- = TRADUTOR AUTOMÁTICO ASN.1/X.409 = -
APLICAÇÃO EM SISTEMAS DE TRATAMENTO DE MENSAGENS X.409

AUTORES: Daniel Maurice Perpignan (EMBRATEL-GTC-RIO DE JANEIRO-RJ)
José Lucas Mourão Rangel Netto (PUC-RJ - DEPARTAMENTO DE INFORMÁTICA)

RESUMO

Um dos problemas fundamentais em comunicação de dados tem sido o tratamento das diversas formas utilizadas para representar a informação em diferentes sistemas de computação. No Modelo da ISO para interconexão de sistemas abertos (Modelo OSI), cabe à camada de apresentação a tarefa de resolver as diferentes formas de representação dos dados (a sintaxe) para que a camada de aplicação possa efetuar a correta interpretação semântica dos mesmos. Para tal, é preciso que haja um mecanismo para descrever os dados de forma independente de sua representação (sintaxe abstrata) e, eleger uma forma de codificação consistente e mutuamente aceitável para transmissão efetiva de dados entre aplicações remotas (sintaxe de transferência). A "Notação para Sintaxe Abstrata 1" (ASN.1 - Abstract Sintax Notation One) e as "Regras Básica de Codificação" (BER - Basic Encoding Rules) definidas nas normas internacionais ISO DIS 8824, ISO DIS 8825 e CCITT X.409 fornecem tais mecanismos que vem sendo utilizados pela grande maioria dos protocolos camada de aplicação tais como: MHS - Message Handling Systems, FTAM - File Transfer, Access and Management, ODA - Overall Document Architecture, etc.

Para passar de uma especificação de dados efetuada segundo a Notação ASN.1 para a codificação da sintaxe de transferência, foi desenvolvido um tradutor automático utilizando-se técnicas de construção de compiladores e as especificações do Modelo OSI, mais especificamente na sua aplicação para Sistemas de Tratamento de Mensagens baseados no padrão definido nas recomendações da série X.400 do CCITT.

Este tradutor foi desenvolvimento para ser uma ferramenta bastante versátil, que poderá facilitar o desenvolvimento e teste de validação de protocolos para a camada de aplicação do Modelo OSI.

1) INTRODUÇÃO

Este artigo irá descrever uma ferramenta que foi desenvolvida numa tese de mestrado (Perp.88) visando facilitar a tarefa de especificação de protocolos de comunicação. Esta ferramenta permite verificar que Unidades de Dados de Protocolos de Aplicação (APDU´s - Aplication Protocol Data Units) estejam corretamente especificadas através da Notação ASN.1, bem como gerar, de forma semi-automática, o código correspondente segundo as regras de codificação associadas.

Para a construção da feramenta foi implementado um compilador para a linguagem ASN.1 e suas regras de codificação associadas, padronizadas pela ISO e CCITT. A solução adotada consistiu na pesquisa e definição de uma estrutura de dados genérica, que atendesse aos requisitos necessários para representar qualquer APDU de protocolo de aplicação especificada em ASN.1. Em seguida, para geração do código para cada instância de comunicacão de um determinado protocolo, foi desenvolvido um Editor, orientado por sintaxe,que permite ao usuário a entrada de valores, de modo assistido, para os diversos elementos de dados constitutivos da APDU de um determinado protocolo.

2) CONCEITOS GERAIS E CAMPO DE APLICAÇÃO:

Para se entender o objetivo do presente trabalho, é necessário conhecer o contexto em que ele se aplica e, ter uma noção dos conceitos do Modelo OSI, Sistemas de Tratamento de Mensagens X.400, e de protocolos de Apresentação e de Aplicação, que delimitam e definem o ambiente de aplicabilidade de seus resultados.

3) INTRODUÇÃO À "NOTAÇÃO PARA SINTAXE ABSTRATA UM" (ASN.1 - ABSTRACT SINTAX NOTATION ONE"

Nos sistemas de informação distribuídos, as Aplicações necessitam transmitir informações entre estações remotas, sem se preocupar em determinar os detalhes de representação da informação no sistema remoto ou de sua codificação durante a fase de transmissão (sintaxe de transferência). Cabe à Camada de Apresentação adaptar a sintaxe dos dados do originador à sintaxe do destinatário, preservando-se os significados da informação (isto é, a semântica) em favor das Entidades e/ou Processos de Aplicação. Para tal, é necessário que, a nível da Camada de Aplicação, os dados a serem trocados estejam especificados através do que se denomina de "sintaxe abstrata", isto é, uma sintaxe em que os detalhes de codificação das informações não sejam considerados.

Um conjunto de definições de Tipos de Dados, utilizando uma Notação bem definida constitui o que se convencionou chamar de Sintaxe Abstrata para a informação que pode ser expressa por valores atribuídos a ocorrências (variáveis) desses Tipos de Dados. Esta "Notação bem definida", ou "Notação Padrão" pode ser, teoricamente, a de uma linguagem de programação que permita a definição de estruturas complexas de dados (por ex.: Pascal, ADA, Modula, etc.). A "Notação Padrão" define os aspectos e as regras a serem usadas na especificação formal dos tipos de dados e valores, de forma independente da técnica de codificação a ser empregada para representar estes mesmos tipos e valores.

