Utilizando LOTOS na concepção formal de uma aplicação para gerência de redes: Especificação e Verificação

Braulio Adriano de Mello, Murilo Silva de Camargo e Elizabeth Sueli Specialski
E-mail: bmello@missoes.com.br; {murilo, beth}@inf.ufsc.br

Universidade Federal de Santa Catarina
Curso de Pós-Graduação em Ciência da Computação
Departamento de Informática e de estatística
Campus Universitário Trindade - Caixa Postal 476
88040-900 Florianópolis - SC
janeiro, 1997

Resumo - *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, Simulação, Verificação, Gerência de Redes, Gateway CMIP-SNMP.*

Abstract - *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. 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, are presents. With the use of abstract data types, mostly to the verification task, the tolls presented restrictions. These restrictions are approached by their capacities and incompatibilities.*

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 correção e confiabilidade. Em conseqüência, as técnicas para a descrição formal de sistemas estão sendo cada vez mais utilizadas. Tais técnicas apresentam-se como uma solução para vários problemas de concepção.

A utilização de uma técnica formal permite descrever sistemas com precisão, livres de ambigüidades, e facilita a detecção e correção de erros de concepção. Também possibilita o uso de ferramentas de software no processo de análise, teste, simulação e verificação das especificações.

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 explorar a aplicabilidade da técnica de descrição formal LOTOS na concepção e verificação de sistemas,

especificamente, uma aplicação Gateway [Oliv 96] para gerência de redes de computadores. Também é objetivo deste trabalho conceber a descrição formal correta desta aplicação, até o momento inexistente. De forma geral, esta aplicação segue os padrões do modelo OSI (*Open Systems Interconnection*) [ISO1 91] ao executar operações de gerenciamento sobre objetos que seguem os padrões do modelo SNMP(*Simple Network Management Protocol*)[Stal 93].

Neste trabalho, a ênfase é dada a uma avaliação de possíveis problemas durante o processo de concepção. Tal avaliação envolve tarefas tais como especificação, simulação e verificação, associadas às vantagens, restrições e disponibilidade de ferramentas afins.

Neste trabalho, o termo Gateway será usado para denotar o Gateway CMIP-SNMP aqui especificado e verificado.

A seção 2 apresenta sucintamente os requisitos informais da aplicação gateway. A seção 3 mostra os principais tipos de dados e os principais processos da especificação LOTOS do gateway CMIP-SNMP, a seção 4 apresenta o trabalho de validação e verificação da especificação. Finalmente, é comentada a experiência no uso de ferramentas de software para LOTOS, suas capacidades, restrições e incompatibilidades (seção 5), conclusões e perspectivas futuras (seção 6), e referências bibliográficas (seção 7).

2. O Gateway CMIP-SNMP: Requisitos informais

O uso do padrão OSI na gerência de redes é uma tendência baseada na necessidade de gerenciamento mais eficaz devido ao crescimento e heterogeneidade das redes, sendo confirmada pela disponibilidade de produtos comerciais que já implementam tais recursos. Assim, o uso de produtos que seguem padrões distintos (OSI e Internet) numa mesma rede, resulta na necessidade de comunicação entre aplicações que seguem padrões distintos. Esta comunicação pode ser fornecida por um gateway.

A aplicação especificada e verificada neste trabalho tem o propósito de permitir 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.

Além de realizar a tradução de PDUs (Protocol Data Unit), esta aplicação, denominada gateway, também implementa no lado SNMP funcionalidades existentes apenas no modelo OSI, como por exemplo, a possibilidade do uso de filtros durante a seleção de objetos. Esta característica diferencia esta aplicação das demais semelhantes, porém, adiciona complexidade ao sistema.

Abaixo é feita uma apresentação sucinta dos principais requerimentos do gateway. Os requisitos informais detalhados desta aplicação gateway podem ser encontrados em [Oliv 96]. Neste trabalho, entende-se por gateway, como o caso específico do gateway CMIP-SNMP aqui especificado e verificado.

Funcionalidade: Como comentado acima, a aplicação gateway visa a tradução e mapeamento de funcionalidade entre os protocolos CMIP (*Common Management Interconnection Protocol*) e SNMP. Esta facilidade permite que um gerente no domínio OSI possa gerenciar, sem alterações, nós Internet (Figura 1).

