1,1)+ 1‘(1,2)+ 1‘(1,3)
  SN’Saco4 1           1‘(ACK,1)+ 1‘(ACK,3)+ 1‘(NAK,0)+ 1‘(NAK,2)
  SN’Saco5 1           1‘(ACK,0)+ 1‘(ACK,1)+ 1‘(NAK,2)+ 1‘(NAK,3)
  SN’Setup_SNC 1       1‘((1,1,[],1),0)+ 1‘((1,1,[],1),1)+
1‘((1,1,[],1),2)+
1‘((1,1,[],1),3)
  SN’atmSubNetwork 1   1‘(1,{ulabel = "",charInf = VC,cLL = [],cSNL = [],
sLTPL = []})+ 1‘(2,{ulabel = "",charInf = VC,cLL = [],cSNL = [],
sLTPL = []})
  SN’wait1 1           1‘(1,0)+ 1‘(1,1)+ 1‘(1,2)+ 1‘(1,3)
  SN’wait2 1           empty
  SN’wait3 1           empty

  Best Lower Multi-set Bounds
  SN’Create2 1         empty
  SN’Delete2 1         empty
  SN’Feed 1            1‘[1,2]
  SN’Final1 1          1‘(1,0)+ 1‘(2,0)
  SN’Final2 1          empty
  SN’Final3 1          empty
  SN’ID_gen2 1         1‘(3,0)
  SN’P1 1              empty
  SN’P11 1             empty
  SN’P2 1              empty
  SN’P3 1              empty
```

```
  SN'P4 1              empty
  SN'P6 1              1`([],0)
  SN'P8 1              empty
  SN'SNCCreate1LP 1    empty
  SN'SNCCreate1UP 1    empty
  SN'SNCSetupMPLP 1    empty
  SN'SNCSetupMPUP 1    empty
  SN'SNCSetupPPLP 1    empty
  SN'SNCSetupPPUP 1    empty
  SN'SNTPCreate1LP 1   empty
  SN'SNTPCreate1UP 1   empty
  SN'SNTPCreateLP 1    empty
  SN'SNTPCreateUP 1    empty
  SN'SN_notify1 1      1`(Created_SN,1,0)+ 1`(Created_SN,2,0)
  SN'SN_notify2 1      1`(Created_SN,1,0)+ 1`(Created_SN,2,0)
  SN'Saco1 1           1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)+ 1`(2,0)+
1`(2,1)+
1`(2,2)+ 1`(2,3)+ 1`(3,0)+ 1`(3,1)+ 1`(3,2)+ 1`(3,3)
  SN'Saco2 1           empty
  SN'Saco22 1          empty
  SN'Saco23 1          empty
  SN'Saco3 1           1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)
  SN'Saco4 1           1`(ACK,1)+ 1`(ACK,3)+ 1`(NAK,0)+ 1`(NAK,2)
  SN'Saco5 1           1`(ACK,0)+ 1`(ACK,1)+ 1`(NAK,2)+ 1`(NAK,3)
  SN'Setup_SNC 1       empty
  SN'atmSubNetwork 1   1`(1,{ulabel = "",charInf = VC,cLL = [],cSNL = [],
sLTPL = []})+ 1`(2,{ulabel = "",charInf = VC,cLL = [],
cSNL = [],sLTPL = []})
  SN'wait1 1           empty
  SN'wait2 1           empty
  SN'wait3 1           empty


 Liveness Properties
------------------------------------------------------------------------
  Dead Markings:  81 [28561,28560,28559,28558,28557,...]
  Dead Transitions Instances:

  SN'T1 1
  SN'T10 1
  SN'T11 1
  SN'T12 1
  SN'T13 1
  SN'T14 1
  SN'T15 1
  SN'T17 1
  SN'T19 1
  SN'T20 1
  SN'T21 1
  SN'T22 1
  SN'T5 1
  SN'T6 1
  SN'T7 1
```

SN'T9 1

Relatório da simulação terceiro cenário:

- método: Create da rede SNTP
- ficha: 2`(snid,o1)