Para ser útil aos propósitos da Camada de Apresentação, uma Notacão Padrão para definição de uma sintaxe abstrata necessita de um conjunto de regras de codificação que determine, de forma algorítmica, para qualquer conjunto e estruturas de dados que eteja definido através da Notação Padrão, a Representação Padrão dos valores dos tipos de dados sob a forma de cadeias de bits, isto é, a Sintaxe de Transferência da Camada de Apresentação. A notação das linguagens de programação expressa através de uma variante de BNF (Backus-Naur Form), para definição de estruturas de dados, geralmente não especificam estas regras, que dependem também da máquina e do compilador utilizados.

Assim sendo, a ISO e o CCITT definiram uma Notação Padrão para sintaxes abstratas chamada: "Notação para sintaxe Abstrata Um" - ASN.1 e um conjunto de regras básicas de codificação associado. As normas ISO DIS 8824 e ISO DIS 8825 descrevem a notacão e regras de codificação, respectivamente, que no CCITT estão contidas numa única recomendação, a X.409 publicada em 1984 no assim chamado "Livro Vermelho". A possibilidade de existirem outras notações e regras de codificação é explicitamente reconhecida pela ISO e pelo CCITT.

A Notacão ASN.1 oferece bastante flexibilidade para especificacão e criação de novos tipos de dados, havendo frequentemente várias maneiras de expressar a mesma sintaxe abstrata. Devido a isto, uma ferramenta que permita testar a eficiência de uma especificação em ASN.1 é importante. Resumindo os principais pontos da notação ASN.1 e das regras de codificação, temos:

- Uma notação geral e padronizada para definição de sintaxes abstratas, expressa por uma gramática em BNF.

- Uma série de tipos de dados básicos, pré-definidos na linguagem (p.ex.: INTEGER, BOOLEAN, SET, etc.)

- Uma série de mecanismos de estruturação de dados.

- Uma série de tipos para suportar os diferentes tipos de códigos de cadeias de caracteres (ASCII, TELETEX, VIDEOTEXTO, etc.)

- Codificação segundo o formato TLV (Type, Length, Value).

- Possibilidade de criação de novos tipos.

- Tamanhos indefinidos, porém bem delimitados. (A limitação só é feita por considerações pragmáticas).

- O fato de gerar sempre sequências com um número inteiro de octetos ("Octet Aligned").

Cabe aqui ressaltar que as técnicas de especificação formal de protocolos através de linguagens padronizadas pelo CCITT e pela ISO, tais como: SDL (Specification and Description Language) ESTELLE (Extended State Transition Language) e LOTOS (Language for Temporal Ordering Specification), estão baseadas em modelos de sistemas de transições e incorporam facilidades para definir Operações com estruturas de dados, inspiradas nas construções encontradas em linguagens de programação de alto nível. Assim sendo, as técnicas de Especificação Formal são úteis tanto para definir os serviços de uma camada (N), isto é, as interações permitidas entre uma Entidade (N+1) e o provedor dos serviços da camada (N), como para um protocolo da Camada (N), isto é, as interações entre Entidades (N) remotas correspondentes.

Desta forma, a ASN.1 não pode ser considerada , e nem foi desenvolvida para ser uma técnica de Especificação Formal comparável com a linguagens citadas no parágrafo anterior. Faltam à ASN.1 facilidades para expressar a ordem temporal das interações entre as Entidades (N). Mesmo assim, a maioria dos Padrões para Protocolos de Aplicação estão usando a Notação ASN.1 para descrever a sintaxe de seus blocos PDU, o que torna imprescindível o seu conhecimento para aqueles que desejem compreender e implementar um desses protocolos.

A linguagem ASN.1 possui várias semelhanças com os aspectos de definições de tipos de dados das linguagens convencionais, como Pascal, C e ADA. Ela é também similar a linguagens mais antigas para descrição de tipos de dados em protocolos de comunicação como a linguagem definida pela Xerox - "Xerox's Courier Protocol".

A linguagem possui uma sintaxe formal para definição e nomeação de tipos, bem como mecanismos que permitem aos usuários

a construção de tipos complexos a partir de tipos mais simples ou predefinidos. Ela está especificada utilizando-se uma variante de BNF. (CCITT409).

Basicamente, são definidos "tipos abstratos de dados" e "objetos abstratos de dados" que são identificáveis por nomes independentemente de sua forma de implementação. O tipo torna-se conhecido através de sua declaração, sob forma de uma produção escrita através da BNF (Notação Padrão), e o objeto é definido por uma declaração de atribuição, também sob forma de produção BNF, de um valor para um tipo conhecido. Através de tipos predefinidos, como inteiros, booleanos etc. podem ser definidos outros tipos, utilizando-se inclusive definições recursivas.

3.1. Descrição Resumida da Notação ASN.1

Uma Notação Padrão é um conjunto de convenções utilizadas na especificação de Tipos e Valores associados a um tipo. Estas convenções serão utilizadas nas especificações dos protocolos de Aplicação.

Lembramos que a informação é representada por um tipo e um valor associado e que cada tipo possui uma Notação Padrão e uma Representação (codificação) Padrão. Vamos examinar como isto é efetuado na linguagem ASN.1.