O gateway é composto por dois módulos. O primeiro realiza a tradução das PDUs, e o segundo é responsável pelo controle das operações. Para realizar a tradução, é preciso conhecer o formato das PDUs de ambos os modelos (este formato pode ser encontrado definido em ASN.1 (*Abstract Sintax Notation One*)

[ASN.1 87] em [Oliv 96], anexo A). A Unidade de Controle visa garantir a consistência das informações que são repassadas pelo Gateway.

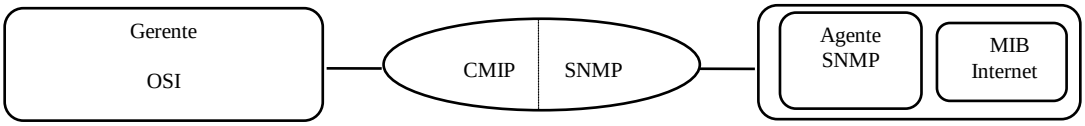

Figura 1 - Aplicação Gateway

Outra tarefa necessária ao funcionamento do gateway é a tradução das MIBs, ou seja, a atualização da MIB OSI, da estrutura de identificação dos agentes, e da estrutura de mapeamento do nome. Entretanto, esta tarefa não é desenvolvida pelo Gateway.

Durante a execução de uma operação de gerenciamento três fases são observadas. A primeira é o estabelecimento da associação entre os processos *gerente* e *agente*. 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, que por sua vez, podem transmitir notificações ao gerente [BRIS 93].

A segunda fase envolve a troca e execução das operações de gerenciamento. A última fase é a liberação da associação. Como no SNMP não ocorre associação, ao contrário do modelo OSI, o gateway "emula" para o gerente o estabelecimento de uma associação (enviando as primitivas apropriadas), possibilitando a execução de operações de gerenciamento requisitadas por um gerente OSI sobre recursos (objetos) SNMP.

Estrutura de mapeamento: Para efeito de mapeamento da informação, considera-se o nome (agente) e serviço fornecido. O mapeamento do nome é usado para referenciar a informação de gerenciamento. O mapeamento do serviço é usado para efeitos de tradução.

Mapeamento funcional: Como as funções de escopo, filtro e sincronização disponíveis no lado OSI não possuem contrapartida no lado SNMP, o gateway, além de realizar a funções normais de tradução, também implementa o mapeamento de tais funções. Isto adiciona complexidade ao processo. A função de escopo permite que múltiplos objetos sejam selecionados. Com o filtro, apenas objetos que atendem a condições pré determinadas são selecionados. E finalmente, a sincronização, que pode ser atômica ou de melhor esforço. Se atômica, a operação deve ser realizada sobre todos os objetos selecionados, ou não é realizada. Se de melhor esforço, para os objetos selecionados sobre os quais a operação não obteve sucesso, retorna uma informação de erro.

Associação: Para estabelecer uma associação, inicialmente o gateway reconhece o ambiente (OSI/SNMP) do agente usando a informação contida no parâmetro CAPT (Called AP_Title). Se o agente é OSI, a operação é repassada ao ACSE (*Association Control Service Element*). Se SNMP, o gateway verifica se o agente esta ativo, confirmando ou não o estabelecimento da associação ao gerente OSI.

Get: A operação get é usada na recuperação de valores dos objetos gerenciados. Se a operação é executada sobre objetos SNMP, ela pode ser mapeada pelo gateway em uma ou mais operações get. Este fator depende da funcionalidade requerida através dos valores especificados nos parâmetros de escopo, filtro e sincronização, recebidos juntamente com uma requisição.

Set: Esta operação é usada para alterar os valores dos objetos gerenciados. A operação set tem o procedimento relativo ao mapeamento da funcionalidade semelhante ao da operação get. Porém, quando um set é requisitado, inicialmente os valores originais são recuperados, permitindo que estes sejam restabelecidos caso a operação set seja rejeitada.

