Universidade Federal de Santa Catarina
Curso de Pós-Graduação em Ciência da Computação

# Utilizando LOTOS na concepção formal de uma aplicação para Gerência de Redes: Especificação e Verificação

por
Braulio Adriano de Mello

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação.
Orientador: Prof. Murilo Silva de Camargo
Co-Orientadora: Profa. Elizabeth S. Specialski

Florianópolis, fevereiro de 1997.

# Utilizando LOTOS na concepção formal de uma aplicação para Gerência de Redes: Especificação e Verificação

Braulio Adriano de Mello

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, Especialidade Sistemas de Computação, e aprovada em sua forma final pelo Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina.

Prof. Murilo Silva de Camargo, Dr.
Orientador, INE, UFSC

Profa. Elizabeth Sueli Specialski, M.Sc.
Co-Orientadora, INE, UFSC

Prof. Murilo Silva de Camargo, Dr.
Coordenador do CPGCC, INE, UFSC

Banca Examinadora:

Prof. Murilo Silva de Camargo, Dr.
Presidente, INE, UFSC

Profa. Elizabeth Sueli Specialski, M.Sc.
INE, UFSC

Prof. Rosvelter João Coelho da Costa, Dr.
INE, UFSC

# Agradecimentos

Agradeço aos amigos e colegas e aos professores do CPGCC da UFSC.

À UFSC e à CAPES, pelo suporte material e apoio financeiro.

À Universidade Regional Integrada (URI) pela oportunidade.

Ao professor Afonso Canisio Jung (URI - Campus de Santo Ângelo), pela revisão do texto.

À Verinha e à Valdete pela dedicação no atendimento aos alunos do curso, mesmo nos múltiplos pedidos para ontem.

Aos professores Vitório Bruno Mazzola e Rosvelter João Coelho da Costa pelas contribuições ao presente trabalho.

Ao Alexandre Vitoreti de Oliveira, pelas discussões e contribuições para este trabalho.

À professora Elizabeth Sueli Specialski, pelo incentivo e pela co-orientação durante o curso de Mestrado.

Ao professor Murilo Silva de Camargo, pela orientação, apoio e motivação no desenvolvimento deste trabalho.

À *diretoria,* sinônimo de amizade verdadeira solidificada por tempos de dificuldades, alegrias e comemorações.

Ao Carlos Margutti, meu tio, pelo exemplo de vida, determinação e superação.

À minha noiva pela compreensão e amor.

Aos meus pais e ao meu irmão, pelo incentivo, pelas orações e pela presença em todos os momentos.

# Sumário

Lista de Figuras ..... vi
Lista de Tabelas ..... vi
Lista de Siglas ..... viii
Resumo ..... xi
Abstract ..... xii
1. Introdução ..... 1
2. Uma visão geral sobre Concepção Formal de Sistemas ..... 3
2.1 Um método para a concepção de sistemas ..... 3
2.2 Uso de TDFs na concepção e verificação de sistemas ..... 5
2.2.1 Sistemas de Transições Rotulados ..... 7
2.2.2 Redes de Petri ..... 7
2.2.3 Cálculo de Processos ..... 8
2.2.4 Escolha de LOTOS ..... 9
2.3 Validação e Verificação ..... 9
2.3.1 Formas de verificação ..... 10
2.3.2 Métodos de verificação ..... 11
2.3.3 Relações de equivalência ..... 12
2.3.4 Análise de alcançabilidade ..... 14
2.4 Comentários finais ..... 15
3. O Gateway CMIP-SNMP: Modelo OSI x Modelo Internet ..... 16
3.1 Gerenciamento de Redes ..... 16
3.2 Modelo de Gerenciamento OSI ..... 18
3.2.1 Gerentes, Agentes e Objetos Gerenciados ..... 18
3.2.2 Informação de Gerenciamento ..... 19
3.3 Modelo de Gerenciamento Internet ..... 20
3.4 OSI x Internet ..... 21
3.5 Requisitos informais da aplicação Gateway CMIP-SNMP ..... 21
3.5.1 Funcionalidade do Gateway ..... 21
3.5.2 Estrutura de Mapeamento ..... 23
3.5.3 Mapeamento Funcional ..... 23

3.5.4 Tamanho do pacote ..... 25
3.5.5 Operações de gerenciamento da aplicação ..... 25
3.6 Comentários finais ..... 30
4. Concepção Formal do Gateway CMIP-SNMP usando LOTOS ..... 32
4.1 Estrutura geral do Gateway ..... 32
4.2 Modelo abstrato da aplicação ..... 35
4.3 Modelo detalhado da aplicação ..... 36
4.4 Tipos de Dados ..... 39
4.5 Comportamento ..... 45
4.6 Análise da aplicação especificada ..... 56
4.6.1 Análise sintática e semântica ..... 57
4.6.2 Simulação ..... 57
4.6.3 Verificação ..... 61
4.7 Comentários finais ..... 72
5. Algumas considerações sobre as ferramentas LOTOS usadas na concepção do Gateway ..... 74
5.1 Ferramentas computacionais para LOTOS ..... 74
5.2 Incompatibilidades ..... 79
5.3 Comentários finais ..... 80
6. Conclusões e perspectivas futuras ..... 82
7. Referências bibliográficas ..... 85
Anexo A - A TDF LOTOS ..... 90
Anexo B - Ferramentas Computacionais para LOTOS ..... 96
Anexo C - Implementação Automática de Especificações LOTOS ..... 104

# Lista de Figuras

Figura 2.1 - Processo de concepção ..... 4
Figura 2.2 - Equivalência observacional forte ..... 12
Figura 2.3 - Equivalência observacional fraca ..... 13
Figura 2.4 - Equivalência de teste ..... 14
Figura 2.5 - Não são equivalência de teste ..... 14
Figura 3.1 - Operações e Informações de gerenciamento ..... 19
Figura 3.2 - Modelo Internet ..... 20
Figura 3.3 - Aplicação Gateway ..... 22
Figura 4.1 - Módulos do Gateway ..... 32
Figura 4.2 - Contexto do Gateway ..... 33
Figura 4.3 - Comportamento geral do Gateway ..... 34
Figura 4.4 - Funcionalidades mapeadas para GET e SET ..... 35
Figura 4.5 - Arquitetura da aplicação Gateway ..... 35
Figura 4.6 - *Sorts* para a função de associação ..... 41
Figura 4.7 - Operações para a função de associação ..... 41
Figura 4.8 - *Sorts* das operações de gerenciamento ..... 42
Figura 4.9 - Operações de gerenciamento (*request,confirm,response*) ..... 43
Figura 4.10 - *Sort signal* para identificar eventos gerais ..... 43
Figura 4.11 - *Sorts* para controle das funcionalidades ..... 44
Figura 4.12 - *Sort signal_set* para identificar variações da operação SET ..... 44
Figura 4.13 - Tipo de dado para conjunto de objetos ..... 45
Figura 4.14 - Comportamento geral do Gateway ..... 46
Figura 4.15 - Estabelecimento da associação com agente SNMP ..... 47
Figura 4.16 - Comportamento do processo G_ASSOCIATE ..... 47
Figura 4.17 - Tratamento de uma requisição de associação ..... 48
Figura 4.18 - Instanciação das operações de gerenciamento e interrupção ..... 48
Figura 4.19 - Interrupção operações de gerenciamento ..... 49
Figura 4.20 - Processo que instancia as operações de gerenciamento ..... 49
Figura 4.21 - Processo principal da operação G_SET ..... 50
Figura 4.22 - Operação G_SET no modo confirmado ..... 52
Figura 4.23 - Operação G_SET no modo não confirmado ..... 52
Figura 4.24 - Operação G_GET ..... 53
Figura 4.25 - Operação TRAP ..... 53
Figura 4.26 - Processo que trata do comportamento do relatório de evento ..... 53
Figura 4.27 - Estabelecimento de associação com agente OSI ..... 54

Figura 4.28 - Operação M_GET ..... 54
Figura 4.29 - M_SET no modo confirmado ..... 55
Figura 4.30 - M_SET no modo não confirmado ..... 55
Figura 4.31 - Relatório de Evento de agente OSI ..... 55
Figura 4.32 - G_GET com escopo, filtro e sincronização especificados ..... 60
Figura 4.33 - G_SET com escopo, filtro e sincronização especificados ..... 60
Figura 4.34 - Ilustração do uso de dados ..... 65
Figura 4.35 - Resultados apresentados pela ferramenta LOLA em um teste ..... 69
Figura A.1 - O tipo de dado Boolean ..... 94
Figura B.1 - Sintaxe das anotações ..... 102

# Lista de Tabelas

Tabela 2.1 - Operadores CCS ..... 8
Tabela 2.2 - Operadores CSP ..... 9
Tabela 3.1 - Tabela de identificação dos Agentes ..... 26
Tabela 3.2 - Estrutura de Mapeamento ..... 27
Tabela 4.1 - Teste de G_GET para a seleção de um objeto ..... 70
Tabela 4.2 - Teste de G_GET para a seleção de um objeto (em separado) ..... 70
Tabela 4.3 - Teste de G_GET para a seleção simples de múltiplos objetos ..... 71
Tabela 4.4 - Teste de G_GET com funcionalidade de sincronização especificada ..... 72
Tabela 6.1 - Funcionalidades de algumas ferramentas ..... 75
Tabela 6.2 - Incompatibilidades entre TOPO e CAESAR ..... 80
Tabela A.1 - Operadores LOTOS ..... 91

# Lista de Siglas

| | |
|---|---|
| ACSE | - Association Control Service Element |
| API | - Application Program Interfaces |
| ASE | - Application Service Element |
| ASN.1 | - Abstract Syntax Notation.One |
| BRISA | - Sociedade Brasileira para a Interconexão de Sistemas Abertos |
| CCITT | - Consultative Committee for International Telegraph and Telephone |
| CCR | - Concurrency Control and Recovery |
| CCS | - Calculus of Communicating Systems |
| CMIP | - Common Management Information Protocol |
| CMIPDU | - Common Management Information Protocol Data Units |
| CMIS | - Common Management Information Service |
| CMISE | - Common Management Information Service Element |
| CPGCC | - Curso de Pós-Graduação em Ciência da Computação |
| CSP | - Communicating Sequential Processes |
| DN | - Distinguished Name |
| ESTELLE | - Extended Finite State Machine Language |
| FDT | - Formal Description Techniques |
| IP | - Internet Protocol |
| ISO | - International Organization for Standardization |
| ITU | - International Telecommunications Union |
| LITE | - LOTOSphere Integrated Tool Environment |
| LOLA | - LOtos LAboratory |
| LOTOS | - Language of Temporal Ordering Specification |

| | |
|---|---|
| LTS | - Labeled Transition System |
| MIB | - Management Information Base |
| MIS | - Management Information Service |
| MIM | - Management Information Model |
| MO | - Managed Object |
| NMF | - Network Management Forum |
| NMS | - Network Management Station |
| OSI | - Open Systems Interconnection |
| OID | - Object Identifier |
| PDU | - Protocol Data Unit |
| RDN | - Relative Distinguished Name |
| ROSE | - Remote Operations Service Element |
| RTSE | - Reliable Transfer Service Element |
| SDL | - Specification and Description Language |
| SMAE | - System Management Application Entity |
| SMASE | - System Management Application Service Element |
| SMI | - Structure Management Information |
| SNMP | - Simple Network Management Protocol |
| TCP | - Transmission Control Protocol |
| TDA | - Tipo de Dado Abstrato |
| TDF | - Técnica de Descrição Formal |
| UDP | - User Datagram Protocol |
| UFSC | - Universidade Federal de Santa Catarina |

# Resumo

LOTOS é uma técnica de descrição formal (TDF) que, através de um conjunto de operações algébricas, permite a especificação de sistemas com alto nível de abstração, facilitando a detecção e resolução de variados problemas de concepção. Existem inúmeras ferramentas de software que dão suporte às tarefas de análise, simulação, teste e verificação de especificações baseadas nesta TDF. A TDF LOTOS foi padronizada pela ISO em 1988 e é utilizada principalmente na descrição formal de sistemas distribuídos.

Este trabalho apresenta um estudo sobre a aplicação da técnica de descrição formal LOTOS na concepção formal de um sistema para gerência de redes denominado Gateway CMIP-SNMP. São apresentados os resultados obtidos durante o trabalho de descrição formal do sistema em LOTOS e durante o desenvolvimento das tarefas de análise, simulação, teste e verificação. Devido ao uso de dados, principalmente para a tarefa de verificação, variadas restrições são impostas pelas ferramentas utilizadas. Tais restrições são abordadas segundo as capacidades e incompatibilidades dessas ferramentas.

# Palavras chave

Técnicas de Descrição Formal, LOTOS, Análise, Simulação, Verificação, Gerência de Redes, Protocolos, Gateway CMIP-SNMP.

# Abstract

LOTOS is a formal description technique (FDT) that, by a set of algebraic operations, allows the specification of system with high level of abstraction, meaning the detection and the resolution of conception problems. There are tools that suport tasks of analisys, simulation, test and verification of LOTOS specifications. The FDT LOTOS is an ISO standard since 1988, and it is mostly used on the distributed systems description.

This work presents a study about the application of the LOTOS formal description technique on the formal conception of a network management system called CMIP-SNMP Gateway. This work show the results reached during the work of the formal description of system using full LOTOS, and during the development of analysis, simutation, test and verification of the specification. With the use of abstract data types, mostly for the verification task, the tolls present restrictions. These restrictions are approached by their capacities and incompatibilities.

# Keywords

Formal Description Techniques, LOTOS, Analysis, Simulation, Verification, Network Management, Protocols, CMIP-SNMP Gateway.

# 1. Introdução

O desenvolvimento de sistemas pode se tornar uma tarefa complexa, principalmente em aplicações mais críticas, onde existe a necessidade de um alto grau de corretude e confiabilidade. Em conseqüência, as técnicas para a descrição formal de sistemas estão sendo cada vez mais utilizadas. Com o uso de uma técnica formal, o processo de correção do sistema, antes de sua implementação final, é facilitado. As TDFs (Técnicas de Descrição Formal) apresentam-se como uma solução para vários problemas de concepção de sistemas em geral.

Através de uma técnica formal, é possível descrever sistemas livres de ambigüidades. A detecção e correção de erros de concepção também é facilitada a partir do uso de ferramentas automáticas de análise, teste, simulação e verificação das especificações.

A técnica de descrição formal utilizada neste trabalho é a TDF LOTOS (*Language of Temporal Ordering Specification*) [ISO 8807]. Esta técnica possui uma base formal bem definida e oferece um grande poder de expressividade e abstração. O componente comportamental de LOTOS segue uma linha baseada nos cálculos de processos CCS [Miln 80] e CSP [Hoar 85]. O tratamento de dados é feito utilizando-se da linguagem de especificação algébrica ACT ONE [EhMa 85].

A especificação LOTOS de um sistema pode não ser uma tarefa simples. A interpretação de descrições em linguagem natural geralmente é difícil devido à sua natureza ambígua e sua abstração forte. Assim, durante o processo de especificação formal de um sistema, as ambigüidades inerentes à linguagem natural devem ser resolvidas. Isto requer a tomada de decisões durante esse processo. Também, não há um conjunto de ferramentas que desenvolva, sem restrições, tarefas tais como a verificação, principalmente para especificações LOTOS que utilizam tipos de dados.

Este trabalho apresenta um estudo visando a explorar a aplicabilidade da técnica de descrição formal LOTOS na concepção e verificação de sistemas. Mais especificamente, uma aplicação Gateway CMIP-SNMP [Oliv 96] para gerência de redes

de computadores. De forma geral, esta aplicação permite que uma aplicação gerente que segue os padrões do modelo OSI (*Open Systems Interconnection*) [ISO1 91] execute operações de gerenciamento sobre objetos que seguem os padrões do modelo SNMP (*Simple Network Management Protocol*) [Stal 93].

Neste estudo, a ênfase é dada a uma avaliação de possíveis problemas durante o processo de concepção. Tal avaliação envolve tarefas tais como especificação, simulação, teste e verificação, associadas ao uso de ferramentas afins. Quanto às ferramentas, é apresentada uma discussão sobre suas capacidades, restrições e incompatibilidades, principalmente, no que ser refere às dificuldades impostas pelo uso de dados.

No restante desta monografia, o termo Gateway será usado para denotar o Gateway CMIP-SNMP, objeto principal de estudo deste trabalho.

A organização desta monografia é apresentada a seguir.

No capítulo 2, é apresentada uma visão geral sobre a concepção formal de sistemas. Em seguida, são abordados alguns formalismos e apresenta-se uma discussão sobre a conceituação dos termos validação e verificação.

No capítulo 3, inicialmente, identifica-se o contexto em que se insere a aplicação Gateway. Em seguida, é apresentado um breve comentário sobre necessidades de gerenciamento, justificando o uso do Gateway. Finalmente, são apresentados os requisitos informais (em linguagem natural) da aplicação Gateway.

No capítulo 4, apresenta-se a arquitetura do modelo abstrato do Gateway e de seus componentes principais. O comportamento interno de cada módulo é então comentado informalmente. Logo após, é apresentada a especificação LOTOS do Gateway e os resultados da análise, simulação, teste e verificação da especificação.

No capítulo 5, são apresentadas algumas considerações sobre as ferramentas LOTOS usadas na concepção do Gateway. São comentadas suas capacidades, restrições e incompatibilidades.

Por fim, no capítulo 6, são apresentadas as conclusões e discute-se os resultados alcançados, as limitações da metodologia adotada e perspectivas futuras.

# 2. Uma visão geral sobre Concepção Formal de Sistemas

Técnicas de descrição formal (TDFs) têm sido usadas como uma solução para variados problemas na concepção de sistemas, principalmente quando a complexidade é mais acentuada. As TDFs capacitam o especificador a conceber especificações precisas, fornecendo fundamentação necessária para o trabalho de análise em busca da correção do sistema.

Neste capítulo, um método de concepção de sistemas é apresentado. Este método é dividido em fases distintas, onde aspectos de uma metodologia são considerados. Em seguida, é discutido o uso de TDFs na concepção formal e verificação de sistemas. Alguns formalismos são sucintamente abordados.

## 2.1 Um método para a concepção de sistemas

Um método para a concepção de sistemas pode ser dividido em várias fases consecutivas. O método aqui apresentado tem suas fases representadas na Figura 2.1, onde cada fase representa um determinado nível do projeto, nas quais são desenvolvidas atividades afins [EFHJ 91]. As fases são as seguintes: levantamento de requisitos, projeto do serviço e protocolo e, especificação da implementação. Para as tarefas desenvolvidas em cada uma das fases, foram adotados alguns aspectos da metodologia definida em [Mazz 91].

Os requisitos ou requerimentos de um sistema são as especificações em linguagem natural que definem as propriedades desejáveis deste sistema. Normalmente, os requisitos possuem ambigüidades e inconsistências, o que impossibilita qualquer tipo de prova de corretude.

Na fase de levantamento dos requisitos, são identificados os aspectos principais do sistema. A arquitetura é então definida, identificando-se os módulos, seus relacionamentos e canais utilizados que irão compor a arquitetura. Neste nível, pode-se fazer a diagramação em blocos, o que facilita o entendimento. Também é feita a definição

do comportamento destes módulos, onde as operações são descritas, e determinada a ordem em que estas operações podem ocorrer.

Na fase de projeto do serviço e protocolo, com base na arquitetura do sistema, é concebida a *especificação abstrata* (serviço) que representa os aspectos observáveis do sistema. Em seguida, o funcionamento completo do sistema é descrito formalmente quando concebida a *especificação detalhada* (protocolo). Este segundo nível da especificação trata de aspectos internos, e deve manter as propriedades identificadas na especificação abstrata do sistema. Ou seja, deve existir uma relação de equivalência entre os sistemas de transições de ambas as especificações (abstrata e detalhada). A análise (sintática e semântica), teste, simulação e verificação da especificação também são realizadas neste estágio.

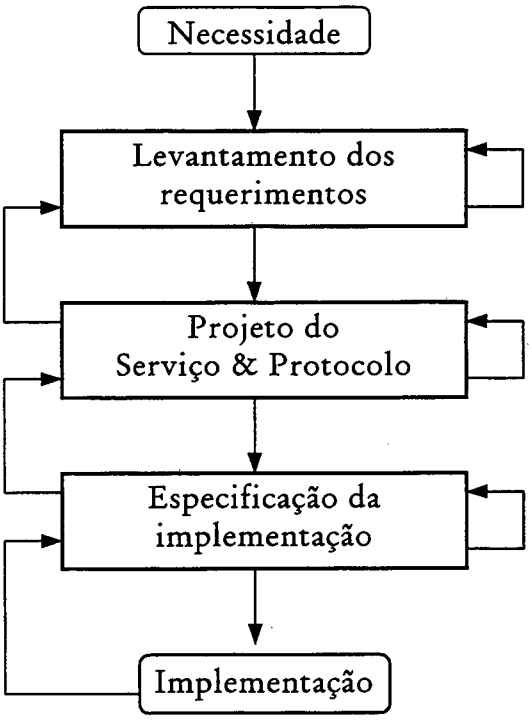


Figura 2.1 - Processo de concepção

Na fase de especificação da implementação, o resultado deve ser uma especificação que permita sua tradução, o mais diretamente possível, para a implementação em código final.

A escolha do formalismo é importante, pois diferentes técnicas de descrição formal apresentam características distintas. Estas características podem ser mais ou menos adequadas às propriedades que se deseja representar ou funcionalidades requeridas (por exemplo, implementação final automatizada). Principalmente durante as análises das especificações, o uso de ferramentas automatizadas como apoio ao trabalho humano é bastante desejável.

Em resumo, o uso de técnicas na concepção de sistemas, associadas a uma seqüência de passos previamente estabelecidos, pode resultar na construção de sistemas melhores em todos os aspectos. Essa seqüência de passos caracteriza o método (ou metodologia) utilizado que visa a possibilitar a sistematização do processo de desenvolvimento.

A seguir, o uso de técnicas formais na concepção e verificação de sistemas é melhor discutido.

## 2.2 Uso de TDFs na concepção e verificação de sistemas

TDFs são utilizadas para representar, através de uma notação formal, os requisitos de um sistema computacional, ou seja, especificar formalmente um sistema.

O termo *especificação* pode ser entendido como a descrição das propriedades que um sistema deve apresentar. Já a *especificação formal* segue sintaxe e semântica definidas e associadas a uma linguagem para descrever essas propriedades. A especificação formal difere de linguagens naturais que geralmente são ambíguas e muito expressivas. A especificação formal inicial de um sistema pode ser considerada como a primeira representação precisa (não ambígua) do comportamento desse sistema.

É importante lembrar que uma especificação em linguagem natural fornece os requisitos para a especificação formal de um sistema. Portanto, deve mostrar o que o sistema deve fazer, e não como fazer. A representação de detalhes não visíveis ao usuário neste estágio de desenvolvimento do sistema pode resultar em excessiva e desnecessária complexidade.

Quanto ao primeiro nível da representação formal, não há como garantir que os requisitos do sistema estão representados correta e completamente. O mesmo problema

ocorre na tradução de uma especificação detalhada para a implementação final (codificação). Algumas técnicas disponibilizam ferramentas que geram código final automaticamente, porém, permanece a falta de garantia (prova) de que o código gerado mantém a correção.

De modo geral, a importância do uso de técnicas de descrição formal é justificada pela facilidade fornecida ao especificador na detecção de problemas. Estes problemas podem proliferar durante o desenvolvimento de um sistema, podendo causar transtornos até mesmo irreversíveis após a sua implementação final. Como conseqüência do uso de uma TDF, maior grau de corretude é agregado ao sistema.

A seguir, são listadas algumas vantagens geralmente oferecidas no uso de TDFs.

- *Abstração:* A abstração permite descrever sistemas na sua essência, ou seja, sem preocupação com detalhes irrelevantes à representação do comportamento desejado (detalhes de implementação).

- *Simulação:* Com a simulação, é possível observar o comportamento das especificações com base no formalismo fornecido pela técnica de descrição utilizada. Possibilita uma avaliação interativa onde erros podem ser detectados mais facilmente. A simulação não é exaustiva, por isso o grau de eficácia depende da intensidade em que é aplicada.

- *Testes:* Através dos testes sobre as especificações, é possível verificar se uma determinada seqüência de ações é válida ou não. Para isso, usa-se um processo em composição paralela com o processo que se deseja testar. Neste processo de teste é especificada a seqüência desejada de eventos além de um outro evento que indica o sucesso do teste. Assim como a simulação, o teste não é exaustivo, portanto sua eficácia também depende da intensidade em que é aplicado.

- *Verificação:* A verificação de propriedades das especificações é utilizada para provar exaustivamente que uma determinada propriedade da especificação é verdadeira. As TDFs fornecem métodos e técnicas para a verificação, como por exemplo, as relações de equivalência entre especificações com níveis de abstração diferentes. Também permite análise exaustiva para a detecção de situações indesejáveis (deadlocks, por exemplo).

- *Ferramentas automatizadas:* O uso de ferramentas agiliza as tarefas de análise das especificações. A utilização de métodos automatizados nessas tarefas também tende a reduzir a ocorrência de erros humanos.

### *2.2.1 Sistemas de Transições Rotulados*

Os sistemas de transições rotulados são utilizados na modelagem do comportamento de processos e a semântica operacional de LOTOS é definida em termos de sistemas de transições rotulados [ISO 8807]. A transição é interpretada como a atividade realizada quando ocorre uma troca de estados no sistema, por exemplo, o envio ou recebimento de mensagens. Os rótulos das transições representam essas ações.

Um STR (*Labeled Transition System*) é uma quádrupla $<S, So, E, T>$, onde [EFHJ 91]:

- $S$ representa o conjunto de estados;
- $So$ representa o estado inicial, onde $So \in S$;
- $E$ representa o conjunto de nomes de transições;
- $T$ representa o conjunto de transições rotuladas. Cada transição consiste de um estado $s$, um rótulo $a$, e um estado $s'$. Na transição $s \xrightarrow{a} s'$, o estado corrente s pode evoluir para $s'$ pela realização da ação $a$.

Os sistemas de transições podem ter um número infinito de estados dependendo do sistema modelado. Para sistemas finitos, existe um grande número de ferramentas computacionais que realizam a análise automática de comportamento.

### *2.2.2 Redes de Petri*

A teoria de Redes de Petri [Bram 83] foi um dos primeiros formalismos introduzidos para tratar concorrência, indeterminismo e relações causais entre eventos. As redes de Petri têm sua estrutura representada por lugares e transições. Os lugares representam atividades do sistema e as transições, ligadas a lugares de entrada e saída, representam a ocorrência de eventos. A ocorrência de eventos no sistema se dá através do disparo das transições, com a retirada de fichas dos lugares de entrada e seu depósito nos lugares de saída. Nas redes de Petri interpretadas, é dado um sentido concreto à um modelo, associando lugares, transições e fichas às entidades externas. A ferramenta

CAESAR gera uma rede de Petri interpretada de uma especificação LOTOS, sobre a qual realiza a detecção de deadlocks nas especificações.

Uma apresentação mais detalhada das propriedades e aplicações das redes de Petri pode ser encontrada em [Mura 89].