3.1.1. Notação Padrão

A Notação Padrão para cada tipo e/ou Valor associado é definida através de uma BNF. Esta descrição compreende uma série de Produções, que nada mais são do que regras de substituição de símbolos. Existem três classes de símbolos que podem aparecer numa Produção: Terminais, Não-Terminais e Operadores. Os Terminais são símbolos que têm significado predefinido na linguagem. Os Não-Terminais são símbolos definidos através de outros símbolos, terminais ou não. Os Operadores são utilizados nas definições dos Não-Terminais.

```
Ex.: BooleanType  :: = BOOLEAN
     BooleanValue :: = TRUE | FALSE
```

Os símbolos à esquerda do operador de atribuição (::=) são Não-Terminais, enquanto os símbolos à direita são Terminais. A barra vertical é o Operador de Alternativa.

O operador "::=" aparece, às vezes, entre aspas. Isto se deve ao fato de que, neste caso trata-se de um símbolo terminal e as aspas são colocadas para não confundí-lo com o

Operador de atribuição da BNF.

Para introduzir uma Notação Padrão utilizaremos um exemplo. Os detalhes completos da Notação poderão ser encontrados nas referências (CCITT409),(ISO1) e (ISO2).

Exemplo: Vamos considerar um bloco PDU fictício cuja definição, segundo a Notação Padrão (X.409 ou ASN.1), está na Figura 1.

Identificação de Tipos

Um tipo pode ser visto como uma coleção específica de objetos distintos, aos quais é atribuído um significado. Por exemplo, o tipo INTEGER (inteiro) inclui todos os valores numéricos que podem ser expressos como números inteiros. A linguaguem ASN.1 define três gêneros de Tipos: Tipos predefinidos (built-in), tais como inteiros, booleanos etc.: Tipos "Cadeias de Caracteres" ("Character Set String Types"), que contêm caracteres de um determinado conjunto de caracteres, tal como ASCII, TELETEX, BAUDOT...; e um grupo denominado "Tipos Úteis", que inclui dois tipos para representação de Data e Hora.

Para distinguir um tipo de outro, a linguagem atribui um rótulo (tag) a cada tipo. O rótulo possui duas partes: uma classe e um Número. A classe pode assumir quatro valores: "Universal", "Aplicação", "Privado" e "Especfífico a um Contexto" (denominado simplesmente "Contexto"). Os rótulos Universais são atribuídos aos tipos definidos na Norma ASN.1, incluindo os tipos predefinidos, os tipos de "Cadeias de Caracteres" e os Tipos Úteis. Os rótulos de aplicação podem ser utilizados pelos projetistas das aplicações a serem padronizadas, tal como os Sistemas de Tratamento de Mensagens, FTAM etc. Os rótulos Privados podem ser atribuídos a tipos definidos por quaisquer outros usuárioes da ASN.1, tal como formatos de blocos PDU´s de protocolos (aplicações) internos a uma determinada companhia. Finalmente, os rótulos Específicos a Um Contexto são utilizados para distinguir tipos em contextos específicos da ASN.1, tal como dentro dos tipos estruturados SEQUENCE, SET e CHOICE, de modo a identificar univocamente cada componente.

Estas quatro classes dividem o espaço de numeração dos rótulos em quatro partes. A segunda parte do rótulo é um número que identifica um tipo específico dentro de sua classe.

Por exemplo, vamos listar na tabela seguinte os tipos universais definidos na ASN.1.

OBS.: Existem diferenças entre a ASN.1 e a X.409 que segundo notícias mais recentes, serão eliminadas quando da publicação, em 1988, das novas recomendações do CCITT.

Os tipos assinalados com (*) na tabela não foram tratados nesta versão do Tradutor, por não haver aplicação prática até o momento, segundo pesquisa dos autores.

Neste ponto vale ressaltar algums convenções adotadas na ASN.1, que faz distinção entre letras maiúsculas e letras minúsculas nos nomes atribuídos aos tipos e objetos. Assim sendo, temos que:

a) Todos os nomes de tipo devem iniciar com letras maiúsculas;

RÓTULO UNIVESAL	TIPO
1	BOOLEAN
2	INTEGER
3	BIT STRING
4	OCTECT STRING
5	NULL
6	OBJECT IDENTIFIER (*)
7	OBJECT DESCRIPTOR (*)
8	EXTERNAL (*)
9 A 15	VALORES RESERVADOS PARA EXTENSÕES FUTURAS
16	SEQUENCE E SEGUENCE OF
17	SET E SET OF
18 A 22	TIPOS "CADEIAS DE CARACTERES"
23	UTC TIME
24	GENERALIZED TIME

b) os nomes dos tipos predefinidos são escritos somente com letras maiusculas.

É importante notar que o rótulo (tag) meramente identifica um tipo, não definindo a estrutura ou o conteúdo de qualquer valor deste tipo. O projetista de blocos PDUs para protocolos de aplicação deve tomar cuidado na atribuição de rótulos aos tipos que criar, de modo a não atribuir o mesmo rótulo a dois tipos diferentes, conduzindo a problemas na decodificação, oriundos de possíveis ambiguidades.