Trap: A operação trap tem o objetivo de informar a ocorrência de um evento em um objeto. Quando o gateway recebe um trap, deve traduzir a informação recuperada para o formato de um relatório de evento que é enviado ao gerente. Como o modelo OSI é orientado à conexão, caso não exista uma associação estabelecida entre o agente emissor da informação e um gerente, esta deve ser providenciada antes do envio do relatório.

A especificação LOTOS resultante do trabalho de descrição formal destes requisitos é apresentada na seção seguinte. Apenas as partes relevantes referentes às operações possíveis são mostradas, pois o volume da especificação não permite sua apresentação completa neste. A especificação formal completa pode ser encontrado em [Mell 97].

3. Especificação LOTOS da aplicação gateway

Esta seção apresenta os principais tipos de dados e os principais processos definidos na especificação LOTOS da aplicação gateway. Algumas considerações sobre aspectos relevantes tais como o uso de anotações e algumas decisões de projeto resultante do formalismo usado também são discutidas nesta seção. Na seção 7 são abordadas as dificuldades e restrições encontradas durante o processo de especificação LOTOS do Gateway, simulação e verificação.

3.1 Tipos de dados

A definição dos tipos de dados da especificação apresentada é composta por 18 tipos, sendo que destes, 2 fazem parte da biblioteca fornecida em [IS 8807], anexo A (A*nnex 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.

Os *sorts* do tipo de dado que define o estabelecimento e encerramento de uma associação entre um gerente OSI e um agente SNMP são mostrados na Figura 2. Observa-se que *sorts* e operações (Figura 3) possuem comentários especiais denominados anotações. Estas anotações são necessárias para o uso da ferramenta CAESAR.ADT na compilação dos tipos de dados abstratos, gerando tipos concretos (código C).

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

Figura 2 - Sorts para a função de associação

```
...
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
```

Figura 3 - Operações para a função de associação

As operações possíveis no processo de associação são apresentadas na Figura 3. 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.

As operações GAssocReq, GReleaseReq, e GAbortReq definem as primitivas de requisição para o estabelecimento, encerramento normal e encerramento abrupto de uma associação disponíveis no Gateway, respectivamente. Quando as primitivas de requisição são recebidas pelo Gateway, este executa o mapeamento adequado, confirmando ou não a execução das operações através das primitivas GAssocCnf e GReleaseCnf, que representam as primitivas de confirmação do estabelecimento e encerramento normal de uma 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  (*!   ...   *),
      operation2_gw (*!   ...   *)
...
```

Figura 4 - Sorts das operações de gerenciamento

A Figura 4 apresenta os sorts relativos ao tipo que define operações de gerenciamento sobre os objetos gerenciados. O sort *operation_gw* identifica as requisições para a execução de operações de gerenciamento. O sort *operation2_gw* identifica tanto as primitivas que enviam as respostas das operações requisitadas, bem como 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.

```
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 *)
```

Figura 5 - Operações de gerenciamento

As operações mapeadas pelo Gateway (Figura 5) no lado SNMP são GET, SET e EVENT-REPORT. Cada uma destas operações é discutida na apresentação da parte comportamental.

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, para essas definições de tipo, é usada uma característica de especificação estruturada denominada renomeação de tipos de dados, demonstrada na Figura 6.

Na seção 5 estas características de estruturação para tipos de dados são comentadas, identificando as restrições impostas por ferramentas para algumas destas características.

Foram usadas definições de dados no controle das operações de gerenciamento, permitindo que as funções adicionais (escopo, filtro e sincronização) sejam mapeadas. Tais definições permitem a realização de escolhas determinísticas (uso de guardas) na execução das operações.

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

Figura 6 - Exemplo de renomeação de tipos

Devido ao uso de tipos de dados, optou-se por usar uma única porta de comunicação para as diferentes operações. A identificação do agente e da associação são então especificadas pelo valor dos dados associados. Da mesma forma, para a identificação das operações, foi necessário definir três tipos de dados. Um para identificar eventos em geral (signal), outro para identificar as diferentes variações da operação GET (signal_get), e o terceiro para identificar as diferentes variações da operação SET (signal_set).

