

UNIVERSIDADE FEDERAL DE SANTA CATARINA  
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

UMA FERRAMENTA PARA ESPECIFICAÇÃO DE PROTOCOLOS DENTRO DO  
CONTEXTO ESTELLE: UM TRADUTOR ASN.1/ESTELLE

DÍSSERTAÇÃO SUBMETIDA A UNIVERSIDADE FEDERAL DE SANTA CATARINA  
PARA OBTENÇÃO DO GRAU DE  
MESTRE EM ENGENHARIA ELÉTRICA



0.205.302-6

UFSC-BU

MARIA MARTA LEITE

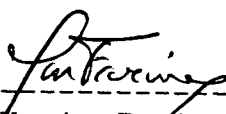
FLORIANÓPOLIS, ABRIL 1992

UMA FERRAMENTA PARA ESPECIFICAÇÃO DE PROTOCOLOS DENTRO DO  
CONTEXTO ESTELLE: UM TRADUTOR ASN.1/ESTELLE.


MARIA MARTA LEITE

ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA OBTENÇÃO DO TÍTULO DE  
MESTRE EM ENGENHARIA ELÉTRICA

ESPECIALIDADE CONTROLE E AUTOMAÇÃO INDUSTRIAL E APROVADA EM SUA  
FORMA FINAL PELO PROGRAMA DE PÓS-GRADUAÇÃO



-----  
Prof. Jean-Marie Farines, Dr.Ing.  
ORIENTADOR

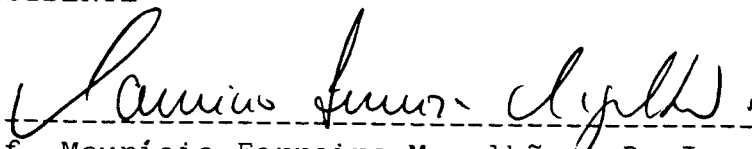


-----  
Prof. João Pedro Assumpção Bastos, Dr.Etát  
COORDENADOR DO CURSO

BANCA EXAMINADORA



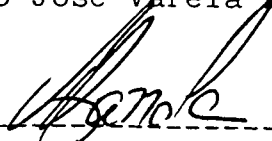
-----  
Prof. Jean-Marie Farines, Dr.Ing.  
PRESIDENTE



-----  
Prof. Mauricio Ferreira Magalhães, Dr.Ing.



-----  
Prof. Olinto José Varela Furtado, M.Sc.



-----  
Prof. Vitório Bruno Mazzola, Dr.

*Ao Paulinho*

## AGRADECIMENTOS

Ao Professor Jean-Marie Farines pela orientação e oportunidade de realização deste trabalho.

Aos meus queridos amigos Carmen e Paulo Lacerda, Elizabeth, Leandro, Olinto e De Lucca pela companhia e inestimável incentivo.

Ao meu marido Paulo pelo constante apoio e carinho.

Aos meus filhos Danilo e Morgana pelos seus doces sorrisos.

Aos meus pais pela força e dedicação.

A todos que de forma direta ou indireta contribuíram para a realização deste trabalho.

# ÍNDICE

RESUMO..... 1

ABSTRACT..... 3

CAPÍTULO 1: INTRODUÇÃO..... 5

CAPÍTULO 2: A UTILIZAÇÃO DE TÉCNICAS DE DESCRIÇÃO FORMAL  
PARA PROTOCOLOS DE COMUNICAÇÃO

2.1 Introdução..... 8

2.2 A Arquitetura de Sistemas Interconectados..... 8

2.3 A Concepção de Sistemas de Comunicação..... 10

    2.3.1 Aspectos a serem Especificados..... 11

    2.3.2 As Técnicas de Descrição Formal..... 13

    2.3.3 As Etapas de Concepção de Protocolos..... 15

2.4 Conclusão..... 17

CAPÍTULO 3: A TÉCNICA DE DESCRIÇÃO FORMAL ESTELLE

3.1 Introdução..... 19

3.2 A Utilização de Técnicas de Descrição Formal..... 19

3.3 Principais Características Estelle..... 21

    3.3.1 Conceitos Básicos..... 21

    3.3.2 Estruturação..... 22

    3.3.3 Comunicação..... 28

    3.3.4 Paralelismo..... 29

3.4	A Representação dos Dados em Estelle.....	30
3.4.1	Discussão sobre o uso de Pascal em Estelle.....	31
3.5	Ferramentas para Simulação Estelle.....	32
3.6	Conclusão.....	32
CAPÍTULO 4: A NOTAÇÃO PARA SINTAXE ABSTRATA ASN.1		
4.1	Introdução.....	34
4.2	O Problema da Representação de Dados.....	34
4.3	As Estruturas de Dados ASN.1.....	38
4.3.1	Regras para a Notação.....	38
4.3.2	As Estruturas de Dados.....	39
4.3.2.1	Os Tipos Primitivos.....	39
4.3.2.2	Os Tipos Construtores.....	42
4.3.2.3	Os Tipos Cadeia de Caracteres.....	44
4.3.2.4	Os Tipos Úteis.....	44
4.3.2.5	Os Atributos Especiais.....	45
4.3.2.6	A Utilização de Rótulos.....	45
4.4	A Codificação de Valores ASN.1.....	46
4.4.1	Codificação do Tipo.....	47
4.4.2	Codificação do Tamanho.....	48
4.4.3	Codificação do Conteúdo.....	49
4.5	Conclusão.....	49
CAPÍTULO 5: A INTEGRAÇÃO ASN.1/ESTELLE		
5.1	Introdução.....	50
5.2	Motivações para a Integração.....	50

5.3	Abordagens para a Integração.....	52
5.4	Premissas para a Tradução.....	55
5.4.1	Informações Relacionadas a Codificação de Valores.....	55
5.4.2	As Restrições da Linguagem Estelle.....	57
5.4.3	As Dificuldades de Relacionamento de Estruturas.....	57
5.5	Proposta de Tradução ASN.1/Estelle.....	58
5.5.1	Estruturas ASN.1 que Permitem Tradução Direta.....	58
5.5.2	Estruturas ASN.1 que Necessitam de Formas Especiais de Tradução.....	62
5.6	Considerações Finais.....	68
5.7	Conclusão.....	69

## CAPÍTULO 6: DESENVOLVIMENTO DA FERRAMENTA DE TRADUÇÃO

6.1	Introdução.....	70
6.2	As Limitações da Ferramenta de Tradução.....	70
6.2.1	Sensibilidade ao Contexto.....	71
6.2.2	Não Obrigatoriedade de Nomes de Referência... ..	72
6.3	Etapas de Desenvolvimento.....	74
6.4	As Ferramentas de Implementação.....	76
6.4.1	A Ferramenta LEX.....	77
6.4.2	A Ferramenta YACC.....	77
6.5	A Utilização da Ferramenta num Ambiente de Concepção Estelle.....	79
6.6	Conclusão.....	81

CAPÍTULO 7: CONCLUSÕES.....	83
REFERÊNCIAS BIBLIOGRÁFICAS.....	85
APÊNDICE 1.....	92
APÊNDICE 2.....	95
APÊNDICE 3.....	103



## RESUMO

As especificações de protocolos e serviços fornecidas pelas normas ISO são geralmente descritas em linguagem natural, com máquinas de estado associadas, e as PDUs ("Protocol Data Unit") apresentadas através da notação ASN.1 ("Abstract Syntax Notation One").

A notação ASN.1 visa compatibilizar e tratar as diversas formas de representação da informação em sistemas heterogêneos. Através dela é possível definir estruturas de dados complexas de forma compacta e abstrata. Esta notação é utilizada principalmente para representação de dados dos protocolos da camada de aplicação do modelo de referência OSI/ISO.

Por outro lado, a utilização de linguagem natural para especificar serviços e protocolos pode levar a ambigüidades, incompletitude e inconsistência destes, e conseqüentemente a implementações que apresentam comportamentos indesejados. O uso de Técnicas de Descrição Formal (TDF) permite evitar estes problemas, além de facilitar a construção de ferramentas de concepção automatizadas. A ISO propôs e normalizou duas técnicas de descrição formal, denominadas Estelle ("Extended State Transition Language") e LOTOS ("Language of Temporal Ordering Specification").

Neste trabalho nos interessamos particularmente pela linguagem de especificação Estelle por ser de fácil uso e por dispor de ferramentas automatizadas para simulação, verificação e geração de código. Entretanto, a estrutura de dados da

linguagem Estelle é praticamente a mesma da linguagem Pascal, não sendo muito adequada para a representação de PDUs.

Esta dissertação apresenta uma proposta de integração da Notação de Sintaxe Abstrata ASN.1 com a Técnica de Descrição Formal Estelle. A integração proposta baseia-se numa abordagem de tradução de estruturas definidas através de ASN.1 para estruturas Estelle. A integração destas duas linguagens foi implementada através de uma ferramenta de tradução ASN.1/Estelle. Esta ferramenta deve ser incluída num ambiente de desenvolvimento e implementação de especificações Estelle. Seu principal objetivo é auxiliar a tarefa de especificação de protocolos em Estelle quando as unidades de dados do protocolo são definidas através da notação ASN.1.

## ABSTRACT

Protocol and service specifications provided by ISO rules are usually described in natural language associated with state machines and PDU (Protocol Data Units) presented in ASN.1 (Abstract Syntax Notation One).

Notation ASN.1 aims to provide compatibility and to deal with the variety of kinds of information representation in heterogeneous systems. It makes it possible to define complex data structures in a compact and abstract manner. This notation is used mainly to represent data of protocols of the application layer of the OSI/ISO reference model.

On the other hand, the use of natural language to specify services and protocols may lead to ambiguities, incompleteness and inconsistencies, and consequently to implementations of undesirable behavior. The use of Formal Description Techniques (FDT) allows the avoidance of these problems and enables the construction of automatic conception tools. ISO has proposed and normalized two Formal Description Techniques denoted Estelle (Extended State Transition Language) and LOTOS (Language of Temporal Ordering Specification).

In this work we are particularly interested in the specification language Estelle because it is easy to use and has got automatic tools for simulation, verification and code generation. Nevertheless, Estelle's data structure is almost the same as Pascal's, and not quite adequate for PDU representation.

This dissertation proposes an integration of the ASN.1 Abstract Syntax Notation with Estelle Formal Description Technique. The integration proposed is based in an approach for translation of structures defined in ASN.1 for Estelle structures. The integration of these languages was implemented in a tool for ASN.1/Estelle translation. This tool will make part of an environment for development and implementation of Estelle specifications. Its main objective is to aid the job of protocol specification in Estelle when the protocol data units are defined in ASN.1 notation.

## Capítulo 1

### INTRODUÇÃO

O uso de Sistemas Distribuídos tem sido, nestes últimos anos, a grande tendência na informática. A larga utilização das redes locais e de longa distância evidencia esta tendência. Os Sistemas Distribuídos proporcionam facilidades que vão desde o compartilhamento de recursos até a melhoria no desempenho, devido ao paralelismo.

Para que a comunicação entre sistemas interconectados seja realizada de forma segura e confiável é necessária a definição das regras de diálogo, ou *protocolos*, que definem os mecanismos para a troca de dados e sincronização entre os componentes de um sistema. Estas necessidades levaram a ISO ("International Standard Organization") a propor o Modelo de Referência para Interconexão de Sistemas Abertos ("Reference Model - Open System Interconnection"), ou RM-OSI [ISO 84].

Para garantir a completitude e não ambigüidade destes protocolos têm sido empregadas Técnicas de Descrição Formal (TDF) para sua especificação. O grupo FDT ("Formal Description Techniques") da ISO foi criado com o objetivo de estudar e propor técnicas de descrição formal que possam ser aplicadas aos protocolos e serviços sendo padronizados. Atualmente, as técnicas de descrição formal padronizadas são Estelle [ISO 89] e LOTOS [ISO 88]. Outrossim, o uso de abordagens formais, desde a fase de especificação até a fase de implementação da mesma, possibilita a verificação de correção da implementação e das

propriedades do sistema especificado. Além disso, um sistema descrito formalmente pode servir de referência comum para implementações em ambientes distintos e também para testar estas implementações.

Outro ponto importante em Sistemas Distribuídos é a forma de representação dos dados, ou seja, o modo como eles são codificados para transmissão e decodificados na recepção em diferentes sistemas. Cada sistema tem sua forma própria de representação de dados e é necessária a compatibilização entre elas. Para que seja dada a mesma interpretação para os dados trocados entre diferentes sistemas são necessários mecanismos comuns para descrição dos dados. Com o objetivo de padronizar uma forma de especificação e codificação das unidades de dados trocadas por sistemas interconectados, foi definida a notação ASN.1 ("Abstract Syntax Notation One") [ISO 87a] [ISO 87b], que inclui um conjunto de regras básicas de codificação. Esta notação está sendo utilizada principalmente para definir as unidades de dados nas normas de padronização dos protocolos do nível de aplicação do modelo OSI.

Neste contexto, o uso de ferramentas automatizadas para as várias etapas do desenvolvimento de protocolos e de ambientes de concepção que as integrem, é uma necessidade. Isto serve para tratar de forma mais econômica o problema da concepção de sistemas de comunicação, cada vez mais complexos.

O presente trabalho apresenta uma proposta de integração da notação abstrata ASN.1 com a técnica de descrição formal Estelle. Esta integração se fará através de um tradutor de especificações ASN.1 para definições em Estelle. A ferramenta de tradução será integrada a um ambiente de concepção Estelle. Sua

utilização se justifica principalmente pelo crescente uso, tanto de técnicas de descrição formal, tal como Estelle, quanto da notação ASN.1, para a concepção de sistemas de comunicação.

No capítulo 2 é discutido o uso de técnicas de descrição formal para protocolos de comunicação. Nos capítulos 3 e 4 são apresentadas, respectivamente, a técnica de descrição formal Estelle e a notação ASN.1. A seguir, no capítulo 5, são discutidas algumas das possíveis abordagens para integração ASN.1/Estelle, destacando-se a abordagem escolhida. Nesta etapa são também apresentadas as correspondências entre estruturas ASN.1 e estruturas Estelle, que servirão de base para o tradutor desenvolvido. Enfim, no capítulo 6, são apresentados alguns tópicos relevantes sobre a implementação do tradutor, assim como o ambiente no qual ele se insere.

## Capítulo 2

### A UTILIZAÇÃO DE TÉCNICAS DE DESCRIÇÃO FORMAL PARA PROTOCOLOS DE COMUNICAÇÃO

#### 2.1 Introdução

Em sistemas interconectados as diversas funções estão localizadas em componentes distintos na rede. A comunicação entre estes componentes é feita através de *protocolos de comunicação*. A definição precisa destes protocolos de comunicação facilita a sua implementação. Para melhor atingir este objetivo deve-se ter um cuidado especial com a especificação de protocolos.

Neste capítulo serão apresentados conceitos relativos às atividades de especificação, validação e implementação de protocolos. Dentre estas atividades é discutida mais detalhadamente a especificação de protocolos e o uso de técnicas formais.

#### 2.2 A Arquitetura de Sistemas Interconectados

A grande utilização de sistemas de computação e a necessidade de compartilhamento de recursos computacionais foram os principais fatores que geraram a necessidade de interligação de máquinas, muitas vezes heterogêneas. Devido à complexidade na



ligação em rede destes equipamentos, é necessário que exista uma metodologia para especificar as funções de comunicação.

A arquitetura do modelo de referência OSI da ISO [ISO 84] é apresentada na figura 2.1. Este modelo tem sido largamente utilizado e é reconhecido, dentro do domínio de redes heterôgeneas, como uma base eficiente para especificar e implementar, com sucesso, sistemas muito complexos.

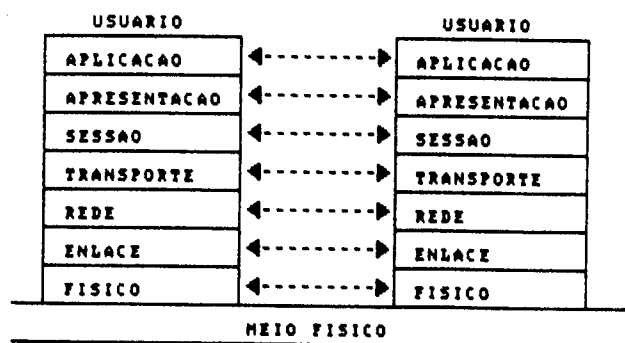


Figura 2.1 - A arquitetura do RM-OSI

Para este modelo hierárquico são utilizados os conceitos de *Serviços* e *Protocolos*. Um *Protocolo* constitui-se de um conjunto de regras que as entidades comunicantes devem seguir para que a comunicação entre elas possa ocorrer de forma adequada. *Entidades* são elementos ativos em cada nível hierárquico e entidades de mesmo nível em máquinas diferentes são chamadas entidades pares. Um *Serviço* é um conjunto de funcionalidades que são fornecidas a um usuário.