### *2.2.3 Cálculo de Processos*

Das várias álgebras de processos existentes na literatura, existem três que se destacam pela sua relevância: CSP (*Communicating Sequential Processes)* [Hoar 85], CCS (*Calculus of Communicating Systems)* [Miln 80], e LOTOS (*Language of Temporal Ordering Specification*) [ISO 8807].

CCS é um cálculo de processos para a especificação de sistemas concorrentes. Mesmo com um conjunto pequeno de operadores, CCS tem um alto nível de expressividade, possibilitando descrever sistemas em diferentes níveis de abstração. O conjunto de operadores CCS são listados na Tabela 2.1, onde $P$ e $Q$ representam processos, $a$ representa uma ação e L um conjunto de nomes de ações.

| Descrição | Notação |
|---|---|
| ação interna | $\tau$ |
| processo inativo (deadlock) | 0 |
| prefixo de ação | $a.P$ |
| escolha | $P + Q$ |
| paralelismo / sincronização | $P \mid Q$ |
| restrição | $P \setminus L$ |

Tabela 2.1 - Operadores CCS

Assim como CCS, CSP é um cálculo de processos para a especificação de sistemas concorrentes. Comparado a CCS, as principais diferenças são baseadas na possibilidade de sincronização múltipla e no maior número de operadores. O conjunto de operadores CSP são listados na Tabela 2.2, onde $P$ e $Q$ representam processos, $a$ representa uma ação.

| Descrição | Notação |
|---|---|
| deadlock | stop |
| divergência | chaos |
| prefixo de ação | $a \rightarrow P$ |
| escolha determinística baseada em ações (binárias) | $(a \rightarrow P) \mid (b \rightarrow P)$ |
| (e sobre conjuntos de ações) | $(x{:}A \rightarrow P(x))$ |
| escolha geral sobre processos | $P \,[]\, Q$ |
| escolha indeterminística demoníaca sobre processos | $P \sqcap Q$ |
| paralelismo | $P \,\|\|_A\, Q$ |
| intercalação | $P \,\|\|\|\, Q$ |
| evento de comunicação (entrada) | c ? w |
| evento de comunicação (saída) | c ! 1 |
| ocultação de eventos | $P \setminus C$ |
| recursão | $\mu X : A.\ F(X)$ |

Tabela 2.2 - Operadores CSP

A TDF LOTOS é baseada principalmente em CCS e CSP. Também herdou conceitos de outras linguagens tais como o de tipos algébricos de dados da linguagem de especificação algébrica ACTONE [EhMa 85] . A TDF LOTOS é brevemente apresentada no Anexo A.

### *2.2.4 Escolha de LOTOS*

LOTOS possui mecanismos de abstração que permitem a especificação de aplicações em diferentes níveis de abstração. Possui um formalismo bem definido e oferece meios para estruturar as especificações. Por isso, optou-se pela linguagem LOTOS para concepção formal do Gateway CMIP-SNMP desenvolvida a partir do capítulo 3 deste trabalho.

## 2.3 Validação e Verificação

Em [EFHJ 91], o termo validação é usado para designar atividades de detecção e correção de erros de uma dada especificação ou implementação. Três atividades são: verificação, simulação e teste. A atividade de verificação visa a provar que uma especificação satisfaz determinadas propriedades. Na simulação, o comportamento de um modelo é observado em busca de erros, embora de forma não exaustiva. No teste, um

processo é submetido a uma seqüência de ações já conhecida, devendo fornecer o mesmo resultado (para a seqüência de ações dada).

Já em [Kirk 94], verificação é entendido como prova formal das propriedades pela manipulação de axiomas, e validação como uma demonstração convincente por experimentos de que uma dada especificação está correta. Tais experimentos podem incluir simulação ou testes, por exemplo.

Neste trabalho, o termo verificação é assumido como prova formal das propriedades de uma especificação através de um método (relações de equivalência, por exemplo). E validação, como um conjunto de procedimentos visando a estabelecer a satisfabilidade de um sistema com respeito a um conjunto de propriedades relativas à sua corretude.

Tomando como exemplo um protocolo de comunicação, a verificação tem como meta provar a equivalência entre a especificação do protocolo proposto e a especificação de seu respectivo serviço. Ela também visa a provar que a especificação possui ou não propriedades desejáveis, tais como de segurança(*safety*) e de vivacidade (*liveness*).

Um dos objetivos da validação é de examinar se o serviço especificado representa o comportamento que o usuário tem em mente. Durante este processo, geralmente, o sistema é apresentado ao usuário que desempenha um papel de fiscal.

A aplicação de uma técnica formal na descrição de sistemas pode ser bastante dificultada quando sujeita ao julgamento do usuário e também do próprio especificador. O uso de um formalismo força o projetista a estruturar e simplificar suas especificações bem como formular e formalizar possíveis suposições quanto ao sistema.

Em [Pehr 88], é apresentado um tutorial sobre a verificação de protocolos.

### *2.3.1 Formas de verificação*

O processo de verificação de uma especificação pode ser executado de duas formas gerais [Kirk 94], por comparação ou pela prova individual. Na primeira, é feita a comparação formal entre duas especificações. Na segunda, as propriedades individuais de uma especificação são verificadas. Para cada caso, são utilizados diferentes métodos.

Na comparação formal entre duas especificações, pretende-se provar que uma especificação satisfaz outra. Tais especificações podem usar o mesmo formalismo (técnica de especificação formal), ou formalismos diferentes.

Quando usado o mesmo formalismo, as especificações devem ter diferentes níveis de abstração, ou seja, uma deve ser um detalhamento da outra. Quando usados dois formalismos diferentes, é necessário uma prova que relacione a semântica da técnica usada em uma especificação com a semântica da técnica usada na outra especificação.

Na verificação de uma especificação individual, as propriedades desejáveis devem ser identificadas. Como exemplo de propriedade, pode-se citar a de segurança e a consistência interna. A propriedade de segurança define o que pode e o que não pode acontecer. A consistência deve permitir que uma alteração possa ser feita sem comprometer outras partes da especificação [EFHJ 91]. As propriedades de um sistema estão relacionadas aos seus requisitos informais.

### *2.3.2 Métodos de verificação*

A tarefa de verificação pode ser realizada por comparação, redução, prova individual, entre outros, como comentado no início do capítulo. Nesta seção, são apresentados alguns métodos utilizados no processo de verificação, entre eles, a análise de alcançabilidade, as relações de equivalência observacional e a prova de propriedades individuais.

A análise de alcançabilidade significa uma exploração exaustiva de todas as interações possíveis entre processos [Holz 87]; ela visa apenas propriedades de segurança, como por exemplo deadlocks e mensagens que chegam fora de seqüência e é viável para sistemas de transições finitos.

O método de verificação usando equivalência observacional consiste, basicamente, na comparação de duas especificações. Através desta comparação, é possível provar se o conjunto de estados do sistema de transição de uma especificação e o conjunto de estados do sistema de transição da outra especificação pertencem à mesma classe de equivalência. Ou seja, se comportam de forma equivalente. Existem diferentes tipos de relações que tratam de diferentes aspectos da especificação. A seguir, são apresentados alguns tipos de relações de equivalência observacional.

As propriedades gerais que um sistema deve satisfazer podem ser classificadas em propriedades de segurança e de vivacidade [Pehr 88], de consistência e de desempenho [EFHJ 91]. A verificação das propriedades de uma especificação individual consiste na sua exploração baseada em um raciocínio formal para a detecção de situações indesejáveis.

### *2.3.3 Relações de equivalência*

Na seção anterior, foram feitas algumas considerações sobre verificação e seus propósitos. Nesta seção, são apresentadas algumas relações de equivalência utilizadas na verificação de especificações LOTOS.

**Equivalência forte**: Informalmente, dois processos $P$ e $Q$ são ditos fortemente equivalentes se sempre que $P$ realizar uma ação $a$ (onde $a$ pode ser uma ação observável, uma ação de terminação com sucesso ou uma ação invisível) e se transformar em $P'$, então $Q$ também poderá realizar a mesma ação $a$ e se transformar em $Q'$ onde $P'$ têm a mesma propriedade acima apresentada para $P$ e $Q$. Quando $P$ e $Q$ são fortemente equivalentes, então escreve-se $P \sim Q$.

Uma relação $R \subseteq P \times P$ uma bissimulação forte se $(P,Q) \in R$ implica, $\forall a \in Act$:

1. *se* $\exists P' : P \rightarrow^{a} P'$ *então* $\exists Q' : Q \rightarrow^{a} Q'$ *e* $(P',Q') \in R$, e

2. *se* $\exists Q' : Q \rightarrow^{a} Q'$ *então* $\exists P' : P \rightarrow^{a} P'$ *e* $(P',Q') \in R$

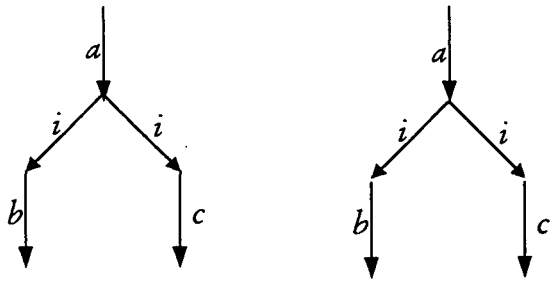


Figura 2.2 - Equivalência observacional forte

**Equivalência observacional fraca**: Informalmente, dois processos $P$ e $Q$ são ditos fracamente equivalentes se sempre que $P$ realizar uma ação $a$ (onde $a$ é uma ação observável ou uma ação de terminação com sucesso) e se transformar em $P'$, então $Q$ também poderá realizar a mesma ação $a$ e se transformar em $Q'$ onde $P'$ têm a mesma propriedade acima apresentada para $P$ e $Q$. Nessa equivalência, as ações internas $i$ são absorvidas e apenas as ações visíveis são consideradas, ou seja, uma ação $a$ em $P$ corresponde à uma ação $a$ associada à zero ou mais eventos internos $i$ em $Q$. Quando $P$ e $Q$ são fracamente equivalentes, então escreve-se $P \approx Q$.

Dados os estados $p$ e $p'$, escreve-se $p \Rightarrow^{\hat{a}} p'$ , se $p$ executa uma ação $a$ e chega a um estado precedida ou sucedida por zero ou mais eventos internos $i$, e chega a um estado $p'$. Uma relação $R \subseteq P \times P$ é uma bissimulação fraca se $(P,Q) \in R$ implica, $\forall a \in Act$:

1.*se* $\exists P': P \rightarrow^{a} P'$ *então* $\exists Q': Q \Rightarrow^{\hat{a}} Q'$ *e* $(P',Q') \in R$, e

2.*se* $\exists Q': Q \rightarrow^{a} Q'$ *então* $\exists P': P \Rightarrow^{\hat{a}} P'$ *e* $(P',Q') \in R$

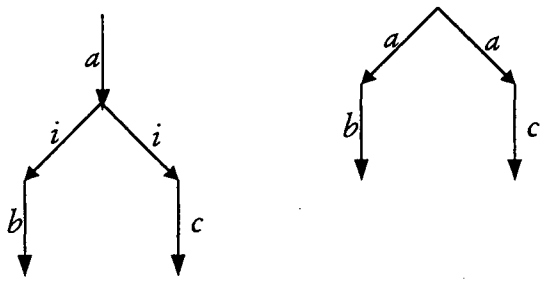


Figura 2.3 - Equivalência observacional fraca

**Equivalência de teste**: A equivalência de teste [Nico 87] é baseada em experimentos sobre processos, ou seja, dois processos possuem esta equivalência se ambos são capazes de passar pela mesma seqüência de testes. Para defini-la, é preciso um conjunto de processos observadores, um modo de observação e um critério para avaliação dos resultados da observação. Em [EFHJ 91], é apresentada uma definição para equivalência de teste utilizando uma notação denominada conjunto de aceitação. Por exemplo, de

acordo com a definição apresentada em [EFHJ 91], os grafos da Figura 2.4 são equivalência de teste. Pois, o conjunto de aceitação é {{b},{c}} em ambos os casos. Já para os grafos apresentados na Figura 2.5 não existe equivalência de teste. Pois, o conjunto de aceitação do primeiro grafo é {{b,c}}, e do segundo é {{b},{c}}.

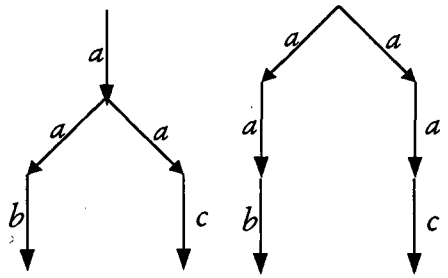


Figura 2.4 - Equivalência de teste

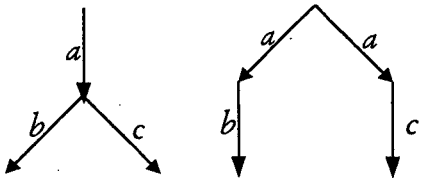


Figura 2.5 - Não são equivalência de teste

**Hierarquia de algumas relações de equivalência**

As principais equivalências observacionais apresentadas possuem a ordem hierárquica abaixo, em ordem decrescente na direção da seta.

equivalência forte

↓

equivalência observacional (fraca)

↓

equivalência de teste

### *2.3.4 Análise de alcançabilidade*

Uma das primeiras formas de verificação de protocolos utilizada foi a análise de alcançabilidade. Neste método, é realizada uma exploração exaustiva de todas as possíveis interações entre um conjunto de processos que se comunicam, modelados como um sistema de transição de estados finito. A análise de alcançabilidade permite detectar apenas erros de segurança.

Este tipo de análise é relativamente fácil de ser automatizado através de uma rotina de exploração. Porém, na prática, existem problemas relativos à explosão de estados. O conjunto de estados modelados deve ser finito e seu tamanho limitado ao espaço e à capacidade de processamento disponíveis no recurso utilizado.

Estudos têm sido realizados com o propósito de buscar melhoramentos para este método, envolvendo automatização em termos de limites e possibilidades. Exemplos de aplicação deste tipo de técnica podem ser encontrados em [Holz 87] e [Holz 94].

## 2.4 Comentários finais

Neste capítulo, foi discutido um processo de concepção de sistemas associando alguns aspectos de uma metodologia. Foi apresentada uma breve abordagem sobre o uso de técnicas de descrição formal no processo de concepção e alguns formalismos.

Em seguida, diferentes interpretações, principalmente do termo verificação, foram apontadas, identificando o que deve-se entender por verificação neste trabalho. Alguns métodos e formas de verificação também foram apresentadas.

O capítulo seguinte apresenta resumidamente o gerenciamento de redes de computadores. O contexto da aplicação Gateway e sua finalidade são então identificados, apontando também algumas características do modelo de gerenciamento OSI e do modelo Internet. Em seguida, os requisitos informais do Gateway são apresentados.

# 3. O Gateway CMIP-SNMP: Modelo OSI x Modelo Internet

Este capítulo apresenta uma introdução sobre gerenciamento de redes e uma visão geral de algumas das principais diferenças entre os dois modelos de gerenciamento mais conhecidos. São eles: o modelo OSI (*Open System Interconnection*), que utiliza o protocolo CMIP (*Common Management Information Protocol*), e o modelo Internet que utiliza o protocolo SNMP (*Simple Network Management Protocol*).

Em seguida, são apresentados os requisitos informais do Gateway CMIP-SNMP onde, inicialmente, sua funcionalidade geral é apresentada. Após, é mostrada a estrutura de mapeamento dos agentes e objetos gerenciados no modelo Internet. É apresentado o mapeamento funcional do sistema que permite a interoperabilidade entre o ambiente OSI e o ambiente Internet. Finalmente, são descritas as atividades fornecidas pelo Gateway. São elas: G_ASSOCIATE, G_GET, G_SET e G_TRAP. O Gateway CMIP-SNMP é definido em [Oliv 96].

## 3.1 Gerenciamento de Redes

As redes de computadores oferecem diversas vantagens comparadas aos sistemas centralizados. Assim, o uso desta tecnologia está em contínuo crescimento, tanto em número de componentes (tamanho da rede), como em número de redes. Em decorrência destes fatores, as redes também tornam-se mais complexas e difíceis de serem gerenciadas, resultando na maior ocorrência de problemas diversos. Estes problemas podem causar danos tais como nível de desempenho abaixo das necessidades, ou mesmo a inoperância da rede, o que é inaceitável.

Como forma de reduzir os problemas e garantir um nível de qualidade no mínimo suficiente, as redes devem ser gerenciadas [BRIS 93]. O gerenciamento de redes busca assegurar que as aplicações disponíveis na rede estejam operacionais durante o maior tempo possível ou todo o tempo [SoLe 95].

Reconhecida a necessidade de gerenciar as redes, aplicações com esta finalidade passaram a ser desenvolvidas. Porém estas aplicações eram proprietárias, ou seja, não se comunicavam com aplicações desenvolvidas por diferentes fornecedores. Isto obrigava o usuário a utilizar produtos de um único fornecedor.

Para resolver este problema, foram adotadas as APIs (*Application Program Interfaces*) que desempenham a tarefa de ler as informações de um recurso específico e fornecê-las ao sistema para que possam ser analisadas. Porém, as APIs exigem grande esforço de desenvolvimento, pois, para cada recurso, uma API deveria ser construída.

Desta forma, as aplicações passaram a ser desenvolvidas segundo padrões tais como o SNMP (padrão *de facto*), visando a integrar os diferentes sistemas de gerenciamento. No entanto, a falta de integração persistiu, pois o SNMP possui um conjunto restrito de funções. Estas funções estão voltadas aos aspectos físicos tais como roteadores e hubs, obrigando os fornecedores a criar extensões não padronizadas para os seus produtos.

Os padrões do modelo OSI foram, então, propostos para resolver esta falta de integração e as limitações em geral do gerenciamento de redes. Contudo, o padrão mais difundido é o SNMP devido à sua simplicidade e facilidade de implementação, comparado ao modelo OSI.

Observa-se, então uma tendência no uso de sistemas de gerenciamento que seguem padrões distintos na mesma rede (redes heterogêneas). Este fator tem estimulado pesquisas no sentido de conceber soluções que permitam a interoperabilidade entre estes sistemas.

Este trabalho aborda uma destas soluções, denominada Gateway. Este Gateway permite a interoperabilidade entre sistemas de gerenciamento que seguem o padrão SNMP e sistemas de gerenciamento que seguem o padrão OSI. Entende-se por interoperabilidade como a possibilidade de efetuar monitoramento e controle sobre objetos SNMP através de uma aplicação gerente OSI.

Esta operação ocorre de forma transparente. Ou seja, as informações (PDUs) resultantes das operações executadas sobre objetos SNMP são enviadas ao gerente OSI já no formato por ele reconhecido. Esta tradução do formato é realizada pelo Gateway em tempo de processamento (*on the fly*).

O Gateway especificado neste trabalho também possui um diferencial perante outras soluções semelhantes existentes. Além das funções normais de tradução, esta aplicação adiciona as funcionalidades de escopo, filtro e sincronização. Estas funcionalidades não possuem contrapartida no ambiente SNMP.

## 3.2 Modelo de Gerenciamento OSI

O modelo OSI foi projetado considerando um nível muito superior de detalhamento e capacidade de tratamento de informações que o modelo SNMP. Possui um conjunto maior de operações de gerenciamento. Isto permite grande versatilidade no controle exercido sobre uma rede, porém o torna mais complexo. Segue o conceito de gerente, agente e objeto gerenciado.

No modelo OSI, para que as informações sejam trocadas entre gerentes e agentes, são necessários alguns serviços de comunicação e suas primitivas de serviço associadas. O conjunto destes serviços e primitivas chama-se *CMIS (Common Management Information Service)*. Neste modelo, o protocolo de gerenciamento é o *CMIP (Common Management Information Protocol)* [X710 91] [X711 91]. O modelo OSI segue uma abordagem orientada a objetos.

No modelo OSI, um objeto gerenciado é definido em termos de seus atributos, comportamento, notificações e operações. Os atributos representam as características do objeto gerenciado. O comportamento representa as mudanças no objeto devido à operações nele executadas. As notificações indicam os eventos nos objetos. Por fim, as operações são executadas nos objetos gerenciados.

### *3.2.1 Gerentes, Agentes e Objetos Gerenciados*

Define-se o *gerente* como um processo iniciado pelo usuário que obtém informações atualizadas sobre objetos gerenciados. Para isso, transmite operações de gerenciamento ao processo chamado *agente*. O agente executa estas operações sobre os objetos gerenciados, podendo ainda transmitir notificações ao gerente [BRIS 93].

Agentes de gerenciamento podem ser de diferentes tipos, dependendo do relacionamento com seus objetos gerenciados. Os agentes também podem acessar, direta ou indiretamente, os objetos gerenciados. Um agente implementado consiste de um

código que manipula informação de gerenciamento (como variáveis de status, contadores, temporizadores).

Os objetos gerenciados representam os recursos sujeitos ao gerenciamento. Um objeto pode ser de um recurso, e um recurso pode ter mais de um objeto gerenciado. O conjunto de objetos gerenciados, seus atributos, operações e notificações, constitui a MIB [Stal 93].

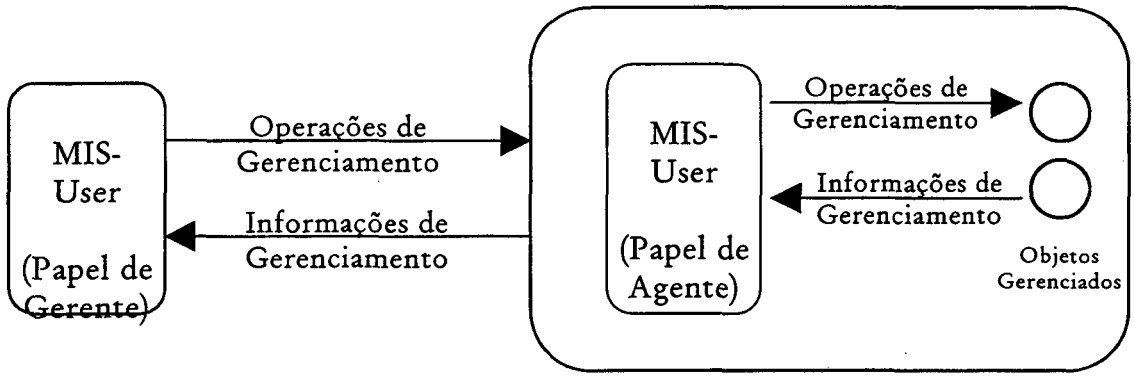


Figura 3.1 - Operações e Informações de gerenciamento

### *3.2.2 Informação de Gerenciamento*

Um requisito para manter um gerenciamento de rede satisfatório é a disponibilidade de informações completas sobre todos os elementos da rede. Estas informações de gerenciamento são coletadas e armazenadas por agentes, e então repassadas para um ou mais gerentes.

Na comunicação entre gerentes e agentes, podem ser usadas as técnicas de *polling* ou *event-reporting*. A técnica de *polling* consiste em uma interação do tipo *request/response* entre um gerente e um agente. Desde que o gerente tenha autorização, ele pode solicitar a um agente o envio de valores de elementos de informação. O agente responde com valores da *MIB (Management Information Base)*. Na técnica *event-reporting* o gerente espera pela chegada de informações enviadas por um agente, que pode gerar relatórios periódicos para fornecer ao gerente seu estado atual. Este relatório também pode ser gerado quando ocorre um evento significativo específico [Stal 93].

## 3.3 Modelo de Gerenciamento Internet

Este modelo consiste em um esquema centralizado, onde pelo menos uma estação de gerenciamento de rede *(NMS - Network Management Station)* é configurada como gerente. Os demais elementos da rede desempenham papel de agentes. São três os elementos que compõem este modelo: nós gerenciados, estações de gerenciamento, e protocolo de gerenciamento. A NMS que executa as aplicações de gerenciamento, também suporta o protocolo de gerenciamento. As informações de gerenciamento são armazenadas na MIB.

No modelo Internet, os papéis do gerente e do agente são fixos. Quanto ao aspecto de comunicação, as operações de gerenciamento, disponíveis para o gerente, restringem-se às requisições de leitura e troca de valores dos objetos gerenciados. Ao agente é disponibilizado o serviço de TRAP, através do qual ele informa ao gerente a ocorrência de eventos nos objetos gerenciados.

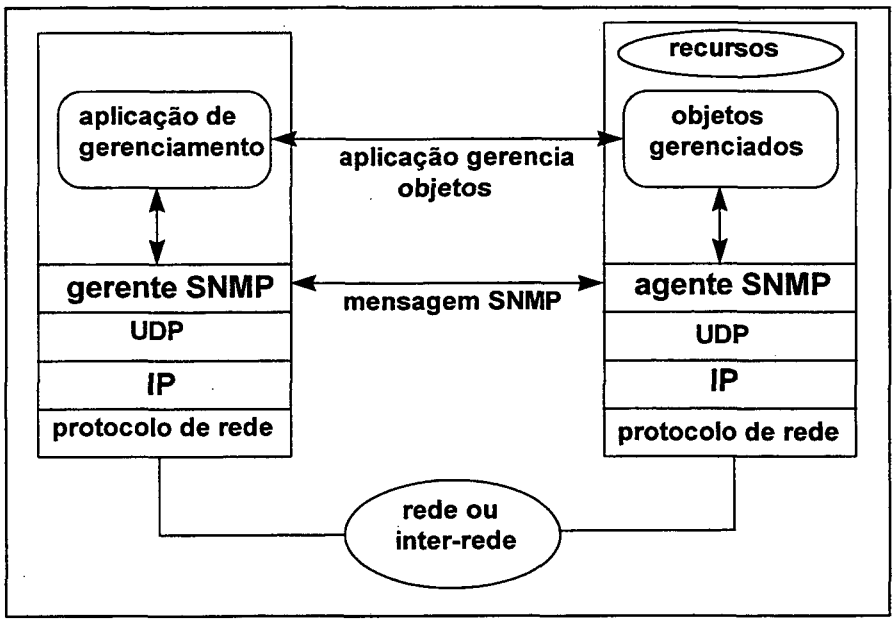


Figura 3.2 - Modelo Internet

Neste modelo, os objetos gerenciados têm as seguintes características: acesso, estado, nome e sintaxe. O acesso controla escrita e leitura, o estado classifica os objetos gerenciados como opcionais, obrigatórios e obsoletos. O nome significa o nome do

objeto gerenciado e a sintaxe define a sintaxe do objeto. O objeto gerenciado é identificado através do OID (*Object Identifier)* [ASN.1 87].

## 3.4 OSI x Internet

Ambos os protocolos, CMIP e SNMP, têm a finalidade de transferir informações entre sistemas de gerenciamento de redes. Nesta seção, são brevemente apresentadas as principais diferenças entre estes dois protocolos no que se refere à filosofia de acesso, à funcionalidade, à complexidade, ao desempenho e ao suporte de comunicação [BRIS 93].

Quanto à filosofia de acesso, ambos os protocolos trabalham através de *polling* (acesso periódico ao recurso). No SNMP, um recurso pode informar ao gerente que precisa ser submetido ao *polling* através de um TRAP. No CMIP, ocorre a notificação, onde o recurso é capacitado de enviar informações (quando ocorre um evento) sem necessidade de *polling*.

O CMIP possui funcionalidade maior que o SNMP, destacando a criação e deleção dinâmica de objetos. O Gateway, além de realizar a tradução das PDUs (*Protocol Data Units*), adiciona parte da funcionalidade OSI no lado SNMP (Capítulo 6).