```
type Set is Boolean, Naturalnumber
sorts
      Set (*! implementedby ADT_SET comparedby ADT_CMP_SET
                  enumeratedby ADT_ENUM_SET printedby ADT_PRINT_SET  *)
opns
      {} (*! implementedby ADT_EMPTY constructor *) : -> Set
      Add   (*! implementedby ADT_ADD constructor *),
      insert  (*! implementedby ADT_INSERT  *),
      remove  (*! implementedby ADT_REMOVE *): Nat,Set -> Set
      removeall (*! implementedby ADT_REMOVEALL *)  : Set,Set -> Set
      IsEmpty   (*! implementedby ADT_ISEMPTY *)    : Set  -> bool
      NotEmpty  (*! implementedby ADT_NOTEMPTY *)   : Set  -> bool
      _eq_      (*! implementedby ADT_EQ_SET  *),
      _ne_      (*! implementedby ADT_NE_SET  *)    : Set,Set -> bool
      _isin_    (*! implementedby ADT_ISIN *),
      _NotIn_   (*! implementedby ADT_NOTIN *)      : Nat,Set -> bool
...
```

Figura 7 - Tipo de dado para conjunto de objetos

Como visto na seção 4, 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 define um conjunto de objetos. É permitido inserir, remover, remover todos (*removeall*), usar operadores booleanos, e verificar se um determinado valor pertence ou não pertence ao conjunto de valores de objetos (Figura 7). Estas operações são essenciais na representação do comportamento quando especificadas as funcionalidades de escopo, filtro, e sincronização, ou ambas.

3.2 Comportamento da especificação

Nesta seção é apresentada a especificação do comportamento da aplicação Gateway. A parte comportamental é composta por 31 processos, sendo que estes podem ser divididos em 6 módulos principais. A Figura 8 apresenta a arquitetura do sistema, onde pode-se observar a comunicação entre esses módulos. Das 1850 linhas do código da especificação, 1250 foram utilizadas para a especificação do comportamento.

Na arquitetura mostrada na Figura 8, o módulo Controle de Associação trata principalmente da identificação do ambiente do agente (OSI/SNMP) com o qual o gerente deseja fazer uma associação. O módulo Controle de Operação trata do mapeamento de uma requisição recebida, realizando a chamada da respectiva operação requisitada. Define o modo (*confirmed/noconfirmed*) da operação.

Como o próprio nome dos módulos denotam, 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 no objetos e envio das informações ao gerente no formato de relatório de evento, quando uma associação é existente. Quando uma associação não está estabelecida, o módulo EVENT_REPORT realiza a mesma função que o módulo TRAP, porém trata também do estabelecimento e encerramento da associação necessária ao envio do relatório de evento ao gerente.

Feita apresentação do modelo mais abstrato do Gateway, a seguir são apresentados os processos mais importantes que compõem a arquitetura (Figura 8).

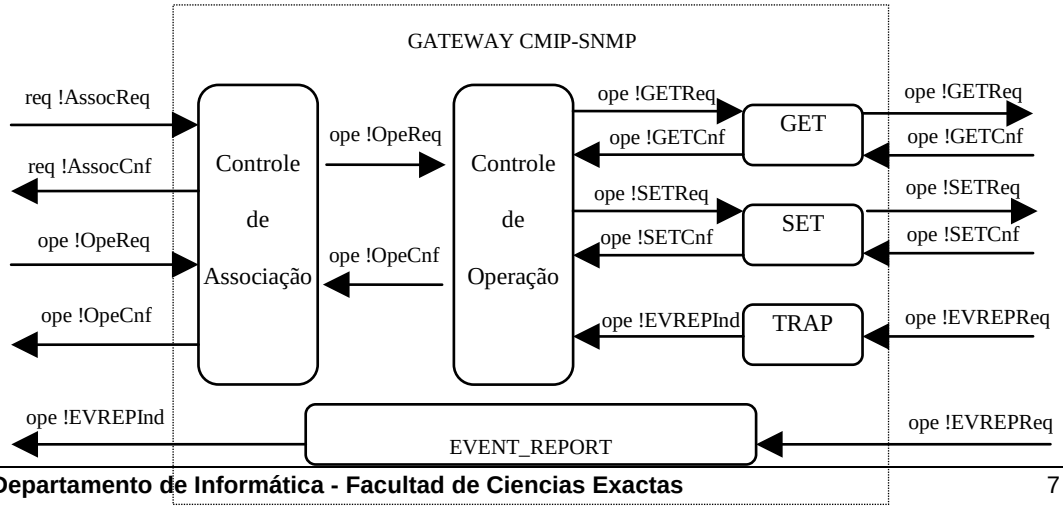

Figura 8 - Arquitetura da aplicação gateway

Inicialmente, o comportamento geral é representado por dois processos que sincronizam independentemente (Figura 9). O processo EVENT_REP (comentado acima) representa o comportamento do sistema quando ocorre um relatório de evento sem que uma associação esteja previamente estabelecida. O processo G_ASSOCIATE representa o comportamento do sistema quando uma associação é solicitada, habilitando a execução das operações de gerenciamento que podem ser requisitadas pelo gerente.

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

Figura 9 - Comportamento geral do gateway

A Figura 10 mostra o comportamento do Gateway quando este recebe uma requisição de um gerente para que uma associação seja estabelecida. Como o modelo SNMP não é orientado a conexão, a aplicação gateway deve realizar o mapeamento desta função, devolvendo ao gerente requisitante a resposta adequada, aceitando ou não o estabelecimento desta associação.

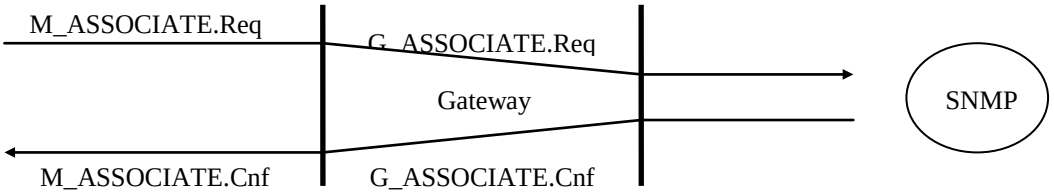

Figura 10 - Estabelecimento da associação com agente SNMP

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 associação com um novo agente é habilitada.

O processo ASSOC_RECOGNIZE_AG reconhece o domínio do agente. Este comportamento caracteriza uma escolha indeterminística. Se o domínio 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).

Quando o domínio 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
```