A função real de cada nível, ou camada, do modelo OSI é proporcionar serviços ao nível superior. Um nível genérico (N) oferece um serviço (N) ao nível (N+1) e usa para tal os serviços

fornecidos pelo nível (N-1). A figura 2.2 ilustra uma parte da arquitetura em níveis e de seus relacionamentos.

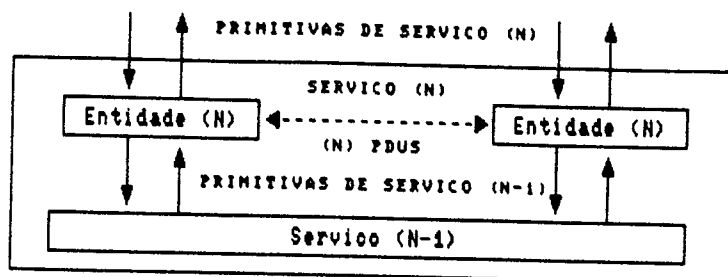


Figura 2.2 - Arquitetura em níveis

### 2.3 A Concepção dos Sistemas de Comunicação

Conceber um sistema de comunicação interconectado consiste em: especificar um modelo arquitetural e um conjunto de classes de serviços/protocolos; validar estes serviços e protocolos e suas interfaces (protocolo (N)/serviço (N-1)); implementar os serviços/protocolos validados; e, finalmente, testar a conformidade desta implementação.

De uma maneira geral, a especificação de um sistema comporta dois aspectos: um *qualitativo*, que consiste em expressar os mecanismos e funções que o sistema deve apresentar, e um *quantitativo* que descreve as restrições de desempenho e de segurança.

Um ponto comum dentro da concepção de qualquer sistema interconectado é a necessidade de definir de maneira completa e não ambígua os protocolos de comunicação e a sincronização entre os componentes deste sistema. Os primeiros estudos sobre a

concepção de protocolos abordavam de forma independente os problemas de especificação e validação e o problema de geração automática de implementações. A introdução de métodos formais de descrição permitiu a geração de ambientes que, partindo da especificação e através de ferramentas automatizadas, permitem chegar à implementação.

### 2.3.1 Aspectos a Serem Especificados

No contexto do modelo de referência OSI, é necessário especificar:

- os *serviços de comunicação* oferecidos pelos níveis;
- os *protocolos* de cada entidade de um dado nível;
- a *interface* através da qual o serviço de comunicação é oferecido ao usuário deste serviço.

A *especificação do serviço* de um nível (N) descreve o serviço de comunicação proporcionado às entidades usuárias do nível (N+1). A *especificação do protocolo* de nível (N) descreve as regras de comportamento para cada entidade de protocolo de nível (N). Ela define o comportamento de uma entidade de protocolo em termos de suas interações com o usuário do serviço local, chamadas *primitivas de serviço*, e as interações trocadas com a entidade remota, chamadas *unidades de dados do protocolo* ou *PDUs*.

Cada *primitiva de serviço* usualmente inclui vários parâmetros que são trocados entre os usuários do serviço e as entidades de protocolo durante a execução desta primitiva. As

*PDU*s não são trocadas diretamente com a entidade par, elas são codificadas e enviadas através dos níveis subjacentes até a entidade par em outra máquina. A figura 2.3 representa a PDU em uma comunicação.

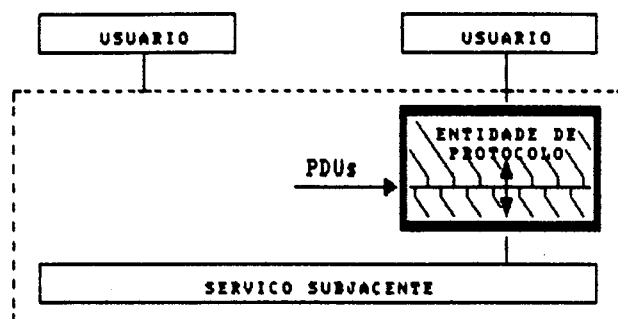


Figura 2.3 - O papel da PDU na comunicação

A parte de cima da representação da entidade de protocolo, na figura anterior, corresponde às suas propriedades abstratas e a parte de baixo corresponde ao mapeamento da PDU em primitivas do serviço de comunicação subjacente.

Pode-se destacar alguns dos aspectos relevantes que devem ser considerados na especificação, segundo [Bochmann 90a]:

- (a) *ordenação temporal das interações;*
- (b) *conjunto de valores possíveis para os parâmetros de interação: primitivas de serviço e PDUs;*
- (c) *regras para interpretar e seleccionar valores de parâmetros de interação: esta regra representa um refinamento da regra (b); ela define, (i) para cada interação que pode ser iniciada pela entidade de protocolo, os valores de parâmetros permitidos, como uma função das interações passadas desta entidade e,*

- (ii) para cada interação recebida, o significado dos valores de parâmetro recebido para o processamento subsequente da entidade de protocolo;
- (d) *codificação das PDUs e de seus parâmetros*: para transmissão através do serviço de comunicação;
- (e) *propriedades de vivacidade ("liveliness")*: ou propriedades qualitativas, especificam que certas interações devem sempre ocorrer para que haja sentido na comunicação, ou seja, assumem que eventos desejados possivelmente ocorrerão; elas são definidas em contraste com as propriedades de *segurança* (aspectos (a) até (d) citados anteriormente), que asseguram que eventos indesejáveis não ocorrerão;
- (f) *propriedades de tempo-real*: são propriedades quantitativas relacionadas com atrasos na comunicação, desempenho, confiança, tempo entre falhas e probabilidades de requerimentos de serviço recusados.

Neste trabalho nos interessamos mais particularmente pelos aspectos (c) e (d) no que diz respeito à codificação e interpretação das PDUs.

### 2.3.2 As Técnicas de Descrição Formal

A especificação de protocolos apresenta-se, usualmente, na forma de uma descrição em linguagem natural acrescida de máquinas de estado. O uso de linguagem natural pode levar a ambigüidades, inconsistências ou a não-completitude nas

especificações e, conseqüentemente, a implementações de protocolos com comportamentos indesejados ou não esperados sem possibilidade de correção destes problemas no decorrer do processo de concepção. Em conseqüência, tornou-se fundamental utilizar técnicas formais de especificação.

Com os trabalhos de padronização dos protocolos do modelo OSI, foi reconhecida a necessidade de utilização de especificação formal. De trabalhos da ISO e do CCITT, com respeito às Técnicas de Descrição Formal, surgiram três linguagens de especificação: Estelle [ISO 89], LOTOS [ISO 88] e SDL [CCITT 84]. Estelle e LOTOS vem sendo utilizadas a nível acadêmico, e mais recentemente a nível industrial, como resultado do esforço dispendido na concepção de ferramentas para geração automática de implementações.

Muitas técnicas de descrição formal foram propostas para especificação de sistemas complexos e em particular os de comunicação. Os formalismos mais importantes são os seguintes:

- (1) máquinas de estado finito [Bochmann 78] [Danthine 80]
- (2) redes de Petri [Danthine 77] [Diaz 82,83] [Courtiat 84] [Ayache 85]
- (3) cálculos algébricos (CCS) [Milner 80]
- (4) gramáticas formais [Haragozo 78]
- (5) linguagens de programação de alto nível
- (6) tipos abstratos de dados
- (7) lógica temporal

As linguagens de especificação hoje normalizadas são extensões aos formalismos (1), no caso de Estelle e SDL, e (3), no caso de LOTOS. A estes formalismos foi acrescida a

representação de dados utilizada em linguagens de programação clássicas (Pascal, por exemplo) ou tipos abstratos algébricos. Constata-se que os aspectos de *codificação de PDUs* não são tratados diretamente por nenhum destes formalismos. Com o objetivo de ter uma forma abstrata e simples para descrição de PDUs foi também definida a notação ASN.1 [ISO 87a] [ISO 87b]. Esta notação permite definir regras para codificação e decodificação de dados. Neste trabalho a preocupação principal vai no sentido de integrar a notação ASN.1 com as linguagens de especificação existentes.

### 2.3.3 As Etapas da Concepção de Protocolos

Vimos que somente uma abordagem fundamentada sobre uma técnica de descrição formal permite que as atividades inerentes à concepção de protocolos possa ser realizada de forma automatizada ou semi-automatizada. As principais atividades a serem desenvolvidas, após a especificação formal, são:

- validação: consiste em demonstrar que um protocolo oferece realmente o serviço desejado; esta atividade pode incluir operações de verificação e simulação;
- implementação: consiste em obter um código executável a partir da especificação validada, se possível, de forma automatizada ou semi-automatizada.

Alguns métodos tem sido utilizados para *validação* de protocolos. Eles se subdividem em: *análise estática* e *análise*

*dinâmica*. Outrossim a *análise quantitativa* é ainda possível para estudar as questões de desempenho e segurança.

A *análise estática* está baseada na análise do texto da especificação. A *análise dinâmica* leva em conta também a execução do sistema. Devido ao grande número de situações que podem ocorrer durante a execução de um sistema, este método é muito mais difícil de implementar, porém a possibilidade de encontrar os erros é muito maior. Os métodos dinâmicos são classificados em *exaustivo* e *não exaustivo*. Os métodos exaustivos consideram todas as situações que podem ocorrer durante a execução do sistema especificado. Estão classificados nesta categoria os métodos de análise de alcançabilidade exaustiva, a verificação de asserções, e a prova de teoremas. Uma restrição destes tipos de métodos é a dificuldade de sua aplicação em especificações muito grandes. Por outro lado, os métodos *não exaustivos* validam apenas alguns caminhos selecionados entre todos os possíveis para execução.

O desenvolvimento de implementações é realizado seguindo vários passos que iniciam pela especificação do protocolo a ser implementado. O processo de geração automática de código pode ser parcial ou totalmente automatizado. Os seguintes tipos de ferramentas podem ser utilizadas [Bochmann 90a]:

- geração automática de código fonte para especificações em alguma TDF;
- geração automática de rotinas de codificação/decodificação de especificações de PDUs descritas em ASN.1;
- pacotes de suporte para gerenciamento de "buffers", comunicação inter-tarefas, entre outros.



Para validar uma implementação através de *testes*, é importante selecionar alguns *casos de testes* para cobrir todos os aspectos de comportamento a serem testados. A maioria das ferramentas existentes para a automatização da seleção de casos de teste, para uma dada especificação de protocolo, está baseada nas especificações formais obtidas.

Neste trabalho nos interessamos particularmente pela linguagem de especificação Estelle por ser largamente utilizada e por dispor de ferramentas automatizadas para simulação, verificação e geração de código. É preocupação também o estudo de técnicas e ferramentas que apoiam a geração de PDUs representadas em ASN.1.

#### 2.4 Conclusão

A vantagem em utilizar técnicas de descrição formal é a possibilidade de utilizar métodos automáticos ou semi-automáticos para o desenvolvimento dos protocolos de comunicação. Algumas destas técnicas possuem um grande número de ferramentas automatizadas já desenvolvidas e disponíveis.

É importante notar que o modelo de máquina de estado finito e a notação ASN.1 cobrem aspectos diferentes da especificação. Enquanto o modelo de máquina de estado finito descreve os estados "principais" e tipos de interações de entrada e saída, ignorando parâmetros, a notação ASN.1 descreve os possíveis conjuntos de valores de parâmetros de interação e a codificação destes parâmetros em PDUs. A junção destes dois aspectos é

objeto deste trabalho. Antes de apresentar esta etapa de junção, será apresentada a seguir a linguagem de especificação Estelle, que é baseada em máquinas de estado, e a notação ASN.1, que serão utilizadas neste trabalho.

## Capítulo 3

### A TÉCNICA DE DESCRIÇÃO FORMAL ESTELLE

#### 3.1 Introdução

Neste capítulo são apresentados os principais conceitos da Técnica de Descrição Formal Estelle, conforme o documento de padronização [ISO 89].

Após mencionar alguns aspectos sobre a utilização de técnicas de especificação formal, são descritas as principais características da linguagem Estelle no que se refere a estruturação, comunicação e paralelismo. Finalmente é apresentada e discutida a utilização da linguagem Pascal para representação de dados nesta linguagem.

#### 3.2 A Utilização de Técnicas de Descrição Formal

A especificação de protocolos de comunicação, e dos sistemas interconectados em geral, apresenta um elevado grau de complexidade. Esta complexidade existe devido à necessidade de descrever vários componentes sequenciais que podem ser executados em paralelo e cooperar entre si, o que leva a um grande número de estados. Os protocolos de comunicação devem ser especificados de forma completa, precisa e sem ambigüidades; sua

implementação deve ser conforme a especificação para que o mesmo apresente o comportamento desejado.

Com o aumento na complexidade e no número de protocolos de comunicação, surgiu a necessidade de especificações mais precisas para descrevê-los. No Modelo de Referência OSI/ISO [ISO 84] os protocolos são estruturados em um conjunto de níveis. Cada nível deste modelo tem interfaces bem definidas para os níveis acima e abaixo do mesmo. Dentro de cada nível são definidas entidades e estas utilizam-se de um protocolo para comunicarem-se umas com as outras. Uma especificação de protocolo deve consistir da especificação de interfaces e entidades de protocolo. A figura 3.1 ilustra estes conceitos.

A ISO desenvolveu e padronizou duas técnicas de descrição formal: LOTOS e Estelle. LOTOS ("Language of Temporal Ordering Specification") [ISO 88] está baseada em CCS ("Calculus of Communicating Systems"), para definir a parte de controle, e nos tipos abstratos de dados de ACT ONE para definir as estruturas de dados. Estelle ("Extended State Transition Language") [ISO 89] está baseada em um modelo de máquinas de estado finita estendida que se comunicam por filas FIFO, em sua versão padronizada, e utiliza um subconjunto da linguagem Pascal para representar dados e descrever ações. Estas técnicas foram desenvolvidas com o objetivo de especificar formalmente sistemas abertos concorrentes e interconectados.

Conforme já mencionado, nosso interesse pela linguagem Estelle está baseado em sua larga utilização e devido a existência de várias ferramentas de apoio às fases de concepção de protocolos quando esta linguagem é utilizada.

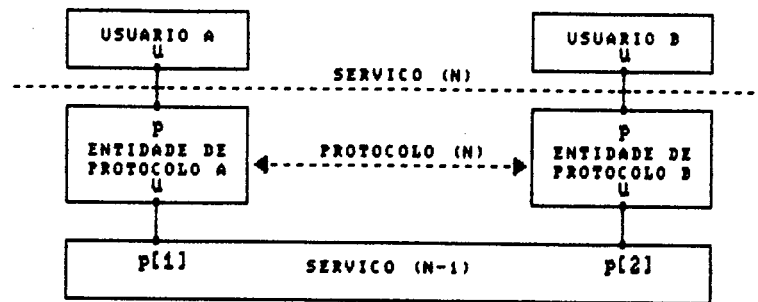


Figura 3.1 - Entidades de protocolos e suas interfaces

### 3.3 Principais Características Estelle

#### 3.3.1 Conceitos Básicos

Uma especificação Estelle possui algumas características básicas [Courtiat 87b]:

- as especificações são constituídas por módulos e a linguagem possibilita a definição clara das interfaces entre estes módulos;
- a descrição do comportamento de cada módulo é suficientemente refinada e precisa de modo a garantir a inexistência de ambigüidades, e que qualquer ambigüidade em si, como por exemplo o não-determinismo, seja explícita;
- o conceito de tipagem (definição do tipo, seguido de criação de cópias) da linguagem Pascal é estendido às

estruturas Estelle, como por exemplo módulos e canais, e algumas delas são parametrizáveis.

### 3.3.2 Estruturação

Serão introduzidos a seguir conceitos relacionados a estruturação em Estelle tais como os de módulo, de ponto de interação, de canais de comunicação e de instância de módulo [Budkowski 87] [Linn 86].

#### O MÓDULO

O componente básico de uma especificação Estelle é o módulo ("module"). Em sua estrutura um módulo pode conter a definição de sub-módulos e de suas interconexões. Esta regra aplicada repetidamente leva a uma estrutura hierárquica de definição de módulos. A figura 3.2 [Budkowski 87] ilustra o conceito de relacionamento pai/filho entre módulos. A raiz da árvore, na figura 3.2 (a), ou a maior caixa externa, na figura 3.2 (b), representa o módulo principal, ou seja, o sistema especificado. As caixas B e C são módulos filhos do módulo A. Os módulos D, E e F são módulos filhos de B, e G e H são módulos filhos de C.