Pelo exemplo, pode-se ver que os tipos criados "Segurança" e "Endereço" foram rotulados de modo a evitar ambiguidade a nível de aplicação, enquanto que os campos do tipo estruturado "Endereço" foram rotulados a nível de contexto. (Outra convenção da ASN.1 estabelece que os rótulos devem aparecer entre colchetes e, se a classe for omitida, assume-se que é específica ao contexto). Obviamente, rótulos da classe universal não podem aparecer nas especificações das PDUs. Entretanto pode-se mudar o rótulo de um tipo universal, criando um novo tipo, através do tipo "Tagged", que veremos um pouco mais adiante, como já podemos depreender do exemplo da Figura 1, que continuaremos a utilizar.

Forma Geral de Definições ASN.1

Um grupo de definições ou produções em ASN.1 deve estar contido dentro de um Módulo. Estes Módulos possuem a seguinte forma:

ModuleDefinition ::=

modulereference DEFINITIONS "::=" BEGIN ModuleBody END

No nosso exemplo, o mome do módulo (modulereference) é "Exemplo" e as definições contidas entre as palavras reservadas da linguagem, BEGIN e END, irão compor o corpo do módulo (ModuleBody).

O nome do módulo é um exemplo de identificador em ASN.1. Identificadores são usados para nomear módulos, tipos, componentes de tipos e valores. Os caracteres permitidos em identificadores ASN.1 são as letras maíusculas e minúsculas, os dígitos e o hífen. O identificador deve começar sempre por uma letra. Se a primeira letra for maiúscula trata-se de um nome de Módulo ou de tipo; caso contrário trata-se de um identificador de valor ou de um campo em um tipo estruturado. Convém ressaltar aqui que os identificadores de valor ou de campo (ou componente) de um tipo mais complexo são atribuídos somente para facilitar a especificação e legibilidade por pessoas, em nada contribuindo para as regras de condificações da sintaxe de transferência.

No nosso exemplo, tomemos o caso de identificador de transação ao qual foi atribuído o identificador (nome do campo, neste caso) "id-transação", sem o qual ficaria difícil ao projetista saber onde se acha tal campo na hora de lhe atribuir um valor.

Continuando o nosso exemplo, se tomarmos o tipo

"Segurança", fica muito mais fácil e legível atribuir nomes aos graus de segurança desejados para a transmissão da PDU, do que lembrar do valor numérico associado (identificador de valor).

Como qualquer outra linguagem, a ASN.1 possui uma série de palavras reservadas (incluindo-se os nomes dos tipos predefinidos, de cadeias de caracteres e úteis) que não podem ser redefinidas pelo projetista. São permitidos também comentários dentro de um módulo. Comentários em ASN.1 começam por dois hifens "--") e terminam, seja com outro par de hifens, seja com o final da linha.

Definição de Tipos Simples

Os módulos devem conter basicamente definições de tipos. A definição de valores está também prevista na notação.

```
TypeAssignment  ::= typereference "::=" Type
ValueAssignment ::= valuereference Type "::=" Value
```

Onde "typereference" é um identificador de tipo e "valuereference" um identificador de valor. Como foi dito, as definições de tipo podem ser recursivas. O Não-Terminal "Type" pode ser substituído por tipos predefinidos ou por outros tipos definidos no mesmo módulo ou num outro módulo qualquer . Neste último caso, o identificador deve vir precedido do nome do módulo em que foi inicialmente definido, seguido de um ponto. Em BNF temos::

```
Externaltypereference ::= modulereference . typereference
```

O mesmo se aplica ao Não-Terminal "Value".

Desta forma, uma definição de tipo ou valor pode ser muito simples ou mais complicada, dependendo da complexidade da estrutura de dados que está sendo especificada. Não existe limite para o nível de aninhamento das definições, podendo ir de uma única linha a várias páginas de definições aninhadas.

No nosso exemplo, a definição do tipo "Classe" pode ser vista como simplesmente a atribuição de um novo nome ao tipo predefinido INTEGER. Isto é, este novo tipo possui o mesmo rótulo que o tipo INTEGER.

A ANS.1 permite também que inteiros e cadeias de bits possam ser utilizados como Tipos Enumeração do Pascal ou "C". No nosso exemplo, podemos observar o tipo "Segurança" e o tipo referido pelo identificador de campo "log" do tipo estruturado

"PDUPedido", como exemplos. Neste caso, os identificadores que aparecem entre chaves são identificadores de valor e podem ser utilizados dentro do módulo, sempre associados ao tipo que os definiu.

Os tipos "PrinttableString" e "NumericString" fazem parte dos Tipos de Cadeia de Caracteres, cuja classe é universal e que estão previstos na ASN.1. A lista completa destes tipos é dada na referência (CCITT 409). Aqui citaremos apenas as principais, que são:

RÓTULO UNIVERSAL	TIPO
18	NumericString
19	PrintableString
20	T61String ou TeletexString
22	IA5String (ASCII)

Estes tipos definem conjuntos ou subconjuntos de sinais gráficos ou de controle de determinados conjuntos padronizados de caracteres. Por exemplo, uma "NumericString" só pode ser composta dos dígitos e do ponto, representados segundo o código ASCII.

Uma "PrintableString" é uma cadeia de caracteres imprimíveis na maioria das impressoras existentes no mercado. Este conjunto de caracteres é formado pelas letras maiúsculas e minúsculas, pelos dígitos e por alguns símbolos de pontuação, codificados segundo o padrão ASCII ou IA5 ("International Alphabet Number 5" - segundo o CCITT). (ver referência (CCITT 409).