O CMIP requer mais memória e capacidade de processamento que o SNMP, devido à sua robustez, flexibilidade.

O CMIP é orientado à conexão e exige um serviço confiável de transporte (TCP - *Transmission Control Protocol*, por exemplo), enquanto o SNMP requer um serviço simples de datagrama (UDP - *User Datagram Protocol*).

## 3.5 Requisitos informais da aplicação Gateway CMIP-SNMP

Como comentado na introdução deste capítulo, esta seção apresenta os requisitos informais da aplicação Gateway. Entende-se por Gateway, como a aplicação Gateway CMIP-SNMP [Oliv 96] para gerenciamento de redes, aqui especificada e verificada.

### *3.5.1 Funcionalidade do Gateway*

Uma aplicação Gateway tem como objetivo implementar a tradução entre diferentes protocolos de comunicação. O Gateway abordado neste trabalho visa à tradução (fazendo interface com o ACSE, o SMASE e o TCP/IP) entre os protocolos

CMIP e SNMP. A escolha destes dois modelos de gerenciamento justifica-se pelo seu uso mais difundido. Esta facilidade permite que um gerente no domínio OSI possa gerenciar, sem alterações, nós Internet (Figura 4.3). Nesta seção, são comentados os módulos de tradução e de controle e as fases de uma operação de gerenciamento.

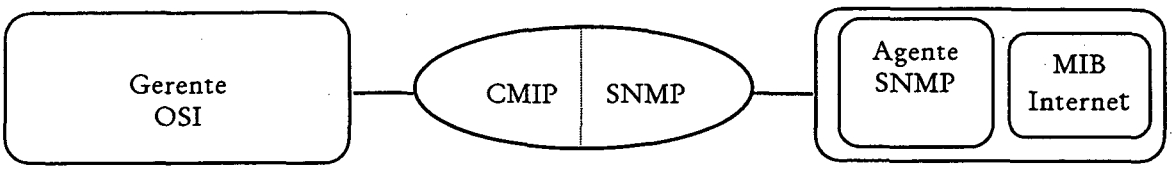


Figura 3.3 - Aplicação Gateway

O Gateway é composto por dois módulos. O primeiro realiza a tradução das PDUs e ao segundo cabe o controle destas PDUs. O Tradutor de PDUs é responsável pela transformação das PDUs para o formato reconhecido pelo ambiente destino. No ambiente destino, pode estar um gerente ou agente tanto OSI quanto SNMP; depende da operação realizada. Para realizar esta tradução, é preciso o conhecimento do formato das PDUs de ambos os modelos (este formato pode ser encontrado definido em ASN.1 em [Oliv 96]). A Unidade de Controle garante a consistência das informações que são repassadas pelo Gateway.

Além destes dois módulos definidos, é necessário realizar a tradução das MIBs. Para isso deve-se atualizar a MIB OSI, a estrutura de identificação dos agentes e a estrutura de mapeamento (estas estruturas são apresentadas adiante). Entretanto, esta tarefa deve ser realizada manualmente, pois não é implementada pelo Gateway.

Durante a execução de uma operação de gerenciamento, três fases são observadas. A primeira é o estabelecimento da associação entre o processo *gerente* e o processo *agente*. A definição para agente e gerente de gerenciamento é apresentada no início deste capítulo.

A segunda fase envolve a troca e execução das operações de gerenciamento. A última fase consiste na liberação da associação. Como no modelo SNMP não ocorre associação, ao contrário do modelo OSI, o Gateway fornece a função de associação que

possibilita a execução de operações de gerenciamento requisitadas por um gerente OSI sobre recursos SNMP.

### *3.5.2 Estrutura de Mapeamento*

O Gateway realiza o mapeamento da informação de gerenciamento do modelo de informação OSI para o modelo de informação Internet, bem como no sentido contrário. Para isso, considera os aspectos relativos ao mapeamento do nome e o mapeamento do serviço.

Através do mapeamento do nome é que uma determinada informação de gerenciamento na MIB Internet é referenciada, usando para isso o nome OSI especificado. A estrutura usada para o mapeamento do nome é apresentada mais a frente, juntamente com a descrição das atividades da aplicação.

O mapeamento do serviço fornecido pelo Gateway trata das diferenças dos serviços fornecidos pelos dois protocolos, CMIP e SNMP. Estes serviços são mapeados, levando em consideração as funcionalidades de escopo, filtro e sincronização que não tem contrapartida no modelo Internet. Deste modo, uma operação no lado OSI pode resultar em uma ou mais operações no lado Internet. Este é o mapeamento funcional que é definido a seguir.

### *3.5.3 Mapeamento Funcional*

O Gateway permite que os serviços fornecidos por um domínio sejam traduzidos para o outro através do mapeamento funcional. Um dos principais problemas referente a esta tarefa tem base na maior complexidade do modelo OSI. Isto dificulta o mapeamento das funções disponíveis no modelo OSI para o modelo Internet que é mais simples. O tamanho do pacote, limitado no protocolo SNMP, também exige tratamento no processo de mapeamento.

A transformação funcional pode ser realizada de duas formas. Na primeira, uma operação simples pode resultar em uma ou mais operações no ambiente destino. Na segunda, várias operações podem ser mapeadas para uma simples operação no ambiente destino.

Uma das principais causas das diferenças nas operações estão baseadas nas três funcionalidades que são definidas no modelo OSI e que não possuem contrapartida no modelo Internet. São elas: escopo, filtro e sincronização. Estas funções adicionam complexidade ao mapeamento para o domínio Internet. Deste modo, sua execução deve estar prevista no mapeamento funcional. Estas funções podem ser aplicadas individualmente, ou combinadas. Por exemplo, aplicando filtro nos objetos selecionados pelo escopo, duas funcionalidades são combinadas.

A seguir, são apresentados os aspectos funcionais do Gateway para estas funcionalidades. Na seção que descreve as atividades da aplicação, estas funções também são abordadas.

**Escopo:** Quando um valor é especificado no parâmetro *scope*, múltiplos objetos podem ser selecionados. Deste modo, a operação requisitada deve ser executada sobre este conjunto de objetos selecionados. No domínio Internet, esta operação pode ser mapeada pela repetição da operação requisitada para cada um dos objetos.

No escopo, são possíveis as seguintes escolhas: somente o objeto base, *n* níveis subordinados ao objeto base, ou o objeto base e todos os seus subordinados. No primeiro caso, o Gateway realiza o mapeamento da função escopo como uma simples requisição GET para recuperar o objeto base. No segundo caso, o Gateway deve percorrer a MIB e retornar com todos os objetos gerenciados que estão no nível definido pelo objeto especificado. Para isso, o Gateway usa a operação *GetNext* para extrair a informação sobre todos os objetos no nível especificado. A estrutura de mapeamento dos objetos já apresenta o nível, justamente para o uso na função escopo. No último caso, o Gateway extrairá informação de todos os objetos definidos na MIB abaixo do objeto especificado, também usando a função *GetNext*, identificando o objeto através de seu OID (*Object Identifier*).

**Filtro:** O filtro especifica condições que devem ser satisfeitas pelo objeto para que uma operação possa ser executada sobre ele. Ou seja, o filtro impõe restrições para um objeto selecionado, ou para múltiplos objetos selecionados pela função escopo. O escopo usa a

hierarquia de objetos na árvore de instâncias para selecionar um objeto, já a função filtro usa o estado de uma instância de objeto.

Quando recebida uma requisição do lado OSI, os filtros especificados no parâmetro *filter* são armazenados pelo Gateway. Após a seleção dos objetos, as condições armazenadas são aplicadas sobre os objetos e a operação requisitada é, então, executada somente sobre aqueles objetos que atendem às condições do filtro.

**Sincronização:** Existem dois tipos de sincronização, atômica ou de melhor esforço. Na sincronização atômica, a operação deve obter sucesso em todos os objetos selecionados, caso contrário, deve retornar ao estado anterior. Já na sincronização de melhor esforço, a operação é executada em tantos objetos selecionados quanto possível, retornando uma mensagem de erro referente ao objeto sobre o qual não foi possível executar a operação.

No caso do Gateway, quando o gerente requer atualização de múltiplos objetos na linha de uma tabela, deve permitir apenas sincronização atômica. Porém, se o gerente requer uma operação em vários objetos escalares, então o Gateway pode implementá-las tanto como sincronização atômica ou de melhor esforço. A função de sincronização é melhor apresentada quando descrita a operação SET, nas atividades da aplicação.

### *3.5.4 Tamanho do pacote*

No protocolo CMIP, o tamanho do pacote não é limitado. Já o protocolo SNMP usa o protocolo de transporte UDP que tem restrições quanto ao tamanho do pacote que é transmitido. Esta restrição também adiciona complexidade à máquina de estados do Gateway. Em casos onde a quebra é necessária, o Gateway deve manter o estado de cada requisição no lado OSI, coletar todas as respostas do domínio Internet e formatá-las numa única resposta que é enviada ao gerente OSI.

### *3.5.5 Operações de gerenciamento da aplicação*

O Gateway não faz o mapeamento, para o lado Internet, de todas as operações ou atividades fornecidas pelo modelo OSI. A seguir, estão descritas as operações ou atividades cuja tradução é implementada pela aplicação Gateway.

**Associação:** Na execução de uma operação de gerenciamento, inicialmente, o Gateway recebe uma requisição do gerente. Então, o Gateway deve estabelecer uma associação do processo de aplicação gerente com o processo de aplicação agente. O Gateway dispõe da função G_ASSOCIATE para a execução desta operação. Esta função usa os mesmos parâmetros definidos para a função *A_ASSOCIATE_Request* do ACSE [Borg 95].

Para reconhecimento do agente, é utilizada a informação recebida pelo parâmetro CAPT (*Called AP_Title*). Este parâmetro contém a identificação do agente utilizada para identificar, sobre uma tabela, se o recurso é OSI ou SNMP. Esta tabela de identificação contém apenas os nomes dos agentes SNMP (Tabela 4.1). O processo de identificação ocorre da seguinte forma: se a informação recebida pelo parâmetro CAPT não existir na tabela de identificação, trata-se de um agente OSI. O Gateway, então, faz uma chamada de serviço fornecido pelo ACSE (*request*), confirmando ou não a associação para o gerente. Caso a informação recebida existir na tabela, o agente é SNMP.

| Identificação | Endereço IP |
|---|---|
| "Hub1" | 150.162.1.2 |

Tabela 3.1 - Tabela de identificação dos Agentes

Quando o agente é SNMP, o Gateway deve verificar se o agente está ativo ou não, enviando uma PDU *Get* com o identificador do objeto *System* que existe em todas as MIBs mantidas por agentes SNMP. Se ocorrer *timeout*, o Gateway informa ao gerente que o agente está inativo pelo envio de uma PDU (*A-Associate-Reject*), rejeitando o pedido de associação. Caso contrário, se o agente responder, a associação é confirmada (*A-Associate_confirm*), permitindo a execução de operações de gerenciamento. A confirmação, no que se refere à estrutura de mapeamento dos serviços no Gateway, ocorre através da passagem de um valor, '0' ou '1', por exemplo.

Para notificações dos agentes, também deve ocorrer associação. Neste caso, quando o agente é SNMP, o Gateway faz a tradução da informação para o formato de relatório de evento e a envia ao gerente OSI.

Depois do estabelecimento da associação, o processo de aplicação inicia a fase de emissão das operações de gerenciamento sobre os objetos gerenciados. Todas as operações recebidas serão enviadas para o agente cujo endereço IP foi identificado durante a fase de estabelecimento da associação. Nesta fase, o Gateway permite a execução das operações GET() e SET() sobre agentes SNMP.

Após a execução das operações de gerenciamento, ocorre a liberação da associação. No lado Internet, como não existe realmente uma associação, pois o Gateway apenas verifica a viabilidade da operação (confirmando ou não), a operação é simplesmente finalizada. Então, os mecanismos de controle são atualizados para não mais permitir o envio de operações de gerenciamento. Para isso, o Gateway inviabiliza o acesso do gerente ao endereço IP do agente. Já no lado OSI, a liberação da associação utiliza os serviços do ACSE.

**GET:** A operação GET é usada para recuperar valores de atributos dos objetos gerenciados. O Gateway permite a execução desta operação. Para isso, o gerente faz a chamada da função G_GET, passando os mesmos parâmetros que o M_GET definido em [X710 91].

Inicialmente, verifica-se o domínio do objeto sobre o qual se deseja executar a operação de gerenciamento (se SNMP ou OSI). Para isso, o Gateway usa a informação contida no parâmetro *Base Object Instance* (identificação do objeto) para reconhecer o objeto na estrutura de mapeamento (Tabela 4.2). Se o objeto estiver presente nesta estrutura, é SNMP, caso contrário, é OSI.

Observa-se que, caso o recurso seja SNMP, deve-se verificar o conteúdo do parâmetro *Attribute Identifier List*. Se estiver vazio, então todos os atributos da classe são mapeados para o formato SNMP. Caso contrário, somente os atributos presentes no parâmetro são mapeados.

| Nome-OSI | Nome-Internet | Nível |
|---|---|---|
| | | |

Tabela 3.2 - Estrutura de Mapeamento

Esta tabela contém a identificação dos objetos (recursos) SNMP e o identificador correspondente no ambiente OSI. Sempre que um recurso SNMP é incluso no ambiente de gerenciamento OSI (MIB-OSI), a tabela também deve ser atualizada. O trabalho de atualização da tabela não é implementado pelo Gateway, portanto não será abordado na especificação formal.

Feito o reconhecimento do objeto na estrutura (Tabela 4.2), se este pertence ao domínio OSI, o Gateway faz a chamada da função M_GET definida pelo modelo OSI, repassando ao gerente a resposta recebida.

Se o objeto pertence ao domínio SNMP, faz-se a tradução das PDUs, gerando PDUs SNMP dos atributos definidos no parâmetro *Attribute Identifier List*. Em seguida, estas PDUs são enviadas para o agente associado, a operação é executada, as respostas são traduzidas para o formato reconhecido pelo domínio OSI (CMIP) e enviadas para o gerente. No SNMP, a primitiva de resposta é o *GetResponse* para o SET e para o GET.

Até este momento, foi descrita a funcionalidade da operação G_GET. Porém esta operação pode incorporar as funções de escopo, filtro e sincronização que não possuem contrapartida no lado Internet. Como já comentado, tais funcionalidades são emuladas pelo Gateway através das operações G_GET e G_SET. A seguir, é descrito o procedimento das operações, quando especificadas as funcionalidades.

Quanto à função escopo, deve-se observar o campo NÍVEL na estrutura de mapeamento apresentada na Tabela 4.2. Este campo permite aplicar a função escopo em tempo de mapeamento de PDUs, e não após o mapeamento, como ocorre nas funções de filtro e de sincronização. Para isso, basta verificar se os objetos definidos no parâmetro *Object Instance* e *Attribute List*, estão dentro do escopo especificado no parâmetro *scope*. Assim, somente sobre os objetos que estão dentro do escopo definido é que a execução da operação de gerenciamento solicitada terá efeito.

Quando é identificada a existência de um filtro, inicialmente ocorre o mapeamento das PDUs que são enviadas ao agente associado (*GetRequest*) com os atributos contidos no parâmetro *filter*. Somente quando as respostas são recebidas é que são aplicadas as condições especificadas pelo filtro para cada PDU de resposta (pois o escopo pode selecionar mais de um objeto). Assim, o Gateway repassará para o gerente

somente aquelas respostas que satisfazem as condições impostas pelo filtro. A aplicação das condições do filtro sobre as PDUs recebidas é realizada pela unidade de controle do Gateway.

A função de sincronização, assim como o filtro, é aplicada após o mapeamento. A sincronização pode ser de dois tipos: de melhor esforço e atômica. Caso seja definida a sincronização de melhor esforço no parâmetro *synchronization*, o Gateway poderá informar ao processo gerente uma resposta, independente se esta resposta está completa ou não. No caso da sincronização atômica, a unidade de controle somente repassará ao gerente uma resposta se a operação for executada sobre todos os objetos selecionados. Caso contrário, se pelo menos uma variável não permitir a atualização requerida, nenhuma delas é modificada. O campo *error-status* e *error-index*, na PDU de resposta, indicarão o erro ocorrido.

**SET:** Esta operação, utilizada para requisitar a modificação de valores de atributos. Segue os mesmos procedimentos da operação GET, porém, permite apenas uma das variações existentes no modelo OSI. As variações são: *add values*, *remove values*, *set to default* e *replace* para a operação Set. No domínio SNMP, apenas o *replace* é permitido.

O mapeamento do nome para o objeto base e seus atributos, se SNMP, é realizado de acordo com o parâmetro *Modification List* (são mapeados apenas os atributos contidos neste parâmetro).

Se o escopo é especificado, o mapeamento é realizado da mesma forma que para a função G_GET. Cada objeto do escopo corresponde a uma PDU *SETRequest* do protocolo SNMP.

O campo *variable-bindings* é composto pelos atributos mapeados e os seus respectivos valores, definidos no parâmetro *Modification List*. Deste modo, quando usado o campo *variable-bindings*, apenas uma PDU é enviada, caso contrário, deve ser enviada uma PDU para cada atributo mapeado.

A sincronização procede da mesma forma que na operação GET. Se a sincronização for atômica e todas as variáveis (atributos) puderem ser modificadas pelos

valores especificados, um *GETResponse* é retornado para o Gateway. Esta resposta tem a mesma forma que a resposta resultante de um *SETRequest*. Caso contrário, se pelo menos uma variável não permitir a atualização requerida, nenhuma delas é modificada. Então, ocorre erro indicado na PDU de resposta através do campo *error-status* e *error-index*.

O filtro também é mapeado como na operação G_GET. Porém, quando as respostas são recebidas, um *SETRequest* é enviado para os objetos que satisfazem as condições do filtro. Este *SETRequest* possui os atributos do parâmetro *Attribute Identifier List* mapeados compondo o campo *variable-bindings*,

Se a sincronização for de melhor esforço, o processo receptor da PDU *SETRequest* quebra a PDU em várias, uma para cada variável do campo *variable-bindings*. No processo de modificação, a variável que não é modificada recebe valor NULL. Em seguida, uma PDU *GETResponse* com todas as respostas é montada e enviada ao Gateway que, por sua vez, pode identificar em quais variáveis a operação foi executada com sucesso.

**TRAPs:** Os TRAPs são informes quando da ocorrência de eventos nos objetos que podem ser recebidos pelo Gateway. Este tipo de operação permite ao agente enviar informações ao gerente sobre algumas condições previamente estabelecidas. Da mesma forma que nas outras operações, quando o Gateway recebe informações resultantes de um TRAP, deve mapeá-las em notificações CMIP. Em seguida, estas notificações, no formato de relatório de evento, são enviadas ao gerente no modo não confirmado.

Como o modelo OSI é orientado à conexão, uma associação deve estar estabelecida antes do envio da notificação traduzida para o formato CMIP. Esta situação é particularmente distinta, pois uma notificação pode ocorrer sem que uma associação esteja estabelecida. Contudo, como comentado, para que o relatório de evento gerado seja enviado para um gerente, uma associação é requerida.

## 3.6 Comentários finais

Este capítulo apresentou inicialmente alguns aspectos de gerenciamento de redes, identificando o contexto em que a aplicação Gateway está inserida, bem como sua

finalidade no gerenciamento de redes. São, então, apresentados os modelos OSI e Internet, e os requisitos informais da aplicação Gateway especificados e verificada.

De modo geral, estes requisitos são compostos pelas estruturas de mapeamento do nome, mapeamento do serviço e mapeamento das funcionalidades. As atividades desenvolvidas são: estabelecimento (G_ASSOCIATE) e encerramento (G_RELEASE ou G_ABORT) da associação, recuperação de valores de objetos (G_GET), alteração de valores de objetos (G_SET) e notificação da ocorrência de eventos nos objetos (G_TRAP).

O capítulo seguinte é dedicado à apresentação do processo de especificação LOTOS do Gateway. São apresentados os modelos abstrato e detalhado (informais), e os principais tipos de dados e processos que compõem a especificação LOTOS do Gateway. Finalmente, são discutidas as tarefas de análise da especificação LOTOS, associados ao uso de ferramentas LOTOS.

# 4. Concepção Formal do Gateway CMIP-SNMP usando LOTOS

Este capítulo apresenta o processo de especificação LOTOS e análise do Gateway. Na fase de construção da especificação, várias dúvidas foram levantadas em conseqüência da imprecisão característica do uso de uma linguagem natural na descrição dos requisitos. Durante a busca de uma interpretação precisa destes requisitos informais, medidas corretivas foram tomadas, bem como algumas decisões de projeto.

Inicialmente, a estrutura geral da aplicação Gateway é abordada visando ao processo de construção da especificação LOTOS. É, então, identificado o comportamento do modelo abstrato e o comportamento do modelo detalhado do Gateway. Em seguida, as partes estática e dinâmica da especificação detalhada são apresentadas, mostrando os principais tipos de dados e os principais processos definidos. Após, o processo de validação e verificação da especificação concebida é apresentado. São abordadas principalmente as dificuldades impostas pelo uso de tipos de dados abstratos diante da funcionalidade fornecida pelas ferramentas de software no tratamento destes.

## 4.1 Estrutura geral do Gateway

O Gateway é logicamente dividido em dois módulos, o *tradutor* e a *unidade de controle*. O tradutor é responsável pelo mapeamento do nome e do serviço e a unidade de controle visa a garantir a consistência das informações (Figura 4.1).

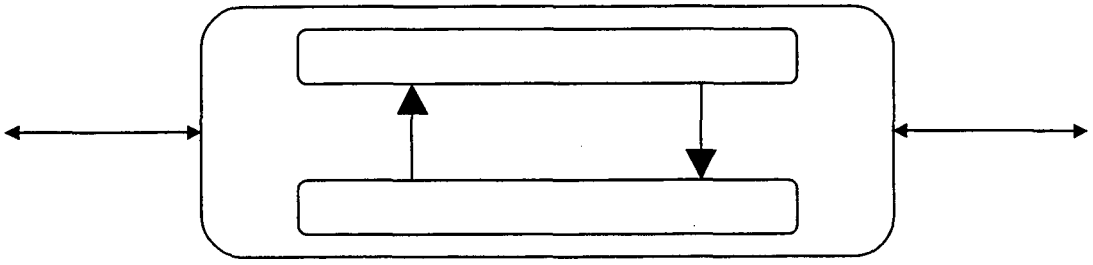

Figura 4.1 - Módulos do Gateway

O motivo pelo qual a abordagem formal não distingue estes dois módulos, baseia-se no fato de que as funções de controle estão implícitas nas funções de tradução (mapeamento). Deste modo, sua abordagem em separado tende a dificultar o processo de concepção formal do sistema.

A Figura 4.2 representa as interações possíveis da aplicação Gateway com o seu ambiente. Nas interações com os agentes de gerenciamento, se OSI, ocorre simplesmente o serviço de *pass-through*. Ou seja, o Gateway simplesmente repassa a requisição recebida à função OSI respectiva. Quando o agente é SNMP, são executadas as funções normais de tradução e mapeamento das funcionalidades.

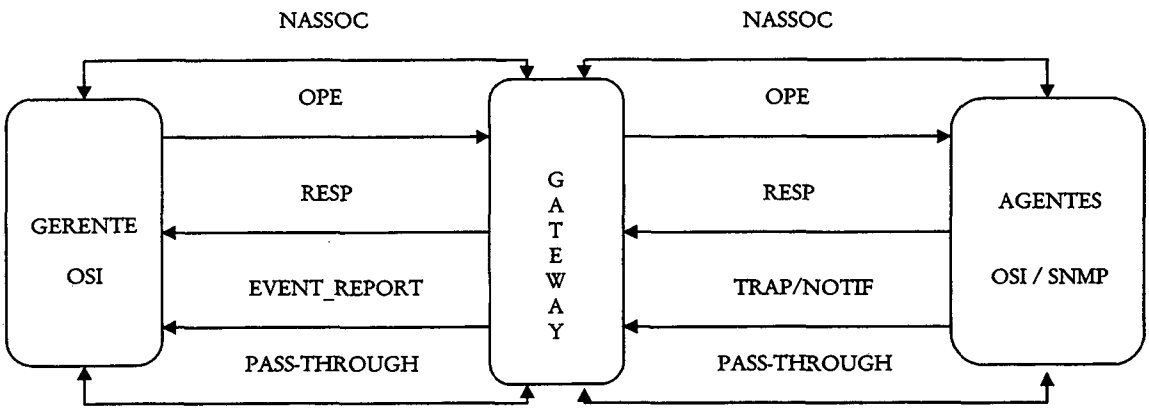


Figura 4.2 - Contexto do Gateway

Na Figura 4.2, o fluxo de informações representado pelas setas tem o seguinte significado:

- NASSOC - representa o fluxo de informações destinadas ao estabelecimento e liberação de uma associação;
- OPE - representa as requisições de operações que podem ser executadas sobre os objetos;
- RESP - respostas recebidas;
- EVENT_REP/TRAP/NOTIF - ocorrência de evento, gerando informações que são enviadas ao gerente; e

- PASS-THROUGH - repasse de informações, quando o agente é OSI (funções do Gateway não são utilizadas).

Em LOTOS, as funções do Gateway são modeladas por duas portas principais. A primeira é a porta *ped* que evolui no processo de estabelecimento das associações e a segunda é a porta *ope* que evolui no processo de execução das operações de gerenciamento. A identificação da associação em uma comunicação é representada pelo valor da variável *nassoc*, e do agente pelo valor da variável *agent*. Além desta variável, durante a execução de operações sobre os objetos, outras informações são associadas à porta *ope* com o objetivo de identificar a operação e suas variações. Tais variações são discutidas neste capítulo, quando apresentada a parte dinâmica da especificação.

A Figura 4.3 representa a ordem em que ocorrem os serviços de associação, operações/relatórios de evento e liberação da associação. Uma operação ou relatório de evento só pode ocorrer após estabelecida uma associação. Para que haja um relatório de evento, uma notificação deve ser recebida pelo agente. A associação pode ser liberada de forma abrupta (a qualquer momento) ou normal. Se o domínio do objeto é OSI, o Gateway simplesmente repassa as primitivas de serviço (*pass-through*). Se for SNMP, executa suas funções de mapeamento do nome e do serviço.

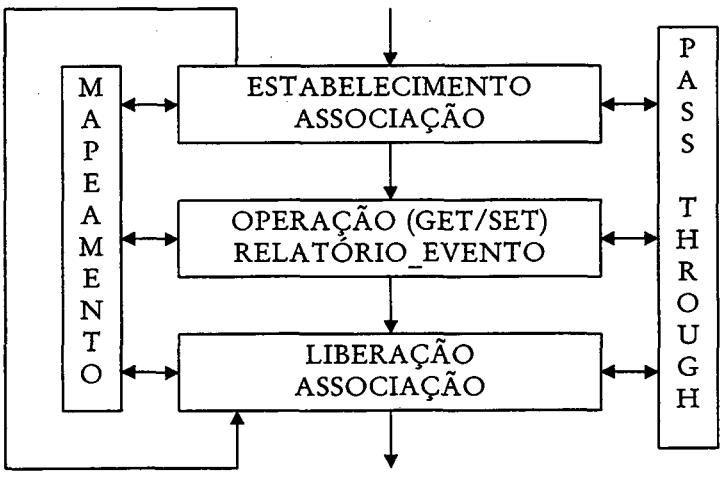


Figura 4.3 - Comportamento geral do Gateway

Nas operações G_GET e G_SET ocorre mapeamento do serviço (tradução), levando em consideração as funcionalidades de escopo, filtro e sincronização. Estas funcionalidades devem ser mapeadas em uma ou mais operações no lado SNMP, visto que não há contrapartida destas funções no modelo Internet (Figura 4.4).