A definição do corpo de módulo ("module body") consiste de um conjunto de transições. Um módulo está ativo se sua especificação inclui ao menos uma transição, caso contrário ele está inativo.

Um módulo Estelle pode possuir um dos seguintes atributos de classe ("system-class"):

- "systemprocess" ou "process"
- "systemactivity" ou "activity"

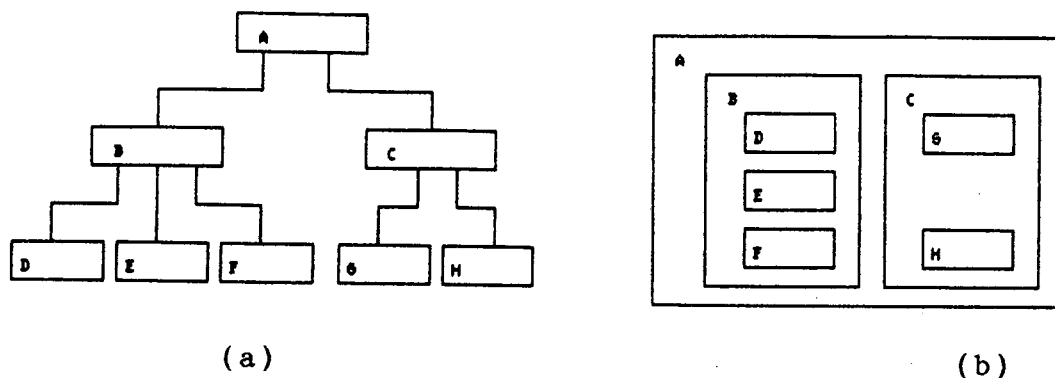


Figura 3.2 - Estrutura de módulos Estelle

Módulos atribuídos "systemprocess" ou "systemactivity" são chamados módulos sistema. Estes atributos estão relacionados à semântica do paralelismo em Estelle que será descrito no item 3.3.4.

Todos os módulos definidos numa especificação devem ser envolvidos por um módulo principal ("specification module") cuja sintaxe está definida no ítem A1.1 do Apêndice 1.

Dentro da hierarquia de módulos os seguintes princípios devem ser obedecidos:

- cada módulo ativo deve ter seu atributo;
- módulos sistema não podem ser aninhados dentro de módulos com atributos;
- módulos com atributos "process" ou "activity" devem ser aninhados dentro de um módulo sistema;
- módulos com atributos "process" ou "systemprocess" podem ser sub-estruturados apenas em módulos com atributos "process" ou "activity";

- módulos com atributos "activity" ou "systemactivity" podem ser sub-estruturados apenas em módulos com atributos "activity".

## OS PONTOS DE INTERAÇÃO

Os sistemas especificados em Estelle são vistos como uma coleção de componentes que se comunicam por pontos de interação ("ip") [Budkowski 87] [Linn 86].

Um ponto de interação é uma interface abstrata usada para troca de mensagens (interações) entre módulos. A cada ponto de interação é associada uma fila FIFO não limitada. Estas filas armazenam interações recebidas de outros módulos. As filas podem ser definidas ("queue-discipline") como "individual", se pertence a apenas um ponto de interação, ou "common", se é compartilhada por vários pontos de interação de um módulo.

Os pontos de interação podem ser internos ou externos. Um módulo pode ser representado graficamente por um retângulo com pontos em seus limites, representando pontos de interação externos, e pontos internos, representando pontos de interação internos. A figura 3.3 [Budkowski 87] ilustra a utilização de pontos de interação e apresenta uma possível configuração de atributos de classe para a arquitetura apresentada na figura 3.2.

Além da estrutura hierárquica, uma estrutura de comunicação é definida na especificação. Apenas dois tipos de ligação entre módulos são permitidas: "attach" e "connect". Dois pontos de interação devem ser ligados por um comando "attach" quando um deles é um ponto de interação externo do módulo pai e o outro é um ponto de interação externo do módulo filho. Dois pontos de



interação devem ser conectados por um comando "connect" se eles ligam um ponto de interação interno de um módulo pai a um ponto de interação externo de seu filho, dois pontos de interação externos de dois módulos irmãos ou ainda dois pontos de interação internos de um módulo. A figura 3.3 ilustra também as ligações entre pontos de interação.

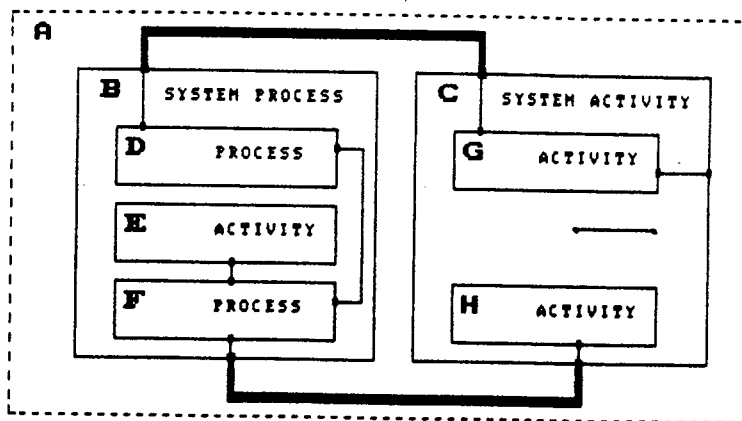


Figura 3.3 - Ligação entre módulos e uso de atributos de classe em Estelle

Em determinado instante, um ponto de interação pode estar ligado por um "connect" a, no máximo, um ponto de interação e não pode estar ligado por um comando "connect" a si mesmo. Um ponto de interação externo de um módulo pode estar ligado através de um comando "attach" a no máximo um ponto de interação externo de seu módulo pai e a no máximo um ponto de interação do módulo filho. Um ponto de interação externo que está ligado por um "attach" a um ponto de interação externo de seu módulo pai não pode estar simultaneamente ligado por um "connect".

Uma ligação ("link") entre dois pontos de interação é uma linha contínua composta por exatamente um segmento "connect" e zero ou mais segmentos "attach".

Uma vez que módulos que incluem dentro de si outros módulos *sistema* são sempre inativos, a estrutura destes sub-sistemas e suas ligações de comunicação, uma vez inicializadas, não podem ser trocadas, constituindo uma estrutura *estática*. Estas ligações podem ser visualizadas pelas linhas cheias na figura 3.3. No entanto, a estrutura interna de cada sub-sistema pode variar, isto porque as ações de um módulo podem incluir comandos criando e destruindo instâncias de módulo filhos e suas ligações, sendo portanto uma estrutura *dinâmica*.

Existem alguns outros atributos relacionados a um ponto de interação tais como papel (ponto de interação de entrada ou de saída) e tipo de canal (identificador associado).

## OS CANAIS DE COMUNICAÇÃO

A definição de canal ("channel") em Estelle especifica um conjunto de interações que podem ser trocadas entre módulos [Budkowski 87] [Linn 86]. Os pontos de interação, através dos quais mensagens entram e saem dos módulos, referem-se a canais de um tipo específico. Um tipo de canal é especificado em duas partes:

- identificador e papel ("role") do canal
- declaração de todas as possíveis interações associadas a cada papel

A sintaxe da definição de um canal de comunicação é dada no ítem A1.2 do Apêndice 1.

## AS INSTÂNCIAS DE MÓDULO

Uma instância de módulo (ou tarefa) é uma instância de um tipo de módulo genérico e várias instâncias de um mesmo tipo de módulo podem existir simultaneamente em uma especificação. Assim, uma especificação Estelle é constituída por um conjunto de módulos, cujo comportamento interno é determinado por um autômato de transição de estados, eventualmente não determinístico, chamado instância de módulo. Deve existir apenas uma instância do módulo principal.

Os componentes primários na definição de um módulo são o cabeçalho ("module header") e o corpo ("module body"). Cada instância de um mesmo tipo de módulo apresenta a mesma *visibilidade externa*, caracterizada pela definição do cabeçalho do módulo, e o mesmo *comportamento interno*, caracterizado pela definição do corpo do módulo.

A definição do cabeçalho especifica o tipo de módulo e podem existir várias definições de corpo associadas ao mesmo cabeçalho, ou seja, módulos podem ter a mesma visibilidade externa e diferentes comportamentos internos. Na definição do cabeçalho estão incluídos atributos de classe, já mencionados, lista de parâmetros formais, pontos de interação e variáveis exportadas. A sintaxe da definição do cabeçalho de um módulo na linguagem Estelle está apresentada no ítem A1.3 do Apêndice 1.

No mínimo, um corpo de módulo é definido para cada definição de cabeçalho. A definição do corpo contém referência ao cabeçalho com o qual está associado. Se o corpo for composto apenas da palavra "external" isso indica que a definição já existe ou será providenciada posteriormente no refinamento do processo de especificação. A sintaxe da definição do corpo de um

módulo na linguagem Estelle está apresentada no ítem A1.4 do Apêndice 1. O corpo do módulo é composto de três partes: declaração, inicialização e transição.

A parte declaração consiste de: definição de constantes e tipos, declaração de variáveis, definição de procedimentos e funções. Estas definições são derivadas do Pascal com sua sintaxe e semântica. As definições não incluídas no Pascal são: tipo de canal, tipo de módulo, conjunto de estados do módulo, pontos de interação internos e variáveis de módulo. Estas últimas servem como referência a instâncias de um certo tipo de módulo.

A parte inicialização especifica os valores de algumas variáveis do módulo com as quais cada instância criada deste módulo inicia sua execução. A inicialização de uma variável de módulo resulta na criação de uma nova instância de módulo do tipo da variável.

A parte transição descreve em detalhes, através de um conjunto de declarações de transições, o comportamento interno do módulo. Cada declaração de transição é composta de duas partes: condição e ação. A sintaxe da declaração de transições de um corpo de módulo na linguagem Estelle está apresentada no ítem A1.5 do Apêndice 1.

### 3.3.3 Comunicação

A comunicação entre tarefas é feita em Estelle através de pontos de interação por dois mecanismos: troca de mensagens e compartilhamento restrito de variáveis [Budkowski 87] [Linn 86].

O módulo principal não possui formas de se comunicar com o ambiente externo.

Um estabelecimento prévio de um "link" de comunicação deve ser feito para troca de mensagens e as mensagens recebidas por uma tarefa em seu ponto de interação são colocadas numa fila FIFO não limitada associada a este ponto de interação.

Apenas uma comunicação fim-a-fim entre tarefas é possível e uma tarefa pode sempre enviar uma mensagem. Este princípio utilizado em Estelle é comumente chamado de envio não-bloqueante (ou assíncrono).

Variáveis podem ser compartilhadas entre uma tarefa e sua tarefa pai e devem ser declaradas como "exported". Esta é a única forma de compartilhamento de variáveis e a exclusão mútua para acesso às mesmas é governada pelo princípio da prioridade pai/filho. Tal princípio diz que uma ação pronta para disparo em uma tarefa inibe a seleção de ações de todas as suas tarefas descendentes.

#### 3.3.4 Paralelismo

Dois tipos de paralelismo entre módulos podem ser expressados em Estelle [Budkowski 87]: *paralelismo síncrono* e *paralelismo assíncrono*. O *paralelismo assíncrono* é permitido apenas entre sub-sistemas, ou mais precisamente, entre ações de diferentes instâncias de módulo pertencentes à diferentes sub-sistemas. O *paralelismo síncrono* é permitido apenas dentro de um sub-sistema, ou mais precisamente, entre ações de diferentes instâncias de módulo pertencentes ao mesmo sub-sistema.

Quando um conjunto de transições prontas para disparo são selecionadas dentro de um sub-sistema (no máximo uma transição por módulo) elas são executadas em paralelo. Deste modo, diz-se que as transições dentro de um sub-sistema podem ser *sincronizadas*. Se o conjunto de transições selecionadas é formado por apenas uma transição a cada passo, então tem-se um comportamento *não-determinístico* dentro de um sub-sistema. Este comportamento pode ser imposto ao sub-sistema com a utilização do atributo "systemactivity".

Sendo assim, a existência do paralelismo ou do não-determinismo na execução de tarefas em Estelle é determinado pelos seguintes aspectos:

- hierarquia de módulos;
- atributos de classe utilizados nos cabeçalhos dos módulos;
- princípio de prioridade pai/filho.

### 3.4 A Representação dos Dados em Estelle

Os tipos de dados Estelle, derivados do Pascal, incluem os tipos simples, os tipos estruturados e os tipos ponteiro. Aos tipos simples foi acrescentada a construção "...", que possibilita a declaração de um identificador sem atribuição de tipo específico ao mesmo.

O uso da linguagem Estelle para a especificação formal de protocolos tem mostrado as limitações desta técnica no que se refere ao poder de expressão das estruturas de dados. Estas limitações serão discutidas posteriormente.

### 3.4.1 Discussão sobre o uso de Pascal em Estelle

Estelle é uma linguagem que permite a distinção clara entre as partes de *dados* e *controle* em uma especificação. A parte de controle, que utiliza-se também de construções próprias de Estelle, representa o comportamento da especificação, enquanto a parte de dados, que se utiliza de construções Pascal, oferece todas as possibilidades desta linguagem para manipular dados. Esta distinção, permitida pela linguagem, é importante pois possibilita a leitura de uma especificação com vários níveis de detalhe [Courtiat 87a]; ela oferece, igualmente, a possibilidade de obter-se de forma relativamente fácil um modelo simples representando um protótipo de implementação, sobre o qual uma primeira validação pode ser feita.

Entretanto, a utilização da linguagem Pascal em Estelle para representar a parte de dados pode acarretar problemas relacionados à forte tipagem do Pascal e influenciar sobre a maneira como são declaradas as estruturas Estelle. Por utilizar Pascal também dentro da definição do corpo dos módulos, seu poder de expressão pode levar o especificador a reduzir ao máximo a parte de controle e descrever o comportamento sob a forma de um programa Pascal. O programa Pascal seria aceito por um compilador Estelle, porém seria difícil extrair um modelo que possa ser validado [Courtiat 87a].

Alguns níveis de protocolos tem a parte de dados exercendo um papel tão importante quanto a parte de controle. Este é o caso dos protocolos de nível mais alto do modelo OSI/ISO. Os dados nestes protocolos são definidos por estruturas complexas e geralmente incluem grandes quantidades de informação. Neste

caso, o uso de estruturas de dados Pascal é prejudicial pois não permite retratar estes dados de forma compacta e abstrata o suficiente e, além disso, algumas vezes levam em si escolhas de implementação. A definição de uma notação mais adequada para representação de PDUs (tal como ASN.1) diminui estes problemas.

Neste trabalho nos interessamos em juntar as facilidades da linguagem de especificação Estelle, com relação à parte de controle, com as vantagens de ASN.1 para representar PDUs.

### 3.5 Ferramentas para Simulação Estelle

Devido à larga utilização da linguagem Estelle, um grande número de ferramentas têm sido construídas para dar suporte a esta linguagem. Estas ferramentas, além da validação e dos testes, ainda automatizam grande parte do processo de implementação.

Existem vários ambientes hoje disponíveis para a linguagem Estelle e entre eles podem ser citados: VEDA [Jard 88], XESAR [Richier 87], VALIRA [Vuong 86], GROPE [New 89] e ESTIM [Saqui 90b].

### 3.6 Conclusão

Neste capítulo foi reafirmada a necessidade de utilizar-se Técnicas de Descrição Formal para especificar sistemas interconectados e, mais particularmente, protocolos de comunicação. Foram apresentadas as características mais



relevantes da linguagem Estelle e foram discutidas as implicações decorrentes da utilização de Pascal para descrição dos dados na linguagem Estelle. Finalmente algumas das ferramentas disponíveis para dar suporte à linguagem de especificação Estelle foram citadas.

## Capítulo 4

### A NOTAÇÃO PARA SINTAXE ABSTRATA ASN.1

#### 4.1 Introdução

Um dos problemas fundamentais nos sistemas abertos interconectados tem sido o tratamento das diversas formas utilizadas para representação de dados.

Este capítulo apresenta a proposta de notação ASN.1 [ISO 87a] [ISO 87b] assumida por organismos de padronização internacionais para a representação de dados. Esta proposta está baseada na definição de uma sintaxe abstrata e de regras básicas para codificação e decodificação de dados.

A apresentação está dividida em três partes. Inicialmente são tratados os problemas relativos à representação dos dados. Em seguida será apresentada a sintaxe da notação abstrata ASN.1. Na última parte é apresentada a forma de codificação de valores para esta notação.