```
Statistics
------------------------------------------------------------------------

  Occurrence Graph
    Nodes:  3
    Arcs:   2
    Secs:   0
    Status: Full


Boundedness Properties
------------------------------------------------------------------------

  Best Integers Bounds     Upper      Lower
  SNTP'Create 1            2          0
  SNTP'Delete 1            0          0
  SNTP'Feedsntp1 1         1          1
  SNTP'Feedsntp2 1         1          1
  SNTP'GP4 1               0          0
  SNTP'GP5 1               0          0
  SNTP'GP6 1               0          0
  SNTP'GP7 1               2          0
  SNTP'ID_gen1 1           1          1
  SNTP'P16 1               0          0
  SNTP'P17 1               0          0
  SNTP'P18 1               0          0
  SNTP'P19 1               0          0
  SNTP'P20 1               0          0
  SNTP'SNTP_notify1 1      2          0
  SNTP'SNTP_notify2 1      0          0
  SNTP'SNTP_notify3 1      0          0
  SNTP'SetupMP 1           0          0
  SNTP'SetupPP 1           0          0
  SNTP'atmSubNetworkTP 1   2          0


Liveness Properties
------------------------------------------------------------------------

  Dead Markings:  [3]
  Dead Transitions Instances:

  SNTP'T16 1
  SNTP'T17 1
```

```
SNTP'T18 1
SNTP'T19 1
SNTP'T20 1
SNTP'T21 1
SNTP'T22 1
SNTP'T24 1
```

Relatório da simulação quarto cenário:

- método: Delete do SNTP
- ficha: 1‘(sntpID,o1)

```
Statistics
------------------------------------------------------------------------

  Occurrence Graph
    Nodes:  36
    Arcs:   84
    Secs:   1
    Status: Full


Boundedness Properties
------------------------------------------------------------------------

  Best Integers Bounds     Upper      Lower
  SNTP'Create 1            0          0
  SNTP'Delete 1            1          0
  SNTP'Feedsntp1 1         1          1
  SNTP'Feedsntp2 1         1          1
  SNTP'GP4 1               1          0
  SNTP'GP5 1               0          0
  SNTP'GP6 1               5          0
  SNTP'GP7 1               2          2
  SNTP'ID_gen1 1           1          1
  SNTP'P16 1               0          0
  SNTP'P17 1               5          0
  SNTP'P18 1               5          3
  SNTP'P19 1               0          0
  SNTP'P20 1               0          0
  SNTP'SNTP_notify1 1      3          2
  SNTP'SNTP_notify2 1      0          0
  SNTP'SNTP_notify3 1      2          0
  SNTP'SetupMP 1           0          0
  SNTP'SetupPP 1           0          0
  SNTP'atmSubNetworkTP 1   2          1

  Best Upper Multi-set Bounds
  SNTP'Create 1        empty
  SNTP'Delete 1        1‘(1,0)
  SNTP'Feedsntp1 1     1‘[1,2]
  SNTP'Feedsntp2 1     1‘[1,2]
```

```
  SNTP'GP4 1           1'(ACK,0)
  SNTP'GP5 1           empty
  SNTP'GP6 1           2'(ACK,0)+ 3'(NAK,0)
  SNTP'GP7 1           1'(1,0)+ 1'(2,0)
  SNTP'ID_gen1 1       1'(3,0)
  SNTP'P16 1           empty
  SNTP'P17 1           2'(2,0)+ 2'(3,0)+ 1'(4,0)
  SNTP'P18 1           5'(1,0)
  SNTP'P19 1           empty
  SNTP'P20 1           empty
  SNTP'SNTP_notify1 1 1'(Created_SNTP,1,0)+ 1'(Created_SNTP,2,0)+
1'(Deleted_SNTP,1,0)
  SNTP'SNTP_notify2 1 empty
  SNTP'SNTP_notify3 1 2'(Modified_SNTP,2,0)
  SNTP'SetupMP 1       empty
  SNTP'SetupPP 1       empty
  SNTP'atmSubNetworkTP 11'(1,{relLC = 0,relLTP = 0,relSNC = 0,refCTP =
0,
userLab = "",charInf = VC,sn = 1})+ 1'(2,{relLC = 0,relLTP = 0,
relSNC = 0,refCTP = 0,userLab = "",charInf = VC,sn = 1})+
1'(2,{relLC = 0,relLTP = 0,relSNC = 1,refCTP = 0,userLab = "",
charInf = VC,sn = 0})