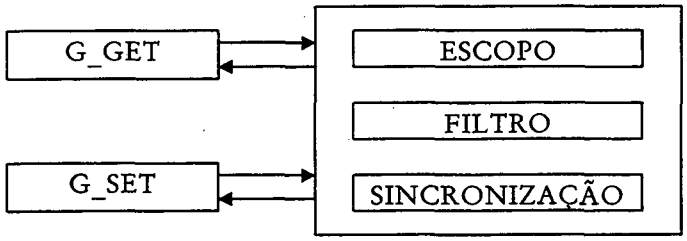


Figura 4.4 - Funcionalidades mapeadas para GET e SET

## 4.2 Modelo abstrato da aplicação

Na arquitetura apresentada na Figura 4.5, o módulo Controle de Associação faz a identificação do ambiente do agente (OSI/SNMP) com o qual o gerente deseja fazer uma associação. Se OSI, repassa as requisições recebidas ao módulo que executa os serviços de *pass-through*. Se SNMP, executa as funções de tradução e mapeamento. O módulo Controle de operação preocupa-se com o mapeamento de uma requisição recebida, realizando a chamada da respectiva operação requisitada.

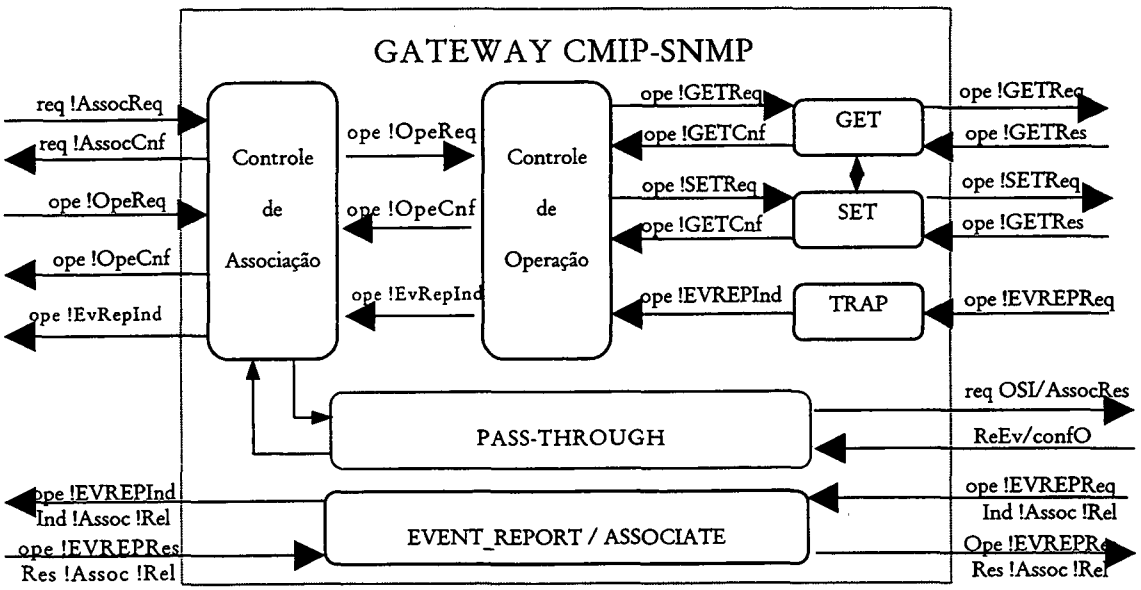


Figura 4.5 - Arquitetura da aplicação Gateway

Como o próprio nome dos módulos denota, GET e SET tratam da operação G_GET e G_SET, respectivamente. As funcionalidades de escopo, filtro e sincronização são então mapeadas e o resultado da operação é retornado.

O módulo TRAP trata da ocorrência de eventos nos objetos e envio das informações ao gerente no formato de relatório de evento, quando uma associação está estabelecida. Quando uma associação não está estabelecida, é possível a ocorrência de um relatório de evento através do módulo EVENT_REPORT que realiza a mesma função que o módulo TRAP, porém trata também do estabelecimento e encerramento da associação.

Quando o destino de uma primitiva é o ambiente OSI, o Gateway apenas repassa a primitiva à função adequada. Esta tarefa é desenvolvida pelo módulo PASS-THROUGH.

## 4.3 Modelo detalhado da aplicação

Na seção anterior, foram identificados os módulos que compõem o modelo abstrato da aplicação Gateway e suas respectivas funções. Nesta seção, o comportamento interno destes módulos é resumidamente descrito visando aos aspectos de sua descrição formal.

### Módulo Controle de Associação

Recebida uma requisição de associação, esta pode ser aceita ou rejeitada. Se rejeitada, o processo encerra com sucesso. Caso contrário, o ambiente do agente é reconhecido. Se OSI, o comportamento passa a ser representado pelo módulo PASS-THROUGH. Se SNMP, é verificada a existência do agente. Ocorrendo *timeout*, o processo encerra com sucesso, caso contrário, a associação é confirmada e as operações de gerenciamento habilitadas.

Assim que uma requisição de associação é recebida, a cada ação possível uma requisição para abortar (*abort*) a associação pode ocorrer, finalizando o processo com sucesso. Quando uma associação é confirmada, a cada ação possível uma requisição para encerramento normal (*release*) da associação também é habilitada.

### Módulo Controle de Operação

Confirmada uma associação, este módulo recebe requisições para execução de operações e instancia o processo adequado. Também controla o modo (*confirmed/noconfirmed*) da operação. Se confirmado, uma primitiva do tipo *confirm* deve ser enviada ao gerente no encerramento da operação executada.

### Módulo GET

Esta operação é sempre confirmada. Recebida uma requisição para a recuperação de valores de atributos, são verificados os parâmetros que ditam as funcionalidades de escopo, filtro e sincronização. A combinação das informações identificadas nestes parâmetros determina mapeamento a ser realizado. Se todas as funcionalidades são requeridas, a operação tem a seguinte seqüência de passos (comportamento):

1. verifica se objeto existe;
2. busca valor do objeto;
3. repete 1 e 2 até que escopo seja satisfeito;
4. recebe a resposta (*response*) com valores selecionados;
5. aplica condição de filtro para cada objeto selecionado;
6. se sincronização é de melhor esforço, confirma operação (*confirm*), enviando código de erro no lugar dos objetos sobre os quais não foi possível realizar operação;
7. se sincronização atômica, executa 8 ou 9;
8. se operação obteve sucesso em todos os objetos selecionados, confirma operação enviando os valores selecionados;
9. Se operação não obteve sucesso em pelo menos um objeto selecionado, cancela operação.

As seguintes combinações de funcionalidades são possíveis a cada operação requisitada:

- um único objeto;
- um único objeto e filtro;
- escopo;
- escopo e filtro;
- escopo e sincronização atômica;
- escopo e sincronização de melhor esforço;
- escopo, filtro e sincronização atômica;
- escopo, filtro e sincronização de melhor esforço.

Por exemplo, se apenas escopo é especificado, os passos 1, 2 e 3 são executados, em seguida a operação é confirmada.

### Módulo SET

Uma operação SET pode ser confirmada ou não confirmada. Recebida uma requisição para a alteração de valores de atributos, inicialmente uma operação GET é executada sobre o conjunto de objetos que terão seus valores alterados. Tal medida permite ao Gateway restabelecer os valores originais caso a operação SET seja cancelada. Em seguida, os mesmos passos descritos para o módulo GET são executados, porém, realizando alteração, e não recuperação de atributos. A operação SET recebe um *GETResponse* com os valores dos objetos sobre o qual a operação foi executada, da mesma forma que na operação GET.

### Módulo TRAP

Recebido um TRAP informando a ocorrência de um evento em um objeto, o valor respectivo é recuperado. Então, o Gateway realiza a tradução da informação para o formato de relatório de evento, enviando ao gerente.

### Módulo PASS_THROUGH

No modelo especificado, este módulo trata o comportamento das requisições no lado OSI para: associação, GET, SET, relatório de evento e encerramento da associação (*release/abort*).

### Módulo EVENT_REPORT

No modelo SNMP, a ocorrência de um evento em um objeto gerenciado é informada ao gerente através de um TRAP. Este gerente, então, pode tomar ou não medidas de gerenciamento. No modelo OSI, o objeto envia informações de gerenciamento ao agente através de uma notificação. O agente, então, gera um relatório de evento que é enviado à aplicação gerente. No entanto, para que um gerente OSI receba um relatório de evento, uma associação deve estar estabelecida.

Quando uma associação não está estabelecida e ocorre um evento em um objeto OSI, todas as primitivas necessárias ao envio do relatório de evento gerado ao gerente passam por este módulo. Ou seja, o módulo recebe um pedido de associação do agente e envia ao gerente. Recebe a resposta do gerente confirmando ou não a associação, enviando esta resposta ao agente. Só então, se a associação é confirmada, é que este módulo recebe o relatório de evento do agente e envia ao gerente. Em seguida, a associação é encerrada pelos mesmos meios.

Quando uma associação não está estabelecida e ocorre um evento em um objeto SNMP, esse módulo, então, recebe a informação referente ao *trap* ocorrido. Ele, então, toma a iniciativa de traduzir o *trap* para o formato de um relatório de evento e de estabelecer uma associação com o gerente OSI. Só, então, o *trap* traduzido é enviado ao gerente. Em seguida, esse módulo trata de encerrar a associação.

## 4.4 Tipos de Dados

A definição dos tipos de dados da especificação LOTOS do Gateway é composta de 18 tipos, sendo que, destes, 2 fazem parte da biblioteca fornecida em [ISO 8807], anexo A (*Annex A - Standard library of data types*). São eles: *Boolean e NaturalNumber*. Das 1850 linhas do código da especificação, 600 foram utilizadas para a definição de tipos de dados. A implementação em código C dos tipos de dados, gerada pela ferramenta CAESAR.ADT, possui 2122 linhas.

O uso da parte estática de LOTOS permite construir especificações menores e mais completas, pois torna-se possível associar um valor a um evento. Deste modo, uma mesma porta pode representar um número de eventos igual ao domínio do *sort* associado.

Por exemplo, dado um *sort SA*, onde *SA = {1,2,3}*, uma variável *nassoc* declarada pelo *sort SA* pode assumir os valores *1, 2* ou *3*.

Considerando um pedido de associação em um ambiente de gerenciamento qualquer, representa-se pela seguinte expressão:

*ped ?nassoc:SA*

A representação acima é uma escolha indeterminística entre os possíveis valores do *sort SA*. Para reescrever este exemplo sem o uso da parte de dados, cada evento deve ser especificado individualmente. Isto aumenta o tamanho e, geralmente, a complexidade da especificação.

Do mesmo modo que o uso de dados fornece maior versatilidade para o especificador, novos problemas podem ocorrer, principalmente na verificação das propriedades da especificação. Isto é justificado pelo fato de que a maioria dos pesquisadores tem concentrado as pesquisas na verificação da parte comportamental de especificações LOTOS. Assim, a disponibilidade de técnicas e ferramentas que tratam especificações LOTOS com tipos de dados é mais restrita.

Devido a algumas restrições impostas pelas ferramentas, algumas características de estruturação da linguagem LOTOS não foram utilizadas. Estes aspectos são discutidos no capítulo 8.

As definições de tipos de dados da especificação do Gateway são apresentadas aqui de forma sucinta. Importância é dada à definição do conjunto de *sorts* e respectivas operações.

Nesta seção, juntamente com a definição dos tipos de dados, também são mostradas as anotações usadas pela ferramenta CAESAR.ADT na tradução dos tipos abstratos de dados para tipos concretos (em código C).

Inicialmente, a Figura 4.6 apresenta tipo de dado que define os *sorts* usados no processo de estabelecimento e encerramento de uma associação entre um gerente OSI e um agente SNMP. Já, neste tipo observa-se o uso das anotações.

```
type assoc_primit_snmp is Boolean
sorts
assoc_anmp      (*! implementedby ADT_ASSOC_SNMP
                    comparedby ADT_CMP_ASSOC_SNMP
                    enumeratedby ADT_ENUM_ASSOC_SNMP
                    printedby ADT_PRINT_ASSOC_SNMP *),
release_snmp    (*! implementedby ADT_RELEASE_SNMP
                        ... *),
abort_snmp      (*! implementedby ADT_ABORT_SNMP
                        ... *),
assoc_primit_snmp   (*!  implementedby ADT_ASSOC_PRIMIT_SNMP
                        ... *)
```

Figura 4.6 - *Sorts* para a função de associação

As operações possíveis no processo de associação são apresentadas na Figura 4.7. Um pedido de associação pode ser aceito ou recusado, caracterizando uma escolha indeterminística. A mesma situação é encontrada quando ocorre uma requisição para o encerramento normal da associação. Se a requisição é aceita (*affirmative*), a associação é encerrada, caso contrário (*negative*) a associação é mantida.

```
...
opns
      accepted     (*!  implementedby  ACCEPTED_OPE  constructor
*),
      refused      (*! implementedby REFUSED_OPE constructor *)
                        :-> assoc_snmp

      affirmative  (*! implementedby AFFIRMATIVE constructor *),
      negative     (*! implementedby NEGATIVE constructor *)
                        : -> release_snmp

      exec_abort   (*! implementedby EXEC_ABORT constructor *)
                        : -> abort_snmp

      GAssocReq    (*! implementedby GASSOCREQ constructor *),
      GAssocCnf    (*! implementedby GASSOCCNF constructor *)
                        : assoc_snmp-> assoc_primit_snmp

      GReleaseReq  (*! implementedby GRELEASEREQ constructor *),
      GReleaseCnf  (*! implementedby GRELEASECNF constructor *)
                        : release_snmp     -> assoc_primit_snmp

      GAbortReq    (*! implementedby GABORTREQ constructor *)
                        : abort_snmp-> assoc_primit_snmp
      ...
```

Figura 4.7 - Operações para a função de associação

As operações *GAssocReq*, *GReleaseReq*, e *GAbortReq* (Figura 4.7) representam as primitivas de requisição para estabelecimento, encerramento normal e encerramento abrupto de uma associação, respectivamente. Quando as primitivas OSI de requisição são

recebidas pelo Gateway, este executa o mapeamento adequado, confirmando ou não tal operação através das primitivas *GAssocCnf* e *GReleaseCnf*. Estas primitivas representam as primitivas de confirmação do estabelecimento e encerramento normal da associação, respectivamente.

```
type operation_gw is boolean
sorts
      operat_gw     (*!  implementedby ADT_OPERAT_GW
                         comparedby ADT_CMP_OPERAT_GW
                         enumeratedby ADT_ENUM_OPERAT_GW
                         printedby ADT_PRINT_OPERAT_GW    *),
      operation_gw   (*! implementedby ADT_OPERATION_GW
                         comparedby ADT_CMP_OPERATION_GW
                         enumeratedby ADT_ENUM_OPERATION_GW
                         printedby ADT_PRINT_OPERATION_GW  *),
      operation2_gw (*! implementedby ADT_OPERATIONTWO_GW
                         comparedby ADT_CMP_OPERATIONTWO_GW
                         enumeratedby ADT_ENUM_OPERATIONTWO_GW
                         printedby ADT_PRINT_OPERATIONTWO_GW *)
```

Figura 4.8 - *Sorts* das operações de gerenciamento

A Figura 4.8 apresenta os *sorts* relativos ao tipo que define as operações de gerenciamento sobre objetos gerenciados. O *sort operation_gw* identifica as requisições para a execução de operações de gerenciamento. O *sort operation2_gw* identifica as primitivas que enviam as respostas das operações requisitadas. Este segundo *sort* também identifica as primitivas através das quais os resultados (valores) da execução destas operações são enviados ao gerente, incluindo a ocorrência de possíveis erros.

Da mesma forma que um pedido de associação pode ser aceito ou recusado, o pedido para a execução de uma operação também pode ser aceito ou não aceito.

As operações mapeadas pelo Gateway (Figura 4.9) no lado SNMP são GET, SET e EVENT-REPORT. O comportamento de cada uma delas é discutido na seção 4.3.

Quando requisitada uma operação sobre objetos OSI, o Gateway realiza serviços de PASS-THROUGH. As mesmas definições de tipos usadas nas funções SNMP também são definidas para as funções OSI. Para isso, é usada uma característica de especificação estruturada denominada renomeação de tipos de dados, demonstrada na Figura 4.10.

```
opns
      accepted     (*! implementedby ACCEPTED_OPE constructor
      *),
      refused      (*! implementedby REFUSED_OPE constructor
      *)
                        : -> operat_gw

      GGetReq      (*! implementedby GGETREQ constructor *),
      GGetNext     (*! implementedby GGETNEXT constructor *),
      GSetReq      (*! implementedby GSETREQ constructor *),
      GEvRepReq    (*! implementedby GEVREPREQ constructor *)
                        : operat_gw -> operation_gw

      GGetCnf      (*! implementedby GGETCNF constructor *),
      GSetCnf      (*! implementedby GSETCNF constructor *),
      GEvRepCnf    (*! implementedby GEVREPCNF constructor *)
                        : -> operation2_gw

      GGetRes      (*! implementedby GGETRES constructor *),
      GSetRes      (*! implementedby GSETRES constructor *),
      GEvRepRes    (*! implementedby GEVREPRES constructor *),
      GEvRepInd    (*! implementedby GEVREPIND constructor *)
                        : -> operation2_gw
      _eq_ (*! implementedby ADT_EQ_PRI_OPEa *)
                        :operat_gw,operat_gw    -> bool
```

Figura 4.9 - Operações de gerenciamento (*request,confirm,response*)

```
type              osi_ident_assoc      is
snmp_ident_assoc
      renamedby
      sortnames
            assoc_osi for assoc_snmp
            ...
      opnnames
            AAssocReq for GAssocReq
            ...
endtype
```

Figura 4.10 - *Sort signal* para identificar eventos gerais

A Figura 4.11 mostra as definições de dados que possibilitam manter controle das operações de gerenciamento. Este tipo de dado permite o mapeamento das funções adicionais de escopo, filtro e sincronização.

```
type stcodes is boolean
sorts
      modo  (*!   implementedby ADT_MODO
                  comparedby ADT_CMP_MODO
                  enumeratedby ADT_ENUM_MODO
                  printedby ADT_PRINT_MODO *),

      capt_base_ob_inst        ...
      result_req               ...
      stfilter                 ...
      satfilter                ...
      type_synchronization     ...
      sinc_be                  ...
      statexist                ...
opns
      confirmed (*! implementedby CONFIRMED constructor *),
      noconfirmed (*! implementedby NOCONFIRMED constructor
*)                        : -> modo
...
eqns
```

Figura 4.11 - *Sorts* para controle das funcionalidades

Devido ao uso de tipos de dados, optou-se por utilizar uma única porta de comunicação para as diferentes operações. A identificação da associação e do agente de gerenciamento são então especificadas pelo valor dos dados associados. Da mesma forma, para a identificação das operações, foi necessário definir três tipos de dados. O primeiro para identificar eventos em geral (*signal*). O segundo para identificar as diferentes variações da operação GET (*signal_get*), e um terceiro tipo para identificar as diferentes variações da operação SET (*signal_set*). A Figura 4.12 apresenta o tipo *signal_set*, exemplificando esta construção.

```
type  signal_set is
sorts
      signal_set ...
opns
      operation_set_one_filter
            (*! implementedby OPERATION_SET_ONE_FILTER constructor *),
      operation_set_one_nofilter
            (*! implementedby OPERATION_SET_ONE_NOFILTER constructor *),
                        : -> signal_set
      ...
endtype
```

Figura 4.12 - *Sort signal_set* para identificar variações da operação SET

Como visto anteriormente, quando uma operação de gerenciamento é executada no lado SNMP, o Gateway permite a seleção de múltiplos objetos. Para possibilitar a representação deste comportamento, foi definido um tipo de dado que especifica um conjunto de objetos. É permitido inserir objetos neste conjunto, remover, remover todos (*removeall*), usar operadores booleanos, e verificar se um determinado valor pertence ou não pertence ao conjunto. Esta operações são essenciais na representação do comportamento quando especificadas as funcionalidades de escopo, filtro e sincronização. Quando apresentada a parte comportamental da especificação, as possíveis variações no uso destas funcionalidades são descritas.

A Figura 4.13 apresenta as operações disponíveis definidas no tipo de dado que trata dos conjuntos.

```
type Set is Boolean, Naturalnumber
sorts
    Set       ...
opns
    {}        ...   :      -> Set
    insert    ... ,
    remove    ...   : Nat,Set -> Set
    removeall...    : Set,Set -> Set
    _eq_      ... ,
    _ne_      ...   : Set,Set -> bool
    _IsIn_    ... ,
    _NotIn_   ...   : Nat,Set -> bool
eqns
    ...
```

Figura 4.13 - Tipo de dado para conjunto de objetos

## 4.5 Comportamento

A parte dinâmica da especificação é composta de 31 processos, sendo que estes podem ser divididos em 7 módulos distintos. A seção 4.2 apresenta a arquitetura do modelo abstrato da aplicação, onde pode-se observar a comunicação entre os módulos, e destes com o ambiente. Das 1850 linhas do código da especificação, 1250 foram utilizadas para a descrição formal do comportamento. A implementação em código C da parte comportamental, gerada pela ferramenta CAESAR, possui 16.974 linhas.

Como comentado no capítulo 2, o comportamento em LOTOS é descrito por expressões de comportamento. Estas expressões representam formalmente a ordem em que os eventos podem ocorrer, ou seja, que ações são possíveis como próximas ações de um processo. A definição de processos e a instanciação de processos são similares aos procedimentos e à chamada destes procedimentos em linguagem de programação normal, respectivamente [LFHa 92].

O comportamento geral da especificação, mostrado na Figura 4.14, é representado por dois processos que sincronizam independentemente. O processo EVENT_REP representa o comportamento do sistema, quando ocorre um relatório de evento sem que uma associação esteja estabelecida. O processo G_ASSOCIATE representa o comportamento do sistema, quando uma associação é requisitada, habilitando a execução das operações de gerenciamento que podem ser requisitadas pelo gerente. Quando existe uma associação estabelecida, a ocorrência de relatórios de eventos também é permitida.

```
Specification Gateway_CMIP_SNMP[...] :noexit
      tipos de dados
behaviour
      EVENT_REP[...]
        |||
      G_ASSOCIATE[...]
where
       processos
endspec
```

Figura 4.14 - Comportamento geral do Gateway

A Figura 4.15 mostra o comportamento do Gateway, quando este recebe uma requisição de uma aplicação gerente para que uma associação seja estabelecida. Como o lado SNMP não é orientado à conexão, a aplicação Gateway realiza o mapeamento da função, devolvendo ao gerente requisitante uma resposta, aceitando ou não o estabelecimento desta associação.

O trabalho de mapeamento desenvolvido pelo Gateway faz com que o gerente, que é OSI, realize esta operação como se estivesse estabelecendo associação com um agente também OSI. Ou seja, de forma transparente.

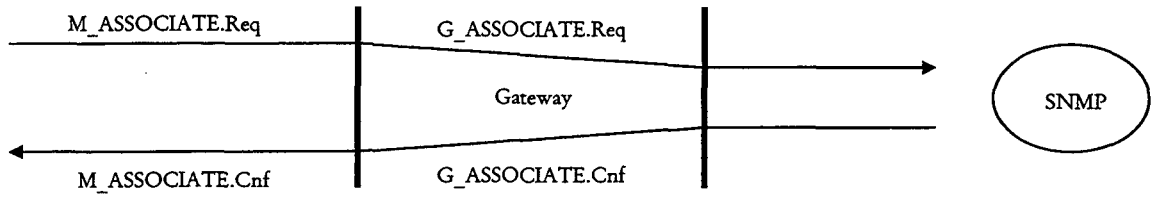


Figura 4.15 - Estabelecimento da associação com agente SNMP

A especificação considera a existência de apenas um gerente, porém permite que este gerente estabeleça associação com mais de um agente concorrentemente. Assim, a cada ocorrência de um evento, uma nova requisição para o estabelecimento de uma nova associação é habilitada.

No entanto, esta representação apresentou problemas no uso de ferramentas de apoio. Por exemplo, o simulador simbólico SMILE aceita a chamada recursiva de um processo tanto à direita como à esquerda de um operador de paralelismo (Figura 4.16). Já a ferramenta CAESAR não permite este tipo de construção. Estes problemas são abordados adiante, juntamente com a apresentação das tarefas de simulação e verificação da especificação LOTOS do Gateway.

```
process G_ASSOCIATE[...] ...
        ...
      ped ?nassoc:Nat ?pri:assoc_primit_snmp [IsGAssocReq(pri)];
          (
      G_ASSOCIATE[...]
                  |||
      ASSOC_RECOGNIZE_AG[...] (nassoc)
          )
where
      ...
endproc
```

Figura 4.16 - Comportamento do processo G_ASSOCIATE

O processo ASSOC_RECOGNIZE_AG (Figura 4.17) reconhece o ambiente do agente. Este comportamento caracteriza uma escolha indeterminística. Se o ambiente é OSI, é instanciado o processo ASSOC_PASS_TR, que representa o comportamento do Gateway, quando as operações são executadas sobre objetos OSI (realiza apenas serviços de PASS-THROUGH).

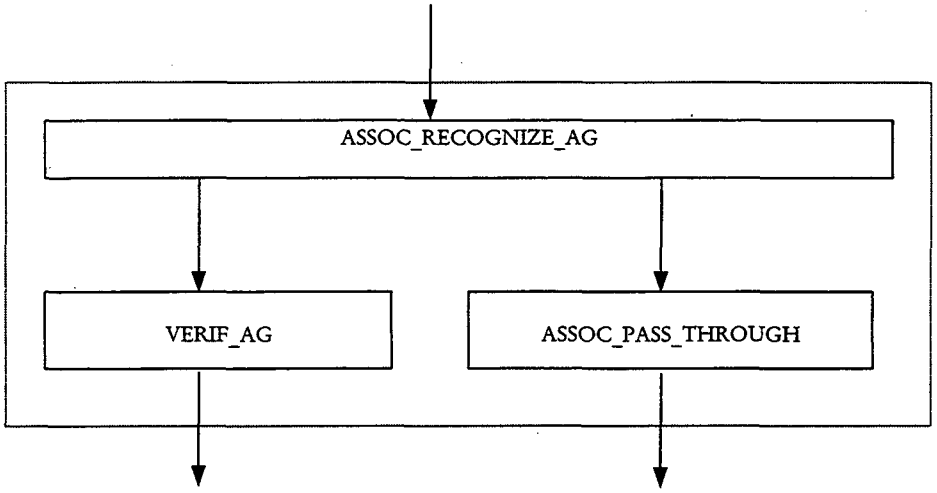


Figura 4.17 - Tratamento de uma requisição de associação

Quando o ambiente é SNMP, os processos subseqüentes realizam alguns controles sobre o comportamento do sistema. Entre eles, a verificação da existência ou inexistência do objeto (ocorrendo *timeout* no segundo caso) e a confirmação ou rejeição do pedido de associação.

```
process GW_FUNC_TRAD[...] ...
    ...
    OPER_NOTIF[...] (nassoc,agent,...)
        [>
    ABORT[...] (nassoc,agent,...)
where
    ...
endproc
```

Figura 4.18 - Instanciação das operações de gerenciamento e interrupção

Se a associação é confirmada, são habilitadas as operações de gerenciamento sobre os objetos gerenciados. Todas as operações são representadas pelo processo geral OPER_NOTIF que pode ser interrompido a qualquer momento mediante o recebimento de uma requisição ABORT (Figura 4.18 e Figura 4.19).