Figura11 - Instanciação das operações de gerenciamento

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 11).

O gateway realiza o mapeamento das operações GET, SET e TRAP sobre objetos SNMP. O processo OPER_NOTIF (Figura 12) instancia os processos respectivos às operações de gerenciamento em composição paralela independente. Ou seja, qualquer operação pode evoluir de forma independente, podendo ser interrompida apenas por uma requisição ABORT .

```
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_GET !nassoc !agent !domain ;
      G_TRAP[...] (nassoc,agent,noconfirmed,SNMP)
where
     ...
```

Figura 12 - Processo que instancia as operações de gerenc.

Uma operação get (G_GET) é executada sempre no modo confirmado, ou seja, é exigido o retorno de uma resposta ao gerente indicando o resultado da operação. Deste modo, a primitiva de confirmação sempre é enviada, cujos parâmetros devem conter as informações respectivas ao resultado da operação.

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.

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

```
process G_SET[...] ...
      (
       ope !G_SET ... ?pri:operation_gw [... refused] ; exit
                  []
       ope !G_SET ... ?pri:operation_gw [... refused] ;
       (
             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 ... ;
          ...
      )
       )
         >>  ope !G_SET ... ;
           (
             G_SET[...] (nassoc,agent,confirmed,SNMP)
                  []
             G_SET[...] (nassoc,agent,noconfirmed,SNMP)
           )
...
```

Figura 15 - Processo principal da operação G_SET

A implementação das funções de escopo, filtro e sincronização no lado SNMP adicionaram complexidade às operações G_GET e G_SET. O comportamento de ambas é relativamente semelhante, porém o G_SET adiciona algumas tarefas particulares. Por exemplo, no início da execução da operação, ocorre a recuperação dos valores originais dos objetos. Isto permite o restabelecimento dos valores originais destes objetos no caso do insucesso da operação SET.