  Best Lower Multi-set Bounds
  SNTP'Create 1        empty
  SNTP'Delete 1        empty
  SNTP'Feedsntp1 1     1'[1,2]
  SNTP'Feedsntp2 1     1'[1,2]
  SNTP'GP4 1           empty
  SNTP'GP5 1           empty
  SNTP'GP6 1           empty
  SNTP'GP7 1           1'(1,0)+ 1'(2,0)
  SNTP'ID_gen1 1       1'(3,0)
  SNTP'P16 1           empty
  SNTP'P17 1           empty
  SNTP'P18 1           3'(1,0)
  SNTP'P19 1           empty
  SNTP'P20 1           empty
  SNTP'SNTP_notify1 1 1'(Created_SNTP,1,0)+ 1'(Created_SNTP,2,0)
  SNTP'SNTP_notify2 1 empty
  SNTP'SNTP_notify3 1 empty
  SNTP'SetupMP 1       empty
  SNTP'SetupPP 1       empty
  SNTP'atmSubNetworkTP 1empty


 Liveness Properties
-----------------------------------------------------------------------------

  Dead Markings:  [36]
  Dead Transitions Instances:

  SNTP'T16 1
  SNTP'T17 1
```

```
SNTP'T20 1
SNTP'T21 1
SNTP'T22 1
SNTP'T23 1
```

Relatório da simulação quinto cenário:

- método: Mudança da informação contida nos TPs, através dos métodos Setup_PP e Setup_MP.
- ficha: um snaTP e uma lista com dois tps em snTPzL

```
Statistics
------------------------------------------------------------------------

  Occurrence Graph
    Nodes:  12
    Arcs:   20
    Secs:   0
    Status: Full


Boundedness Properties
------------------------------------------------------------------------

  Best Integers Bounds     Upper      Lower
  SNTP'Create 1            0          0
  SNTP'Delete 1            0          0
  SNTP'Feedsntp1 1         1          1
  SNTP'Feedsntp2 1         1          1
  SNTP'GP4 1               0          0
  SNTP'GP5 1               1          0
  SNTP'GP6 1               2          0
  SNTP'GP7 1               3          3
  SNTP'ID_gen1 1           1          1
  SNTP'P16 1               0          0
  SNTP'P17 1               2          0
  SNTP'P18 1               2          0
  SNTP'P19 1               1          0
  SNTP'P20 1               1          0
  SNTP'SNTP_notify1 1      3          3
  SNTP'SNTP_notify2 1      1          0
  SNTP'SNTP_notify3 1      2          0
  SNTP'SetupMP 1           0          0
  SNTP'SetupPP 1           1          0
  SNTP'atmSubNetworkTP 1   3          3

  Best Upper Multi-set Bounds
  SNTP'Create 1        empty
  SNTP'Delete 1        empty
  SNTP'Feedsntp1 1     1`[1,2,3]
```