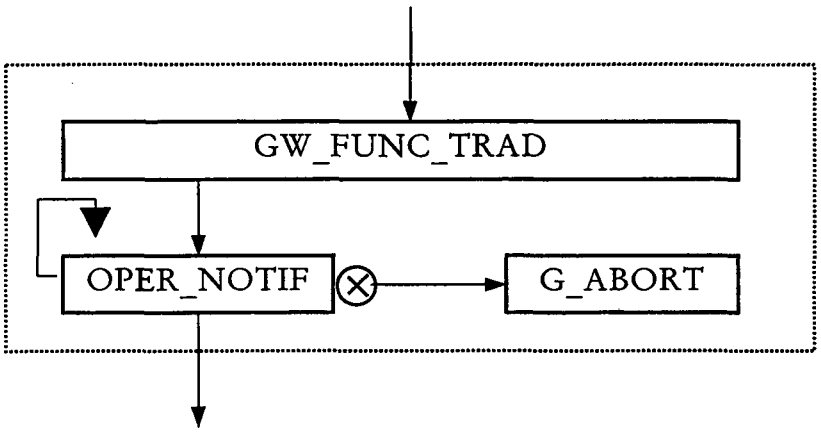


Figura 4.19 - Interrupção operações de gerenciamento

O Gateway realiza o mapeamento das operações GET, SET e TRAP sobre objetos SNMP. O processo OPER_NOTIF instancia os processos, respectivos às operações de gerenciamento, em composição paralela independente. Ou seja, qualquer operação pode evoluir independentemente (Figura 4.20).

```
process OPER_NOTIF[...] ...

            ope !G_GET !nassoc !agent !domain ;
             G_GET[...] (nassoc,agent,confirmed,SNMP)
        |||
            ope !G_SET !nassoc !agent !domain ;
             (
              G_SET[...] (nassoc,agent,confirmed,SNMP)
                []
              G_SET[...] (nassoc,agent,noconfirmed,SNMP)
             )
        |||
            ope !G_TRAP !nassoc !agent !domain ;
             G_TRAP[...] (nassoc,agent,noconfirmed,SNMP)
where
      ...
```

Figura 4.20 - Processo que instancia as operações de gerenciamento

Uma operação *get* (G_GET) é executada sempre no modo confirmado. Já a operação *set* (G_SET) pode evoluir tanto no modo confirmado, como no modo não confirmado. Esta operação ainda possui algumas particularidades destinadas ao controle dos procedimentos utilizados no processo de mapeamento das informações (discutidas adiante).

A operação G_TRAP é executada sempre no modo não confirmado. Neste caso, o objeto gerenciado envia um aviso ao agente da ocorrência de um evento e as informações geradas são então recuperadas. Neste ponto, o Gateway realiza a tradução destas informações, gerando o relatório de evento que deve ser enviado à aplicação gerente.

Quando uma operação é finalizada, esta deve ser novamente disponibilizada. Isto devido ao fato de que em uma mesma associação, múltiplas operações do mesmo tipo (SET, por exemplo) podem ser executadas. Assim, o término com sucesso do processo que representa o comportamento de cada operação habilita recursivamente a operação (Figura 4.21).

```
process G_SET[...] ...
      (
        ope !G_SET ... ?pri:operation_gw [... refused] ; exit
                []
        ope !G_SET ... ?pri:operation_gw [... accepted] ;
          (
            ope !operation_set_one_nofilter ... ;
              SET_ONE[...] (...,nofilter,...)
                []
            ope !operation_set_one_filter ... ;
              SET_ONE[...] (...,filter,...)
                []
            ope !operation_set_scope_filter ... ;
              SET_SCOPE[...] (...,filter,...)
                []
            ope !operation_set_scope_nofilter ... ;
              SET_SCOPE[...] (...,nofilter,...)
                []
            ope !operation_set_scope_sinc_at ... ;
              SET_SINC[...] (...,atomic,nofilter,...)
                []
            ope !operation_set_scope_sinc_be ... ;
              SET_SINC[...] (...,best_effort,nofilter,...)
                []
            ope !operation_set_scope_filter_sinc_at ... ;
              SET_SINC[...] (...,atomic,filter,...)
                []
            ope !operation_set_scope_filter_sinc_be ... ;
              SET_SINC[...] (...,best_effort,filter,...)
          )
      )
      >>  ope !G_SET ... ;
      (
          G_SET[...] (nassoc,agent,confirmed,SNMP)
                []
          G_SET[...] (nassoc,agent,noconfirmed,SNMP)
      )
```

Figura 4.21 - Processo principal da operação G_SET

A implementação das funções de escopo, filtro e sincronização no lado SNMP adicionam complexidade às operações G_GET e G_SET. O comportamento de ambas é relativamente semelhante, porém o G_SET executa algumas tarefas adicionais. A seguir, o comportamento da operação G_SET é descrito, abordando os detalhes relevantes. Em seguida, a operação G_GET é comentada, discutindo os pontos em que difere da operação G_SET.

Uma requisição de operação G_SET pode ser recusada ou aceita. Se recusada, o processo é instanciado recursivamente, tornando-a disponível. Se aceita, inicialmente uma operação G_GET é usada para recuperar os valores originais dos objetos. Caso a operação seja rejeitada, é possível restabelecer o estado original destes objetos.

No processo de recuperação de valores dos objetos, três situações distintas podem ser alcançadas. São elas:

- nenhum objeto é selecionado;
- um objeto é selecionado; ou
- múltiplos objetos são selecionados.

No primeiro caso, o Gateway recebe uma resposta sem nenhum valor recuperado. Se a operação está no modo confirmado, uma confirmação é então enviada ao gerente. Então, a operação é novamente habilitada (recursividade).

No segundo caso, durante a execução do G_GET, verifica-se a existência do objeto. Se não existe, retorna uma resposta sem valor selecionado e o mesmo procedimento de controle do modo (confirmado/não confirmado) é aplicado. Se o objeto existe, este objeto selecionado é submetido a condições estabelecidas (se existentes) no parâmetro filtro, podendo ser ou não ser satisfeitas.

A operação SET pode ainda ser rejeitada ou aceita após a seleção dos objetos. Se rejeitada, o valor original é restabelecido ao atributo correspondente com um novo G_SET e a resposta é enviada ao gerente. Se a operação não for rejeitada, a resposta é enviada ao gerente com os valores que foram atualizados. O procedimento de controle do modo (confirmado/não confirmado) sempre é aplicado.

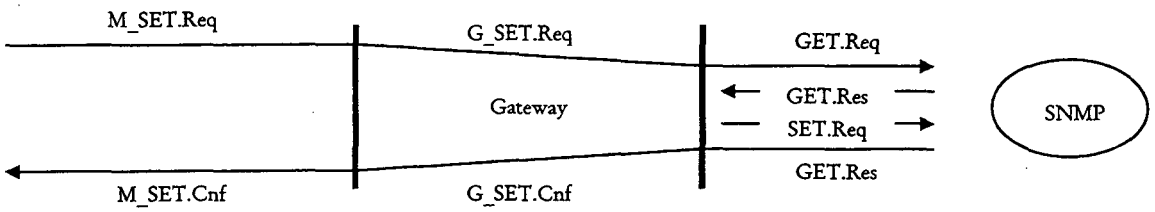


Figura 4.22 - Operação G_SET no modo confirmado

A Figura 4.22 representa uma operação de gerenciamento SET no modo confirmado, e a Figura 4.23, quando executada sobre um objeto SNMP, no modo não confirmado. Observa-se que o G_GET é, inicialmente, executado com o objetivo de buscar os valores originais dos objetos.

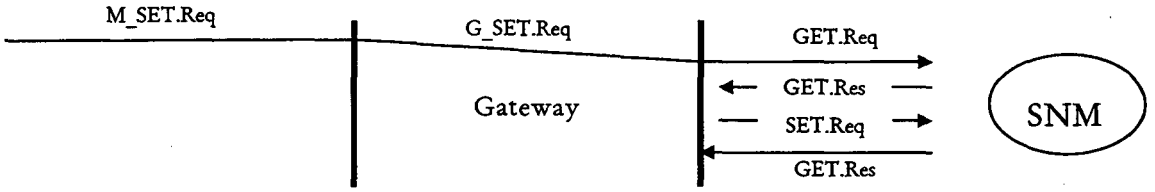


Figura 4.23 - Operação G_SET no modo não confirmado

A operação G_GET, representada na Figura 4.24, tem comportamento semelhante ao da operação G_SET. As principais diferenças são identificadas no modo e no acesso. No modo, pois uma operação G_GET é sempre confirmada. E, no acesso, porque esta operação faz apenas a recuperação de valores de um objeto e envia ao gerente. Como já comentado, no G_SET, esta recuperação é realizada inicialmente para permitir o restabelecimento dos valores originais nos objetos, caso ocorra algum problema.

A terceira operação mapeada pelo Gateway no lado SNMP é o relatório de evento. Ela é a única operação que pode ocorrer sem que uma associação esteja previamente estabelecida, como comentado anteriormente.

O Gateway tanto pode receber um relatório de evento do lado OSI, repassando a informação ao gerente, como receber informações recuperadas a partir da ocorrência de um TRAP no lado SNMP. Neste segundo caso, o Gateway realiza a tradução das

informações para o formato de relatório de evento reconhecido pelo gerente OSI, estabelece a associação (se esta não existir) e envia o relatório ao gerente (Figura 4.25).

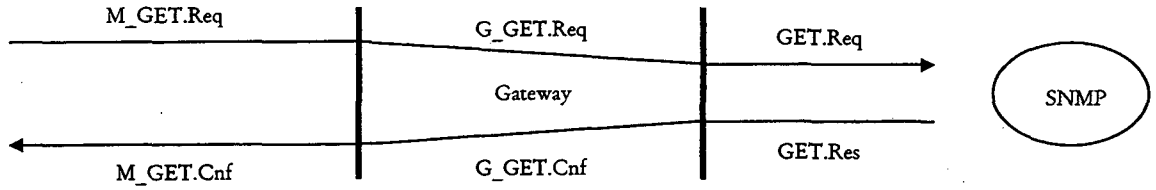


Figura 4.24 - Operação G_GET

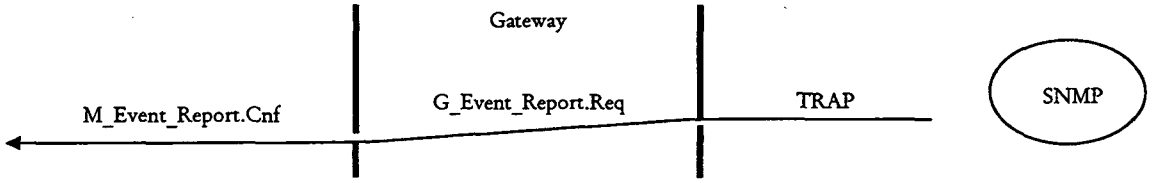


Figura 4.25 - Operação TRAP

A Figura 4.26 mostra o processo TRAP, que é instanciado pelo processo que representa o comportamento do Gateway, quando da ocorrência de eventos em objetos SNMP. Também é instanciado pelo processo que representa o comportamento do Gateway, quando da ocorrência de um evento em um objeto gerenciado OSI.

```
process TRAP[ope] ...
              [domain = SNMP] ->
                (
                     ope ...       [IsGetReq ... refused]; exit
                   []
                     ope ...       [IsGetReq ... accepted];
                           ope ...       [... GEvRepInd]; exit
                )
                     []
              [domain = OSI] ->
                (
                     ope ...       [...   refused]; exit
                     []
                     ope ...       [...   accepted];
                     (
                        ope ...    ;
                            (
                               [modo_evrep = confirmed] ->
                                  ope ... [... MEvResp]; exit
                                        []
                               [modo_evrep = noconfirmed] ->
                                  ope ... ; exit
              ...
```

Figura 4.26 - Processo que trata do comportamento do relatório de evento

Este processo realiza um escolha determinística (uso de guardas), de acordo com o valor do domínio repassado no parâmetro correspondente pelo processo instanciador. Uma requisição de envio de um relatório de evento também pode ser aceita ou recusada.

Quanto ao comportamento da aplicação, quando realizadas operações de gerenciamento sobre objetos também OSI, o procedimento adotado é baseado nas normas do CCITT [X710 91]. Abaixo é apresentado resumidamente o comportamento especificado do Gateway nestes casos.

Quando é requerida uma associação com um agente OSI, o Gateway simplesmente repassa as informações recebidas para o provedor ACSE, e ao gerente OSI, dependendo da primitiva (Figura 4.27).

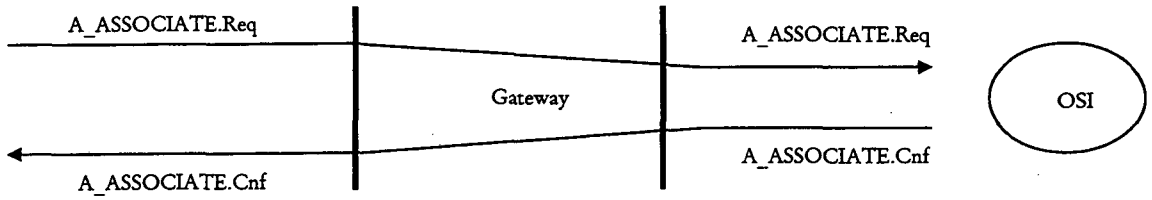


Figura 4.27 - Estabelecimento de associação com agente OSI

Quando é requerida uma operação de gerenciamento tal como GET ou SET, o Gateway faz a chamada do serviço correspondente (M_GET e M_SET) no lado OSI. Em seguida, repassa as informações recebidas ao gerente. A Figura 4.28 mostra o comportamento do Gateway, quando requerida uma operação GET sobre um agente OSI. A Figura 4.29 e a Figura 4.30 mostram o comportamento da operação SET sobre objetos OSI, no modo confirmado e não confirmado, respectivamente.

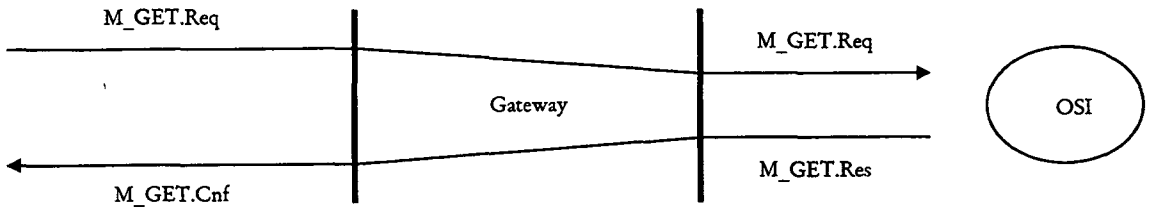


Figura 4.28 - Operação M_GET

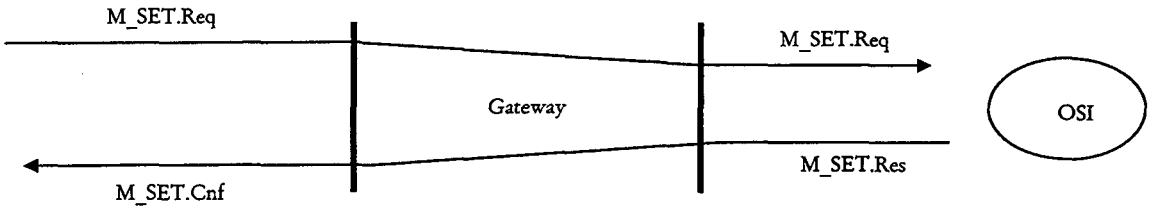


Figura 4.29 - M_SET no modo confirmado

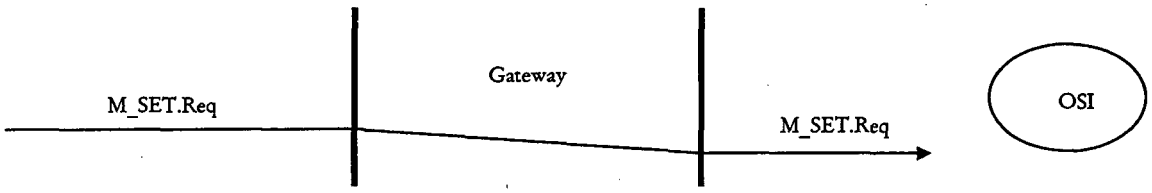


Figura 4.30 - M_SET no modo não confirmado

No caso de um relatório de evento, o Gateway também simplesmente repassa a informação recebida do agente ao gerente OSI (Figura 4.31).

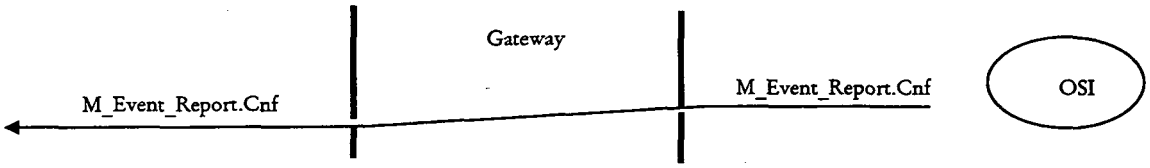


Figura 4.31 - Relatório de Evento de agente OSI

Assim que uma requisição para o estabelecimento de uma associação ocorre, nenhuma outra primitiva pode ser enviada pelo gerente respectivo, exceto a primitiva *A_ABORT.request*, até que retorne a primitiva de confirmação da associação. Quando a associação é estabelecida, qualquer operação pode ser requerida até que ocorra uma requisição de encerramento da associação.

O resultado de um pedido de encerramento normal da associação pode ser afirmativo ou negativo. No primeiro caso, a associação é encerrada, no segundo, permanece estabelecida e novas operações de gerenciamento podem ser executadas.

## 4.6 Análise da aplicação especificada

Nesta seção, são apresentados os resultados obtidos nos trabalhos de análise da especificação LOTOS do Gateway. Este trabalho pode ser dividido nas tarefas de análise sintática, análise semântica, simulação, teste e verificação. O capítulo seguinte apresenta uma avaliação das características das ferramentas LOTOS utilizadas nestas tarefas, abordando aspectos de capacidade, restrições e incompatibilidades dessas ferramentas.

### Considerações gerais sobre requisitos informais e processo de especificação, verificação e implementação

Inicialmente, não há como garantir que uma primeira representação formal de um sistema incorpora de forma correta todos os requisitos informais deste. Esta característica torna o trabalho de interpretação dos requisitos uma arte. O especificador então se mantém convicto de que a especificação representa corretamente as propriedades desejáveis do sistema apenas pela observação e pelos experimentos. Uma forma de otimizar esta tarefa seria pela identificação do maior número possível de ambigüidades ainda sobre as definições em linguagem natural do sistema. Isto pode ser alcançado atribuindo um caráter formal, tanto quanto possível, a esta definição informal.

A escolha de uma TDF e o uso de estilos [ViSc 88] ou técnicas de estruturação durante a construção de especificações, afeta diretamente a verificação. Quando a especificação é bem estruturada, o trabalho de verificação é facilitado. As tarefas de análise sintática e semântica, bem como de simulação, geralmente são mais fáceis de desenvolver, onde inclusive a disponibilidade de meios automatizados é razoável.

Já na transição de uma especificação formal abstrata para uma detalhada, é possível provar as propriedades com base no formalismo fornecido pela técnica utilizada. Esta prova pode ser realizada, por exemplo, através do estabelecimento de uma relação de bissimulação entre dois níveis de abstração de uma especificação. Sobre uma descrição formal individual também é possível realizar verificação. Para isso, existem métodos

formais, como por exemplo, a análise de alcançabilidade na busca de *deadlocks* indesejáveis.

O problema se repete na implementação final do sistema. Não se conhecem meios que possibilitem provar que o código final satisfaz todas as propriedades especificadas na descrição formal detalhada (protocolo). Para algumas TDFs, são disponíveis ferramentas de software que automatizam parte da geração de código final. No entanto, também não há como garantir que a implementação desta ferramenta está correta.

Para LOTOS, não são conhecidas ferramentas que ofereçam funcionalidade para gerar o código final de um sistema, ou mesmo parte dele, a partir de sua especificação formal.

### *4.6.1 Análise sintática e semântica*

A sintaxe de uma especificação é bastante simples de ser analisada. As ferramentas suportam bem esta tarefa, porém possuem particularidades que resultam em pontos de incompatibilidade com outras ferramentas afins.

Já durante a análise semântica, a resolução dos erros pode ser um pouco mais complexa. Assim como em linguagens de programação normal, deve-se preservar a consistência interna, pois uma alteração no código da especificação não significa necessariamente que seus efeitos serão restritos ao local onde a alteração foi realizada. Para TDFs, esta tarefa apresenta maior grau de dificuldade na medida em que o objetivo é a representação de um comportamento e não de um procedimento normal como em linguagens comuns.

Nesta fase, surgiram alguns problemas onde, para alcançar a resolução dos erros sem que se perca o comportamento pretendido, foram necessárias algumas alterações na estrutura da especificação.

### *4.6.2 Simulação*

A simulação pode ser considerada uma técnica que possibilita a realização de experiências em um computador. Estas experiências envolvem modelos lógicos que descrevem o comportamento de um sistema, considerando sua ordem temporal.

Como comentado no capítulo 2, com a simulação, é possível observar o comportamento das especificações com base no formalismo fornecido pela técnica de descrição formal utilizada. Possibilita uma avaliação interativa onde erros podem ser detectados mais facilmente. Se o modelo especificado for simples (pequeno), é possível descobrir grande parte dos erros, o que não acontece quando o modelo é grande ou complexo.

Durante a fase de simulação, pode-se dizer que ocorre uma interação com os requisitos informais do sistema. Estes requisitos ditam as regras que devem ser seguidas pelo modelo formal que, por sua vez, não permite a existência de ambigüidades. A detecção de erros diversos do sistema é então facilitada.

Para a especificação LOTOS do Gateway aqui apresentada, o trabalho de simulação foi executado sucessivas vezes para cada um dos módulos da especificação, até que o comportamento desejado fosse alcançado. Como visto, estes módulos foram definidos durante a descrição da arquitetura do sistema (seção 4.2).

É importante ressaltar que a simulação pode ser considerada como uma das principais ferramentas de auxílio para que o especificador possa especificar o comportamento desejado do sistema de acordo com a sua definição informal. Uma especificação pode ser executada sem erros, contudo isso não significa que o comportamento está de acordo com o esperado.

Durante a tarefa de simulação, aspectos relevantes que passaram despercebidos durante a fase de descrição dos requisitos informais foram detectados. Em alguns casos, para a correção desses erros, novas decisões de projeto foram tomadas.

A seguir, são listados alguns exemplos dos problemas detectados na definição informal do Gateway durante a fase de simulação. São eles:

- inicialmente, o Gateway deveria capacitar agentes OSI de gerenciar objetos definidos em uma MIB SNMP, provendo maior flexibilidade à aplicação. Porém, este requisito foi substituído por outro que objetiva verificar se um objeto SNMP, sobre o qual é requisitada uma operação, é existente ou não. Isto visa ao maior controle e consistência;

- como o modelo OSI é orientado à conexão, uma associação é exigida para que qualquer operação possa ser realizada. Porém, para a ocorrência de um relatório de evento, nos requisitos informais do Gateway não estava previsto o estabelecimento de uma associação, caso esta não exista, para permitir o envio de um relatório de evento;
- no modelo SNMP, quando executada uma operação SET, é gerada uma resposta no formato de um *Get.response*, com as mesmas informações recebidas quando executada uma operação GET normal. Porém, após a execução de uma operação SET, o Gateway envia uma operação GET com o objetivo de recuperar os valores atualizados. Como esta informação já é gerada pelo próprio SET, tal requisito caracteriza uma redundância;

Para exemplificar o trabalho de simulação desenvolvido, a seguir é apresentado, resumidamente, o comportamento alcançado para os módulos GET e SET da especificação. Para ambos os módulos da especificação, foi utilizada uma mesma seqüência de passos no desenvolvimento da simulação (método). De modo geral, este método é dividido em duas partes principais. Na primeira, a simulação foi utilizada para validar a evolução dos eventos que representam a entrada e saída dos dados para o módulo simulado. Para isso, foi utilizada uma seqüência de eventos com resultado já conhecido. Na segunda, o comportamento interno do módulo foi simulado, buscando a forma exaustiva para as diferentes variações no comportamento. Estas variações são determinadas principalmente pelo valor das variáveis utilizadas (escolhas determinísticas) e escolhas indeterminísticas.

A Figura 4.32 ilustra o comportamento da operação G_GET, onde ambos os parâmetros que determinam a funcionalidade (*scope*, *filter* e *synchronization*) possuem valor especificado. Neste caso, ocorre a seleção de múltiplos objetos sobre os quais as condições especificadas no filtro são aplicadas. A sincronização é então empregada, antes do envio dos valores requisitados ao gerente. Observa-se que, após a seleção de objetos realizada pelo escopo, a função G_GET disponível no Gateway é que aplica o filtro e a sincronização sobre este conjunto de objetos.

As demais variações do comportamento da operação G_GET seguem o mesmo princípio apresentado na Figura 4.32. Contudo, o comportamento interno é ditado pelas combinações possíveis das funcionalidades.

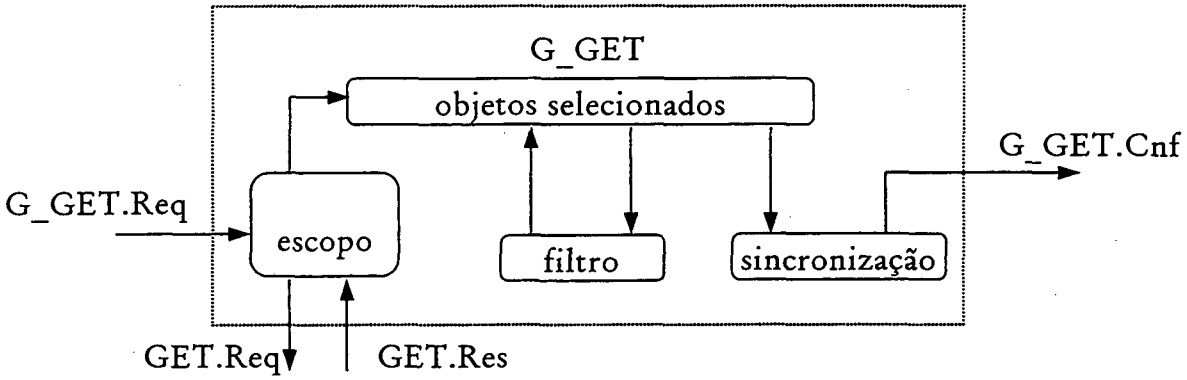


Figura 4.32 - G_GET com escopo, filtro e sincronização especificados

A Figura 4.33 ilustra o comportamento da operação G_SET, também com valor especificado para todas as funcionalidades, e no modo confirmado. Observa-se que esta operação mantém armazenados os valores originais dos objetos. Se por algum motivo a operação for cancelada, estes valores originais são restabelecidos na MIB. Para isso, a funcionalidade de sincronização (visualizada na figura 4.33) pode: ou confirmar a operação com os valores atualizados, ou executar uma nova operação SET, restabelecendo os valores originais dos objetos selecionados.