Quando uma operação é finalizada, esta deve ser novamente disponibilizada. Isto porque 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. Como exemplo, o processo principal da operação G_SET é mostrado na Figura 15. A especificação LOTOS das demais operações são completamente apresentadas em [Mell 97].

O processo principal que representa o comportamento da operação G_SET possibilita a escolha indeterminística entre as diversas variações (Figura 15) da operação. Tais variações têm como principal propósito a possibilidade de combinar as funções de escopo, filtro ou sincronização em uma única requisição de operação.

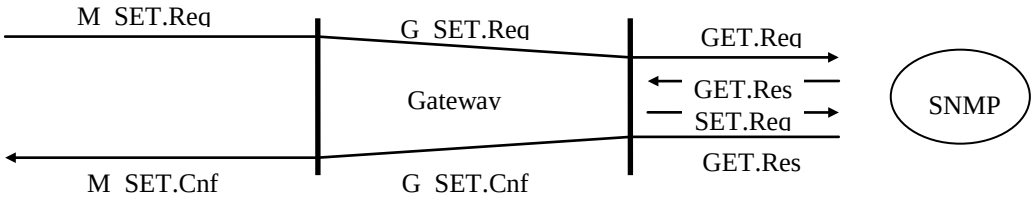

Figura 16 - Operação G_SET no modo confirmado

A Figura 16 representa a operação de gerenciamento SET, quando executada sobre um objeto SNMP, no modo confirmado. Observa-se que, inicialmente, um *GET.request* é executado com o objetivo de buscar os valores originais dos objetos.

A seção seguinte apresenta o trabalho de validação e verificação realizado.

4. Validação e Verificação da aplicação especificada

Nesta seção são apresentados os resultados dos trabalhos de validação e verificação desenvolvidos para a especificação LOTOS do Gateway. Este trabalho pode ser dividido nas tarefas de análise (sintática e semântica), simulação e verificação. A seção 7 apresenta uma análise sobre características de algumas ferramentas para LOTOS utilizadas nestas tarefas, abordando aspectos como a disponibilidade, capacidades, restrições e incompatibilidades.

A sintaxe de uma especificação é bastante simples de ser analisada, já durante a análise semântica, é necessário cuidado, principalmente com relação à consistência interna. Esta consistência é importante, pois uma alteração no código LOTOS da especificação não significa necessariamente que seus efeitos serão restritos ao local onde a alteração foi realizada. Então, 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.

A simulação foi uma das principais técnicas utilizadas durante a construção da especificação LOTOS do Gateway. Com a simulação é possível observar o comportamento dinâmico do sistema, permitindo que erros sejam detectados. Se o modelo for simples (ou pequeno), é possível detectar a maioria dos erros, o que geralmente não acontece quando o modelo especificado é grande e possivelmente complexo. Neste segundo caso, geralmente não há como simular todos os eventos possíveis.

Durante a simulação da especificação aqui apresentada, aspectos relevantes que passaram despercebidos durante a fase de descrição dos requisitos informais foram detectados. Para a correção de alguns destes erros, foram tomadas algumas novas decisões de projeto.

A aplicação Gateway foi especificada em dois níveis distintos de abstração: uma *especificação abstrata e uma especificação detalhada*. A *especificação abstrata* da aplicação trata do fluxo de informações recebidas e enviadas (comportamento observável). Os aspectos internos, principalmente os respectivos ao mapeamento das funcionalidades apresentadas na seção 4 e 5, não são tratados. Já na *especificação detalhada* da aplicação, tais aspectos internos têm seu comportamento representado.

Através de um extensivo trabalho de simulação, foi realizado um trabalho de comparação entre a *especificação abstrata* do sistema (serviço), e da sua *especificação detalhada* (protocolo). Esse trabalho de comparação permitiu analisar a igualdade entre as seqüências de ações realizadas por ambos os níveis de abstração da especificação.

Já a verificação da relação de equivalência entre a *especificação abstrata* do sistema (serviço) e da sua *especificação detalhada* (protocolo) não foi possível para todos os módulos do Gateway devido à vários fatores. Inicialmente, não existência de ferramentas robustas para verificação de especificações que utilizam tipos de dados. Também, as ferramentas existentes impõem restrições que impossibilitam a execução de algumas tarefas.