```
  SNTP'Feedsntp2 1     1`[1,2,3]
  SNTP'GP4 1           empty
  SNTP'GP5 1           1`(ACK,0)
  SNTP'GP6 1           2`(ACK,0)
  SNTP'GP7 1           1`(1,0)+ 1`(2,0)+ 1`(3,0)
  SNTP'ID_gen1 1       1`(4,0)
  SNTP'P16 1           empty
  SNTP'P17 1           1`(2,0)+ 1`(3,0)
  SNTP'P18 1           2`(1,0)
  SNTP'P19 1           1`(1,0)
  SNTP'P20 1           1`(1,0)
  SNTP'SNTP_notify1 1 1`(Created_SNTP,1,0)+ 1`(Created_SNTP,2,0)+
1`(Created_SNTP,3,0)
  SNTP'SNTP_notify2 1 1`(Modified_SNTP,1,0)
  SNTP'SNTP_notify3 1 1`(Modified_SNTP,2,0)+ 1`(Modified_SNTP,3,0)
  SNTP'SetupMP 1       empty
  SNTP'SetupPP 1       1`((1,1),0)
  SNTP'atmSubNetworkTP 11`(1,{relLC = 0,relLTP = 0,relSNC = 0,refCTP =
0,
userLab = "",charInf = VC,sn = 1})+ 1`(1,{relLC = 0,relLTP = 0,
relSNC = 1,refCTP = 0,userLab = "",charInf = VC,sn = 0})+
1`(2,{relLC = 0,relLTP = 0,relSNC = 0,refCTP = 0,userLab = "",
charInf = VC,sn = 1})+ 1`(2,{relLC = 0,relLTP = 0,relSNC = 1,
refCTP = 0,userLab = "",charInf = VC,sn = 0})+ 1`(3,{relLC = 0,
relLTP = 0,relSNC = 0,refCTP = 0,userLab = "",charInf = VC,sn =
1})+
1`(3,{relLC = 0,relLTP = 0,relSNC = 1,refCTP = 0,userLab = "",
charInf = VC,sn = 0})

  Best Lower Multi-set Bounds
  SNTP'Create 1        empty
  SNTP'Delete 1        empty
  SNTP'Feedsntp1 1     1`[1,2,3]
  SNTP'Feedsntp2 1     1`[1,2,3]
  SNTP'GP4 1           empty
  SNTP'GP5 1           empty
  SNTP'GP6 1           empty
  SNTP'GP7 1           1`(1,0)+ 1`(2,0)+ 1`(3,0)
  SNTP'ID_gen1 1       1`(4,0)
  SNTP'P16 1           empty
  SNTP'P17 1           empty
  SNTP'P18 1           empty
  SNTP'P19 1           empty
  SNTP'P20 1           empty
  SNTP'SNTP_notify1 1 1`(Created_SNTP,1,0)+ 1`(Created_SNTP,2,0)+
1`(Created_SNTP,3,0)
  SNTP'SNTP_notify2 1 empty
  SNTP'SNTP_notify3 1 empty
  SNTP'SetupMP 1       empty
  SNTP'SetupPP 1       empty
  SNTP'atmSubNetworkTP 1empty
```

```
Liveness Properties
-------------------------------------------------------------------------
  Dead Markings:  [12]
  Dead Transitions Instances:

  SNTP'T16 1
  SNTP'T17 1
  SNTP'T18 1
  SNTP'T22 1
  SNTP'T23 1
  SNTP'T24 1
```

Relatório da simulação sexto cenário:

- método: Mudança da informação contida nos TPs, através dos métodos Setup_PP e Setup_MP.
- ficha: um snaTP e uma lista com dois tps em snTPzL, sem que um dos tps pertença ao atributo atmSNTP.

```
Statistics
-------------------------------------------------------------------------
  Occurrence Graph
    Nodes:  114
    Arcs:   301
    Secs:   1
    Status: Full
```

```
Boundedness Properties
-------------------------------------------------------------------------
  Best Integers Bounds     Upper      Lower
  SNTP'Create 1            0          0
  SNTP'Delete 1            0          0
  SNTP'Feedsntp1 1         1          1
  SNTP'Feedsntp2 1         1          1
  SNTP'GP4 1               0          0
  SNTP'GP5 1               1          0
  SNTP'GP6 1               4          0
  SNTP'GP7 1               3          3
  SNTP'ID_gen1 1           1          1
  SNTP'P16 1               0          0
  SNTP'P17 1               4          0
  SNTP'P18 1               4          1
  SNTP'P19 1               1          0
  SNTP'P20 1               1          0
  SNTP'SNTP_notify1 1      3          3
  SNTP'SNTP_notify2 1      1          0
  SNTP'SNTP_notify3 1      3          0
  SNTP'SetupMP 1           0          0
```