OBS.: O Teletex é um serviço telemático de transmissão de textos definido pelo CCITT.

Uma observação importante que precisa ser feita é a de que na ASN.1 não existe tamanho máximo para cadeias de caracteres. A única forma de especificar-se um tamanho máximo é através dos comentários. O mesmo vale para cadeias de bits e de octetos.

Para terminar de falar sobre os tipos simples, vamos continuar acompanhando nosso exemplo. Na definição do tipo "PDUPedido" encontramos o campo:

respostaExigida BOOLEAN

que define uma variável booleana que pode assumir os valores TRUE (Verdadeiro) ou FALSE (Falso). Poderíamos, neste ponto, ter definido um novo tipo para este campo:

RespostaExigida ::= BOOLEAN

(Observe que o nome, agora, começa por uma letra maiúscula). A linguagem ASN.1 permite várias maneiras para se expressar a mesma sintaxe abstrata. A escolha é uma questão de estilo de programação do projetista e de suas necessidades específicas, como por exemplo, rapidez na decodificação de um determinado significado. Atribuindo-se tipos específicos a cada variável, fica mais rápido reconhecê-las e, por conseguinte, decodificá-la. Voltaremos a este ponto mais adiante.

Prosseguindo com o exemplo, encontramos o campo:

dados-Usuário OCTET STRING

que define uma variável de tamanho indefinido, composta por uma sequência de octetos, do tipo predefinido OCTET STRING. Somente valores em notação hexadecimal ou binária podem lhe ser atribuídos.

O último tipo simples de que ainda não falamos é o tipo NULL (Nulo). Este tipo aparece no nosso exemplo como uma alternativa para o campo denominado info-Roteamento" da nossa PDU fictícia. O tipo NULL é utilizado para, nos tipos estruturados, indicar a ausência efetiva de um elemento. No nosso exemplo, numa PDU de pedido de operação que transita por uma rede pela primeira vez, esta informação não existe de fato, sendo acrescida informação a cada nó da rede que o bloco PDU atravessa.

Quanto aos Tipos Úteis, que servem para codificar a data e hora local e/ou universal, com diversos graus de precisão, eles seguem as normas ISO 2014, ISO 3307 e ISO 4031 e se encontram especificados nas referências (CCIT409) e (ISO2). Existem dois tipos definidos na ASN.1: "GeneralizedTime" (mais utilizado para representar a data e hora local) e "UTCTime" (mais utilizado para representar a data e hora universal). No nosso exemplo, utilizamos o tipo "GeneralizedTime" para indicar, na PDU Resposta, a data e hora em que a operação foi registrada no "log" remoto.

Definição dos Tipos Estruturados

A linguagem ASN.1 fornece mecanismos para que se possa construir novos tipos que representem estruturas complexas que

permitam especificar os blocos PDUs dos protocolos de aplicação. Estes novos tipos são construídos através dos tipos simples e dos tipos estruturados da linguagem ASN.1. Existem cinco tipos estruturados na linguagem:

1) SEQUENCE - designa uma lista finita e ordenada de tipos. Cada elemento da lista pode ser visto como um campo de uma estrutura "record" do PASCAL.

2) SEQUENCE OF -designa uma lista ilimitada forma pela repetição interativa e ordenada de um mesmo tipo.

3) SET -designa uma lista finita de tipos na qual a ordem de arrumação dos tipos não é significativa, isto é, valores podem ser atribuídos em qualquer ordem.

4) SET OF -designa uma lista ilimitada formada pela repetição interativa de um mesmo tipo, onde nenhum significado pode ser atribuído à ordem em que valores deste tipo serão atribuídos e depois transmitidos.

5) CHOICE -designa uma lista limitada de tipos, dentre os quais apenas um pode ser escolhido numa determinada instância de comunicação, isto é para formar um campo do bloco PDU a ser transmitido.

As definições dos tipos estruturados podem usar qualquer tipo definido na linguagem, no próprio módulo ou num módulo externo. O nível de imbricação e aninhamento de tipos estruturados não é especificado, deixando total liberdade ao projetista da PDU.

Retornando ao nosso exemplo, podemos notar que nossa PDU fictícia é uma sequência ordenada de campos. Como já dissemos, pode-se fazer uma analogia com o tipo "record" do PASCAL. No nosso caso, o início e o fim da estrutura são assinalados pelos caracteres abre-chave ({) e fecha-chave (}), respectivamente. Cada campo possui um tipo associado e, eventualmente, um nome identificador. Como já dissemos, os identificadores servem apenas para fins de documentação dos campos; porém, se colocados, deve-se garantir que os identificadores de campo atribuídos no contexto de um tipo SEQUENCE, SET ou CHOICE sejam distintos, de modo a não se criar ambiguidades na notação padrão dos valores correspondentes.

Vamos dar um exemplo para melhor exemplificar o caso. Suponhamos a seguinte estrutura:

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

Como saber se o número corresponde ao CPF ou à Identidade, olhando-se para a notação? Se reescrevemos adefinição como

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

fica mais claro, embora ambas as formas estejam sintaticamente corretas segundo a ASN.1

O mesmo problema pode ocorrer a nível de codificacão/decodificação dos valores para sua transmissão. Assim sendo, a ASN.1 especifica que todos os tipos, no contexto de uma estrutura do tipo SEQUENCE, SET ou CHOICE, devem possuir rótulos diferentes.