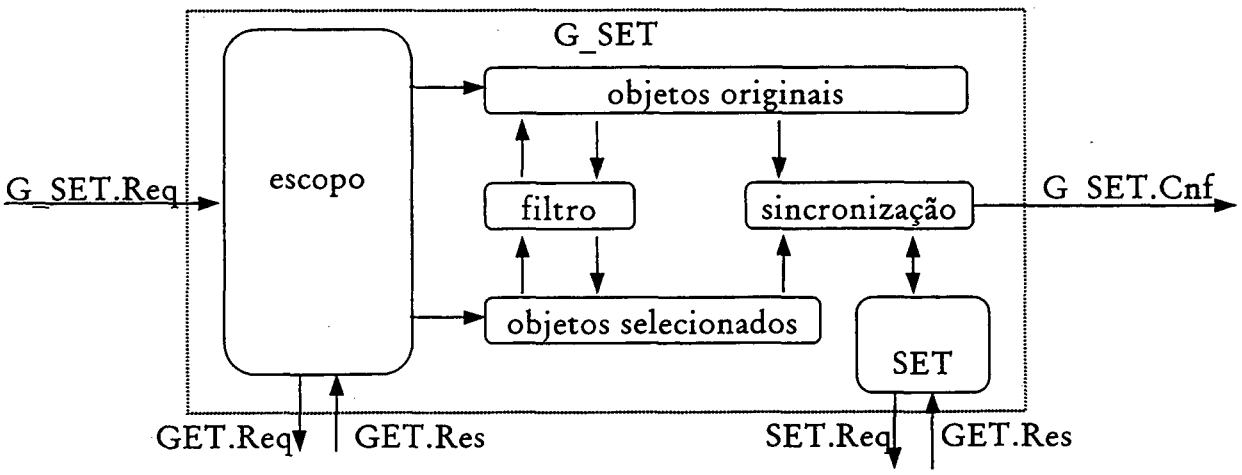


Figura 4.33 - G_SET com escopo, filtro e sincronização especificados

A aplicação Gateway foi especificada em dois níveis distintos de abstração: uma *especificação abstrata* e uma *especificação detalhada*. A *especificação abstrata* da aplicação, cujo objetivo é capturar formalmente a essência do sistema, trata do fluxo de informações recebidas e enviadas (comportamento observável). Os aspectos internos, principalmente os respectivos ao mapeamento das funcionalidades apresentadas no capítulo 4, não são tratados. Já na *especificação detalhada* da aplicação, tais aspectos internos têm seu comportamento representado.

Através de um extensivo exercício de simulação, foi realizado um trabalho de comparação entre a *especificação abstrata* do sistema (serviço) e da sua *especificação detalhada* (protocolo). E, após uma análise quase exaustiva, ficou evidenciado que o comportamento da especificação abstrata é equivalente ao comportamento da especificação detalhada. A verificação de equivalência entre o serviço e o protocolo não foi possível de ser verificada globalmente. Porém, se conseguiu o estabelecimento de uma relação formal de equivalência para alguns módulos. Resultados desta tarefa são apresentados a seguir.

A principal ferramenta utilizada no processo de simulação da aplicação especificada foi o simulador simbólico SMILE. Esta é uma das ferramentas mais robustas destinadas a tarefa de simulação de especificações que utilizam LOTOS completo. O sistema também foi simulado pela ferramenta CAESAR/Aldébaran. Esta segunda ferramenta exige a compilação em separado dos tipos de dados da especificação. O conjunto de ferramentas CAESAR/Aldébaran foi utilizada principalmente no processo de verificação da especificação.

### *4.6.3 Verificação*

Inicialmente, são comentados alguns problemas existentes na verificação de (quaisquer) especificações LOTOS que utilizam LOTOS completo. Em seguida, o processo de verificação da especificação LOTOS do Gateway é apresentado e discutido à luz desses problemas.

**Problemas na verificação de especificações com LOTOS completo:** A verificação de especificações envolvendo LOTOS completo é muito mais complexa comparada à verificação de especificações com LOTOS básico. Quando apenas a parte dinâmica está

envolvida, torna-se mais fácil entender o comportamento especificado. Também, uma variedade de técnicas, métodos e ferramentas são disponíveis neste caso.

Como a parte de tipos de dados na descrição de sistemas em LOTOS vem sendo mais utilizada, algumas ferramentas passaram a permitir também a análise de especificações com dados. Para a análise sintática e semântica, e simulação, existem ferramentas que suportam bem tais tarefas. No entanto, a tarefa de verificação impõe um grau de dificuldade mais acentuado. Além disso, as abordagens utilizadas para verificação em LOTOS básico podem não ser apropriadas para uso com especificações em LOTOS completo [kirk 94]. Contudo, elas são utilizadas.

Para exemplificar a adequabilidade dos métodos utilizados, pode-se citar a equivalência entre dois níveis distintos de abstração de uma especificação. Para que as ações sincronizem em uma relação de bissimulação, os nomes das portas e os valores dos dados associados à esta porta devem ser iguais. Como as variáveis são declaradas em pontos variados da especificação, geralmente na medida da necessidade, é muito complexo tratar os dados associados às portas na busca de uma equivalência. Esta dificuldade compromete inclusive partes do processo de simulação. Por exemplo, torna-se inviável simular o comportamento de um processo em particular, se este depende de variáveis declaradas em outros processos em ordem hierárquica superior.

A literatura apresenta algumas abordagens para a tradução de especificações em LOTOS completo para especificações em LOTOS básico. Esta abordagem foi investigada no projeto LOTOSPhere e implementada no ambiente de ferramentas LITE. Esta tradução consiste de transformações que visam a preservar a correção [Bolo 92] das especificações. Porém, as propriedades inerentes à representação com o uso de dados devem ser sujeitas a representação com LOTOS básico, o que nem sempre ocorre.

### O processo de verificação

Assim como para a simulação, no processo de verificação da especificação LOTOS do Gateway foram utilizadas algumas das técnicas e métodos suportadas por ferramentas automatizadas. Como já comentado, a verificação manual de especificações extensas é

inviável devido, principalmente, ao elevado gasto de tempo e maior possibilidade de ocorrência de erros.

As ferramentas utilizadas na verificação fazem parte do ambiente CAESAR/Aldebaran e do ambiente MiniLite. Ambos os ambientes incorporam ferramentas para variadas finalidades, contudo, mantêm algumas incompatibilidades.

No processo de verificação, foram abordadas as seguintes propriedades:

- verificação da equivalência entre níveis de abstração diferentes da especificação, utilizando para isso o estabelecimento de uma relação de bissimulação;
- propriedade de segurança (*safety*);
- propriedade de vivacidade (*liveness*); e
- verificação de seqüências pré-determinadas de eventos através de testes.

A seguir, são abordadas, isoladamente, cada uma das tarefas que envolveram o trabalho de verificação da especificação LOTOS do Gateway. São abordadas, também, as principais dificuldades e restrições encontradas neste processo.

A verificação completa de relações de bissimulação entre a *especificação abstrata* do sistema e da sua *especificação detalhada* não foi possível devido á vários fatores. Inicialmente, pela inexistência de ferramentas realmente robustas para verificação de especificações que utilizam tipos de dados. As ferramentas existentes impõem restrições que impossibilitaram a execução de algumas tarefas.

Para este trabalho, foi utilizado, principalmente, o conjunto de ferramentas CAESAR/Aldébaran (ou CADP). Inicialmente, no uso deste conjunto de ferramentas, foram encontradas dificuldades na geração de tipos de dados concretos (em código C) a partir dos tipos abstratos da especificação. Também, foi necessário reduzir o conjunto de valores dos *sorts*, visando a conter a explosão de estados.

Porém, o problema maior que impediu o uso da ferramenta no estabelecimento de relações de equivalência, foi a impossibilidade de limitar e definir os valores do *sort* utilizado na recuperação de valores de objetos gerenciados. Este recurso é necessário para se representar o comportamento das operações G_GET e G_SET. Estas operações

incorporam as principais funcionalidades do Gateway, deste modo, também compõem a maior parte da especificação.

Entende-se por limitar e definir os valores de um *sort* como o reconhecimento do conjunto de valores possíveis para este *sort*, que deve ser finito. Esta limitação é necessária para todos os *sorts* da especificação, pois o LTS (*Labeled Transition Systems*) gerado pela ferramenta CAESAR não permite declaração de valor, mas apenas oferta de valor. Por exemplo, sendo *g?x:bool*, o sistema de transições rotuladas deste evento será expandido para *g!false* e *g!true*. Como a ferramenta Aldébaran não consegue realizar a verificação com base em LTS infinitos, o conjunto de valores dos *sorts* deve ser finito.

O *sort* utilizado na recuperação de valores de objetos gerenciados, cuja limitação ou definição não foi possível, tem seu conjunto de valores variável. Ou seja, o número de elementos é indeterminado.

Este problema poderia ser resolvido com a definição do seu conjunto de valores. Contudo isto comprometeria o comportamento da especificação diante das propriedades desejáveis. Neste *sort,* é permitido inserir, remover e testar o conjunto de valores em diferentes momentos durante a evolução do comportamento da especificação. O comportamento subseqüente a um evento que envolve este *sort,* é dependente da operação sobre ele realizada. Por isso, tanto os valores quanto o número de elementos deste *sort* que define um conjunto, deve ser variável.

Uma alternativa para contornar este problema seria a definição das possíveis combinações de valores para o *sort*. Contudo, isto resultaria em uma profunda alteração no código LOTOS da especificação e no aumento considerável de seu tamanho, pois cada caso deveria ser tratado separadamente. Outrossim, desta forma não haveria meios de executar as operações de controle sobre cada objeto gerenciado selecionado. São estas operações que decidem se este objeto deve ou não se inserido no conjunto definido pelo *sort* em questão. Outra conseqüência seria a perda das vantagens fornecidas pelo uso dos dados. Essas conseqüências são suficientes para justificar a inviabilidade desta possível solução.

Outra alternativa seria interferir no código C gerado a partir dos tipos de dados da especificação pela ferramenta CAESAR.ADT. Inicialmente, esta alternativa foge aos objetivos do uso de uma ferramenta que é de automatizar tais tarefas. Também não há

garantias de que seja possível obter sucesso, além de exigir conhecimento mais aprofundado, inclusive de aspectos internos da ferramenta. Concluindo, optou-se pela comparação dos comportamentos via simulação, na medida do possível, ao invés da verificação da equivalência.

Apesar das dificuldades descritas acima, foi possível estabelecer a relação de *equivalência fraca* entre a *especificação abstrata* e a *especificação detalhada* para os módulos do sistema que não utilizam o *sort* variável. Para estes módulos, o sistema de transições rotuladas pode ser gerado por completo. Os módulos são os seguintes: controle de associação, controle de operação, TRAP, PASS-THROUGH e EVENT_REPORT. Estes módulos podem ser melhor visualizados na arquitetura apresentada no início deste capítulo.

Como já comentado, o estabelecimento de uma relação de equivalência entre especificações que utilizam dados é mais complexa, comparado à especificações que utilizam LOTOS básico. Para que as ações combinem em uma relação de bissimulação, os nomes das portas e os valores dos dados da transição também precisam combinar.

No entanto, durante o processo de descrição formal, geralmente o especificador declara variáveis dos *sorts* na medida da necessidade. Então, quando uma variável é declarada associada a uma porta oculta, na especificação abstrata não é possível simplesmente substituí-la por um evento interno *i*. Posteriormente, esta variável pode ser oferecida combinada a uma porta observável, o que requer sua declaração anterior.

```
hide oc_gate in
...
      oc_gate ?x:sort1
      ...
      oc_gate ?y:sort2
      ...
      gate !x !y
...
```

Figura 4.34 - Ilustração do uso de dados

Por exemplo, observando a Figura 4.34, se uma ou ambas as ocorrências da porta oculta `oc_gate` for substituída por uma ação interna *i*, a ação na porta `gate` torna-se inválida. Em especificações complexas, isto pode gerar situações difíceis de tratar.

Esta dificuldade pode ser sentida no estabelecimento da equivalência entre o nível abstrato e o nível detalhado de alguns dos módulos da especificação do Gateway, como comentado anteriormente. A maneira mais simples para resolver os problemas encontrados foi a utilização de portas auxiliares. Ou seja, quando ocorrida a necessidade de se estabelecer uma relação envolvendo um evento que utiliza uma variável cuja declaração está associada à uma porta oculta, dois passos são adotados. Primeiramente, é feita uma tentativa de contornar o problema, associando a declaração desta variável a uma outra porta que não seja oculta. Caso esta medida seja inviável, então usa-se uma nova porta que deve ser observável e sem dados associados. Esta é a porta auxiliar que deve ser inserida em ambos os níveis de abstração da especificação. A relação de equivalência é, então, estabelecida sobre esta porta auxiliar.

Existem outras abordagens para a verificação de especificações com LOTOS completo, como por exemplo, a transformação destas especificações para outras correspondentes em LOTOS básico. Esta abordagem foi utilizada no projeto LOTOSPhere [Bolo 92] (como comentado anteriormente). O objetivo é simplificar a verificação, porém só é justificável se existe a possibilidade de preservar as propriedades originais. Para a especificação aqui proposta, novamente esbarra-se no problema do *sort* utilizado que define o conjunto de objetos. Não há como representar em LOTOS básico um conjunto cujo número de componentes é variável.

O ambiente MiniLite disponibiliza a ferramenta AUTO para a verificação de especificações em LOTOS básico, ou resultantes do processo de transformação para LOTOS básico.

Em resumo, o uso de dados fornece maior poder ao especificador que assim torna-se mais capacitado a conceber especificações LOTOS menores, com maior representatividade e maior agilidade. Por outro lado, torna qualquer tarefa de validação e verificação mais complexa. Principalmente para verificação, devido à inexistência de ferramentas robustas o suficiente.

Quanto à verificação de propriedades de segurança (*safety*) [Pehr 88], foi possível alcançar sucesso no uso de ferramentas afins. A propriedade de segurança especifica somente o que pode e o que não pode acontecer, mas não requer que nada aconteça. Com a ferramenta CAESAR foi possível verificar esta propriedade, apesar do problema

ocorrido com a relação de equivalência. A diferença neste caso, é que a detecção de deadlocks é realizada sobre uma rede de Petri interpretada, que também é gerada, e não sobre o LTS. A própria natureza da propriedade facilita sua verificação. A rede de Petri é considerada um componente interno do conjunto de ferramentas, pois o usuário não mantém interação direta com esta parte. Já o LTS apresenta uma estrutura acessível. Por exemplo, o próprio usuário pode gerar manualmente um LTS e utilizar as ferramentas no tratamento deste.

Com o uso da ferramenta CAESAR, não foi detectado nenhum deadlock na especificação LOTOS do Gateway.

Na execução desta verificação, algumas interações com o usuário são requeridas, entre elas, o fornecimento do tamanho desejado de uma tabela a ser gerada. Para tabelas com tamanho acima de 200, a ferramenta acusou memória insuficiente. O recurso utilizado foi uma estação Sun SPARC 10, com 64 Mb de RAM e 2.8 Gb de disco. Todos os trabalhos foram desenvolvidos neste equipamento.

Para gerar um LTS com 69064 transições e 36531 estados, o CAESAR usou 11 segundos. Para realizar a redução deste LTS por equivalência observacional, foram necessários 8 minutos. Deve-se observar que este número de transições é pequeno. Um sistema grande pode chegar facilmente a 500.000 transições. Comparando os tempos gastos, verifica-se que o tratamento de LTS é um processo demasiadamente demorado para uso em sistemas mais extensos.

O uso de dados compromete ainda mais a explosão de estados. Durante a verificação do Gateway, uma simples adição de uma declaração de variável associada a uma porta já com uma outra declaração e uma oferta de valor, provocou um salto de 34720 transições para 74811 transições no LTS.

Propriedade de vivacidade (*liveness*) foi verificada através da ferramenta LOLA. Esta propriedade assume que eventos desejáveis eventualmente possam acontecer. Essa ferramenta permite definir uma profundidade determinada na árvore da especificação para a verificação da propriedade desejada. Fazendo uso desta facilidade, foi realizada a verificação até a profundidade 18, sendo que nenhuma situação de deadlock foi detectada. Para profundidades maiores e para a exploração exaustiva, a ferramenta acusou memória insuficiente do recurso utilizado.

Tanto a verificação da propriedade de segurança quanto da propriedade de vivacidade são relativamente simples de se implementar. Assim, as ferramentas utilizadas fornecem um suporte suficiente. Contudo, o número de estados deve ser finito e o recurso de processamento utilizado deve ser suficiente para realizar a exploração exaustiva dos estados. A explosão de estados é o maior problema quando utilizadas técnicas de alcançabilidade.

Testes também foram realizados sobre a especificação LOTOS para validar seqüências desejáveis de eventos. Com o uso da facilidade fornecida pelos testes, algumas seqüências de eventos fora das especificações dos propósitos do Gateway foram detectadas.

O teste é uma técnica de detecção de erros bastante trabalhosa. Para cada processo ou uma determinada hierarquia de processos da especificação, um processo de teste deve ser construído e utilizado. Para especificações grandes, o teste exaustivo é possível, contudo pode se tornar inviável diante do número de seqüências de eventos que devem ser testados.

Para realização de um teste, utiliza-se um processo auxiliar (processo de teste) que é inserido no código fonte da especificação em composição paralela com o processo que se deseja validar. Este processo de teste deve sincronizar nos eventos do processo que se deseja testar. O último evento do processo de teste não deve estar no processo da especificação. Este evento é utilizado para indicar a execução com sucesso da seqüência de eventos testada.

Existem ferramentas que automatizam o processo de teste de especificações. Para isso, o processo de teste deve ser construído manualmente de acordo com a situação em questão. Neste trabalho, foi utilizada a ferramenta LOLA para a realização dos testes sobre a especificação LOTOS do Gateway.

A ferramenta LOLA disponibiliza dois comandos de teste, o *testexpand* e o *oneexpand*. Ambos permitem determinar uma profundidade de alcance na árvore da especificação para execução de um teste. O primeiro realiza uma exploração exaustiva dos estados, fornecendo alguns resultados estatísticos, entre eles, o número de vezes que o evento de sucesso ocorreu. O segundo comando escolhe um ramo da árvore da especificação randomicamente, também verificando a ocorrência do evento de sucesso.

O resultado apresentado a seguir foi obtido a partir de um teste realizado para os primeiros eventos do processo EVENT_REP, com profundidade especificada em 10. O resultado *MAY PASS* apresentado em *Test result*, indica que o evento de sucesso ocorreu em pelo menos uma execução. Quando o evento não ocorre, o resultado *REJECT* é fornecido e se o evento de sucesso ocorre em todas as execuções, o resultado *MUST PASS* é fornecido. Para este teste, o evento de sucesso ocorreu 174 vezes. A Figura 4.35 mostra os resultados apresentados pela ferramenta LOLA neste caso.

As tabelas a seguir apresentam os resultados dos testes realizados sobre os módulos GET da especificação. Os processos de teste utilizados em composição paralela sincronizam principalmente nos eventos observáveis, com valores dos dados associados pré-determinados. Ou seja, onde no processo da especificação existe uma declaração de variável como por exemplo `?stfil:stfilter`, no processo de teste esta declaração é substituída pelo fornecimento de um valor (termo) diretamente, neste caso por `!nofilter` ou `!filter`. O valor especificado nestas variáveis determina o comportamento da respectiva hierarquia de processos.

```
Analysed states        = 473
   Generated transitions = 1046
   Duplicated states     = 0
   Deadlocks             = 0


  Test result  = MAY PASS.

                      successes = 174
                          stops = 0
                          exits = 0
                  cuts by depth = 81

lola>
```

Figura 4.35 - Resultados apresentados pela ferramenta LOLA em um teste

A Tabela 4.1 apresenta os resultados obtidos no teste da operação G_GET onde, inicialmente, são mostrados os resultados de um teste sobre o processo que realiza a seleção de apenas um objeto, com ou sem filtro. Na linha 1 e 2 desta tabela, apenas a ocorrência do primeiro evento (não observável) foi testada, porém com profundidades

diferentes. Já na linha 3, também foi testada a ocorrência do evento de resposta (*GGetRes*) da operação. Observa-se a distinção entre o número de eventos de sucesso alcançados entre as linhas 2 e 3.

| | Processo | profund. | Transições | Estados analisados | deadlocks detectados | Eventos de sucesso | resultado |
|---|---|---|---|---|---|---|---|
| 1 | GET_ONE | 10 | 112.168 | 18.505 | 0 | 1 | MAY PASS |
| 2 | GET_ONE | 11 | 718.696 | 112.575 | 0 | 25 | MAY PASS |
| 3 | GET_ONE | 11 | 707.137 | 109.711 | 0 | 1 | MAY PASS |

Tabela 4.1 - Teste de G_GET para a seleção de um objeto

Devido à explosão de estados, os testes passaram a exigir um tempo mais elevado para profundidades maiores. Optou-se então por eliminar a hierarquia de processos que representa o comportamento do módulo SET enquanto realizados os testes sobre os processos que representam o comportamento do módulo GET. Ambos os módulos (GET e SET) evoluem em paralelismo independente, por isso eliminando um deles não há comprometimento do outro para o teste. A tabela 4.2 mostra os resultados dos testes após a eliminação do módulo SET.

O mesmo teste é demonstrado na linha 3 da Tabela 4.1 e na linha 1 da Tabela 4.2. Observa-se que o número de transições geradas reduziu consideravelmente, permanecendo o evento de sucesso esperado. Na linha 2 da Tabela 4.2, também foi testado o sucesso de um terceiro evento, que é a confirmação da execução da operação (*GGetCnf*). Para este teste foram necessárias aproximadamente 2 horas de processamento, gerando 3.045.496 transições.

| | Processo | profund. | Transições | Estados analisados | deadlocks detectados | Eventos de sucesso | resultado |
|---|---|---|---|---|---|---|---|
| 1 | GET_ONE | 11 | 100.238 | 21.647 | 0 | 1 | MAY PASS |
| 2 | GET_ONE | 13 | 3.045.496 | 525.452 | 0 | 1 | MAY PASS |

Tabela 4.2 - Teste de G_GET para a seleção de um objeto (em separado)

Na Tabela 4.3, são apresentados os resultados dos testes realizados quando da seleção simples (sem a funcionalidade de sincronização) de múltiplos objetos. A linha 1

mostra o teste da ocorrência apenas do evento observável inicial. A linha 2 mostra o teste da ocorrência também do evento de resposta da operação.

| | Processo | profund. | Transições | Estados analisados | deadlocks detectados | Eventos de sucesso | resultado |
|---|---|---|---|---|---|---|---|
| 1 | GET_SCOPE | 10 | 20.733 | 5.229 | 0 | 13 | MAY PASS |
| 2 | GET_SCOPE | 11 | 95.570 | 20761 | 0 | 1 | MAY PASS |

Tabela 4.3 - Teste de G_GET para a seleção simples de múltiplos objetos

No teste exaustivo da especificação do Gateway, até a profundidade 8, o número de transições e estados gerados foi baixo. Isto se explica pelo fato de que até esta profundidade o comportamento trata apenas do reconhecimento do ambiente, estabelecimento da associação, entre outras tarefas que não possuem muitas variações. Já na execução das operações de gerenciamento, o número de combinações possíveis ditados pelos dados de controle eleva muito a explosão de estados. Isto pode ser confirmado, por exemplo, na Tabela 4.2, onde a profundidade passou de 11 para 13 e o número de transições foi multiplicado por 30, aproximadamente.

Na Tabela 4.4, são apresentados os resultados obtidos no teste do processo que trata do comportamento da especificação, quando as funcionalidades de escopo e sincronização são determinadas. Na linha 1 dessa tabela, foi testado o sucesso do primeiro evento observável, após requerida e aceita a execução de uma operação para a recuperação de valores de objeto. Na linha 2, o mesmo teste foi estabelecido, porém com profundidade 11, aumentando o número de sucessos do evento requerido. Na linha 3, um evento de resposta (observável) também foi testado, no entanto a profundidade utilizada não foi suficiente. Este teste foi repetido com profundidade 13, gerando aproximadamente 2.600.000 transições em aproximadamente 2 horas, também sem a ocorrência de sucesso. De acordo com a observação do comportamento através da simulação, este segundo evento ocorre corretamente na profundidade 17.

Durante o processo de teste, alguns erros foram detectados na especificação. Por exemplo, na ocorrência de uma primitiva *GetNext*, durante uma operação com escopo, a resposta respectiva com o objeto selecionado não estava ocorrendo. Também, quando requerida uma operação com as funcionalidades de sincronização e escopo especificadas,

a busca dos valores dos objetos estava ocorrendo internamente, quando deveria ser um evento observável de acordo com o especificado no modelo abstrato.

| | Processo | profund. | Transições | Estados analisados | deadlocks detectados | Eventos de sucesso | resultado |
|---|---|---|---|---|---|---|---|
| 1 | GET_SINC | 10 | 21.143 | 5.328 | 0 | 1 | MAY PASS |
| 2 | GET_SINC | 11 | 91.625 | 19.892 | 0 | 15 | MAY PASS |
| 3 | GET_SINC | 11 | 91.631 | 19.897 | 0 | 0 | REJECT |

Tabela 4.4 - Teste de G_GET com funcionalidade de sincronização especificada

Contudo, o teste exaustivo de uma especificação grande, ou seja, testar o sucesso de todos os eventos desejáveis, pode ser um processo demorado [BSSt 87], ou mesmo inviável.

Nesta seção (seção 4.6), foram apresentados os resultados obtidos durante o processo de análise da especificação LOTOS do Gateway. Foram desenvolvidas as tarefas de análise sintática e semântica, simulação, verificação e teste. Principalmente para a verificação, as restrições impostas pelas ferramentas devido ao uso de dados na especificação limitaram as possibilidades da aplicação completa de alguns dos métodos utilizados. Também observa-se que a verificação completa pode ser difícil de ser alcançada, principalmente pelo fato de que não há como julgar quando exatamente esta tarefa pode ser considerada suficientemente completa. A princípio, sempre é possível adicionar melhorias em um sistema.

## 4.7 Comentários finais

Neste capítulo, a estrutura geral da aplicação Gateway, sua arquitetura e os componentes relevantes (tipos e processos) da especificação LOTOS foram mostrados. A apresentação da arquitetura objetivou demonstrar, no modelo abstrato, a comunicação com o ambiente e, no modelo detalhado, o comportamento interno de cada módulo. Em seguida, foi abordado o processo de análise da especificação, onde as dificuldades encontradas foram discutidas e os resultados obtidos apresentados.

O capítulo seguinte apresenta a experiência no uso de ferramentas de software para LOTOS utilizadas nas fases de análise, simulação e verificação da especificação.