```
  SNTP'SetupPP 1          1          0
  SNTP'atmSubNetworkTP 1  3          3

  Best Upper Multi-set Bounds
  SNTP'Create 1        empty
  SNTP'Delete 1        empty
  SNTP'Feedsntp1 1     1`[1,2,3]
  SNTP'Feedsntp2 1     1`[1,2,3]
  SNTP'GP4 1           empty
  SNTP'GP5 1           1`(ACK,0)
  SNTP'GP6 1           3`(ACK,0)+ 1`(NAK,0)
  SNTP'GP7 1           1`(1,0)+ 1`(2,0)+ 1`(3,0)
  SNTP'ID_gen1 1       1`(4,0)
  SNTP'P16 1           empty
  SNTP'P17 1           1`(2,0)+ 2`(3,0)+ 1`(4,0)
  SNTP'P18 1           2`(1,0)+ 2`(2,0)
  SNTP'P19 1           1`(1,0)
  SNTP'P20 1           1`(1,0)
  SNTP'SNTP_notify1 1 1`(Created_SNTP,1,0)+ 1`(Created_SNTP,2,0)+
1`(Created_SNTP,3,0)
  SNTP'SNTP_notify2 1 1`(Modified_SNTP,1,0)
  SNTP'SNTP_notify3 1 1`(Modified_SNTP,2,0)+ 2`(Modified_SNTP,3,0)
  SNTP'SetupMP 1       empty
  SNTP'SetupPP 1       1`((1,1),0)
  SNTP'atmSubNetworkTP 11`(1,{relLC = 0,relLTP = 0,relSNC = 0,refCTP =
0,
userLab = "",charInf = VC,sn = 1})+ 1`(1,{relLC = 0,relLTP = 0,
relSNC = 1,refCTP = 0,userLab = "",charInf = VC,sn = 0})+
1`(2,{relLC = 0,relLTP = 0,relSNC = 0,refCTP = 0,userLab = "",
charInf = VC,sn = 1})+ 1`(2,{relLC = 0,relLTP = 0,relSNC = 1,
refCTP = 0,userLab = "",charInf = VC,sn = 0})+ 1`(2,{relLC = 0,
relLTP = 0,relSNC = 2,refCTP = 0,userLab = "",charInf = VC,sn =
0})+
1`(3,{relLC = 0,relLTP = 0,relSNC = 0,refCTP = 0,userLab = "",
charInf = VC,sn = 1})+ 1`(3,{relLC = 0,relLTP = 0,relSNC = 1,
refCTP = 0,userLab = "",charInf = VC,sn = 0})+ 1`(3,{relLC = 0,
relLTP = 0,relSNC = 2,refCTP = 0,userLab = "",charInf = VC,sn =
0})

  Best Lower Multi-set Bounds
  SNTP'Create 1        empty
  SNTP'Delete 1        empty
  SNTP'Feedsntp1 1     1`[1,2,3]
  SNTP'Feedsntp2 1     1`[1,2,3]
  SNTP'GP4 1           empty
  SNTP'GP5 1           empty
  SNTP'GP6 1           empty
  SNTP'GP7 1           1`(1,0)+ 1`(2,0)+ 1`(3,0)
  SNTP'ID_gen1 1       1`(4,0)
  SNTP'P16 1           empty
  SNTP'P17 1           empty
  SNTP'P18 1           empty
  SNTP'P19 1           empty
```

```
  SNTP'P20 1          empty
  SNTP'SNTP_notify1 1 1`(Created_SNTP,1,0)+ 1`(Created_SNTP,2,0)+
1`(Created_SNTP,3,0)
  SNTP'SNTP_notify2 1 empty
  SNTP'SNTP_notify3 1 empty
  SNTP'SetupMP 1      empty
  SNTP'SetupPP 1      empty
  SNTP'atmSubNetworkTP 1empty