Para tal, a ASN.1 fornece meios para atribuição de novos rótulos, tanto para os tipos predefinidos como para os novos tipos criados dentro do módulo. Isto é feito através da definição de um novo tipo batizado na ASN.1 como "TaggedType", e cuja definição, na BNF, é a seguinte:

```
TaggedType ::= Tag IMPLIC Type |
               tag Type
Tag   ::= [ class number ]
Class  ::= UNIVERSAL |  APPLICATION  |  PRIVATE
           empty
```

onde o Não-Terminal Type pode ser substituído por qualquer tipo definido na linguagem ou no módulo.

Desta forma, no exemplo anterior foram criados dois novos tipos, uma para o CPF e outro para a identidade, cuja classe é específica ao contexto e cujo número é o (zero) e 1 (um), respectivamente. Desta forma, as regras de codificação/decodificação saberão distinguir qual foi a opção escolhida numa determinada instância de comunicação. No nosso exemplo, podemos notar que o mesmo foi feito nos campos do tipo "Endereço" e no campo "parte-variável" do tipo PDU. Observe-se que, no caso do tipo "Endereço", não foi necessário rotular novamente o último campo "entidade", pois seu tipo já permite distinguí-lo dos demais.

Os tipos "Segurança" e "Endereço" foram considerados de uso mais geral, recebendo rótulos da classe aplicação. Isto supõe que eles serão utilizados em mais de um lugar dentro do mesmo módulo ou não, mas dentro da mesma aplicação (STM, FTAM, etc.)

Observando a BNF do tipo "Tagged", notamos que existem duas alternativas de utilização. O uso da palavra reservada IMPLICIT indica que o rótulo original do tipo deve ser substituído pelo novo rótulo especificado. Isto é, o rótulo original é perdido. No segundo caso, o rótulo original é preservado, criando-se um novo tipo estruturado, que irá conter o tipo original, e cujo rótulo será o especificado entre colchetes. É opção do projetista escolher qual forma atende melhor os seus objetivos.

No nosso exemplo, encontramos os dois casos. Cabe observar que o uso do IMPLICIT reduz o overhead de transmissão, devendo-se para tal usar a segunda alternativa somente quando necessário à aplicação, na interpretação dos dados recebidos.

O tipo "Tagged" possui outra característica peculiar, que é a de não possuir um rótulo específico. Isto acontece também com o tipo CHOICE. Isto quer dizer que estes tipos não geram código na sintaxe de transferência, ou, melhor dizendo, assumem os rótulos da alternativa escolhida, no caso do tipo CHOICE, ou do especificado explicitamente na Notação Padrão, no caso do tipo "Tagged".

A necessidade de distinção dos rótulos, a nível de sintaxe de transferência, e de identificadores de campos e de valores, a nível de Notação Padrão para valores, se faz sentir melhor no tipo SET onde os elementos podem aparecer em qualquer ordem. No nosso exemplo, podemos ver que, no tipo "PDUResposta", onde todos os tipos são diferentes, não houve necessidade de utilizar-se rótulos (tags) novos. Assim, uma instância possível de PDUResposta poderia ser, em Notação ASN.1:

```
pdu's PDUResposta ::= [resultado sucesso,
                       dados-usuário 'A0030B020'H,
                       log "8801252030Z" ,
                       id-transação "12345" ]
```

Outro ponto que pode ser observado nos tipos SEQUENCE e SET é a possibilidade de poder omitir-se alguns campos numa instância de comunicação. Isto é possível na ASN.1, especificando-e um valor a ser assumido por omissão (valor default) para um determinado campo, ou declarando-se o campo como

opcional. para isto, utilizam-se as palavras reservada DEFAULT e OPTIONAL, como pode ser visto no nosso exemplo. Quando uma destas últimas for especificada para um campo, e a aplicação não dispuser em contrário, isto é , não atribuir um valor para estes campos, estes não serão gerados na sintaxe de transferência. Assim sendo, se não for especificada uma "Classe" para a PDUPedido do nosso exemplo, este campo não constará do bloco PDU a ser transmitido e o receptor assumirá o valor zero para este campo.

Outros Tipos Definidos na ASN.1

O nosso exemplo apresentou uma amostra representativa dos tipos disponíveis na ASN.1. A norma ASN.1 define alguns outros tipos que listamos rapidamente a seguir:

a) Tipo ANY - serve para indicar que qualquer tipo definido na norma ASN.1 pode aparecer. A norma estabelece que, onde este tipo for utilizado na especificação de um bloco PDU, faz-se necessário uma padronização adicional, no futuro ou em outra norma pertinente, antes que ocorra uma instância de comunicação.

b) Tipo SELECTION - serve para fazer referência, dentro de um Módulo, a um campo (tipo) de uma lista de alternativas de um tipo CHOICE.

c) Tipo EXTERNAL - este tipo é utilizado no Protocolo de Apresentação OSI, conforme especificado na Norma ASN.1.

d) OBJECT IDENTIFIER - define um identificador para um objeto particular, que é codificado como uma série de inteiros. É utilizado para nomear uma série de coisas, incluindo sintaxe abstratas e de transferência, que são alocados por autoridades normativas.