#### 4.2 O Problema da Representação de Dados

A solução para o problema de representação, codificação, transmissão e decodificação de estruturas de dados é ter um modo

de descrever estas estruturas que seja suficientemente flexível para ser utilizada em várias aplicações [Tanenbaum 88].

Como ilustração, consideraremos um exemplo onde existem  $n$  sítios, cada um deles com sua própria representação interna para os dados; no exemplo  $n=5$ , conforme apresentado na figura 4.1 [Costa 89]. Neste caso deve haver  $n-1$  (4, no exemplo) procedimentos diferentes em cada sítio para converter a representação local dos dados para a representação de cada um de seus correspondentes. No total teremos  $n*(n-1)$  (20, no exemplo) procedimentos diferentes para  $n$  sítios. Entretanto, a adoção de uma sintaxe de transferência intermediária permitirá ter apenas dois procedimentos de conversão para cada sítio, conforme ilustrado na figura 4.2. Um procedimento servirá para converter a representação local dos dados para esta sintaxe e outro para realizar a função inversa. Para este segundo caso serão necessários, para  $n$  sítios,  $2*n$  procedimentos de conversão (10, no exemplo). O ganho em relação ao número de procedimentos de conversão é evidente.

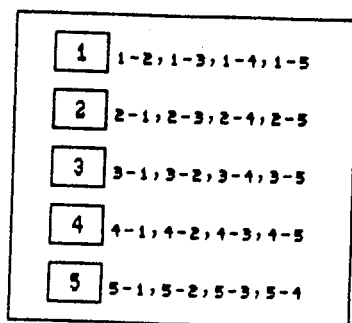


Figura 4.1 - Sem uso de representação intermediária

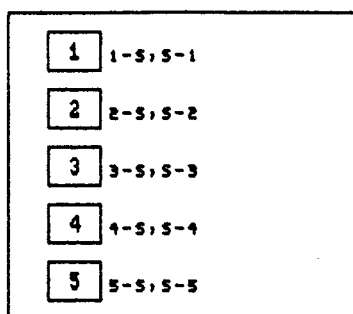


Figura 4.2 - Com uso de representação intermediária

Dentro do contexto do modelo OSI, dois sistemas de comunicação devem poder interpretar da mesma maneira a estrutura e o conteúdo dos dados trocados pelas aplicações que eles suportam. Em conseqüência, deve existir um formato comum para os dados que permita sua transferência em meio heterogêneo. Este formato comum, chamado Sintaxe de Transferência, é negociado pelas entidades de Apresentação que asseguram as funções de codificação/decodificação entre as representações internas locais de dados (Sintaxe Fonte e Sintaxe Destino) e a Sintaxe de Transferência. Estas sintaxes (fonte, destino e de transferência) são três representações concretas diferentes dos mesmos dados, cuja estrutura pode ser descrita através da notação de sintaxe abstrata ASN.1, de forma independente da implementação. A figura 4.3 mostra os formatos de transferência de dados entre duas entidades de Aplicação [Saqui 90a].

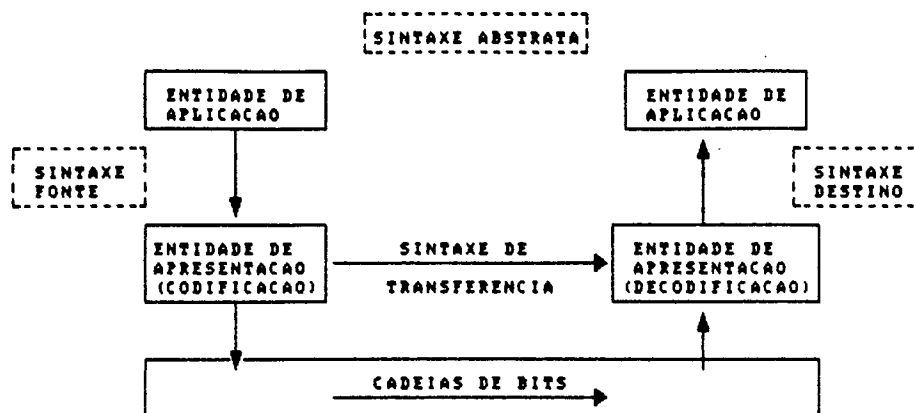


Figura 4.3 - Formato de transferência entre duas entidades de Aplicação

Como conseqüência destas necessidades, a ISO e o CCITT definiram uma notação padrão de sintaxe abstrata. O CCITT foi o precursor lançando, em 1984, a norma X.409 que faz parte da série X.400, onde define uma sintaxe abstrata e a representação padrão, termo correspondente às regras de codificação e decodificação. A ISO definiu "Abstract Syntax Notation One", ou ASN.1. Também foram definidas regras básicas de codificação e decodificação chamadas "Basic Encoding Rules for Abstract Syntax Notation One", ou BER. Em 1987, a ISO lançou os padrões ISO IS 8824 [ISO 87a] e ISO IS 8825 [ISO 87b] que definem, respectivamente, ASN.1 e as BER. No ano seguinte o CCITT lança a série X.200 que inclui, entre outras, as normas X.208 e X.209 que definem também respectivamente, a sintaxe abstrata e a sintaxe de transferência. A notação ASN.1 tem sido bastante utilizada para a definição dos blocos PDU (unidades de dados do

protocolo) dos protocolos do nível de Aplicação padronizados pela ISO.

A seguir será descrita a sintaxe dos tipos de dados ASN.1, que inicialmente estava dirigida exclusivamente para leitores humanos [Tanenbaum 88]. Nesta forma de notação, informações cruciais tais como tamanho de estruturas, não podem ser especificados, sendo apenas mencionado como informações extras a serem colocadas como comentários. Depois que ferramentas de suporte começaram a ser construídas para ASN.1 foram propostas modificações pelos grupos de trabalho da ISO no sentido de tornar a notação mais precisa e adequada para processamento. Estas modificações serão discutidas no capítulo 6.

### 4.3 As Estruturas de Dados ASN.1

#### 4.3.1 Regras para a Notação

A gramática, descrita através da notação BNF, definida para a notação ASN.1 está apresentada no item A2.1 do Apêndice 2. No item A2.2 do mesmo Apêndice são apresentadas as palavras reservadas da notação ASN.1. Uma característica notacional importante é que os nomes de tipos ("typereference") e módulos ("modulereference") definidos pelo usuário devem sempre iniciar com letra maiúscula enquanto que os nomes de identificadores ("identifier") e nomes de valores ("valuereference") devem iniciar com letra minúscula.

## 4.3.2 As Estruturas de Dados

### 4.3.2.1 Os Tipos Primitivos

Os tipos primitivos são definidos por palavras reservadas e devem ser escritos integralmente com letras maiúsculas. Em alguns casos serão apresentados exemplos para melhor compreensão de sua definição.

#### INTEGER

São os cardinais usados para contagem. Estes não possuem tamanho máximo e existe apenas um valor para o 0 (zero). É possível definir nomes para valores enumerados, como em Pascal.

Exemplos de Tipos:

```
NumeroInteiro ::= INTEGER
```

```
DiasDaSemana ::= INTEGER { domingo(1), segunda(2),
terça(3), quarta(4), quinta(5), sexta(6), sábado(7) }
```

```
-- onde domingo(1) indica que o identificador domingo
estará ligado ao valor 1
```

```
IdadePermitida ::= { mínima(18), máxima(25) }
```

```
-- são definidos os valores mínimo e máximo para o tipo
```

Exemplos de Valores:

```
1254, -5308, 0 , domingo, -10000000000001, máxima
```

**BOOLEAN**

Usado para expressar um tipo que pode assumir um dos dois valores TRUE ou FALSE.

**BIT STRING**

É uma cadeia de zero ou mais bits, sem delimitação de tamanho. Não é necessário que o número de bits da cadeia seja múltiplo de 8 ou de qualquer outro número. Este tipo permite a utilização de bits nomeados. Os valores são definidos entre chaves ({ }).

Exemplos de Tipos:

```

Chaves ::= BIT STRING
Identificação ::= BIT STRING { primeiro(1), último(5) }
-- onde os bits que representam o primeiro e o último
valor foram distinguidos dos demais
FinsDeSemanaDoMês ::= BIT STRING { um(1), trintaem (31)
}
-- onde o dia i está incluído no fim de semana se e
somente se o bit i for igual a 1
EstadoCivil ::= BIT STRING { solteiro(1),
                             casado(2),
                             viúvo(3),
                             divorciado(4),
                             outros(5)          }
-- onde apenas um dos campos deve ser setado para 1 e os
demais setados para 0

```



**Exemplos de Valores:**

'010101110011'B (cadeia de binários)  
'D12B'H (cadeia de hexadecimais)  
último  
solteiro

**OCTET STRING**

É uma cadeia de bytes. Pode ser usado para representar caracteres ou qualquer outro dado orientado a byte. Também não possui tamanho máximo delimitado.

**ANY**

Um campo deste tipo pode ser preenchido com qualquer outro tipo válido. Ele equivale à união de todos os tipos.

**NULL**

É o complemento de ANY. Representa a não existência de tipo. Quando um campo de um registro é assinalado com NULL ele não possui valor algum.

**OBJECT IDENTIFIER**

Este tipo permite definir os módulos relativos à sintaxe e às regras de codificação/decodificação negociadas pelas entidades de Apresentação. Por exemplo, o valor ASN.1 abaixo identifica um objeto definido na parte 4 do documento ISO 8571:

```
{iso standard 8571 part 4 ftam-pci(1)}
```

#### 4.3.2.2 Os Tipos Construtores

Os tipos primitivos podem ser combinados para a construção de tipos mais complexos. Os tipos construtores são definidos por palavras reservadas e devem ser sempre escritos em letras maiúsculas.

#### SEQUENCE

Usado para combinar uma coleção limitada de tipos, análogo ao tipo "record" do Pascal. Os campos podem ser de qualquer tipo incluindo os tipos construtores. Deve ser utilizado quando a ordem dos elementos é significativa.

Exemplos de Tipo:

```
Empregado ::= SEQUENCE
```

```

{ nome           VisibleString,
  departamento   VisibleString DEFAULT
                  "Informática",
  numeroIdent    INTEGER,
  idade          INTEGER OPTIONAL }

```

Exemplo de Valor:

```

{ nome           "De TAL",
  numeroIdent    765204 }

```

**SEQUENCE OF**

Usado para construir conjuntos de valores de um mesmo tipo que podem ser de um tipo primitivo ou outro tipo construtor. Este tipo é utilizado quando envolve um grande número ou um número imprevisível de valores e cuja ordem dos elementos é significativa.

**SET**

Tipo similar ao SEQUENCE com a diferença que a ordem dos componentes não é significativa. A ordem dos valores depositados na memória do receptor pode não ser a mesma que existia antes da transmissão no emissor.

**SET OF**

Similar ao SEQUENCE OF com a diferença que a ordem dos elementos não é significativa.

**CHOICE**

Usado quando uma estrutura de dados pode conter qualquer um dentre vários tipos diferentes.

Exemplo de Tipo:

Identificador ::= CHOICE

```
{ nome          VisibleString,
  numeroIdent  INTEGER          }
```

Exemplo de Valor:

```
{ numeroIdent 765204 }
```

#### 4.3.2.3 Os Tipos Cadeia de Caracteres

Juntamente com os tipos primitivos e construtores, ASN.1 possui tipos pré-definidos que são necessários em várias aplicações. Temos como desdobramento do tipo OCTET STRING oito diferentes tipos de cadeias:

NumericString

PrintableString

TeletexString

VideotexString

VisibleString

IA5String

GraphicString

GeneralString

#### 4.3.2.4 Os Tipos Úteis

Constituídos pelos tipos *GeneralizedTime*, *UTCTime*, *EXTERNAL* e *ObjectDescriptor*. Os dois primeiros são utilizados para indicação de datas e horários padronizados internacionalmente. *EXTERNAL* é utilizado para modelar um valor cujo tipo não é especificado ou será definido em outra parte da especificação, sem restrições à notação utilizada. *ObjectDescriptor* consiste de

um texto legível, escrito de forma dirigida ao leitor humano, que descreve um objeto de informação.

#### 4.3.2.5 Os Atributos Especiais

ASN.1 ainda possui alguns atributos especiais, tais como:

##### DEFAULT

Utilizado sempre seguido por um valor que deve ser assumido quando não for recebido pelo receptor.

##### OPTIONAL

Usado quando um tipo pode ou não existir, ou seja, numa transmissão o campo pode ou não ser transmitido.

#### 4.3.2.6 A Utilização de Rótulos

Outro conceito existente na sintaxe ASN.1 é o de rótulo (ou TAG). Eles são utilizados para identificar tipos ou campos.

O rótulo possui duas partes: *classe* e *número*. As classes são representadas por quatro palavras reservadas: UNIVERSAL, APPLICATION, PRIVATE e "Context Specific". O número é representado por qualquer inteiro. A palavra reservada pode ser suprimida no caso de "Context Specific". A utilização de rótulos visa relacionar as regras de codificação com a sintaxe de transferência. Sempre que um item é transmitido, seu tipo, seu tamanho e seu valor são normalmente enviados segundo a sintaxe

de transferência de ASN.1. Quando um tipo ou campo é rotulado, o TAG é também transmitido para melhor identificá-lo. Desta forma pode haver redundância de informações na comunicação quando o tipo rotulado é utilizado. Para suprimir a informação sobre o tipo de um item quando ele é rotulado, deve ser utilizada a palavra reservada IMPLICIT depois do rótulo. Desta forma são reduzidas as perdas ("overhead") na transmissão.

A utilização de rótulos (TAG) deve ser feita segundo algumas regras:

- UNIVERSAL é reservado para tipos primitivos e construtores definidos no padrão ASN.1 e estão apresentados no item A2.3 do Apêndice 2;
- APPLICATION é utilizado para tipos já definidos para protocolos de Aplicação padronizados pela ISO;
- PRIVATE é utilizado para tipos definidos pelos usuários que escrevem seus protocolos;
- "Context Specific" deve ser utilizado para distinguir tipos diferentes dentro de um mesmo módulo.

#### 4.4 A Codificação de Valores ASN.1

Uma sintaxe abstrata não é suficiente para que possa ocorrer uma comunicação. Deve existir um mecanismo que transforme a sintaxe abstrata em sintaxe concreta, ou seja, em cadeias de bits.

As regras de codificação fornecem um algoritmo que especifica como um valor, definido através de ASN.1, pode ser codificado para transmissão e depois decodificado na recepção.

O algoritmo baseia-se numa estrutura de codificação denominada TLV ("type", "length", "value" - tipo, tamanho, valor). A codificação de cada tipo consiste, então, de três campos que aparecem sempre na mesma ordem.

#### 4.4.1 Codificação do Tipo

Para a codificação do tipo são utilizados um ou mais octetos e este campo é chamado *identificador*. O identificador codifica o rótulo utilizado na especificação ASN.1. Como foi visto anteriormente, um rótulo possui duas partes: classe e número. As regras de codificação acrescentam um terceiro campo, denominado *forma*, que indica se a codificação é *primitiva* (atômica) ou *construtiva*. A forma primitiva de codificação é utilizada para os tipos simples, como INTEGER ou BOOLEAN. A forma construtiva é utilizada para tipos que contém outros tipos, como SEQUENCE e SET.

Assim sendo, os identificadores são codificados na forma CCFiiii, onde CC indica a classe: 00 para UNIVERSAL, 01 para APPLICATION, 10 para "Context Specific" e 11 para PRIVATE. F indica a forma: 0 para forma primitiva e 1 para forma construtiva. A seqüência iiii indica o código numérico atribuído ao tipo. Se o código for maior que 31, existe a possibilidade de extensão, que permite utilizar o número necessário de octetos para codificar o identificador.

#### 4.4.2 Codificação do Tamanho

As regras de codificação especificam três formas para codificação do tamanho. A primeira, denominada *forma curta* é utilizada quando o tamanho do conteúdo for menor ou igual a 127 octetos, e consiste de um octeto cujo bit mais significativo contém somente zeros.

A segunda, denominada *forma longa*, pode ser utilizada para tamanhos de conteúdo que podem ir até  $2^{1008}$  octetos. Nesta forma, o bit mais significativo está setado para 1 (um) e os bits restantes especificam o número de octetos necessários para codificar o tamanho.

A terceira forma, denominada *forma indefinida*, é particularmente útil para codificar conteúdos cujo tamanho não é conhecido por ocasião do início da transmissão. A codificação de um tipo de forma construtiva poderá utilizar a forma definida (curta ou longa) ou indefinida de codificação do tamanho do conteúdo. Esta escolha é delegada ao usuário pela norma ASN.1. Na forma indefinida, o tamanho será codificado em um octeto, com o bit mais significativo setado para 1 (um) e os bits restantes em 0 (zero). Logo após este octeto, serão gerados os octetos do conteúdo e, ao final, será colocada uma marca especial, denominada EOC ("end of contents") que consiste de dois octetos com valor zero, ou seja, tipo zero e tamanho zero.