 Liveness Properties
--------------------------------------------------------------------------------
  Dead Markings:  6 [114,113,112,111,110,...]
  Dead Transitions Instances:

  SNTP'T16 1
  SNTP'T17 1
  SNTP'T22 1
  SNTP'T23 1
  SNTP'T24 1
```

Relatório da simulação do sétimo cenário:

- método: Setup_SNC
- ficha: ficha com snatp=0 e sntpzL=[]

```
 Statistics
--------------------------------------------------------------------------------
  Occurrence Graph
    Nodes:  89932
    Arcs:   438259
    Secs:   30001
    Status: Partial


 Boundedness Properties
--------------------------------------------------------------------------------
  Best Integers Bounds    Upper      Lower
  SN'Create2 1            0          0
  SN'Delete2 1            0          0
  SN'Feed 1               1          1
  SN'Final1 1             2          2
  SN'Final2 1             0          0
  SN'Final3 1             0          0
  SN'ID_gen2 1            1          1
  SN'P1 1                 4          0
  SN'P11 1                4          0
```

```
SN'P2 1                 1          0
SN'P3 1                 4          0
SN'P4 1                 4          0
SN'P6 1                 1          0
SN'P8 1                 1          0
SN'SNCCreate1LP 1       1          0
SN'SNCCreate1UP 1       1          0
SN'SNCSetupMPLP 1       1          0
SN'SNCSetupMPUP 1       1          0
SN'SNCSetupPPLP 1       1          0
SN'SNCSetupPPUP 1       1          0
SN'SNTPCreate1LP 1      2          0
SN'SNTPCreate1UP 1      4          0
SN'SNTPCreateLP 1       2          0
SN'SNTPCreateUP 1       4          0
SN'SN_notify1 1         2          2
SN'SN_notify2 1         2          2
SN'Saco1 1              8          4
SN'Saco2 1              8          5
SN'Saco22 1             3          0
SN'Saco23 1             4          0
SN'Saco3 1              4          3
SN'Saco4 1              4          3
SN'Saco5 1              4          3
SN'Setup_SNC 1          4          0
SN'atmSubNetwork 1      2          2
SN'wait1 1              4          0
SN'wait2 1              4          0
SN'wait3 1              1          0

Best Upper Multi-set Bounds
SN'Create2 1        empty
SN'Delete2 1        empty
SN'Feed 1           1`[1,2]
SN'Final1 1         1`(1,0)+ 1`(2,0)
SN'Final2 1         empty
SN'Final3 1         empty
SN'ID_gen2 1        1`(3,0)
SN'P1 1             1`((0,1),0)+ 1`((0,1),1)+ 1`((0,1),2)+ 1`((0,1),3)
SN'P11 1            1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)
SN'P2 1             1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)+ 1`(2,0)+
1`(2,1)+ 1`(2,2)+ 1`(2,3)
SN'P3 1             1`((1,[],1),0)+ 1`((1,[],1),1)+ 1`((1,[],1),2)+
1`((1,[],1),3)
SN'P4 1             1`((1,0),0)+ 1`((1,0),1)+ 1`((1,0),2)+
1`((1,0),3)+ 1`((1,1),0)+ 1`((1,1),1)+ 1`((1,1),2)+ 1`((1,1),3)
SN'P6 1             1`([],0)+ 1`([1],0)+ 1`([2],0)
SN'P8 1             1`([1],0)+ 1`([2],0)
SN'SNCCreate1LP 1   1`(1,0)
SN'SNCCreate1UP 1   1`((1,[2]),0)+ 1`((2,[1]),0)
SN'SNCSetupMPLP 1   1`(ACK,0)
SN'SNCSetupMPUP 1   1`(([1],1),0)+ 1`(([2],1),0)
SN'SNCSetupPPLP 1   1`(NAK,0)
```