Além destes tipos, a linguagem ASN.1 define muitos pelos quais um usuário mais sofisticado pode definir uma nova notação para definir tipos e valores em ASN.1. Isto é geito através de uma notação para "Macros", cujo único uso até hoje está especificado nas recomendações X.410 e X.411 (protocolo P3) para especificação nas "Operações Remotas".

Os tipos e a notação Macro citados neste item não serão tratados na presente versão do Tradutor.

3.1.2. Representação Padrão

A Representação Padrão é o termo equivalente utilizado pelo CCITT para o que a ISO chamada de BER ("Basic Encoding Rules"), ou seja, Regras básicas para codificação da sintaxe de transferência.

Uma sintaxe abstrata não é suficiente para que possa ocorrer uma comunicação. Deve existir um mecanismo para transforma a sintaxe abstrata em sintaxe concreta, ou seja, em cadeias de bits. As regras de codificação fornecem um algoritmo que especifica como, um valor de um tipo qualquer, definido através da ASN.1, pode ser codificado para transmissão e, depois, decodificado no receptor.

Este algoritmo baseia-se numa estrutura de codificação TLV ("Type", "Length", "Value" - Tipo, Tamanho, Valor) para cada objeto a ser gerado. Isto quer dizer que a codificação de cada tipo consiste de três campos que aparecem sempre nesta ordem: um ou mais octetos indicando o tipo - e este campo é chamado Identificador; um ou mais octetos indicando número de octetos necessários para codificar o valor - este campo é chamado Tamanho; um ou mais octetos necessários para codificar o valor - este campo é cnamado Conteúdo, pois pode conter outros tipos (estruturas TLV) como é o caso dos tipos estruturados. A Figura 2 demonstra o mecanismo de codificação explicado.

Codificação do Identificador

O identificador codifica o rótulo especificado na ASN.1. Como vimos, o rótulo possui duas partes: classe o número. As regras de codificação acrescentam um terceiro campo, chamado 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, enquanto a Forma construtiva é utilizada para tipos que contêm outros tipos, como SEQUENCE e SET.

Assim sendo, os identificadores são codificados como segue:

CCFiiiii

onde CC indica a classe que pode ser:
00 - UNIVERSAL
01 - APPLICATION
10 - CONTEXT
11 - PRIVATE

F Indica a forma que pode ser:
0 - primitiva
1 - construtiva

iiiii Indica o código numérico atribuído ao tipo, como já vimos. Se o código for maior que 31, existe especificada uma possibilidade de extensão que permite utilizar o número necessário de octetos para codificar o identificador.

No nosso exemplo, o tipo 'Classe' teria seu identificador codificado como:

00 0 00010 ou 02 hexadecimal

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 está a zero.

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 a um e os bits restantes especificam o número de octetos necessários para codificar o tamanho.

A terceira forma, denominada Forma Indefinida, é particulamente útil para codificar conteúdos cujo tamanho não é connecido por ocasião do início da transmissão. A norma deixa a escolha do usuário se a codificação de um tipo de forma construtiva irá utilizar a forma definida (curta ou longa) ou indefinida de codificação do tamanho do conteúdo. Na forma indefinida, o Tamanho será codificado em um octeto, com o bit mais significativo em 1 e os bits restantes em zero ('80'H). 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 00 00 (Tipo 00 Tamanho 00).

Na implementação do tradutor foi dada preferência à forma indefinida para codificação do tamanho nos tipos de forma construtiva.

Codificação do Conteúdo

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

Talvez o exemplo mais simples seja o caso do tipo BOOLEANO, cujo valor TRUE (verdadeiro) é codificado como segue:

T	L	V
01	01	FF

e o valor FALSE (falso)

T	L	V
01	01	00

3.2. Comentários sobre a Notação ASN.1

Como foi visto, a linguagem ASN.1 oferece bastante flexibilidade para especificação e criação de novos tipos abstratos de dados, havendo frequentemente várias maneiras de expressar a mesma sintaxe abstrata. Devido a isto, podemos concluir que, segundo a especificação da sintaxe abstrata, as regras de codificação podem gerar sequências de bits diferentes. Visto que as regras de codificação geram uma estrutura TLV, que causa um aumento de pelo menos 2 octetos ao número de octetos a ser transferido (OVERHEAD) para cada tipo, certos cuidados devem ser assumidos na definição dos blocos PDU, de modo a se atingir um compromisso razoável entre o OVEHEAD a ser gerado e a flexibilidade e extensibilidade de identificação e reconhecimento dos mesmos pelos protocolos de aplicação que os utilizam.

Este compromisso deve levar em conta: a atribuição de rótulos específicos aos tipos componentes do bloco PDU, via tipo "Tagged", visando sua rápida distinção por ocasião da decodificação; o OVEHEAD que o tipo "Tagged" acarreta em termos de transmissão; e a complexidade da estrutura gerada. Lembramos, ainda, que rótulos dentro de uma mesma classe não podem possuir o mesmo número e que pode ocorrer, numa estrutura complexa, que um mesmo tipo venha a ter dois ou mais rótulos, um mais externo e outro mais interno dentro da estrutura, isto é, aninhamento de tipos "Tagged".