#### 4.4.3 Codificação do Conteúdo

A codificação do conteúdo é feita sempre numa seqüência completa de octetos e depende do valor do tipo que esteja sendo gerado.

Um exemplo simples é o caso do tipo BOOLEAN, cujo valor TRUE é codificado como segue: 01 01 FF. Com o valor setado para FALSE a codificação seria: 01 01 00.

#### 4.5 Conclusão

Este capítulo apresentou a notação ASN.1. Foram apresentados os conceitos associadas à sintaxe abstrata e as regras básicas de codificação e decodificação. A larga utilização desta notação servirá como motivação para o trabalho proposto.

## Capítulo 5

### A INTEGRAÇÃO ASN.1-ESTELLE

#### 5.1 Introdução

Neste capítulo são apresentadas as motivações para a integração da notação ASN.1 [ISO 87a] [ISO 87b] com a Técnica de Descrição Formal Estelle [ISO 89]. São também discutidas algumas abordagens possíveis para esta integração e é detalhada aquela que foi adotada neste trabalho.

Em seguida são relacionadas algumas premissas para que a tradução possa ser realizada e discutidas as relações entre as estruturas da notação ASN.1 e estruturas de dados Estelle. Finalmente são feitas algumas considerações sobre a integração ASN.1/Estelle.

#### 5.2 Motivações para a Integração

Vimos anteriormente que as normas de padronização ISO para os protocolos da camada de Aplicação têm utilizado a notação ASN.1 para a descrição das Unidades de Dados do Protocolo (PDUs). A notação ASN.1 visa principalmente a definição de tipos de dados complexos de forma compacta e abstrata, incluindo ainda uma forma para definir valores para estes tipos.

A notação ASN.1 não possui forma de definir operações sobre dados, nem meios para expressar a ordem temporal das interações entre entidades de um protocolo. Como consequência disto, esta

notação não é adequada para a obtenção de uma especificação completa de protocolo. Por outro lado, uma de suas principais características consiste em ter um conjunto de regras associadas para a codificação dos dados definidos. Trabalhos têm sido realizados para desenvolver formas de implementação das Regras Básicas de Codificação (ou BER) através de ferramentas de suporte que traduzem uma especificação de blocos PDU em ASN.1 para a representação padrão apropriada para transmissão.

Vimos também que, paralelamente, vem sendo desenvolvidas e utilizadas Técnicas de Descrição Formal para a especificação dos protocolos padronizados. Estas técnicas utilizam, em alguns casos, formas de descrição de tipos e dados similares as das linguagens de programação, o que pode dificultar a tarefa de descrição das estruturas complexas de dados dos protocolos da camada mais alta do modelo OSI. Por outro lado, a natureza formal destas especificações permite a utilização de ferramentas automatizadas durante o ciclo de desenvolvimento dos protocolos. Estas ferramentas proporcionam meios para validação, implementação (automática ou semi-automática) e testes de protocolos [Chan 87] [NBS 87] [Vuong 88].

A integração da notação ASN.1 com as TDFs existentes foi objeto de vários trabalhos [Bochmann 89] [Bochmann 90b] [Costa 89] [Hasegawa 88]. A integração com a linguagem Estelle, em particular, é proposta neste trabalho em função da larga utilização desta e da existência de uma série de ferramentas já desenvolvidas para a linguagem.

A integração se justifica pela própria necessidade de coexistência destas linguagens para cobrir todos os aspectos (dados e funcionamento) de uma especificação formal e torná-la

mais completa. No caso deste trabalho a abordagem de tradução foi escolhida para a integração, e desta escolha resultam, ainda, outras vantagens. Entre elas pode-se destacar que a integração de ASN.1 com Estelle dar-se-á sem propor alterações na notação abstrata ou na linguagem de especificação. A especificação obtida, além de servir como base para a utilização de ferramentas já desenvolvidas, constitui-se de um documento que espelha todas as características do protocolo.

A seguir serão apresentadas as possíveis abordagens para a integração da notação ASN.1 com a linguagem Estelle.

### 5.3 Abordagens para a Integração

As três abordagens que podem ser consideradas para a integração da notação ASN.1 com a linguagem Estelle, conforme [Bochmann 90b], são:

#### *Substituição*

Nesta abordagem [Alvey 87] [Costa 89], a notação ASN.1 é utilizada para substituir as construções de tipos de Estelle nas especificações, ou seja, as definições de tipos Pascal não serão mais válidas.

ASN.1 não proporciona formas de acesso a elementos individuais em uma estrutura de dados. Este acesso é necessário para descrever as partes da especificação relacionadas a interpretação e seleção de valores de parâmetros de interação. É então necessário definir novas formas de acesso aos dados na linguagem de especificação.

Esta abordagem requer, portanto, a definição de *extensões* à notação ASN.1 para permitir acesso às estruturas de dados. Neste caso deve ser utilizada uma *notação ASN.1 estendida* ao invés da definição de tipos e formas para acesso a parâmetros proporcionadas pela linguagem Estelle.

### *Combinação*

Nesta abordagem [Hasegawa 88], a situação é similar à anterior, exceto que as construções da linguagem Estelle permanecem válidas e a notação ASN.1 é utilizada como um método de descrição alternativa dentro da TDF. As especificações de tipos ASN.1 são tratadas como definição de tipos na sintaxe Estelle. Os tipos definidos por ASN.1 são introduzidos em adição às declarações de tipos originais de Estelle.

Esta abordagem leva a uma duplicação de funções na linguagem de especificação. Outro problema está relacionado com o impacto que as mudanças na linguagem podem provocar na descrição do comportamento do protocolo. Pode-se afirmar que a utilização desta abordagem resultará na proposição de um *novo método* de especificação baseado em ASN.1 e Estelle.

### *Tradução*

Nesta abordagem construções ASN.1 são traduzidas para construções equivalentes em Estelle. As construções resultantes da tradução devem fazer parte da especificação completa do protocolo descrito em Estelle. Nesta abordagem não são necessárias extensões a ASN.1 nem são impostas modificações na

definição da TDF e, por conseguinte, nenhuma modificação é necessária nas ferramentas já existentes.

O fator que dificulta a utilização desta abordagem é a não existência de tipos Estelle com correspondência direta com os tipos ASN.1. Porém, no caso da escolha da linguagem Estelle para a especificação de algum protocolo, como as definições de PDUs tem sido feitas geralmente em ASN.1, o especificador encontrará o mesmo problema.

Tal abordagem pode ser utilizada tanto para a integração ASN.1/LOTOS quanto ASN.1/Estelle [Bochmann 89] [Bochmann 90a].

Observando as abordagens apresentadas, pode-se notar que tanto a *substituição* quanto a *combinação* exigem alterações na linguagem Estelle, além de extensões à ASN.1, no caso da substituição. A principal característica da abordagem de *tradução* é que tanto a linguagem Estelle quanto a notação ASN.1 permanecem inalteradas. A escolha desta abordagem teve como razões principais:

- integração de ASN.1 e Estelle sem propor alterações na notação abstrata ou na linguagem de especificação;
- utilização das ferramentas já disponíveis para Estelle.

Para que a abordagem de tradução seja convenientemente utilizada é necessário que as definições de PDU resultantes da tradução sejam simples e legíveis para o especificador de protocolo. Estas definições devem ser completadas, se necessário, e integradas às partes *transição* e *inicialização* da especificação em Estelle, para formar a especificação completa

do protocolo. Tal especificação pode ser utilizada em atividades que vão desde a simulação e verificação até a implementação e testes do protocolo especificado.

Contudo, ferramentas têm sido desenvolvidas de forma independente para ASN.1 e para Estelle. A integração destas linguagens deve levar também à integração de ferramentas associadas. No caso da abordagem de tradução o principal objetivo da integração destas ferramentas é compatibilizar as estruturas de dados utilizadas para representar as PDUs.

#### 5.4 Premissas para a Tradução ASN.1-Estelle

Nesta etapa são discutidas algumas características da notação ASN.1 relativas às informações que podem ser desconsideradas no processo de tradução. São também discutidas algumas restrições da linguagem Estelle.

##### 5.4.1 Informações Relacionadas à Codificação de Valores em ASN.1

Existem algumas informações incluídas em definições ASN.1 relacionadas com a *codificação* de valores. Estas informações não são requeridas para a descrição do comportamento lógico da entidade de protocolo e portanto não deverão ser representadas nas definições Estelle obtidas pela tradução. Estas informações devem ser incluídas sob forma de comentários para que a

correspondência com a definição original em ASN.1 seja mantida. As informações relacionadas com a codificação são:

- utilização de rótulos (TAG);
- indicação IMPLICIT;
- distinção entre SEQUENCE e SET;
- distinção entre SEQUENCE OF e SET OF.

A utilização de rótulos em ASN.1 tem por objetivo distinguir tipos de determinados campos numa instância de comunicação, ou seja, a identificação dos campos sendo utilizados pode ser feita através de rótulos diferentes, sem necessidade de nomes de identificação. Os identificadores servem apenas para fins de documentação dos campos e podem ser utilizados no contexto de um tipo SET, SEQUENCE ou CHOICE. Estes identificadores, se utilizados, devem ser diferentes, de modo a não criar ambigüidades na notação dos valores correspondentes.

O uso da palavra IMPLICIT indica que o rótulo original do tipo deve ser substituído por um novo rótulo especificado, ou seja, o rótulo original é perdido. Sua utilização está relacionada aos rótulos e indica apenas informação para codificação.

Os tipos SEQUENCE e SET possuem como diferença básica a ordem dos elementos na seqüência. Esta diferença não tem influência na tradução para especificações em Estelle.



#### 5.4.2 As Restrições da Linguagem Estelle

O maior problema encontrado na tradução para Estelle é que os tipos ASN.1 têm um poder de expressão maior que os tipos da linguagem Estelle. Os principais problemas encontrados em Estelle para a definição de tipos mais complexos exigidos nos níveis mais altos dos protocolos OSI são os seguintes [Courtiat 87a]:

- Pascal é uma linguagem com tipagem forte o que impõe uma definição explícita de todos os tipos utilizados;
- a forma de declarar certas estruturas é dirigida pelo uso do Pascal como base; ela impõe a declaração estática de estruturas que tem caráter essencialmente dinâmico, tais como os pontos de interação de um módulo.

Além disso, a linguagem Pascal é utilizada também na parte de controle da especificação. Uma consequência desta característica é que uma especificação Estelle conterà "escolhas de implantação" [Courtiat 87a]. Porém, podemos considerar que se o objetivo é produzir um protótipo de implantação, estas escolhas não constituem propriamente um problema.

#### 5.4.3 As Dificuldades de Relacionamento de Estruturas

Comparando as estruturas ASN.1 com as estruturas da linguagem Estelle temos:

- o tamanho para os tipos em ASN.1 é indefinido; as limitações de tamanho existentes em Estelle estão relacionadas apenas à linguagem utilizada nos ambientes de implementação de especificações;
- a definição de campos OPTIONAL em uma seqüência não possui similar em Estelle;
- a definição de valores DEFAULT não existe em Estelle pois os componentes de um tipo Pascal não são definidos a não ser pelos seus tipos.

Estas dificuldades podem ser sanadas com o uso de alguns artifícios que seriam provavelmente utilizados por um especificador de protocolo.

## **5.5 Proposta para Tradução ASN.1/Estelle**

Algumas estruturas ASN.1 são bastante similares às estruturas existentes em Estelle e são traduzidas de forma direta. Porém, algumas outras não possuem esta correspondência. Nestes casos levou-se em consideração as opções possíveis e foram escolhidas as estruturas Pascal que mais se assemelham com a estrutura original de ASN.1. Neste item serão definidas as relações entre os tipos de dados ASN.1 e os tipos Estelle.

### **5.5.1 Estruturas ASN.1 que Permitem Tradução Direta**

## INTEGER e BOOLEAN

Estes tipos pré-definidos tem representação similar em Estelle. Um inteiro em ASN.1 não tem limite de tamanho. Em Estelle também não é definido este limite, porém, a maioria dos compiladores Estelle impõem um tamanho máximo dos inteiros devido a restrições impostas pela linguagem utilizada para geração de código.

## ANY

Este tipo ASN.1 corresponde a uma escolha entre os vários tipos definidos na notação; a escolha será feita em função da implementação. Em Estelle este tipo pode ser representado pela notação "...", que significa que o tipo não será definido até que mais informações sejam fornecidas para a implementação.

## SEQUENCE e SET

Os tipos SEQUENCE e SET de ASN.1 contém uma quantidade de elementos de tipos diferentes. A única diferença entre as duas construções é a ordem de transmissão destes elementos que é significativa no caso de SEQUENCE e não definida para o caso de SET. Esta diferença será observada apenas na codificação, não tendo significado na definição da estrutura. Sendo assim, elas podem ser traduzidas para a mesma estrutura em Estelle, que é o tipo *record* do Pascal. Por exemplo, a definição ASN.1:

```
Tipol ::= SEQUENCE { b BOOLEAN, i INTEGER }
```

Será traduzida para Estelle como:

```

type Tipo1 = record                                     (*SEQUENCE*)
    b : BOOLEAN ;
    i : INTEGER ;
end;

```

O comentário (\*SEQUENCE\*) indica que a tradução foi obtida de um estrutura SEQUENCE e deve ser utilizado também no caso da tradução ter sido feita de uma estrutura SET (\*SET\*). Esta informação poderá ser útil para certos tipos de processamento.

A tradução de tipos estruturados de ASN.1 para Estelle também determina como os elementos de um tipo de dado podem ser acessados pelo corpo da especificação. Uma vez que Estelle inclui notação para acessar estruturas de dados, nada precisa ser providenciado pelo processo de tradução [Bochmamm 90a]. No caso do exemplo acima, com a definição adicional de tipo:

```
Tipo2 ::= SET { t Tipo1, b1 BOOLEAN }
```

e a definição de uma variável X do tipo Tipo2; a notação Pascal para acessar a estrutura t seria feita através da expressão "X.t" e o elemento inteiro da estrutura t por "X.t.i".

#### CHOICE

O tipo CHOICE é utilizado para modelar uma variável que é selecionada a partir de uma coleção de variáveis cujo número é conhecido e pequeno. A tradução para Estelle é feita através da declaração de tipo *record* com variantes do Pascal. No exemplo seguinte Tipo3 pode ser inteiro ou booleano, e é assim declarado em ASN.1:

```
Tipo3 ::= CHOICE {
                i  INTEGER
                b  BOOLEAN }

```

A tradução do exemplo acima será:

```
type Tipo3 = record
                                (*CHOICE*)
        case Tipo3_choice of
                i_Tipo3: (i : INTEGER);
                b_Tipo3: (b : BOOLEAN);
        end;
```

e considera a seguinte definição:

```
type Tipo3_choice = (i_Tipo3, b_Tipo3);
```

#### TAGGED

Os rótulos são utilizados nas definições de tipos ASN.1 para especificar os valores dos identificadores que são inseridos na forma codificada dos dados. Eles não tem influência no significado das estruturas de dados, seus valores podem ser ignorados na tradução ou podem ser representados por comentários. Entretanto eles são muito importantes para as rotinas de codificação e decodificação e para a distinção entre os diferentes tipos em estruturas de dados. Por este motivo os identificadores de campos foram introduzidos, representando os possíveis valores dos rótulos. Isso se faz necessário pois a linguagem Pascal não possui meios para testar o tipo de um valor. Estes identificadores podem ser usados no corpo de uma especificação Estelle para determinar, por exemplo, o valor de uma variável CHOICE recebida.

### 5.5.2 Estruturas ASN.1 que Necessitam de Formas Especiais de Tradução

Enquanto as traduções apresentadas anteriormente são relativamente diretas, as estruturas apresentadas a seguir não possuem correspondência imediata em Estelle e necessitam ser modeladas por estruturas de dados compostas.

#### OCTET STRING e BIT STRING

Em ASN.1 os tipos OCTET STRING e BIT STRING são utilizados para modelar dados binários onde a quantidade de elementos não é especificada. O tamanho em bits de cada elemento é um múltiplo de 8 (oito) para OCTET STRING e qualquer para BIT STRING. Existem várias especializações para OCTET STRING diferindo entre si nos tipos de valores de octetos permitidos. Por exemplo, um *IA5String* permite a utilização dos caracteres padrão ASCII usual e um *NumericString* permite a utilização apenas dos dígitos numéricos. Entretanto, em Estelle não existe um tipo próprio definido para cadeias.