Ênfase é dada à apresentação das dificuldades, de modo geral, estabelecendo um levantamento comparativo entre necessidades, facilidades e o real suporte fornecido pelas ferramentas utilizadas em termos de capacidade e restrições.

# 5. Algumas considerações sobre as ferramentas LOTOS usadas na concepção do Gateway

Este capítulo apresenta as ferramentas computacionais de LOTOS utilizadas nas tarefas de análise, simulação e verificação realizadas sobre a especificação LOTOS do Gateway. São enfatizadas as dificuldades encontradas, principalmente ligadas à capacidade, restrições e incompatibilidades.

## 5.1 Ferramentas computacionais para LOTOS

Para o tratamento de especificações LOTOS, diversas ferramentas, para variadas finalidades, foram desenvolvidas. Como exemplo, pode-se citar o ambiente LITE [LITE 92] de ferramentas, desenvolvido durante projeto LOTOSPhere [Maña 92]. Este ambiente agrupa um conjunto de ferramentas para edição, análise, teste, simulação e verificação de especificações LOTOS.

Neste trabalho, as principais ferramentas utilizadas foram: TOPO, SMILE (versão 4.0.2), GLOTOS, LOLA (versão 2.1) [LITE 92], CAESAR (versão 5.0) [Gara 94], Aldébaran (versão 5.9) [FeMo 90], e CAESAR.ADT (versão 4.4) [Gara 89]. As quatro primeiras fazem parte do ambiente MiniLite que incorpora as ferramentas mais robustas disponíveis no ambiente LITE. As últimas três ferramentas fazem parte do conjunto CAESAR/Aldébaran.

A Tabela 6.1 apresenta uma análise comparativa da funcionalidade de algumas destas ferramentas. Em seguida, são comentadas algumas restrições e incompatibilidades de cada uma delas.

### TOPO

TOPO é um conjunto de ferramentas para análise e compilação de especificações LOTOS. O compilador TOPO permite o uso de anotações às especificações para possibilitar a adição de características de implementação, permitindo o uso de

mecanismos de controle sobre os módulos C resultantes. O Anexo B apresenta esta característica.

| | TOPO | SMILE | LOLA | GLOTOS | CAESAR/Aldébaran |
|---|---|---|---|---|---|
| LOTOS completo | Suporta | Suporta | Suporta | Suporta | Suporta |
| Compilação | Compila TODA como reescrita e comportamento como LTS | Não | Não | Não | TDA - parte de controle e de dados<br>Comportamento - LTS |
| Análise (sintática e semântica) | Sintática e semântica | Não | Não | Não | Sintática e semântica |
| Simulação | Não | Passo-a-passo, em profund., gera EFSM, instanciação de proc. em tempo de execução | Interativa, escolha na linha de comando | Não | Interativa, escolha na linha de comando |
| Verificação | Não | Não | Prop. de vivacidade (livness) | Não | Detecção de deadlock (safety), equivalência |
| Geração de código | Módulo de dados e módulo de comport. | Não | Não | Não | Tipos concretos em '.h'<br>Comport. em '.c' |
| Represent. gráfica | Não | Não | Não | Diag. de blocos | Não |
| Teste | Não | Não | Processo de teste | Não | Não |

Tabela 6.1 - Funcionalidades de algumas ferramentas

O compilador TOPO possui as seguintes restrições:

- não permite outra funcionalidade a não ser *noexit*;
- não resolve recursão não guardada na instanciação de processos, podendo ocorrer overflow em tempo de execução; e
- rendez-vous apenas com geração de valor não são suportados.

Uma expressão guardada é uma expressão de comportamento onde as substituições recursivas das instanciações sempre levam a uma expressão de comportamento onde todas as instanciações são precedidas de pelo menos uma ação. Uma instanciação recursiva ao lado de um operador de paralelismo sem uma ação precedente é uma expressão não guardada.

Neste trabalho, TOPO foi principalmente utilizado na análise sintática e semântica da especificação e na geração do formato representação comum (*.cr*) da especificação do Gateway, utilizado pelas demais ferramentas. As incompatibilidades entre TOPO e o compilador do conjunto CAESAR/Aldébaran de ferramentas são apresentadas na seção 6.2.

Mesmo com o uso de tipos de dados, tanto a análise sintática como a semântica não oferecem maiores dificuldades.

SMILE

A ferramenta SMILE é um simulador simbólico que requer o arquivo com extensão '*.cr*' gerado por TOPO. Com SMILE, é possível fazer a simulação de um único processo da especificação, simulação passo a passo e simulação exaustiva em profundidade (*depth*) pré- determinada. Também suporta tipos de dados. Como a ferramenta SMILE usa o arquivo com extensão '*.cr*' gerado por TOPO, herda também suas restrições.

De modo geral, este simulador comporta funcionalidade suficiente para a simulação de especificações LOTOS com tipos de dados. É uma das ferramentas mais robustas em sua classe.

Entre suas facilidades, a possibilidade de instanciar novos valores para os tipos de dados em tempo de processamento pode ser útil na busca de comportamentos indesejáveis. Também o modo como é realizada a simulação passo-a-passo é bastante otimizado. Durante a simulação, é possível escolher um evento anterior e dar prosseguimento à tarefa de simulação, sem que para isso tenha de ser feito um retrocesso ao estado desejado.

A dependência do simulador SMILE de um arquivo gerado por outra ferramenta poderia ser um ponto negativo. Contudo não é relevante, visto que trata-se de um ambiente integrado onde os recursos necessários são suficientemente fornecidos. Esta característica de dependência entre as ferramentas computacionais para LOTOS exige mais cuidado durante seu uso.

## LOLA

A ferramenta LOLA permite transformar, executar e testar especificações LOTOS. Não suporta exploração do conjunto de estados de um único processo. Suas principais aplicações são:

- operações de expansão - calcula simbolicamente todas as possíveis execuções de uma especificação LOTOS;
- operações de teste - calcula a resposta de uma especificação para um teste, especificado através de um processo com um evento que indica sucesso.

O suporte para teste de especificações LOTOS fornecido por esta ferramenta permite testar um único processo ou testar uma hierarquia de processos. No entanto, pode ser inviável para todas as seqüências possíveis de eventos em uma especificação, dependendo de sua complexidade e tamanho. Trata-se de uma tarefa trabalhosa, pois para cada comportamento que se deseja testar, a seqüência desejada de eventos deve ser especificada em um processo de teste.

Assim como SMILE, a ferramenta LOLA também utiliza o arquivo com extensão '*.cr*' nas operações de expansão e de teste, o que significa que as restrições da ferramenta que gera este arquivo são herdadas.

## O conjunto CAESAR/Aldebaran de ferramentas

Este conjunto de ferramentas fornece funcionalidade para análise, compilação, simulação e verificação de especificações LOTOS. Na Tabela 6.1, consta que LOTOS completo é suportado, porém não diretamente. Tipos abstratos devem ser convertidos em tipos concretos (código C) em um arquivo com extensão '*.h*'. Para isso, este conjunto de ferramentas disponibiliza a ferramenta CAESAR.ADT que implementa o algoritmo proposto em [Schn 88]. Este algoritmo traduz a parte de dados para a linguagem C, ou seja, gera tipos concretos a partir de tipos abstratos.

A tarefa de geração do código C a partir dos tipos abstratos pode se tornar um tanto complexa, pois em alguns casos, uma parcela de sua definição deve ser codificada manualmente.

Parte das funcionalidades fornecidas por este conjunto de ferramentas é desenvolvida sobre um sistema de transições rotuladas. Contudo, este LTS deve ser finito, ou seja, a especificação não pode ter comportamento infinito e os sorts definidos na parte de dados devem ter domínio igualmente finito. Esta restrição impede o uso de determinadas estruturas que poderiam facilitar o processo de especificação.

Outra restrição é identificada no tratamento de tipos que não possuem construtores. Neste caso, a ferramenta CAESAR.ADT considera tais objetos como externos. O código C correspondente deve ser fornecido pelo usuário no arquivo com extensão *'.t'* para as definições dos tipos e no arquivo com extensão *'.f'* para as definições das funções correspondentes. O arquivo '.h' gerado pela ferramenta apresenta uma lista dos objetos que não foram tratados automaticamente.

Um experimento com a compilação da parte de dados de LOTOS usando a ferramenta CAESAR.ADT é apresentado em [Gara 89].

Na tradução para código C, com o uso da ferramenta CAESAR.ADT, cada *sort* da especificação LOTOS resulta em um conjunto de operações que são divididas em duas classes: os construtores e os não-construtores. No Anexo A, são apresentadas essas classes, seu significado e como identificá-las na especificação.

Uma restrição relevante também existe para o processo de implementação em código C dos tipos de dados com o uso da ferramenta CAESAR.ADT. O tratamento de tipos que utilizam parametrização ou atualização não é suportado pela ferramenta. Caso a especificação use uma destas características (ou ambas), tais tipos devem ser completamente atualizados manualmente, para que possam ser compilados pela ferramenta.

Estas características de especificação LOTOS visam a estruturar as especificações e permitir a reusabilidade de código.

## 5.2 Incompatibilidades

Quanto à análise sintática e semântica, foram detectadas incompatibilidades entre as ferramentas TOPO e CAESAR. Enquanto TOPO permite recursividade no mesmo processo à direita ou à esquerda de um operador de paralelismo ( ||| ), a ferramenta CAESAR não permite tal recursividade nem à direita nem à esquerda de um operador de paralelismo. Quanto ao operador de habilitação (> >) e interrupção ([>), a ferramenta TOPO não detectou uma chamada recursiva à esquerda deste operador. Já CAESAR acusou tal construção como erro.

Isto justifica-se pelo fato de que especificações que não utilizam recursão sobre um operador de paralelismo, ou à esquerda de um operador de habilitação ou de preempção, permitem gerar um LTS finito. Como a ferramenta CAESAR gera apenas LTS finitos, ela não aceita as construções anteriores.

A ferramenta CAESAR não aceita recursão não guardada. A ferramenta TOPO não resolve este tipo de recursão, porém continua tratando da parte correta da expressão de comportamento. No uso de SMILE, que executa a simulação sobre arquivos gerados por TOPO, quando uma situação que envolve uma expressão não guardada é alcançada, uma mensagem de expressão não bem definida é mostrada e a simulação continua para as expressões corretas.

Após gerada a parte de dados em código C, no uso da ferramenta CAESAR, foi acusado um erro no *sort assoc_primit_snmp*, como *sort* com comportamento infinito ou *sort* muito complexo. Este tipo de erro geralmente ocorre com tipos que definem listas ou filas. Porém, neste caso, o *sort* especifica as requisições relativas aos pedidos de associação, onde cada operação possui apenas dois construtores possíveis. Este erro foi corrigido com a eliminação de uma operação sobre o *sort*. A função desta operação foi compensada com algumas alterações na parte comportamental da especificação. O simulador simbólico SMILE não acusou tal erro, executando o comportamento correto da especificação, segundo o observado durante o processo de simulação. Este problema está relacionado à impossibilidade de limitar o domínio de *sorts* infinitos, como comentado no capítulo 4 (4.6.3).

As principais incompatibilidades entre TOPO e CAESAR são identificadas na Tabela 6.2. Incompatibilidades também são identificadas entre simuladores ou outras

ferramentas de finalidade variada. No entanto, estas outras ferramentas geralmente utilizam as informações geradas por um determinado compilador, herdando também suas características. Por exemplo, no ambiente MiniLite, grande parte das ferramentas (SMILE, LOLA,...) utilizam o arquivo com extensão `.cr` gerado por TOPO. Já no conjunto CADP, grande parte do trabalho, inclusive simulação, é realizado sobre o LTS gerado por CAESAR.

| | TOPO | CAESAR |
|---|---|---|
| >> | permite recursão à esquerda | não permite recursão à esquerda |
| [> | permite recursão à esquerda | não permite recursão à esquerda |
| \|[...]\| | permite recursão tanto à esquerda quanto à direita | não permite recursão nem à esquerda nem à direita |
| *sort* infinito | trata *sorts* infinitos simbolicamente, usa formato de representação comum | não gera LTS infinito |
| recursão não guardada | não resolve, porém não interrompe compilação | não permite |

Tabela 6.2 - Incompatibilidades entre TOPO e CAESAR

Desta forma, tomando como exemplo os simuladores, suas incompatibilidades dependem diretamente da ferramenta que gera os dados por eles utilizados. Demais diferenças são relativas às funcionalidades fornecidas por uma e outra ferramenta. A Tabela 6.1 mostra algumas informações a este respeito.

A construção da especificação formal apresentada neste trabalho foi influenciada pelas restrições identificadas nas ferramentas usadas. Este fator não interferiu na busca do comportamento desejado do sistema, mas sim na estruturação das construções, na limitação do uso de dados e no desenvolvimento das tarefas de validação e, principalmente, de verificação.

## 5.3 Comentários finais

Neste capítulo, foi apresentada uma breve discussão sobre a capacidade, restrições e incompatibilidades das ferramentas de software utilizadas neste trabalho. Observa-se que, principalmente para a verificação, não existem ferramentas realmente robustas para

tratamento de especificações que utilizam LOTOS completo. As ferramentas existentes implementam os mesmos métodos utilizados na verificação de especificações que incorporam apenas a parte comportamental de LOTOS. Tais métodos nem sempre se fazem adequados quando utilizados tipos de dados.

# 6. Conclusões e perspectivas futuras

Este trabalho apresentou uma experiência sobre a aplicação da técnica de descrição formal LOTOS na concepção formal de um Gateway CMIP-SNMP para gerência de redes. Nessa experiência, inicialmente foi realizado o levantamento informal dos requisitos do Gateway. Em seguida, foi concebida a descrição LOTOS desses requisitos, utilizando a simulação como apoio na busca do comportamento desejado. Por fim, foi realizada a verificação e teste da especificação LOTOS, visando à sua correção. Para os trabalhos de análise da especificação, foram utilizadas ferramentas computacionais LOTOS, sobre as quais foram apresentados alguns comentários a respeito de suas capacidades, restrições e incompatibilidades.

Na descrição LOTOS do Gateway, o comportamento determinado nos requisitos informais foi alcançado. No decorrer deste trabalho, foram observados aspectos referentes à possíveis restrições impostas pelas ferramentas computacionais utilizadas. Nessa fase, foi utilizada principalmente a simulação para a busca do comportamento desejado do sistema e também na detecção de alguns erros.

Foi realizada a verificação da especificação a partir de diferentes métodos e ferramentas computacionais, sendo detectados alguns erros ocorridos durante a fase de descrição formal e simulação. Foram verificadas propriedades de segurança (busca de deadlocks indesejáveis), de vivacidade, e verificada a relação de equivalência observacional fraca para alguns dos módulos do Gateway. Também foram realizados testes para verificar a ocorrência com sucesso de seqüências de execução desejáveis.

Em resumo, a partir do trabalho aqui desenvolvido demonstra-se que sistemas reais são adequadamente tratados com o uso de LOTOS. Em contrapartida, este experimento reforça a idéia de que é necessária uma boa estruturação das especificações. Esta estruturação permite um melhor entendimento do sistema e, principalmente, facilita o alcance das metas desejáveis.

Sobre os tipos de dados, este trabalho demonstrou que seu uso adiciona funcionalidades que permitem ao projetista descrever sistemas maiores, utilizando, para isso, especificações menores e melhor estruturadas.

A detecção de falhas no Gateway, durante a validação e verificação, reforça a afirmação de que estas tarefas são parte fundamental do processo de concepção formal de um sistema. Nestas fases, comportamentos indesejáveis foram encontrados e corrigidos segundo os requisitos informais do Gateway.

Entre as limitações do trabalho, a mais relevante foi a impossibilidade de verificar a contento a especificação como um todo. As causas desta limitação são o uso de métodos voltados à verificação de especificações em LOTOS básico para verificar especificações em LOTOS completo e as restrições impostas pelas ferramentas computacionais. Isto foi principalmente identificado quando da impossibilidade de tratar *sorts* cujo domínio não pôde ser definido. Essa limitação afetou diretamente a possibilidade de estabelecer a relação de equivalência entre os módulos da especificação abstrata e os módulos da especificação detalhada que utilizam um *sort* com domínio variável. Em resumo, pode-se afirmar que a inexistência de ferramentas realmente robustas, principalmente para a verificação de especificações LOTOS com dados, limita o uso das facilidades fornecidas por esta técnica.

Para a continuidade do trabalho, existem perspectivas que pretende-se explorar. Estas perspectivas são aqui divididas nas tarefas de verificação, implementação automatizada do sistema e uso de outra técnica de descrição formal neste mesmo contexto. A seguir, estas tarefas são listadas discriminando as seguintes atividades:

- dar continuidade ao processo de teste da especificação LOTOS do Gateway;
- utilizar a abordagem transformacional implementada durante o projeto LOTOSPhere [LITE 92] para a verificação da especificação LOTOS do Gateway;
- com base nos experimentos realizados nos trabalhos de verificação, fazer uma análise comparativa entre os diferentes métodos para verificação existentes. Nesta análise podem ser considerados aspectos tais como esforço necessário, grau de dificuldade, requisitos do método, restrições (relativos ao método ou técnica em si e à ferramentas respectivas), entre outros;

- realizar a implementação final do Gateway a partir de sua especificação detalhada, buscando a automatização de parte deste processo. Uma das ferramentas que pode ser utilizada para alcançar essa automatização pode ser o compilador TOPO e a funcionalidade por ele fornecida denominada anotações. Estas anotações permitem que o código gerado pela ferramenta mantenha interação com o ambiente externo;
- especificar o Gateway, utilizando outra técnica de descrição formal (Estelle por exemplo), realizando uma análise comparativa para os resultados obtidos entre a especificação LOTOS e a especificação com uso de outra técnica e, também, sobre as análises realizadas em ambas.

# 7. Referências bibliográficas

[ASN.1 87] Information Processing - Open Systems Interconnection - Specification of Abstract Syntax Notation One, 1987. International Standard 8824, 1987.

[ACML 96] Anido, R.; Cavalli, A.; Macavei, T.; Lima, L. P.: "Testing a real protocol with the aid of verification techniques", XXIII Seminário Integrado de Software e Hardware, pp. 237-248, 1996.

[BoBr 87] Bolognesi, T.; Brinksma, E.: "Introduction to the ISO specification language LOTOS". In: The Formal Description Technique LOTOS, pp. 23-73, 1989.

[Bolo 92] Bolognesi, T., editor: "Catalogue of LOTOS Correctness Preserving Transformations", Techincal Report Lo/WP1/T1.2/N0045, The LOTOSPHERE Esprit Project, 1992.

[Borg 95] - BORGES, A. L.: "Especificação e Implementação de Elementos de Serviço do modelo OSI - ACSE e ROSE", Departamento de Informática e de Estatística - UFSC, dezembro, 1995.

[Bram 83] Brams, G. W.: "Réseaux de Petri: Théorie et Pratique", Ed. Masson, Tomos 1 et 2, Paris, 1983.

[Brin 88] Brinksma, E.: "A tutorial on LOTOS". In: IS8807 - Annex C, 1988.

[BRIS 93] BRISA: "Gerenciamento de Redes - Uma Abordagem de Sistemas Abertos". Makron Books - São Paulo, 1993.

[BSSt 87] Brinksma, E.; Scollo, G.; Steenbergen, C.: "LOTOS Specifications, their Implementations and their Tests", Protocol Specification, Testing, and Verification, VI, pp. 349-360, Elsevier Science Publishers B.V, North-Holland, 1987.

[ClJo 92] Clark, R. G.; Jones, V.M.: "Use of LOTOS in the formal development of an OSI protocol". Computer Communications, V.15, N.2. pp. 86-92, março 1992.

[DCBl 91] Draylon, L.; Chelwynd, A.; Blais, G.: "An introduction to LOTOS through a worked example.". Distributed Multimedia Research Group, School of Engineering, Computing and Mathematical Sciences, Sancaster University, 1991.

[Eert 92] Eertink, H.: "Executing LOTOS Specifications: The SMILE Tool", Telematics Research Centre, Enschede, Netherlands, 1992.

[EFHJ 91] Ernberg, P.; Fredlund, L.; Hansson, H.; Jonsson, B.; Orava, F.; Pehrson, B.: "Guidlines for Specification and Verification of Communication Protocols", Swedish Istitute of Computer Science, Kista, Sweden, 1991.

[EhMa 85] Ehrig, H.; Mahr, B.: "Fundamentals of Algebraic Specification 1". volume 6 of EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1985.

[EHMo 92] Ernberg, P.; Hovander, T.; Monfort, F.: "Specification and Implementation of na ISDN Telephone System Using LOTOS", Swedish Institute of Computer Science - SICS, technical report T92:04, 1992.

[EMCO 92] Ehrig, H.; Mahr, B.; Classen, I.; Orejas, F.: "Introduction to Algebraic Specification. Part 1: Formal Methods for Software Development", The Computer Journal, Vol. 35, No. 5, pp. 460-467, 1992.

[EMOr 92] Ehrig, H.; Mahr, B.; Orejas, F.: "Introduction to Algebraic Specification. Part 2: From Classical View to Foundations of System Specifications", The Computer Journal, Vol. 35, No. 5, pp. 468-477, 1992.

[FeMo 90] Fernandez, J.; Mounier, L.: "Aldébaran: User's manual", IMAG - LGI, Grenoble, France, 1990.

[Gara 89] Garavel, H.: "Compilation of LOTOS Abstract Data Types", Laboratoire de Génie Informatique, Institut I.M.A.G., Grenoble, France, 1989.

[Gara 94] Garavel, H.: "CAESAR Reference Manual", Laboratoire de Génie Informatique, Institut I.M.A.G., Grenoble, France, 1994.

[Gara 96] Garavel, H.: "An overview of the Eucalyptus Toolbox", Inria Rhône-Alpes (*Draft version*), Verimag, France, 1996.

[Gotz 87] Gotzhein, R.: "Specifying Abstract Data Types with LOTOS", In B. Sarikaya and G.V. Bochmann, editors, Protocol Specification, Testing, and Verification, VI, pp. 15-26. Elsevier Science Publishers B.V. (North-Holland), 1987.

[Hoar 85] Hoare, C. A. R.: "Communicating Sequential Processes", Prentice-Hall International, 1985.

[Holz 87] Holzmann, G.J.: "On limits and possibilities of automated protocol analysis". In Proc. IFIP WG 6.2 Symp. on Protocol Specification, Testing, and Verification VII, 1987.

[Holz 94] Holzmann, G.J.: "Na Improvement in Formal Verification". Proc. Forte 94, Bern, Switzerland, 1994.

[ISO 8807] Information Processing Systems - Open Systems Interconnection - "LOTOS - A formal description technique based on the temporal ordering of observational behavior". IS8807, 1988.

[ISO 8807b] ISSO/IECJTC1/SC21N4228, "Proposed Draft Addendum to ISSO 8807:1988 on G-LOTOS", 1988.

[ISO1 91] ISO/IEC DIS 10040 - Information Technology - Open Systems Interconnection - Systems Management Overview, International Organization for Standarzation/International Electrotechnical Commission, maio 1991.

[ISO 96] ISO-IEC/JTC1/SC21/WG7/N1053, "Revised Working Draft on Enhancement to LOTOS (V3)", Output of the Liege meeting, May, 1996.

[Kirk 94] Kirkwood, C. E.: "Verification of LOTOS Specifications using Term Rewriting Techniques". Submitted for the Degree of Doctor of Philosophy. Department of Computing Science, University of Glasgow, June, 1994.

[LCVu 92] Loureiro, A. A. F.; Chanson, S. T.; Vuong, S. T.: "FDT Tools For Protocol Development", Department of Computer Science, University of British Columbia, Vancouver, Canada, 1992.

[LFHa 92] Logrippo, L.; Faci, M.; Haj-Hussein, M.: "An Introduction to LOTOS: Learning by Examples". University of Ottawa, Department of Computer Science, Ottawa, Ontario, Canada, 1992.

[LITE 92] Caneve, M.; Salvatori, E. (editors): "LITE User Manual", Technical Report Lo/WP2/N0034/V08, The LOTOSPHERE Esprit Project, 1992.

[Maña 92] Mañas, J. A.: "Getting to Use the LOTOSPHERE Integrated Tool Environment (LITE)", Dpt. Ingeniería Telemática, Technical University of Madrid, 1992.

[Mazz 91] Mazzola, V. B.: "Contribution A La Conception De Systemes Flexibles De Production: Application De La Technique De Description Formelle Estelle", Tese de Doutorado, Université Paul Sabatier, Tolouse, France, 1991.

[Miln 80] Milner, R.: "A Calculus of Communication Systems", Lectures and Noter in Comp. Science, No. 92, Springer Verlag, 1980.

[Mura 89] Murata, F. T.; IEEE: "Petri Nets: Properties, Analysis and Applications", Proceedings of the IEEE, Vol. 77, N. 4, April 1989.

[Nico 87] Nicola, R. D.: "Extensional Equivalences for Transition Systems", Theoretical Computer Science, 17, pp. 279-301, 1982.

[Oliv 96] Oliveira, A. V.: "Definição de um Gateway CMIP-SNMP", Dissertação de Mestrado, Departamento de Informática e de Estatística - UFSC, 1996.

[Pehr 88] Pehrson, B.: "Tutorial on Verification of Protocols", Swedish Institute of Computer Science, Stockholm, Sweden, 1988.

[SaVi 93] Saloña, A. A.; Vives, J. Q.; Gómez, S. P.: "An Introduction to LOTOS". Dpto. Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, 1993.

[Schn 88] Schnoebelen, P.: "Refined Compilation of Pattern-Matching for Functional Languages", Science of Computer Programming, 11:133-159, 1988.

[Sjöd 91] Sjödin, P.: "From LOTOS Specifications to Distributed Implementadions", A Dissertation submitted for the Degree of Doctor of Technology, Uppsala University, Dpartmente of Computer Systems, 1991.

[SoLe 95] Soares, L.F.G.; Lemos, G.; Colcher, S.: "Redes de Computadores: Das LANs, MANs e WANs às Redes ATM". Rio de Janeiro, Editora Campus, 1995.

[Stal 93] Stallings, W.: "SNMP, SNMPv2 and CMIP: the practical guide to network management standards". Addison_Wesley Publishing Company, 1993.

[VeMa 94] Veiga, M.; Mañas, J. A.: "How to Use LOTOS Data Types from C Code and How to Implement them by Hand Using TOPO - verision 3R6", Dpt. Ingeniería Telemática, Univ. Politécnica de Madrid, Spain, 1994.

[Vasc 89] Vasconcelos, W.W.M.P.: "O Tempo como Modelo: A Aplicação de Lógicas Temporais na Especificação Formal de Sistemas Distribuídos", Dissertação de Mestrado, Universidade Federal do Rio de Janeiro, setembro, 1989.

[ViSc 88] Vissers, C.A.; Scollo, G.; Sinderen, M. van.: "Architecture and Specification Style in Formal Descriptions of Distributed Systems", 1988.

[Viss 89] Vissers, C. A.: "LOTOS Backgrounds", The Formal Description Technique LOTOS, Elseevier Science Publishers B.V.(North_holland), 1989.

[X710 91] Recommendation X.710 and ISO/IEC 9595, Information Technology - Open Systems Interconnection - Common Management Information Service Definition for CCITT Applications, 1991.

[X711 91] Recommendation X.711 and ISO/IEC 9596, Information Technology - Open Systems Interconnection - Common Management Information Service Definition for CCITT Applications, 1991.