Para o estabelecimento das relações de equivalência foi utilizado o conjunto de ferramentas CAESAR/Aldébaran. Com este conjunto de ferramentas foram encontradas dificuldades na geração de tipos de dados concretos (em código C) a partir dos tipos abstratos da especificação, bem como na redução do conjunto de valores dos *sorts* visando conter a explosão de estados sem que, contudo, se perdesse a generalidade necessária. Porém, o problema maior que impediu o uso da ferramenta no estabelecimento de relações de equivalência, foi a impossibilidade de definir o domínio do *sort* utilizado na recuperação de valores de objetos gerenciados, o que era absolutamente necessário para se representar o comportamento da aplicação.

A definição do domínio de um *sort* significa o reconhecimento do conjunto de valores possíveis para este *sort*. Isso é necessária para todos os sorts da especificação, pois o LTS (*Labeled Transition Systems*) gerado 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 devem ser finitos.

O *sort* utilizado na recuperação de valores de objetos gerenciados, cuja definição do domínio não foi possível, tem seu conjunto de valores variável. 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 à um evento que envolve este sort é dependente da operação sobre ele realizada. Resumindo, este sort não pode ter seu domínio definido pois é requisito para a especificação que seu conjunto de valores seja dinâmico.

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 haveriam 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.

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.

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 gerada, e não sobre LTS.

Na execução desta verificação, algumas interações com o usuário são requeridas, entre elas, o fornecimento do tamanho desejado de uma tabela a ser gerada. O recurso utilizado foi uma estação SPARCStation 10, com 64 Mb de memória. Todos os trabalhos foram desenvolvidos neste equipamento.

Propriedade de vivacidade (*liveness*) foi verificada através da ferramenta LOLA. Esta propriedade assume que eventos desejáveis eventualmente possam acontecer. Geralmente requerem raciocínio temporal.

Como ilustração, a implementação dos tipos concretos (código C) da aplicação gateway, que implementa os tipos concretos, possui 2122 linhas. Já o código C da especificação completa gerado pela ferramenta CAESAR possui 16.974 linhas.

O trabalho completo de análise, simulação e principalmente verificação da especificação LOTOS da aplicação gateway, abordando as restrições e demais problemas detectados para cada ferramenta utilizada, é apresentado em [Mell 97].

É importante frisar que os resultados obtidos nas tarefas de verificação, assim como nas outras tarefas afins, caracterizam informações úteis que devem ser interpretadas pelo usuário (especificador ou projetista) e revertidas em ações corretivas. Também, a verificação completa pode ser difícil de ser alcançada, principalmente pelo fato de que não há como saber quando exatamente esta tarefa pode ser considerada suficientemente completa. A princípio, sempre é possível adicionar melhorias em um sistema.

5. Experiência no uso de ferramentas para LOTOS

Esta seção apresenta comentários e críticas sobre as ferramentas utilizadas neste trabalho. Não trata-se de uma mostra de ferramentas, mas de uma avaliação das incompatibilidades, capacidades e restrições detectadas nestas ferramentas durante o tratamento da especificação.

5.1 Ferramentas 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. Este ambiente agrupa um conjunto de ferramentas para edição, análise, teste, simulação e verificação de especificações LOTOS.

Neste trabalho, as principais ferramenta 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 2 apresenta uma análise comparativa da funcionalidade de algumas destas ferramentas. Em seguida, são comentadas algumas restrições e incompatibilidades de cada uma delas.

A ferramenta TOPO permite o uso das chamadas anotações, que são usadas para possibilitar a adição de características de implementação, permitindo o uso de mecanismos de controle sobre os módulos C resultantes. Esta ferramenta apresenta as seguintes restrições:

- não permite outra funcionalidade a não ser *noexit*;

- não permite 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.