O padrão [ISO 87a] define, através de uma *nota*, que comentários podem seguir uma definição de tipo OCTET STRING ou BIT STRING definindo seu tamanho máximo. Como esta decisão deve ser tomada em função da implementação, o tamanho destas estruturas se manterá indefinido na especificação. Propõem-se utilizar uma estrutura "array" para a tradução por ser uma forma simples e existente em qualquer linguagem alvo gerada a partir de Estelle.

Ilustrando tal solução, os seguintes tipos:

```
Tipo4 ::= OCTET STRING
```

```
Tipo5 ::= BIT STRING
```

serão traduzidos para Estelle como:

```
type Tipo4 = array [1..tam_Tipo4_OCTETST] of octeto;
```

```
type Tipo5 = array [1..tam_Tipo5_BITST] of boolean;
```

onde *tam\_Tipo4\_OCTETST*, *tam\_Tipo5\_BITST* e *octeto* serão constantes assim definidas:

```
const tam_Tipo4_OCTETST = any integer;
```

```
      tam_Tipo5_BITST = any integer;
```

```
      octeto = 0..255;
```

#### As Cadeias de Caracteres

Para os tipos Cadeia de Caracteres serão definidos os conjuntos de valores possíveis para cada uma das especializações. Será utilizado para isso o tipo enumerado do Pascal. A utilização do tipo *NumericString* pode ser ilustrada através do seguinte exemplo:

```
Tipo6 ::= NumericString
```

e sua tradução seria:

```
type Tipo6 = array [1..tam_Tipo6_NS] of Char;
```

considerando que comentários devem definir os valores possíveis para cada um dos tipos Cadeia de Caracteres associado.

## NULL

É utilizado nos tipos estruturados de ASN.1 para indicar a possibilidade de ausência efetiva de algum elemento. Uma vez que a utilização do tipo NULL está sempre relacionada com o tipo CHOICE, a forma de tradução utilizada conterá um tipo sem definição associado em um tipo *record* com variantes. Por exemplo temos:

```
Tipo7 ::= CHOICE {
    b  BOOLEAN
    i  INTEGER
    NULL }
```

cuja tradução será:

```
type Tipo7 = record
    (*CHOICE*)
    case Tipo7_choice of
        b_Tipo7      : ( b : BOOLEAN ) ;
        i_Tipo7      : ( i : INTEGER ) ;
        null_Tipo7   : ;
    end;
```

e considera a seguinte definição:

```
type Tipo7_choice = (b_Tipo7, i_Tipo7, null_Tipo7);
```

## OPTIONAL

Um elemento dentro de uma estrutura SEQUENCE ou SET pode ser declarado opcional. Neste caso a tradução consiste em adicionar um componente booleano imediatamente anterior (ou



posterior) ao campo definido como OPTIONAL indicando se o campo existe ou não. Por exemplo temos a definição de tipo ASN.1:

```
Tipo8 ::= SEQUENCE { i INTEGER OPTIONAL,
                    b BOOLEAN }
```

cuja tradução será:

```
type Tipo8 = record                                     (*SEQUENCE*)
    i_Tipo8_presente: boolean;
    i: INTEGER;
    b: BOOLEAN;
end;
```

#### SEQUENCE OF e SET OF

A tradução destas estruturas é a que apresenta mais dificuldade em virtude da não existência de um tipo correspondente em Estelle. Um valor do tipo SEQUENCE OF <tipo> é uma sequência ilimitada de valores, cada um do tipo especificado. Um valor do tipo SET OF <tipo> tem definição similar, exceto que a ordem dos elementos não é significativa. A mesma forma de tradução para Estelle será utilizada por não existir diferença fundamental entre elas. Neste caso a tradução será feita utilizando uma estrutura de *lista encadeada*. Por exemplo:

```
Tipo9 ::= SEQUENCE OF INTEGER
```

será traduzido para:

```

type Tipo9 = ^Lista_Tipo9;           (*SEQUENCE OF*)
type Lista_Tipo9 = record
    item: INTEGER;
    próximo: ^Lista_Tipo9
end;

```

#### DEFAULT

Esta opção deve ser considerada apenas na fase de codificação de valores e deve ser introduzida apenas como comentários na tradução.

#### SELECTION

Utiliza-se este tipo para modelar uma variável cujo tipo equivale a alguma alternativa particular de um tipo CHOICE previamente definido. Se existe, por exemplo, a seguinte definição ASN.1:

```

Tipo10 ::= CHOICE {
    i INTEGER
    b BOOLEAN }

```

A seguinte definição é também possível:

```

Tipo11 ::= SEQUENCE {
    i < Tipo10
    b < Tipo10 }

```

A tradução de Tipo11 para Estelle consideraria a seguinte definição como tradução de Tipo10:

```

type Tipo10 = record                                     (*CHOICE*)
    case Tipo10_choice: integer of
        i_Tipo10: (i : INTEGER);
        b_Tipo10: (b : BOOLEAN);
    end;

```

e como consequência a tradução de Tipo11 será:

```

type Tipo11 = record                                     (*SEQUENCE*)
    i: INTEGER;
    b: BOOLEAN;
end;

```

#### EXTERNAL

Este tipo é utilizado para modelar uma variável cujo tipo não é especificado ou será especificado em outro ponto, sem restrições à notação utilizada. A tradução deste tipo deve ser igual à utilizada para o tipo ANY, ou seja, será traduzida para o tipo "...".

#### Constantes Inteiras

ASN.1 permite a definição de mnemônicos para representar constantes inteiras como neste exemplo:

```
Tipo12 ::= INTEGER { sucesso(0),falha(3) }
```

A tradução desta definição não é trivial e por esta razão alguns artifícios são necessários para que a tradução ofereça condições de uma boa forma de acesso para as estruturas. A ordem

que os valores são alocados não é obrigatoriamente a ordem de contagem dos números naturais o que tornaria simples a tradução para Estelle. Neste caso a tradução será:

```
type Tipo12_comp = ( sucesso , falha );  
type Tipo12 = array [ Tipo12_comp ] of INTEGER;
```

```
...
```

```
var x : Tipo12;
```

```
...
```

e na parte de inicialização:

```
begin  
    x[sucesso] = 0;  
    x[falha] = 3;
```

## 5.6 Considerações Finais

As construções que apresentam problemas de compatibilidade com estruturas Estelle são SEQUENCE OF e SET OF, além de campos definidos com OPTIONAL e DEFAULT. As soluções adotadas foram limitadas pela pouca potencialidade das estruturas de dados utilizadas em Estelle.

As propostas apresentadas geraram a construção de uma ferramenta de tradução. As características desta ferramenta serão abordadas no próximo capítulo.

Nesta primeira versão do tradutor definições recursivas e macros não serão incluídas. Da mesma maneira, extensões recentes de ASN.1 não foram abordadas neste trabalho.

## 5.7 Conclusão

As vantagens da integração da notação ASN.1 com a TDF Estelle foram discutidas neste capítulo, destacando a possibilidade de obter uma especificação completa, o que apresenta vantagens para o especificador de protocolos.

Foram apresentadas as possíveis abordagens para integração e foi justificada a abordagem escolhida, ou seja, a abordagem de tradução. Em seguida foram apresentadas as relações entre estruturas ASN.1 e estruturas Estelle.

## Capítulo 6

### DESENVOLVIMENTO DA FERRAMENTA DE TRADUÇÃO

#### 6.1 Introdução

Este capítulo apresenta a metodologia utilizada para o desenvolvimento do tradutor, bem como as ferramentas escolhidas para a sua implementação. Inicialmente são discutidas algumas das dificuldades encontradas durante o processo, principalmente no que se refere a problemas de ambigüidade e sensibilidade ao contexto na notação ASN.1 [ISO 87a] [ISO 87b]. Na parte final deste capítulo mostra-se como a ferramenta de tradução se integra num ambiente Estelle [ISO 89].

#### 6.2 As Limitações da Ferramenta de Tradução

As principais dificuldades encontradas durante o processo de tradução dizem respeito à sensibilidade ao contexto e a ambigüidades na notação ASN.1. Alguns destes problemas da notação ASN.1 são levantados no documento [ISO 87c].

Os aspectos que tem influência direta sobre a definição da ferramenta de tradução estão relacionadas a problemas na sintaxe de ASN.1, tais como:

- ambigüidade decorrente da notação para valor nos tipos CHOICE, ANY e NULL;

- ambigüidade decorrente da não obrigatoriedade de uso de nomes de referência ("reference names").

### 6.2.1 Sensibilidade ao Contexto

As regras definidas em [ISO 87a] possuem problemas de sensibilidade ao contexto. A gramática definida necessita de um número ilimitado de verificações posteriores ("look-ahead") durante a análise sintática ("parsing"), não permitindo a automatização deste processo. Isto é verificado para a notação de *valor* para os tipos CHOICE e ANY aliado ao fato de não existir delimitadores para as declarações ASN.1. Um exemplo dado em [ISO 87c] ilustra este problema:

```
i1 I2 ::= i3 i4 I5 ::= I6 ..
```

onde I2, i4 e I5 não foram definidos anteriormente. Neste caso não seria possível decidir, neste ponto da análise, se a definição equivale a:

1. um valor "i1" do tipo (CHOICE) "I2", com nome de referência "i3" e valor "i4", e um tipo "I5";
2. um valor "i1" do tipo "I2", com valor "i3" e outro valor "i4" do tipo "I5";
3. se "i3" é NULL
  - (a) um valor "i1" do tipo (ANY) "I2" e um valor "i4" (NULL), e um tipo "I5";

- (b) um valor "i1" do tipo "I2" definido como NULL e um outro valor "i4" do tipo "I5";

Os últimos dois casos ocorrem porque para o tipo NULL a mesma notação é utilizada para tipo e valor. A razão da existência de tais problemas é que o analisador sintático não pode reconhecer (a menos que seu tipo tenha sido definido previamente) onde colocar o separador entre dois comandos.

Para sanar este problema, o grupo da ISO responsável pelos serviços comuns de Aplicação, Apresentação e Sessão propôs algumas alternativas de modificações na gramática ASN.1 que são definidas no documento [ISO 87c]. Dentre as alternativas, as modificações propostas incluem:

- a utilização de colchetes adicionais para as notações de valor nos tipos CHOICE e ANY, ou

*ChoiceValue ::= { NamedValue }*

*AnyValue ::= { Type Value }*

- utilização de uma nova notação para valores do tipo NULL, ou

*NullValue ::= NIL*

### 6.2.2 Não Obrigatoriedade de Nomes de Referência

Outro tipo de ambigüidade se deve a não utilização obrigatória dos nomes de referência. Em ASN.1 utilização de rótulos tem por objetivo distinguir campos de tipos iguais numa



instância de comunicação. Esta distinção pode ser feita através de nomes de referência de campos. Para ilustrar vejamos o exemplo [Perpignan 89]:

```
Registro ::= CHOICE { [0] NumericString    --CPF
                    [1] NumericString } --Identidade
numRegistro Registro ::= "316120437"
```

A única forma de saber se o número declarado corresponde ao *CPF* ou a *identidade* é fazendo a nomeação dos campos, ou seja:

```
Registro ::= CHOICE { cpf          [0] NumericString
                    identidade [1] NumericString }
numRegistro Registro ::= cpf "316120437"
```

Desta forma a declaração fica clara, embora ambas estejam sintaticamente corretas segundo o padrão ASN.1.

A não utilização de nomes de referência pode então tornar a notação ambígua. O exemplo abaixo demonstra esta possibilidade:

```
I1 ::= CHOICE {
    x [0] INTEGER,
    [1] CHOICE { x INTEGER } }
i1 I1 ::= x 5
```

Os problemas de ambigüidade ou sensibilidade ao contexto podem ocorrer também:

- entre valores dos tipos OCTET STRING e BIT STRING, usando as notações 'bstring' ou 'hstring';

- entre nomes de referência de tipo componentes de um tipo CHOICE, nomes de referência de valor e valores definidos para os tipos INTEGER ou BIT STRING.

A solução proposta para este caso envolve a utilização obrigatória de nomes de referência para as notações de todos os tipos estruturados de ASN.1. Esta solução exigiria alterações na notação ASN.1 e também modificações em algumas especificações já existentes. Grande parte dos protocolos sendo padronizados pela ISO apresentam as especificações de PDUs com a nomeação precisa de todos os campos utilizados. A ferramenta desenvolvida nomeia automaticamente os campos pertencentes a tipos estruturados. Esta nomeação é feita através da junção de parte do nome do tipo primário e do número de rótulo utilizado.

### 6.3 As Etapas de Desenvolvimento

O ferramenta de tradução proposta utiliza como entrada a definição de PDUs em ASN.1. Sua função é traduzí-las para definições de tipos e valores Estelle que possam ser incorporadas a uma especificação completa de protocolo. O passo seguinte é a integração desta ferramenta em um ambiente de concepção a linguagem Estelle.

A ferramenta de tradução foi dividida em 3 módulos:

### Módulo de Análise Léxica

Nesta fase são validadas as seqüências de caracteres de entrada os quais são agrupados em "tokens" e servem de entrada para a próxima fase.

### Módulo de Análise Sintática e Verificações Semânticas

Durante esta fase são detectados os erros sintáticos da especificação anterior. A análise sintática é feita a partir da gramática BNF definida para ASN.1. As verificações de consistência e completitude da definição ASN.1 são também realizadas nesta fase. A utilização de rotinas auxiliares e tabelas de armazenamento foram necessárias para a etapa de verificações semânticas. O resultado fornecido pela implementação informa qual o erro semântico e sua localização.

### Módulo de Tradução

Nesta fase é gerada a definição Estelle correspondente à entrada ASN.1. A execução deste módulo só acontece se não houve erros nas fases anteriores.

O primeiro módulo foi desenvolvido através da ferramenta LEX ("Lexical Analyzer Generator") [Lesk 75]. Para a implementação dos demais módulos foi utilizada a ferramenta YACC ("Yet Another Compiler-Compiler") [Johnson 75]. As ferramentas utilizadas serão descritas a seguir. O Módulo de Tradução foi desenvolvido separadamente, apesar de utilizar a mesma

ferramenta, em função da possibilidade de reutilização do Módulo de Análise Sintática e Verificações Semânticas para outros trabalhos sobre a notação ASN.1. A estrutura geral da ferramenta de tradução pode ser vista na figura 6.1.

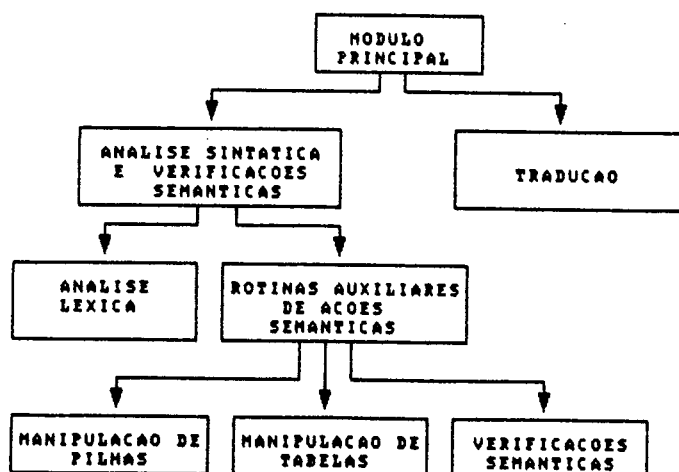


Figura 6.1 - Os módulos do Tradutor

Um exemplo de aplicação da ferramenta de tradução é apresentado no Apêndice 3.

#### 6.4 As Ferramentas de Implementação

Durante a fase de desenvolvimento do tradutor a linguagem Pascal foi utilizada. Esta escolha tem como justificativa a facilidade e larga utilização da mesma e a disponibilidade de ferramentas como LEX e YACC, que podem ser utilizados também em conjunto com a linguagem Pascal.

#### 6.4.1 A Ferramenta LEX

A ferramenta LEX auxilia a concepção de programas cujo fluxo de controle é dirigido por instâncias de expressões regulares, e é adequado para segmentar a entrada preparando-a para uma rotina de análise sintática ("parsing").

O código fonte do LEX é uma tabela de expressões regulares e fragmentos de programa correspondentes. Essa tabela é transformada num programa que divide a entrada de acordo com os padrões estabelecidos nas expressões regulares. Cada vez que um padrão é reconhecido, o fragmento de programa é executado. O reconhecimento das expressões é realizado por um autômato finito gerado pela ferramenta LEX.