# Anexo A - A TDF LOTOS

A TDF LOTOS (*Language of Temporal Ordering Specification*) [ISO 8807] foi desenvolvida durante a década de 80 e normalizada pela ISO (*International Organization for Standardization*) no ano de 1988. Inicialmente usada no projeto e no desenvolvimento de serviços e protocolos de comunicação do modelo de referência OSI (*Open Systems Interconnection*), vem tendo aplicação principalmente na definição de sistemas distribuídos [ClJo 92].

Este capítulo apresenta resumidamente a TDF LOTOS, suas partes dinâmica e estática, e sua semântica operacional.

## LOTOS Básico

A parte dinâmica da linguagem LOTOS foi baseada principalmente em CCS (*Calculus of Communicating Systems*) [Miln 80] e CSP (*Communication of Sequencial Processes*) [Hoar 85] e representa apenas do comportamento [Viss 89] dos sistemas. LOTOS básico é um conjunto da parte dinâmica de LOTOS que é capaz de representar apenas o comportamento de processos sem passagem de valores.

### *Comportamento*

Em LOTOS, as expressões de comportamento determinam que ações são passíveis de serem realizadas. Ou seja, a expressão de comportamento representa o estado de um processo. Existem em LOTOS duas expressões pré-definidas, `stop` e `exit`, que denotam o deadlock e a inação, respectivamente.

Em LOTOS, um processo é uma entidade capaz de executar ações internas e interagir com outros processos, que formam o seu ambiente. Em um processo, as ações podem ser executadas de modo independente (*i*), ou dependente de uma sincronização com o ambiente. A sincronização ocorre através de pontos denominados portas.

Uma das características principais na representação do comportamento em LOTOS está nos operadores de composicão paralela que permitem implementar o

conceito de processo rendez-vous (sincronização de processos). O conceito da sincronização multiway (participação de múltiplos processos) foi herdado da linguagem CSP. No entanto, a maior parte da linguagem é baseada em CCS.

Em LOTOS, as expressões de comportamento são escritas, utilizando processos e os operadores da linguagem. A tabela 3.1 lista os operadores na ordem decrescente de prioridade.

| OPERADOR | DESCRIÇÃO |
|---|---|
| ; | prefixo-de-ação (seqüenciamento de eventos) |
| [] | escolhas indeterminísticas entre comportamentos |
| \|\|\| | composição de processos independentes |
| \|\| | composição de processos dependentes |
| \|[ ]\| | composição geral |
| [> | interrupção de processos |
| >> | habilitação de processos |
| hide ... in ... | ocultação de eventos |

Tabela A.1 - Operadores LOTOS

Em [BoBr 87], são apresentados os operadores LOTOS e as regras para a formação das expressões de comportamento.

*Semântica Operacional de LOTOS*

A expressão $B = a; B1$ representa um processo que pode executar uma ação $a$ e se comportar como $B1$.

$$a; B \xrightarrow{a} B$$

A expressão de escolha $B1$ [ ] $B2$ representa um processo que pode se comportar como $B1$ ou como $B2$.

$$\frac{B_1 \xrightarrow{a} B'_1}{B_1 [] B_2 \xrightarrow{a} B'_1} \qquad \frac{B_2 \xrightarrow{a} B'_2}{B_1 [] B_2 \xrightarrow{a} B'_2}$$

A composição paralela representa o comportamento concorrente de dois processos. A expressão *g;B1*||*g;B2* representa dois processos que interagem simultaneamente na porta *g*. O paralelismo pode ser expressado também pelo operador geral ||[ *g1,...,gn*]|| que define as portas onde ocorre interação, e pelo operador de paralelismo independente |||.

$$\frac{B_1 \xrightarrow{a} B'_1, B_2 \xrightarrow{a} B'_2, a \in \{g_1,\ldots,g_n\} \cup \{\delta\}}{B_1 |[g_1,\ldots,g_n]| B_2 \xrightarrow{a} B'_1 |[g_1,\ldots,g_n]| B'_2}$$

$$\frac{B_1 \xrightarrow{a} B'_1, a \notin \{g_1,\ldots,g_n\} \cup \{\delta\}}{B_1 |[g_1,\ldots,g_n]| B_2 \xrightarrow{a} B'_1 |[g_1,\ldots,g_n]| B_2}$$

$$\frac{B_2 \xrightarrow{a} B'_2, a \notin \{g_1,\ldots,g_n\} \cup \{\delta\}}{B_1 |[g_1,\ldots,g_n]| B_2 \xrightarrow{a} B_1 |[g_1,\ldots,g_n]| B'_2}$$

Na expressão *hide g1, ... ,gn in B,* o operador de ocultação declara o conjunto (*g1, ... ,gn*) tal que as interações nestas portas não podem ser observadas pelo ambiente. Interações em portas ocultas são ações internas.

$$\frac{B \xrightarrow{a} B', a \in \{g_1,\ldots,g_n\}}{hide\ g_1,\ldots,g_n\ in\ B \xrightarrow{i} hide\ g_1,\ldots,g_n\ in\ B'}$$

$$\frac{B_1 \xrightarrow{a} B', a \notin \{g_1,\ldots,g_n\}}{hide\ g_1,\ldots,g_n\ in\ B \xrightarrow{a} hide\ g_1,\ldots,g_n\ in\ B'}$$

Dados dois processos *B1 e B2*, a composição seqüencial é representada pela expressão *B1* >> *B2* onde o processo se comporta como *B1* até seu término e, então, se comporta como *B2* . A expressão exit representa a terminação que define um processo que pode executar uma ação interna na porta $\delta$.

$$exit \xrightarrow{\delta} stop$$

$$\frac{B_1 \xrightarrow{a} B'_1 , a \neq \delta}{B_1 >> B_2 \xrightarrow{a} B'_1 >> B_2} \qquad \frac{B_1 \xrightarrow{\delta} B'_1}{B_1 >> B_2 \xrightarrow{i} B_2}$$

Dados dois processos *B1* e *B2*, a expressão *B1* [>*B2* define um processo onde em cada ação de *B1*, um evento de *B2* pode ocorrer, então o processo passa a se comportar como *B2*. Se *B1* encerrar com sucesso, *B2* não pode mais ser executado.

$$\frac{B_1 \xrightarrow{a} B'_1 , a \neq \delta}{B_1 [> B_2 \xrightarrow{a} B'_1 [> B_2} \qquad \frac{B_2 \xrightarrow{a} B'_2}{B_1 [> B_2 \xrightarrow{a} B'_2} \qquad \frac{B_1 \xrightarrow{\delta} B'_1}{B_1 [> B_2 \xrightarrow{\delta} B'_1}$$

## LOTOS completo

LOTOS completo incorpora a parte de dados [Gotz 87] da linguagem que é baseada na linguagem de especificação algébrica ACT ONE [EhMa 85]. Com isso, os processos de uma especificação adquirem a capacidade de passar valores de dados entre si. A aplicação desta linguagem de especificação algébrica em métodos formais é apresentada em [EMCO 92] e [EMOr 92], onde também são desenvolvidos alguns exemplos mostrando a forma da representação de tipos de dados.

A semântica de um processo de LOTOS completo é dada por um sistema de transições rotuladas estruturado. Ao invés de ser rotulada apenas por um nome de uma porta, a transição passa a ser rotulada por um nome de porta associado a um valor. As transições passam a ter a seguinte forma:

$$P \xrightarrow{gw} Q$$

Na transição acima, *g* significa o nome da porta e *w* o conjunto de valores associados à porta. Por exemplo, as ações são capacitadas a permitir que valores sejam recebidos ( *gate ?x:Nat* ), ou repassados ( *gate !1* ), podendo associar um ou mais valores a uma única ação.

Um tipo de dado é composto de um nome para o tipo de dado, uma lista de *sorts* utilizados pelo tipo, declarações de operações sobre os *sorts* e de equações que especificam o comportamento das operações. Os *sorts* e *operations* de um tipo de dado são referenciados como assinatura (*signature*). Para cada *sort*, é definido um conjunto de valores. A idéia de que diferentes notações podem representar o mesmo conceito é aplicada nas equações de ADTs (*Abstract Data Type*). A Figura 3.1 apresenta parte do tipo Boolean como exemplo. Uma ação em LOTOS completo é representada por uma porta mais uma oferta de valor (*!x*) ou de variável (*?x:Nat*).

```
Type  Boolean is
sorts Bool
opns  true, false :      -> Bool
      not         : Bool -> Bool
      _and_,_or_  : Bool, Bool -> Bool

eqns  forall x:Bool
      ofsort Bool
      not(true)    = false;
      not(false)   = true;
      x and true   = x;
      x and false  = false;
      x or true    = true;
      x or false   = x;
endtype
```

Figura A.1 - O tipo de dado Boolean

Em [ISO 8807], [BoBr 87] e [Brin 88], são apresentados os conceitos de tipos de dados LOTOS, descrevendo as assinaturas e as equações associadas. Também são apresentadas as características para a produção de especificações de dados estruturadas, sendo elas: uso de biblioteca, combinação, renomeação, parametrização, atualização e extensão de especificações de dados.

Alguns trabalhos tais como [DCBl 91] e [LFHa 92] apresentam a linguagem de forma mais didática com o uso de exemplos. Uma introdução para LOTOS é apresentada em [SaVi 93].

## Comentários finais

Este capítulo apresentou a TDF LOTOS, seus operadores e semântica operacional. Alguns aspectos da parte estática (tipos de dados) da linguagem foram também comentados.

# Anexo B - Ferramentas Computacionais para LOTOS

Este anexo apresenta uma breve discussão sobre ferramentas LOTOS de suporte para a concepção formal de sistemas. Aponta algumas características desejáveis e mostra uma classificação destas segundo os estágios do ciclo de vida do sistema [LCVu 92].

Também são apresentadas as ferramentas computacionais LOTOS utilizadas neste trabalho, identificando sua funcionalidade.

### Ferramentas de software

No desenvolvimento de aplicações, principalmente distribuídas, as técnicas de descrição formal têm tido uso crescente. Porém, este fator é diretamente ligado à disponibilidade de ferramentas de software afins. A descrição formal de sistemas grandes, geralmente, é inviável sem a assistência de ferramentas computacionais.

Este fator pode ser justificado pela complexidade da teoria aplicada aos métodos formais comparada ao tempo necessário para processá-la. As ferramentas automatizam a aplicação da teoria formal, assim o usuário pode utilizar métodos formais sem conhecer completamente essa teoria. Também evita que se gaste longo tempo de desenvolvimento (aumento de produtividade) e a ocorrência de erros humanos que geralmente são inevitáveis.

Na concepção de sistemas, as técnicas de descrição formal não fazem parte dos objetivos, mas do arsenal de meios utilizados pelo projetista para alcançar objetivos pré determinados. Este fator engrandece a importância das ferramentas computacionais que, associadas aos métodos formais, permitem melhorias nos resultados obtidos.

Existe um número considerável de ferramentas, principalmente para as três técnicas padronizadas (LOTOS, Estelle e SDL). Tais ferramentas vêm sofrendo atualizações com o propósito principal de solucionar problemas. Algumas delas se

destacam pela sua robustez e funcionalidade. Contudo, deficiências permanecem, principalmente no suporte à verificação de especificações LOTOS que implementam tipos de dados, como discutido neste trabalho.

A seguir, são apresentadas algumas características que essas ferramentas deveriam incorporar. São elas:

- *Facilidade de uso:* a facilidade de uso incorpora alguns aspectos, tais como uma interface amigável com o usuário, o que pode melhorar a produtividade. Esta facilidade deve incluir permissão para realizar alterações na interface ou inclusão de novos comandos, manter ajuda online e tratar o maior número possível de erros do usuário.
- *Performance:* esta característica está ligada ao desempenho ou agilidade na execução das tarefas e no número de tarefas que desempenha (análise, simulação, ...). Esta característica também considera a possibilidade de comunicação com outras ferramentas.
- *Robustez:* inclui facilidades para suportar o trabalho de múltiplas pessoas sobre diferentes aspectos de uma especificação, sem que com isso se perca a consistência necessária. A ferramenta deve ser robusta o suficiente para permitir alterações, mantendo compatibilidade com novas versões. Ela também deve ser capaz de se recuperar de falhas no ambiente durante sua execução e manter algum instrumento que facilite a detecção de erros da própria ferramenta.

Em [LCVu 92], é apresentada um esquema de classificação das ferramentas baseado nas fases de desenvolvimento de sistemas. Este esquema é dividido em quatro classes: especificação, validação e verificação, implementação e teste.

A classe de especificação engloba ferramentas de edição, sustentação, impressão, análise sintática e semântica da especificação. A validação e verificação inclui a análise das propriedades sintáticas e análise das propriedades semânticas (simulação) da aplicação especificada, respectivamente. Quanto à classe de implementação, ela inclui compiladores, tradutores e ferramentas que permitam a geração de código final ou similares, e também ferramentas simbólicas tais como interpretadores e simuladores. E uma última classe que engloba as ferramentas de teste.

Em [LCVu 92], são apresentadas algumas ferramentas para LOTOS, comentando sua evolução. Em [Gara 96], é apresentado o ambiente Eucalyptus de ferramentas que incorpora funcionalidades para análise, simulação, compilação, verificação e teste de especificações LOTOS.

A seguir, são apresentadas as características e funcionalidades das ferramentas computacionais para LOTOS utilizadas neste trabalho.

**O compilador TOPO**

TOPO é um conjunto de ferramentas que fornece suporte para análise sintática e semântica e compilação de especificações LOTOS. Com TOPO, também é possível a realização de testes sobre as especificações, porém não diretamente.

O compilador TOPO é dividido em duas partes, o compilador do comportamento e o compilador de dados. A primeira gera um módulo em linguagem C que implementa a parte comportamental como um sistema de transição rotulada. A segunda gera um módulo em linguagem C que implementa a parte de dados.

TOPO gera o arquivo no formato de representação comum (*Common Representation*), com extensão '*.cr*'. Esta representação é um formato para especificações LOTOS utilizado pela maioria das ferramenta do ambiente MiniLite.

No capítulo 6, foram comentadas algumas restrições deste compilador. A seguir, são então listadas as funcionalidades suportadas pelo compilador TOPO [LITE 92]:

- implementa automaticamente tipos de dados LOTOS como um sistema de reescrita;
- fornece facilitades para gerenciamento de memória da implementação de estruturas de dados resultantes em código C ;
- permite substituir partes das implementações dos tipos de dados por ele geradas, por implementações codificadas manualmente (busca de maior eficiência);
- oferece facilidades para utilizar operações de tipos de dados definidas parcialmente;
- implementa comportamento randômico para especificações indeterminísticas;
- oferece facilidades para controlar externamente a oferta de eventos;
- rendez-vous múltiplos são quase sempre completamente suportados;

- suporta a criação dinâmica e recursiva de processos.

No Anexo B, são apresentadas as anotações que podem ser utilizadas pelo compilador TOPO.

## O simulador SMILE

A ferramenta SMILE fornece funcionalidade necessária para simular simbolicamente especificações LOTOS que utilizam tipos de dados ou não. A diferença básica entre um avaliador (simulador) simbólico e um avaliador não simbólico é a seguinte: um avaliador simbólico permite adicionar declarações de novas variáveis (não instanciadas) para os modelos que ele gera, enquanto um avaliador não simbólico não permite. Uma demonstração do uso de SMILE pode ser encontrada em [Eert 92].

Quando este simulador é executado, oferece uma interface gráfica onde a especificação completa ou apenas o nome dos processos, indicadores de transições e estados, entre outras informações, são apresentados.

Para recursão não guardada, a ferramenta SMILE informa construção da expressão de comportamento mal definida. Então, desconsidera a recursão e continua a simulação do comportamento da parte correta da expressão.

SMILE possui uma série de facilidades, como por exemplo, a geração da árvore da especificação, gerar a máquina de estados finitos, possui uma interface para tipos abstratos de dados, entre outros [LITE 92]. O usuário também pode interferir no comportamento, adicionando restrições ou instanciando novos eventos em tempo de simulação.

Para que uma especificação possa ser simulada com o uso de SMILE, ela deve estar correta sintática e semanticamente. Oferece as seguintes funcionalidades:

- observação das expressões de comportamento individualmente através de um único passo;
- execução simbólica automática de expressões de comportamento;
- análise da parte de tipos abstratos de dados da especificação; e
- tradução da expressão de comportamento em máquina de estado finito.

### A ferramenta LOLA

Em resumo, com a ferramenta LOLA é possível realizar depuração de especificações LOTOS através da sua execução passo-a-passo, a realização de testes e a verificação da propriedade de vivacidade.

A ferramenta LOLA fornece dois comandos com os quais é possível verificar a propriedade de vivacidade, o comando *expand* e o comando *varexpand*. Para ambos os comandos, é possível estabelecer ou não uma profundidade (*depth*) de busca, informada através de um inteiro que representa o número de ações geradas desde o início da exploração. Se esta não é estabelecida, uma exploração exaustiva é realizada, ocorrendo o problema da explosão de estados.

O comando *expand* transforma a especificação original, mantendo bissimulação forte, em uma especificação equivalente. A especificação gerada contém apenas ações visíveis e internas, prefixo de ação, expressões de soma, escolha, operadores `exit` e `stop`, guardas e processos.

O comando *VarExpand* expande a especificação da mesma forma que o comando *expand*, porém guarda variáveis simbólicas e detecta estados duplicados parametrizados. A representação gerada do sistema de transição é menor que a gerada pelo comando *expand*.

Também é disponível o comando *freeExpand* que realiza o mesmo procedimento que o comando *expand*, porém não detecta duplicação de estados.

Esta ferramenta também permite a realização de testes, onde ocorre uma exploração dos estados de uma especificação, observando a ocorrência de eventos de sucesso previamente especificados. Esta exploração pode ser exaustiva, ou apenas de um ramo da especificação escolhido randomicamente.

### O conjunto CAESAR/Aldébaran de ferramentas

Este conjunto de ferramentas fornece funcionalidade para análise (sintática e semântica), compilação, simulação e verificação de especificações LOTOS. Tanto a simulação como a verificação das relações de equivalência entre dois níveis de abstração

de uma especificação são realizadas sobre sistemas de transições rotuladas finitos. Esta característica impõe algumas restrições, comentadas no capítulo 6 deste trabalho.

A seguir, é apresentada a forma pela qual este conjunto de ferramentas trata especificações LOTOS que utilizam tipos de dados.

Inicialmente, a compilação dos tipos de dados de uma especificação deve ser realizado em separado. Para isso, este conjunto de ferramentas disponibiliza a ferramenta CAESAR.ADT que implementa automaticamente os tipos de dados. Contudo possui restrições, como por exemplo, o não tratamento de tipos parametrizados (comentado no capítulo 6). Neste processo de compilação, cada *sort* da especificação resulta em um conjunto de operações que são divididas em duas classes: os construtores e os não-construtores. A segunda opção na anotação de uma operação (*constructor*) indica se uma operação é ou não é um construtor.

Uma operação *F* é um construtor se existe um termo na forma normal sem nenhuma variável contendo *F*. Isto significa que algumas ocorrências de *F* não podem ser reduzidas porque a semântica da operação não é completamente definida pelas equações.

Como nem sempre é possível dar a um objeto C o mesmo nome implementado em LOTOS, o CAESAR.ADT exige que os *sorts* e operações dos tipos sejam fornecidos. Isto é alcançado através do uso de anotação, como comentado acima. Basicamente, usando as informações contidas nas anotações, cada *sort* é traduzido em um tipo C correspondente e cada operação em uma função C correspondente.

Na Figura A.1, que apresenta a sintaxe das anotações que devem ser inseridas na especificação, o "*name*" representa um identificador C. O CAESAR não aceita nomes iguais para a mesma palavra chave C, nem o uso do prefixo CAESAR_ nos nomes. Este prefixo é uma *string* reservada.

```
Sort_name (*!   [implementedby name1]
                [comparedby name2]
                [enumeratedby name3]
                [printedby name4]   *)
```

Figura B.1 - Sintaxe das anotações

As palavras reservadas *'implementedby'*, *'comparedby'*, *'enumeratedby'*, e *'printedby'*, definem os quatro atributos que são ligados a cada *sort*. Os nomes têm a seguinte representação:

- *name1* - identifica um tipo C que implementa o *sort*;
- *name2* - identifica uma função C que implementa a operação de igualdade entre os possíveis valores do *sort;*
- *name3* - identifica uma macro C que trata do domínio do *sort* (conjunto de valores) para a implementação de construções LOTOS tais como ?X:S, onde S representa um *sort*;
- *name4* - identifica uma função C que mostra os valores do *sorte* em um arquivo.

Uma operação *F* é um não-construtor se todas as ocorrências de *F* nos termos não têm variáveis que podem ser eliminadas pela reescrita, ou seja, não pode ocorrer redução.

Durante a geração do código, nenhuma expressão pode ser reescrita para um termo na forma normal, contendo somente operações do tipo construtor. O conceito de construtor é importante, pois representa a base para a representação e implementação a estrutura de dados.

Se o conjunto correto de construtores não é fornecido, alguns erros podem ocorrer. Por exemplo, se uma operação do tipo não construtor é declarada como construtor, os termos resultantes deverão conter uma ocorrência da operação. O

especificador deve verificar se o resultado é ou não é o esperado. Também, se uma operação do tipo construtor não é declarada como tal, o erro ocorrerá durante a geração do código C no uso da ferramenta CAESAR.ADT.

Por exemplo, se uma operação é executada sobre outra operação do tipo não construtor, a ferramenta devolve um aviso anunciando o erro. Na definição equacional, uma operação só pode ser executada sobre construtores ou variáveis.

### A ferramenta gráfica GLOTOS

O uso da ferramenta GLOTOS é bastante simples. Esta ferramenta permite, a partir de uma especificação LOTOS, representar a especificação, uma hierarquia de processos, ou um único processo graficamente (diagrama de blocos). Pode ser muito útil na compreensão do comportamento da especificação. Esta ferramenta suporta tipos de dados. Utiliza a sintaxe gráfica definida para LOTOS em [ISO 8807b].

### As ferramentas GLD e GLA

Existem outras ferramentas semelhantes a GLOTOS, tais como GLA e GLD. GLA é uma ferramenta gráfica que oferece um conjunto de mecanismos para analisar o comportamento dinâmico de uma especificação, bem como de um processo específico. Ajuda na detecção de problemas, principalmente em especificações grandes.

A ferramenta GLD facilita o trabalho de projeto arquitetural, ou seja, na descrição do conjunto de componentes e suas interconexões. Utiliza uma linguagem visual chamada DART, permitindo tradução automática de DART para LOTOS e vice versa.

# Anexo C - Implementação Automática de Especificações LOTOS

A princípio, a implementação automática completa de especificações LOTOS não é possível. Os principais fatores são os seguintes [EHMo 92]:

- não fornece um mapeamento entre eventos abstratos e eventos reais; e
- em situações de escolha, faltam meios para expressar prioridade.

LOTOS possui um alto poder de abstração, o que também dificulta a automatização da implementação.

Não há ferramentas conhecidas para a geração completa de uma implementação final a partir de uma especificação LOTOS. No entanto, existem formas de automatizar partes do processo de implementação final de uma especificação. Por exemplo, através do uso da ferramenta TOPO sobre especificações modificadas com anotações. Neste caso, o usuário deve construir programas na linguagem C que farão a comunicação com o código C gerado pela ferramenta.

Como trabalho futuro, será realizada implementação da especificação detalhada do Gateway. Alterações nesta especificação estão previstas com o objetivo de apurar detalhes importantes de implementação, facilitando, assim, o uso de meios automatizados no processo.

Um exemplo do uso da ferramenta TOPO na implementação semi-automática de uma especificação LOTOS de um sistema telefônico ISDN pode ser encontrado em [EHMo 92].

Em [VeMa 94], é mostrado como usar o código gerado (da parte de dados) pelo compilador TOPO e como adicionar implementações de tipos de dados codificadas manualmente.

## Anotações com TOPO

Anotações são comentários da forma (*| palavra chave ou valor |*) que podem ser inseridos no código da especificação. As anotações são utilizadas para incrementar os módulos C, resultantes do uso de um compilador, com facilidades para permitir que mecanismos de controle possam ser inseridos e utilizados.

Como as anotações utilizam os símbolos normais para comentário, são invisíveis para ferramentas que não implementam esta característica. Para outras ferramentas que também implementam este tipo de representação, a forma interna ao comentário variar.

A seguir, são identificados os tipos de anotações [LITE 92] disponíveis para a parte comportamental de especificações LOTOS reconhecidas pelo compilador TOPO. O conteúdo em negrito é reservado à anotação.

(*| **lbc** *C_code* |*) - O código C (*C_code*) é reproduzido no arquivo com o mesmo nome da especificação e extensão `.lbc.c`. É utilizado para referenciar outros arquivos que fornecem funções usadas em outras anotações.

(*| **C** *C_statments* |*) - *C_statments* são declarações em código C que são executadas após uma ação envolvida em um rendez-vous com sucesso. Também pode ser usada depois de uma palavra reservada *endspec* com o objetivo de fornecer dentro do código da especificação, funções usadas por outras anotações de comportamento.

(*| **delay** *C_expression* |*) - *C_expression* é um valor inteiro que define um retardo para que a ação seja oferecida.

(*| **wait** *C_expression* |*) - Implementa uma situação de espera. A expressão *C_expression* é repetidamente avaliada até que uma resposta verdadeira é retornada, permitindo, então, que a ação correspondente seja oferecida.

(*| **default** *C_expression* |*) - Quando em uma negociação de rendez-vous não existe oferta(!) disponível, um default pode ser fornecido.

(*| **eval** *C_identifier* |*) - Quando em uma negociação de rendez-vous não existe oferta(!) disponível, uma função pode ser fornecida. Esta função é chamada repetidamente até que retorne um valor válido.

Abaixo são identificados os tipos de anotações disponíveis para a parte de dados de especificações LOTOS reconhecidas pelo compilador TOPO. O conteúdo em negrito é reservado à anotação.

(*| **ldc** *C_code* |*) - O código C (*C_code*) é reproduzido em um arquivo com extensão `.ldc.hh`. Utilizado para referenciar outros arquivos que são usados para implementar partes de tipos de dados.

(*| **name** *C_identifier* |*) - Força o compilador de dados a utilizar o nome representado por *C_identifier* na geração de código. É utilizada pelo fato de que nem sempre um identificador LOTOS pode ser um identificador C.

(*| **extern** |*) - Quando um *sort* é marcado com **extern**, então toda operação que envolve este *sort* também deve ser externa.

(*| **equal** *C_identifier* |*) - Identifica uma função C que é esperada do usuário. Esta função deve decidir se dois valores do *sort* respectivo são iguais ou não.

(*| **draw** *C_identifier* |*) - Idendifica uma função C que é esperada do usuário. Esta função deve apresentar um nodo do *sort* respectivo.

(*| **free** *C_identifier* |*) - Identifica uma função C que é esperada do usuário. Esta função deve ser capaz de liberar memória alocada para um nodo do respectivo *sort*.

(*| **partial** *C_expression* |*) - A expressão *C_expression* é avaliada quando uma operação é chamada.

Em [LITE 92], é apresentado um exemplo simples do uso de anotações. Mostra o caso de um produtor e um consumidor, conectados por dois buffers.