```
  SN'SNCSetupPPUP 1    1`((1,1),0)+ 1`((2,1),0)
  SN'SNTPCreate1LP 1   1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)+ 1`(2,0)+
1`(2,1)+ 1`(2,2)+ 1`(2,3)
  SN'SNTPCreate1UP 1   1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)
  SN'SNTPCreateLP 1    1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)+ 1`(2,0)+
1`(2,1)+ 1`(2,2)+ 1`(2,3)
  SN'SNTPCreateUP 1    1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)
  SN'SN_notify1 1      1`(Created_SN,1,0)+ 1`(Created_SN,2,0)
  SN'SN_notify2 1      1`(Created_SN,1,0)+ 1`(Created_SN,2,0)
  SN'Saco1 1           1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)+ 1`(2,0)+
1`(2,1)+ 1`(2,2)+ 1`(2,3)
  SN'Saco2 1           1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)+ 1`(2,0)+
1`(2,1)+ 1`(2,2)+ 1`(2,3)
  SN'Saco22 1          1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)+ 1`(2,0)+
1`(2,1)+ 1`(2,2)+ 1`(2,3)
  SN'Saco23 1          1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)+ 1`(2,0)+
1`(2,1)+ 1`(2,2)+ 1`(2,3)
  SN'Saco3 1           1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)
  SN'Saco4 1           1`(ACK,1)+ 1`(ACK,3)+ 1`(NAK,0)+ 1`(NAK,2)
  SN'Saco5 1           1`(ACK,0)+ 1`(ACK,1)+ 1`(NAK,2)+ 1`(NAK,3)
  SN'Setup_SNC 1       1`((1,0,[],1),0)+ 1`((1,0,[],1),1)+
1`((1,0,[],1),2)+ 1`((1,0,[],1),3)
  SN'atmSubNetwork 1   1`(1,{ulabel = "",charInf = VC,cLL = [],cSNL =
[],sLTPL = []})+ 1`(2,{ulabel = "",charInf = VC,cLL = [],cSNL = [],sLTPL
= []})
  SN'wait1 1           1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)
  SN'wait2 1           1`(1,0)+ 1`(1,1)+ 1`(1,2)+ 1`(1,3)
  SN'wait3 1           1`((1,[2]),0)+ 1`((2,[1]),0)

  Best Lower Multi-set Bounds
  SN'Create2 1         empty
  SN'Delete2 1         empty
  SN'Feed 1            1`[1,2]
  SN'Final1 1          1`(1,0)+ 1`(2,0)
  SN'Final2 1          empty
  SN'Final3 1          empty
  SN'ID_gen2 1         1`(3,0)
  SN'P1 1              empty
  SN'P11 1             empty
  SN'P2 1              empty
  SN'P3 1              empty
  SN'P4 1              empty
  SN'P6 1              empty
  SN'P8 1              empty
  SN'SNCCreate1LP 1    empty
  SN'SNCCreate1UP 1    empty
  SN'SNCSetupMPLP 1    empty
  SN'SNCSetupMPUP 1    empty
  SN'SNCSetupPPLP 1    empty
  SN'SNCSetupPPUP 1    empty
  SN'SNTPCreate1LP 1   empty
  SN'SNTPCreate1UP 1   empty
  SN'SNTPCreateLP 1    empty
```

entre si ou através de *funções de processamento de transporte.* Os pontos a que estão delimitados são os *pontos de referência* da *rede de transporte.*

topological entity - entidade topológica - Um componente arquitetural que descreve a *rede de transporte* em termos da relações topológicas entre os conjuntos de *pontos de referência.* Uma descrição topológica em termos destes componentes descrevem as possibilidades de roteamento da rede e, desse modo, sua habilidade para dar suporte a *entidades de transporte.*

reference point - ponto de referência - Um componente arquitetural do sistema que descreve a ligação entre entradas e saídas das *funções de processamento de transporte* e *entidades de transporte.* É caracterizado pela informação que passa através do mesmo.

transport network layer (or layer network) - camada da rede de transporte - Um componente topológico cuja a única finalidade é a geração e tranferência da *informação característica* particular.