Os analisadores léxicos gerados por LEX aceitam especificações ambíguas e escolhem a maior cadeia de caracteres em cada caso. Isto faz com que um número significativo de verificações posteriores ("look-ahead") seja necessário.

O programa gerado por LEX pode ser em Pascal ou outras linguagens. Para o presente trabalho optou-se pela versão que gera código Pascal.

#### 6.4.2 A Ferramenta YACC

YACC é uma ferramenta geral para geração de analisadores sintáticos. O usuário do YACC especifica a estrutura de sua entrada, em forma de regras gramaticais, junto com o código a ser invocado toda vez que uma estrutura é reconhecida.

Como saída, YACC produz uma subrotina que implementa uma máquina de estado finita, acrescida de uma pilha (autômato de pilha), para realizar a análise sintática da entrada. Esta subrotina, no entanto, não lê a entrada diretamente e sim através de uma outra rotina de nome *yylex()* fornecida pelo usuário ou pelo gerador LEX.

As ferramentas LEX e YACC podem ser usadas separadamente porém sua utilização em conjunto é particularmente interessante. O analisador gerado por LEX reconhece somente expressões regulares; YACC escreve compiladores que aceitam uma ampla classe de gramáticas livres de contexto, e requer um analisador léxico para fornecer os "tokens" de entrada. Em consequência, a combinação entre LEX e YACC é muitas vezes apropriada. O fluxo de controle nesse caso pode ser visto na figura 6.2.

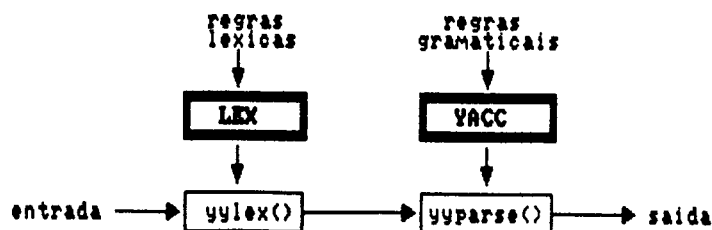


Figura 6.2 - Utilização de LEX e YACC

A saída fornecida pela função *yyparse()* depende do código que o usuário especifica junto com as regras gramaticais. No caso da ferramenta de tradução construída, a saída é a definição de tipos em Estelle.

## 6.5 A Utilização da Ferramenta num Ambiente de Concepção Estelle

Várias ferramentas têm sido desenvolvidas de forma independente tanto para ASN.1 quanto para Estelle. É importante que a integração destas linguagens mantenha a utilização de outras ferramentas já desenvolvidas. Como mostrado na figura 6.3, a definição de PDU em ASN.1, tomada do documento de padronização, deve servir como entrada para uma ferramenta constituída de rotinas de codificação e decodificação que transforme esta definição no formato apropriado para transmissão. Esta mesma definição pode ser transformada também em uma estrutura de dados interna, em linguagem C por exemplo, utilizada pela ferramenta de suporte ASN.1.

Por outro lado, a mesma definição, tomada do documento de padronização, serve como entrada para a ferramenta de tradução desenvolvida neste trabalho. O resultado obtido pelo tradutor servirá de base, juntamente com as outras partes de especificação do protocolo, para ferramentas de concepção Estelle. Para que a integração destas ferramentas seja possível é importante que as estruturas de dados (representando a PDU) obtidas pelas ferramentas em questão sejam iguais.

O presente trabalho aborda apenas a tradução de ASN.1 para Estelle por considerar que a obtenção da especificação completa do protocolo é um documento importante e que existem já algumas ferramentas que realizam as outras funções mencionadas, bastando integrá-las. Esta tarefa deve ser alvo de outros trabalhos que

visem a integração completa das ferramentas de apoio a ASN.1 e Estelle.

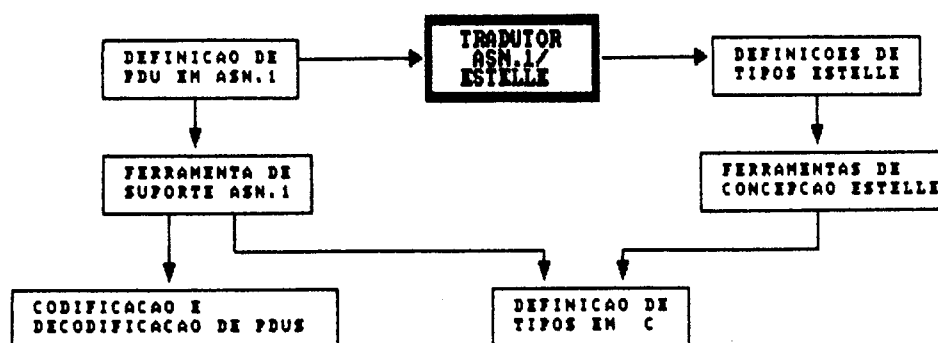


Figura 6.3 - Ambiente integrando ASN.1 e Estelle

A inclusão do tradutor desenvolvido num ambiente de verificação, simulação e geração de código executável também é possível. Neste caso a concepção de protocolos pode-se tornar automatizada parcial ou totalmente. O resultado obtido pelo tradutor de PDUs ASN.1 para Estelle pode ser agregado à parte de controle da especificação Estelle do sistema de comunicação permitindo obter uma especificação detalhada, orientada à implementação, que pode servir de entrada para as ferramentas já desenvolvidas para Estelle. Um ambiente de desenvolvimento deste tipo é similar ao apresentado na figura 6.4.



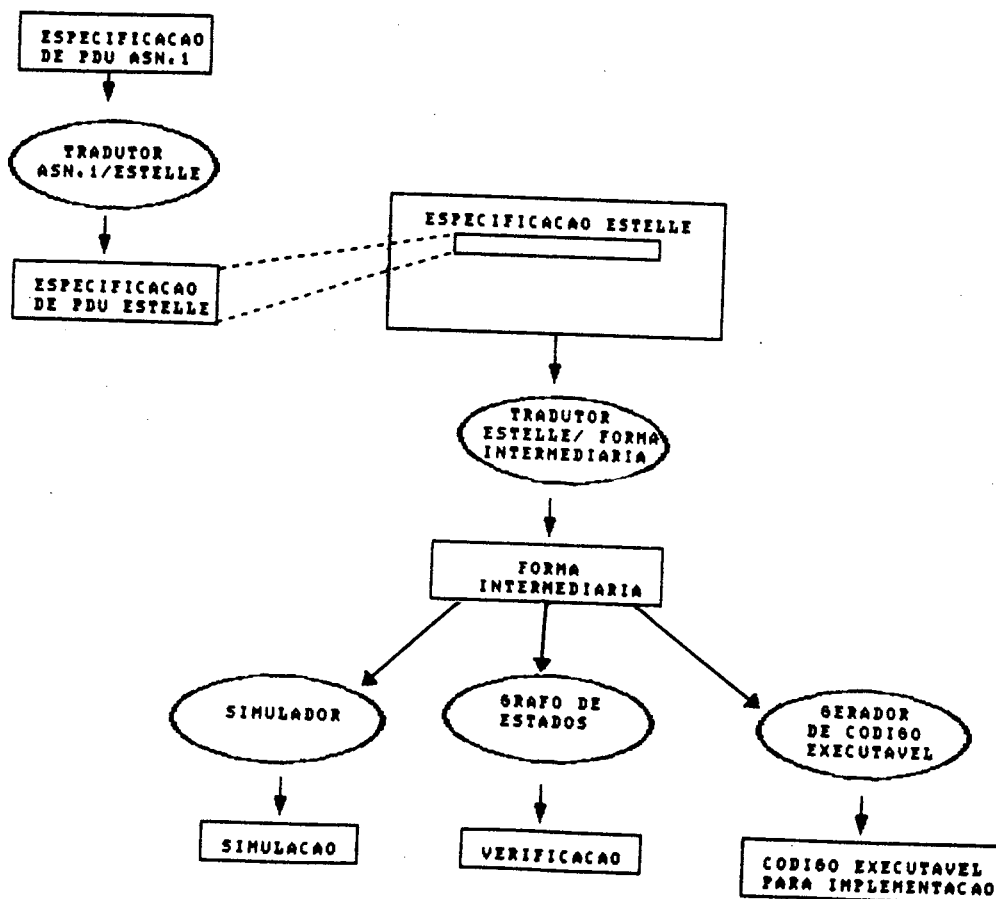


Figura 6.4 - O ambiente de concepção Estelle

## 6.6 Conclusão

Neste capítulo vimos que as principais dificuldades na construção da ferramenta de tradução estão associadas à existência de ambigüidades e sensibilidade ao contexto em ASN.1. Estes problemas apresentam soluções tais como alternativas discutidas e propostas pelos organismos de padronização e em

breve algumas delas, senão todas, devem ser adotadas para a notação ASN.1.

As principais características da ferramenta de tradução foram ainda apresentadas neste capítulo. O uso das ferramentas LEX e YACC e da linguagem Pascal facilitaram o desenvolvimento do tradutor que poderá ser integrado em ambientes de concepção de protocolos baseados em especificações Estelle e ASN.1.

## Capítulo 7

### CONCLUSÕES

A larga utilização, tanto da Notação de Sintaxe Abstrata ASN.1, quanto da Técnica de Descrição Formal Estelle, para a descrição de sistemas de comunicação justificam a necessidade de sua integração.

Esta dissertação apresentou uma proposta de integração da notação ASN.1 com Estelle e os resultados obtidos através do desenvolvimento de uma ferramenta de tradução. A ferramenta de tradução desenvolvida neste trabalho possibilita a obtenção de parte da especificação do protocolo de forma automática. A especificação completa do protocolo resultante serve como base para a automatização parcial do processo de implementação.

Da utilização da abordagem de integração proposta resultam as seguintes vantagens:

- a especificação Estelle completa do protocolo, incluindo a definição de PDUs obtida a partir do processo de tradução, permite aumentar o grau de automatização da implementação;
- as ferramentas de apoio à implementação já desenvolvidas para ASN.1 e Estelle podem continuar a ser utilizadas, e a integração entre elas e o tradutor desenvolvido possibilita a criação de um ambiente de concepção ASN.1/Estelle.

A tradução de estruturas ASN.1 para estruturas Estelle é relativamente simples apesar de algumas dificuldades encontradas que se devem aos seguintes pontos:

- alguns aspectos da notação não tem significado semântico e são relevantes apenas para a codificação das PDUs;
- Estelle não possui facilidades para tipos de dados genéricos, para distinção de campos opcionais e para manipulação de cadeias ("strings"); estas características limitam as possibilidades em termos de relacionamento das estruturas ASN.1 com as estruturas Estelle;
- as soluções propostas incluem características orientadas à implementação, o que não vem a ser um problema; a especificação formal completa com tais características poderia ser obtida por processo automático ou não de tradução.

A versão atual do tradutor não inclui alguns aspectos tais como macros e extensões recentes de ASN.1. Esta tarefa poderia ser alvo de outro trabalho assim como a integração da ferramenta desenvolvida em um ambiente de suporte para a notação ASN.1.

Enfim, este tradutor será integrado em um ambiente de concepção Estelle atualmente em uso na UFSC.

## REFERÊNCIAS BIBLIOGRÁFICAS

- [Alvey 87] "Formal Methods Applied to Protocols (FORMAT): A Framework for the Underlying Concepts of Communications Protocols". Report no.4, Alvey Directorate, UK, 1987.
- [Azema 85] P. Azema, G. Papapanagiotakis. "Protocol Analysis by Using Predicate Nets". 5th Int. Workshop on Protocol Specification, Verification and Testing, França, 1985.
- [Ayache 85] J. M. Ayache, J. P. Courtiat, M. Diaz, G. Juanole. "Utilisation des réseaux de Petri pour la modélisation et la validation des protocoles". Revue TSI vol 4, no. 1, 1985.
- [Bochmann 78] G. v. Bochmann, C. A. Sunshine. "Finite State Description of Communication Protocol". Computer Networks, 1978.
- [Bochmann 80] G. v. Bochmann, C. A. Sunshine. "Formal Methods in Communication Protocol Design". IEEE Transactions on Communications, no. 28 (4), 1980.
- [Bochmann 89] G. v. Bochmann, M. Deslouriers. "Combining ASN.1 Support with the LOTOS Language". Proc. International Symposium on Protocol Specification, Testing and Verification IX, North-Holland, 1989.

- [Bochmann 90a] G. v. Bochmann. "Protocol Specification for OSI".  
Computer Networks and ISDN Systems, no. 18, 1989/1990.
- [Bochmann 90b] G. v. Bochmann, et alli. "Implementation Support  
Tools for OSI Application Layer Protocols". Technical  
Report no. 748. Université de Montréal, 1990.
- [Budkowski 87] S. Budkowski, P. Dembinski, "An introduction to  
Estelle: a specification language for distributed systems".  
Computer Networks, vol 14, 1987.
- [CCITT 84] CCITT/SGXI/WP3-1. "SDL, Specification and Description  
Language". CCITT Recommendations Z.100 - Z.104, 1984.
- [Chan 87] R. I. M. H. Chan. "An Estelle-C Compiler for Automatic  
Protocol Implementation". Technical Report 87-36, University  
of British Columbia, Vancouver, Canada, novembre, 1987.
- [Costa 89] R. J. C. da Costa. "Sur l'integration d'ASN.1 dans  
Estelle". DEA Informatique, Université Paul Sabatier.  
Toulouse, França, 1989.
- [Courtiat 84] J. P. Courtiat, J. M. Ayache, B. Algayres. "Petri  
Nets are good for protocols". Computer Communications  
Review, 14 no. 2, 1984.
- [Courtiat 87a] J. P. Courtiat. "Contribution à la Description  
Formelle de Protocoles". Thèse de Doctorat d'Etat.  
Université Paul Sabatier, Toulouse, 1987.

- [Courtiat 87b] J. P. Courtiat, et alli. "Estelle\*: An ISO language for distributed algorithms and protocols". Technology and Science of Informatics. vol. 6, no. 5, pp. 311-324, 1987.
- [Courtiat 88] J. P. Courtiat. "Estelle\*: A powerful dialect of Estelle for OSI Protocol Descriptions". 8th IFIP International Symposium on Protocol Specification, Verification and Testing. Atlantic City, junho, 1988.
- [Danthine 77] A. S. Danthine. "Petri nets for protocols modeling and verification". COMNET Symposium. Budapest, outubro, 1977.
- [Danthine 80] A. S. Danthine. "Protocol representation with finite-state models". IEEE Transaction on Communications, Abril, 1980.
- [Diaz 82] M. Diaz. "Modelling and analysis of communication and cooperation protocols using Petri net based models". Computer Networks, dezembro, 1982.
- [Diaz 83] M. Diaz, J. P. Courtiat, B. Berthomieu, J. M. Ayache. "Status using Petri net based models for protocols". IEEE International Conference on Communications, Boston, junho, 1983.

- [Diaz 90] M. Diaz. "Environnements Logiciels pour la Conception des Protocoles dans les Systèmes Distribués". Anais de Seminário Franco Brasileiro em Sistemas Informáticos Distribuídos, Florianópolis, 1990.
- [Fernandez 88] J. C. Fernandez. "ALDEBARAN: un système de vérification par reduction de processus communicants". Thèse de Doctorat. Université Joseph Fourier. Grenoble, França, 1988.
- [Harangozo 78] J. Harangozo. "Protocol definition with formal grammars". Symposium on Computer Communication Protocols, Liege, fevereiro, 1978.
- [Hasegawa 88] T. Hasegawa, et alli. "Automatic ADA Program Generation from Protocol Specifications based on Estelle and ASN.1". Proc. of International Conference on Computer Communications (ICCC), Israel, 1988.
- [ISO 84] ISO IS 7498. "Information Processing Systems - Open Systems Interconnection: Basic Reference Model". 1984.
- [ISO 87a] ISO IS 8824. "Information Processing - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)". 1987.
- [ISO 87b] ISO IS 8825. "Information Processing - Open Systems Interconnection - Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)". 1987.



- [ISO 87c] ISO/IEC JTC1/SC21/WG6. "Some ASN.1 Discussion Topics".  
dezembro, 1987.
- [ISO 88] ISO IS 8807. "Information Processing Systems - Open  
Systems Interconnection - LOTOS: A Formal Description  
Technique based on the Temporal Ordering of Observational  
Behaviour". 1988.
- [ISO 89] ISO IS 9074. "Information Processing Systems - Open  
Systems Interconnection - Estelle: A Formal Description  
Technique based on an Extended State Transition Model".  
1989.
- [Jard 88] C. Jard, et alli. "Development of VEDA, a Prototyping  
Tool for Distributed Algorithms". IEEE Transactions on  
Software Engineering, vol.14, no.3, março, 1988.
- [Johnson 75] S. C. Johnson. "YACC - Yet Another Compiler -  
Compiler", Computer Science Technical Report no.32, Bell  
Laboratories, julho, 1975.
- [KNOWBOT 91] PROTEM-CC-S, O Projeto KNOWBOT. Documento  
organizado por L. M. R. Tarouco, agosto, 1991.
- [Lesk 75] M. E. Lesk. "LEX - A Lexical Analyser Generator".  
Computer Science Technical Report no.39, Bell Laboratories,  
outubro, 1975.