Tabela 1 - Funcionalidades de algumas ferramentas

	TOPO	SMILE	LOLA	GLOTOS	CAESAR/Aldébaran
LOTOS completo	Suporta	Suporta	Suporta	Suporta	Suporta
Compilação	Compila TDA como reescrita e comport. como LTS	Não	Não	Não	TDA - parte de controle e de dados Comportamento - LTS
Análise (sint. 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' Comport. em '.c'
Represent. gráfica	Não	Não	Não	Diag. de blocos	Não
Teste	Não	Não	Processo de teste	Não	Não

Como a ferramenta SMILE é um simulador simbólico que requer o arquivo com extensão .cr gerado por TOPO, portanto, herda também as restrições desta ferramenta.

A ferramenta LOLA foi desenvolvida como um ambiente experimental para abordagem transformacional para LOTOS. Permite transformar, executar e testar especificações LOTOS. Suas principais aplicações são:

- operações de expansão - calcula simbolicamente todas as possíveis execuções de uma especificação LOTOS, obtendo o LTS;
- operações de teste - calcula a resposta de uma especificação para um teste.

Quanto à ferramenta CAESAR/Aldebaran, na Tabela 2 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. Para isso, deve-se identificar os chamados construtores no código fonte da especificação através das anotações, que são incluídas na especificação no formato de comentários.

A tarefa de geração do código C a partir dos tipos abstratos pode se tornar um tanto complexa, pois em alguns casos, parte da definição deve ser codificada manualmente. Um experimento com a compilação da parte de dados de LOTOS usando a ferramenta CAESAR.ADT é apresentado em [Gara 89].

Uma limitação relevante existente no processo de geração do código C, com o uso da ferramenta CAESAR.ADT, a partir de uma estrutura de dados de uma especificação LOTOS, é o não tratamento de tipos que utilizam parametrização ou atualização. 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 estruturar as especificações, permitir a reusabilidade, entre outras facilidades. De modo geral, tais restrições denunciam uma certa fragilidade em determinados aspectos no uso de ferramentas LOTOS.

Quanto à análise sintática e semântica, foram detectadas incompatibilidades entre as ferramentas TOPO e CAESAR. Por exemplo, 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 a direita nem à esquerda de um operador de paralelismo. Quanto ao operador de habilitação (>>), a ferramenta TOPO não detectou uma chamada recursiva á esquerda deste operador. Já CAESAR acusou tal construção como erro.

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. O 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.

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 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 necessário uma boa estruturação das especificações. Esta estruturação permite um melhor entendimento do sistema, facilitando 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 (projeto LOTOSPhere) para a verificação da especificação LOTOS do Gateway;
- Realizar a implementação final do Gateway a partir de sua especificação detalhada, buscando a automatização de parte deste processo;
- Especificar o Gateway utilizando outra técnica de descrição formal (Estelle por exemplo), realizando uma análise comparativa para os resultados obtidos.

7. Referências Bibliográficas

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

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

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

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

[Eert 92] Eertink, H.: “Executing LOTOS specifications: the SMILE Tool”, Telematics Research Centre, Enschede, Netherlands, 1992.

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

[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.

[Ernb 91] Ernb, P; Fredlund, L; Hansson, H; Jonsson, B; Orava, F; Pehrson, B.: "Guidelines for Specification and Verification of Communication Protocols". Swedish Institute of Computer Science (SICS), report no. 1, 1991.

[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.

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

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

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

[Kirk 94] Kirkwood, C. E.: "Verification of LOTOS Specifications using Term Rewriting Techniques", Phd Thesis, Department of Computing Science, University of Glasgow, June, 1994.

[LITE 92] Caneve, M.; Salvatori, E. (editors): "LITE User Manual", Technical Report Lo/WP2/N))34/V08, The LOTOSPHERE Esprit Project, 1992.

[LFHa 92] Logrippo, L.; Faci, M.; Haj-Hussein, M.: "An Introduction to LOTOS: learning by examples", Computer Networks and ISDN systems, Vol. 23, pp. 325-342, 1992.

[Mell 97] Mello, B. A.: "Utilizando LOTOS na Concepção Formal de uma Aplicação para Gerência de Redes: Especificação e Verificação", Dissertação de Mestrado a ser defendida em fevereiro/97, Departamento de Informática e Estatística - UFSC, 1997.

[Miln 80] Milner, R.: "A Calculus of Communication Systems", Lecture Notes in Computer Science, No. 92, Springer Verlag, 1980.

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

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

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

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