```
  SN'SNTPCreateUP 1    empty
  SN'SN_notify1 1      1`(Created_SN,1,0)+ 1`(Created_SN,2,0)
  SN'SN_notify2 1      1`(Created_SN,1,0)+ 1`(Created_SN,2,0)
  SN'Saco1 1           empty
  SN'Saco2 1           empty
  SN'Saco22 1          empty
  SN'Saco23 1          empty
  SN'Saco3 1           1`(1,1)+ 1`(1,2)+ 1`(1,3)
  SN'Saco4 1           1`(ACK,1)+ 1`(ACK,3)+ 1`(NAK,2)
  SN'Saco5 1           1`(ACK,1)+ 1`(NAK,2)+ 1`(NAK,3)
  SN'Setup_SNC 1       empty
  SN'atmSubNetwork 1   1`(1,{ulabel = "",charInf = VC,cLL = [],cSNL =
[],sLTPL = []})+ 1`(2,{ulabel = "",charInf = VC,cLL = [],cSNL = [],sLTPL
= []})
  SN'wait1 1           empty
  SN'wait2 1           empty
  SN'wait3 1           empty


 Liveness Properties
--------------------------------------------------------------------------

  Dead Markings:  28051 [89932,89931,89930,89929,89928,...]
```

Apêndice C - Definição de Conceitos e Entidades

Na norma G.803 [IT93a] encontra-se a definição de conceitos e entidades para a hierarquia digital síncrona, também são aplicáveis a redes ATM. A seguir apresentam-se os conceitos e entidades utilizados nesta dissertação.

network - rede - Todas as entidades que juntas provêem comunicação de serviços, tais como: equipamento, planta, facilidades.

transport - transporte - O processo funcional de transferência de informação entre pontos em diferentes localizações.

transmission - transmissão - O processo físico de propagação de sinais de informação através de um meio físico.

transport network - rede de transporte - Os recursos funcionais da *rede* que leva informação do usuário entre diferentes localidades.

characteristic information - informação característica - Um sinal de taxa e formato característico dentro e entre sub-redes e apresentado a uma função de adaptação para transporte pelo servidor da camada de rede.

architectural component - componente arquitetural [2] - Qualquer item necessário para descrever de maneira generalizada a funcionalidade da rede de tranporte independente de implementação tecnológica.

transport processing function - função de processamento de transporte - Um componente arquitetural definido pelo processamento da informação que é executado entre suas saídas e entradas. Contendo um ou mais entradas e uma ou mais saídas, que podem estar associadas com entradas ou saídas de outras funções ou entidades. Tais associações são denominadas de relações de ligação (*binding*).

transport entity - entidade de transporte - Um componente arquitetural que transfere informação de maneira transparente entre diferentes localidades. A *informação* é transferida para a *entidade de transporte* na sua entrada e sai da *entidade de transporte* na sua saída. *Entidades de transporte* podem estar delimitadas

[2] arquitetural aqui significa fazer parte da estrutura do sistema, e não no sentido descrito no Aurélio, de gênero arquitetônico

entre si ou através de *funções de processamento de transporte*. Os pontos a que estão delimitados são os *pontos de referência* da *rede de transporte*.

topological entity - entidade topológica - Um componente arquitetural que descreve a *rede de transporte* em termos da relações topológicas entre os conjuntos de *pontos de referência*. Uma descrição topológica em termos destes componentes descrevem as possibilidades de roteamento da rede e, desse modo, sua habilidade para dar suporte a *entidades de transporte*.

reference point - ponto de referência - Um componente arquitetural do sistema que descreve a ligação entre entradas e saídas das *funções de processamento de transporte* e *entidades de transporte*. É caracterizado pela informação que passa através do mesmo.

transport network layer (or layer network) - camada da rede de transporte - Um componente topológico cuja a única finalidade é a geração e tranferência da *informação característica* particular.