- [Linn 86] R. J. Linn, Jr. "The features and facilities of Estelle", Protocol Specification, Testing and Verification V, M. Diaz. North-Holland, pp. 271-298, 1986.
- [Milner 80] R. Milner. "CCS, a Calculus of Communicating Systems". LNCS 92, Springer-Verlag, 1980.
- [NBS 87] NBS, "User's Guide for the NBS Prototype Compiler for Estelle". Final Report, Report no. ICST/SNA - 87/3, outubro, 1987.
- [New 89] D. New, P. D. Amer. "Adding Graphics and Animation to Estelle". Proceedings of the 9th International Symposium on Protocol Specification, Testing and Verification. Twenty, The Netherlands, junho, 1989.
- [Perpignan89] D. M. Perpignan. "Tradutor Automático ASN.1/X.409 - Aplicação em Sistemas de Tratamento de Mensagens". Anais do 7o. SBRC, Recife, 1989.
- [Richier 87] J. L. Richier, et alli. "Verification in XESAR of the sliding window protocol". 7th IFIP Symposium on Protocol Specification, Testing and Verification, Zurich, maio, 1987.
- [Saqui 90a] P. Saqui-Sannes. "Prototypage d'un Environnement de Validation de Protocoles: Application à l'Approche Estelle". Thèse de Doctorat de l'Université Paul-Sabatier. Toulouse, França, maio, 1990.

- [Saqui 90b] P. Saqui-Sannes. "The ESTIM User Manual". LAAS-CNRS. Toulouse, França, 1990.
- [Tanenbaum 88] A. S. Tanenbaum. "Computer Networks". Prentice-Hall, New York, 1988.
- [Vuong 86] S. T. Vuong, D. D. Hui, D. D. Cowan. "VALIRA - a tool for validation via reachability analysis". 6th IFIP Workshop on Protocol Specification, Testing and Verification, Montréal, junho, 1986.
- [Vuong 88] S. T. Vuong, et alli. "An Estelle-C Compiler for Automatic Protocol Implementation". Proc. IFIP Symposium on Protocol Specification, Testing and Verification, Atlantic City, 1988.

## APÊNDICE 1

## A1.1 Sintaxe do Módulo Especificação

**"specification module"**

```

specification = "specification" IDENTIFIER
                [ system-class ] ";"
                [ default-options ]
                [ time-options ]
                body-definition
                "end" "."

```

```

system-class = "systemprocess" | "systemactivity"

```

```

default-options = "default" queue-discipline ";"

```

```

queue-discipline = "common" "queue" | "individual" "queue"

```

```

time-options = "timescale" IDENTIFIER ";"

```

```

body-definition = declaration-part

```

```

                 initialization-part

```

```

                 transition-declaration-part

```

## A1.2 Sintaxe da Definição de Canal

```

channel-definition = channel-heading channel-block

```

```

channel-heading = "channel" IDENTIFIER "(" role-list ")" ";"

```

```

role-list = IDENTIFIER "," IDENTIFIER

```

```

channel-block = +{ interaction-group }

```

```

interaction-group = "by" role-identifier
                    [ "," role-identifier ] ";"
                    +{ interaction-definition }

role-identifier = IDENTIFIER

interaction-definition = IDENTIFIER
                        [ "(" VALUE-PARAMETER-IDENTIFICATION
                          { ";" VALUE-PARAMETER-SPECIFICATION } ")" ] ";"

```

### A1.3 Sintaxe da Definição de Cabeçalho de Módulo

```

module-header-definition =
    "module" IDENTIFIER [ class ]
    "(" parameter-list ")" ] ";"
    [ "ip" +{interaction-pointer-declaration ";" } ]
    [ "export" +{exported-variable-declaration ";" } ]
    "end" ";"

class = "systemprocess" | "systemactivity"
        | "process" | "activity"

parameter-list = VALUE-PARAMETER-SPECIFICATION
                { ";" VALUE-PARAMETER-SPECIFICATION }

exported-variable-declaration = VARIABLE-DECLARATION

```

#### A1.4 Sintaxe da Definição de Corpo de Módulo

```

module-body-definition = "body" IDENTIFIER "for"
                          header-identifier ";"
                          ( body-definiton "end" ";"
                            | "external" ";" )

```

#### A1.5 Sintaxe da Declaração de Transições

```

transition-declaration = "trans" transition-group
transition-group = +{ clause-group transition-block ";" }
clause-group = [ provided-clause ]
                [ from-clause ]
                [ to-clause ]
                [ any-clause ]
                [ delay-clause ]
                [ when-clause ]
                [ priority-clause ]
transition-block = CONSTANT-DEFINITION-PART
                   TYPE-DEFINITION-PART
                   VARIABLE-DECLARATION-PART
                   PROCEDURE-AND-FUNCTION-DECLARATION-PART
                   [ transition-name ] STATEMENT-PART

```

## APÉNDICE 2

## A2.1 Gramática ASN.1

```

ModuleDefinition ::= modulereference
                  DEFINITIONS
                  " ::= "
                  BEGIN
                  ModuleBody
                  END

ModuleBody       ::= AssignmentList ; empty

AssignmentList  ::= Assignment ;
                  AssignmentList Assignment

Assignment      ::= Typeassignment ; Valueassignment

Externaltypereference ::= modulereference
                    .
                    typereference

Externalvaluereference ::= modulereference
                        .
                        valuereference

DefinedType     ::= Externaltypereference ;
                  typereference

DefinedValue    ::= Externalvaluereference ;
                  valuereference

Typeassignment  ::= typereference
                  " ::= "
                  Type

```

Valueassignment ::= valuereference

Type

"::="

Value

Type ::= BuiltinType | DefinedType

BuiltinType ::= BooleanType |  
 IntegerType |  
 BitStringType |  
 OctetStringType |  
 NullType |  
 SequenceType |  
 SequenceOfType |  
 SetType |  
 SetOfType |  
 ChoiceType |  
 SelectionType |  
 TaggedType |  
 AnyType |  
 ObjectIdentifierType |  
 CharacterStringType |  
 UsefulType

NamedType ::= identifier Type | Type | SelectionType

Value ::= BuiltinValue | DefinedValue



```

BuiltinValue ::= BooleanValue |
               IntegerValue |
               BitStringValue |
               OctetStringValue |
               NullValue |
               SequenceValue |
               SequenceOfValue |
               SetValue |
               SetOfValue |
               ChoiceValue |
               SelectionValue |
               TaggedValue |
               AnyValue |
               ObjectIdentifierValue |
               CharacterStringValue

NamedValue ::= identifier Value | Value

BooleanType ::= BOOLEAN

BooleanValue ::= TRUE | FALSE

IntegerType ::= INTEGER | INTEGER { NamedNumberList }

NamedNumberList ::= NamedNumber |
                   NamedNumberList, NamedNumber

NamedNumber ::= identifier (SignedNumber) |
                identifier (DefinedValue)

SignedNumber ::= number | -number

IntegerValue ::= SignedNumber | identifier

BitStringType ::= BIT STRING |
                BIT STRING { NamedBitList }

NamedBitList ::= NamedBit |
                NamedBitList, NamedBit

```

```

NamedBit      ::= identifier (number)      ;
               identifier (DefinedValue)

BitStringValue ::= bstring ;
               hstring { IdentifierList } ;
               { }

IdentifierList ::= identifier ;
               IdentifierList, identifier

OctetStringType ::= OCTET STRING

OctetStringValue ::= bstring ; hstring

NullType      ::= NULL

NullValue     ::= NULL

SequenceType  ::= SEQUENCE { ElementTypeList } ;
               SEQUENCE { }

ElementTypeList ::= ElementType ;
               ElementTypeList, ElementType

ElementType   ::= NamedType              ;
               NamedType OPTIONAL        ;
               NamedType DEFAULT Value ;
               COMPONENTS OF Type

SequenceValue ::= { ElementValueList } | { }

ElementValueList ::= NamedValue ;
               ElementValueList, NamedValue

SequenceOfType ::= SEQUENCE OF Type | SEQUENCE

SequenceOfValue ::= { ValueList } | { }

ValueList     ::= Value | ValueList, Value

SetType       ::= SET { ElementType } | SET { }

SetValue     ::= { ElementValueList } | { }

SetOfType    ::= SET OF Type | SET

SetOfValue   ::= { ValueList } | { }

```

```

ChoiceType      ::= CHOICE { AlternativeTypeList }
AlternativeTypeList ::= NamedType ;
                    AlternativeTypeList, NamedType

ChoiceValue     ::= NamedValue

SelectionType   ::= identifier Type

SelectionValue  ::= NamedValue

TaggedType     ::= Tag Type ; Tag IMPLICIT Type

Tag            ::= [ Class ClassNumber ]

ClassNumber    ::= number ; DefinedValue

Class         ::= UNIVERSAL ;
                APPLICATION ;
                PRIVATE ;
                empty

TaggedValue    ::= Value

AnyType        ::= ANY

AnyValue       ::= Type Value

ObjectIdentifierType ::= OBJECT IDENTIFIER

ObjectIdentifierValue ::= { ObjIdComponentList } ;
                        { DefinedValue
                          ObjIdComponentList }

ObjIdComponentList ::= ObjIdComponent ;
                    ObjIdComponent ObjIdComponentList

ObjIdComponent  ::= NameForm ;
                NumberForm ;
                NameAndNumberForm

NameForm       ::= identifier

NumberForm     ::= number ; DefinedValue

NameAndNumberForm ::= identifier (NumberForm)

CharacterStringType ::= typereference

```

CharacterStringValue ::= cstring

UsefulType ::= typereference

#### Os Tipos Cadeia de Caracteres (Character String Type)

NumericString            VisibleString

PrintableString        ISO646String

TeletexString         IASString

T61String             GraphicString

VideotexString        GeneralString

#### Os Tipos Úteis (Useful Type)

GeneralizedTime        EXTERNAL

UTCTime                ObjectDescriptor

## A2.2 Palavras Reservadas de ASN.1

BOOLEAN

INTEGER

BIT

STRING

OCTET

NULL

SEQUENCE

OF

SET

IMPLICIT

CHOICE

ANY

EXTERNAL

OBJECT

IDENTIFIER

OPTIONAL

DEFAULT

COMPONENTS

TRUE

FALSE

BEGIN

END

### A2.3 Rótulos Universais de ASN.1

UNIVERSAL 1	Boolean type
UNIVERSAL 2	Integer type
UNIVERSAL 3	Bitstring type
UNIVERSAL 4	Octecstring type
UNIVERSAL 5	Null type
UNIVERSAL 6	Objectidentifier type
UNIVERSAL 7	Objectdescriptor type
UNIVERSAL 8	External type
UNIVERSAL 9 - 15	Reservadas para adendos a este Padrão Internacional
UNIVERSAL 16	Sequence e Sequence-of types
UNIVERSAL 17	Set e Set-of types
UNIVERSAL 18-22,25-27	Character String types
UNIVERSAL 23-24	Time types
UNIVERSAL 28-...	Reservadas para adendos a este Padrão Internacional

## APÊNDICE 3

## EXEMPLO DE UTILIZAÇÃO

Este apêndice apresenta um exemplo de aplicação da ferramenta de tradução. O protocolo utilizado foi o Protocolo do Bit Alternante (ABP). A escolha foi feita pelo fato deste protocolo ser largamente conhecido. No caso, a PDU definida é curta, gerando assim uma definição Estelle também não muito longa.

A especificação completa do protocolo ABP é dada no Anexo B2 de [ISO 89]. Neste protocolo são definidas duas entidades: uma emissora e outra receptora. A entidade emissora envia PDUs de dados à entidade receptora e recebe desta PDUs de "acknowledgement". Cada PDU de dados contém, além dos dados propriamente ditos, um número de seqüência que pode ser 0 ou 1 alternadamente. A PDU de reconhecimento contém apenas um número de seqüência. Uma PDU com valor 0 (zero) indica que houve uma recepção correta de uma PDU de dado se o número de seqüência da mesma foi 0 (zero), ou que é necessária a retransmissão da PDU de dado se o número de seqüência foi 1 (um). A definição de PDU escrita em ASN.1 para este protocolo é dada a seguir [Bochmann 90b]:

```

ABPDU DEFINITIONS ::=
BEGIN
    Npdutype ::= CHOICE
        { datapdu [APPLICATION 1] Datapdutype,
          ackpdu  [APPLICATION 2] Ackpdutype
        }
    Datapdutype ::= SEQUENCE
        { ndata    Udatatype,
          seq      Seqtype
        }
    Ackpdutype ::= SEQUENCE
        { seq      Seqtype }
    Udatatype  ::= SET OF IA5String
    Seqtype    ::= INTEGER { seq(0), seq1(1) }
END

```

A seguir esta apresentada a parte declaração da especificação Estelle gerada pelo tradutor. Ela está dividida em duas partes. A primeira parte, denominada "asn1estelle/geral.e", apresenta as definições de constantes, variáveis e tipos pré-definidos. A segunda parte, denominada "asn1estelle/tipos.e", contém as definições de tipos Estelle correspondentes a ABPDU dada anteriormente.



```
(* asn1estelle/geral.e *)
```

```
type (* tipos pre-definidos *)
  octeto = 0..255 ;
  cstring = ... ;
  hstring = ... ;
  bstring = ... ;
  nil = ... ;
  qualquer = ... ;
  ASN1_DATA_TYPE = record
    (*usado para representar dados codificados pelas BER*)
    octetseq: bstring;
    tamanho : integer;
  end;
  ASN1_TIME = record
    year : integer; (*year :19**ou**, (0..~)*)
    month : integer; (*month : 1 .. 12*)
    day : integer; (*day : 1 .. 31*)
    hour : integer; (*hour : 0 .. 23*)
    minute : integer; (*minute : 0 .. 59*)
    second : real ; (*second : 0 .. 59*)
    diff : real ; (*diferenca entre horario
                  local e UTCTime*)
    zone : integer ; (*flag para a forma de
                    TIME: 0,1,2,3,4 *)
    (*UTC_Z_TIME para UTC Time com Z
      UTC_D_TIME para UTC Time com
      diferencial
      GNL_Z_TIME para Generalized Time
      com Z
      GNL_D_TIME para Generalized Time
      com diferencial
      GNL_L_TIME para Generalized Local
      Time *)
  end ;
  UTCTime = ASN1_TIME ;
  GeneralizedTime = ASN1_TIME ;
  ObjectIdentifier = ... ;
  EXTERNAL = external ;
  (* Os valores pertencentes aos tipos Character String devem
    ser definidos segundo o padrao ISO 2375
    NumericString_value = ...
    PrintableString_value = ...
    TeletexString_value = ...
    VideotexString_value = ...
    VisibleString_value = ...
    IA5String_value = ...
    GraphicString_value = ...
    GeneralString_value = ... *)

type Npdutype_choice = (datapdu_Npdutype,ackpdu_Datapdutype);

const tam_Udatatype_IA5S = ANY INTEGER ;
const seq = 0 ;
const seq1 = 1 ;
```

```
(* asn1estelle/tipos.e *)
```

```
(* MODULO : ABPDU *)
```

```
type Seqtype = INTEGER ;
  (* valores possiveis para Seqtype :
      seq = 0
      seq1 = 1
  *)
```

```
type Udatatype_IA5String = array [1..tam_Udatatype_IA5String]
of IA5String_value ;
```

```
type Udatatype = ^Lista_Udatatype ;
  Lista_Udatatype = record (* SET OF *)
    item: Udatatype_IA5String ;
    next: ^Lista_Udatatype ;
  end;
```

```
type Ackpdutype = record (* SEQUENCE *)
  seq : Seqtype ;
end ;
```

```
type Datapdutype = record (* SEQUENCE *)
  ndata : Udatatype ;
  seq : Seqtype ;
end ;
```

```
type Npdutype = record (* CHOICE *)
  case Npdutype_choice : INTEGER of
    Npdutype_datapdu_tag : ( datapdu : (*
[ APPLICATION 1 ] *) Datapdutype ) ;
    Datapdutype_ackpdu_tag : ( ackpdu :
(* [ APPLICATION 2 ] *) Ackpdutype ) ;
  end ;
```