No nosso exemplo, temos dois blocos PDUs definidos e, de modo geral, é interessante que a aplicação seja capaz de distinguir rapidamemnte um bloco PDU do outro, durante a decodificação, de modo a interpretar corretamente os respectivos conteúdos. Esta identificação pode ser feita baseada no rótulo do primeiro componente de cada um (análise de contexto) ou mais rapidamente, através de um rótulo específico atribuído a cada um, como é mais usual e como adotamos no exemplo. O rótulo específico permite

que as rotinas de decodificação determinem mais facilmente qual o bloco PDU que está sendo processado, independentemente de seu conteúdo.

A combinação da lingagem ASN.1 com as regras de codificação fornece um método eficaz para definição e representação de tipos de dados e valores. A flexibilidade e a clareza que podem ser atingidos pelo seu uso não é totalmente livre, requerendo-se algum esforço no aprendizado e compreenção desta nova ferramenta.

IMPLEMENTAÇÃO DO TRADUTOR ASN.1

A implementação teve dois objetivos principais:

a) Gerar um tradutor que, partindo da especificação, em ASN.1, de blocos PDU (Protocol Data Units) relativos a um determinado protocolo de aplicação (sintaxe abstrata), permita gerar de modo automático, uma representação padrão correspondente (sintaxe de transferência) param qualquer instância de comunicação daquele protocolo.

b) Gerar uma ferramenta para que o projetista de protocolos de aplicação possa verificar se a especificação dos PDUs do protocolo em ASN.1 está sintaticamente correta e não contém ambiguidade de tipos.

Cabe inicialmente observar, que na norma de especificacão dos serviços da Camada de Apresentação (CCITT X.216), a transformação da sintaxe abstrata em sintaxe de transferência ("sintaxe concreta") não afeta os protocolos de apresentação, sendo, portanto, matéria de âmbito local de cada entidade de apresentação, não havendo nenhuma restrição de implememntação. Assim sendo, o tradutor poderá ser chamado para cada instância de comunicação (isto é, cada bloco PDU que a aplicação deseja transmitir), devendo, portanto, ser rápido e eficiente. Entretanto, no caso desta implementação, tivemos de simular a Camada de Aplicação de modo a fornecer os valores sobre os quais o tradutor irá trabalhar, de modo a gerar as instâncias dos blocos PDU a serem transmitidos.

Visto que a notação ASN.1 é utilizada para especificar estruturas de dados referentes a blocos PDUs trocados entre Entidades de Aplicação, decidiu-se representar internamente a sintaxe abstrata do bloco PDU através de uma estrutura de dados do tipo "Árvore", que será denominada "Árvore do Protocolo Px".

Esta forma de representação corresponde a uma árvore sintática da teoria de compiladores, que, uma vez gerada,

permitirá codificar as várias instâncias de comunicação de blocos PDU de um determinado protocolo de aplicação. Desta forma, ganha-se eficiência, pois não será necessário chamar e executar o tradutor inteiro para cada instância de comunicação.

Uma vez definida a estrutura de dados para representar a sintaxe abstrata, vamos descrever o esquema geral para geração dos blocos PDU para um deteminado protocolo. A figura 3 mostra esquematicamente o processo.

Partindo-se da especificação em ASN.1 de blocos PDU de um determinado protocolo de aplicação, será necessário fazer uma adaptação na especificação original do PDU (como será visto mais adiante), de modo a torná-la compatível com o compilador ASN.1, e gerar um arquivo contendo esta especificação. Este procedimento é bastante simples consistindo basicamente em colocar-se um ponto-e-vírgula ao final de cada produção, menos na última. Assim, o arquivo poderá ser gerado via Editor de Texto.

Feito isto, o próximo passado é chamar o compilador ASN.1. O compilador verificará se a especificação do bloco PDU do protocolo está correta, assinalando ao usuário toda vez que for encontrado algum erro sintático ou semântico (quando possível). O processo será repetido tantas vezes quanto for necessário até que se obtenha uma especificação de PDU livre de erros. Quando o compilador não detectar mais erros, ele avisará ao usuário (Sintaxe ok!) e gravará dois arquivos para o protocolo Px, um contendo a Árvore (Px.ARV) e outro contendo o buffer de Identificadores e Valores constantes associado (Px.BUF). Note-se que, em princípio, este procedimento só precisa ser executado uma única vez para um determinado protocolo.

De posse da Árvore, o usuário pode gerar a diversas instâncias de blocos PDU para transmissão, antes ou durante uma comunicação. Isto é feito através de um Editor/Gerador desenvolvido para este fim. O Editor é orientado pela Sintaxe; isto quer dizer, dada uma árvore de sintaxe Abstrata correspondente a um determinado protocolo, ele percorre a árvore e solicita que o usuário forneça, via telas pré-formatadas para cada tipo de nó, os valores correspondentes àquela instância de comunicação. Os valores capturados da tela do terminal são validados e fornecidos ao módulo gerador que irá gravar um arquivo (Px.COD) segundo as regras de codificação da sintaxe de Transferência (BER). O Editor gera outro arquivo ontendo a notação padrão dos valores correspondentes àquela instância de comunicação para facilidar a interpretação do código gerado.

Finalmente, caso o usuário deseje, são fornecidas listagens dos arquivos gerados como mostrado no anexo 1.

A solução adotada na implantação do Tradutor permitiu avaliar a importância e a viabilidade em utilizar-se as técnicas de geração automática de compiladores, consistindo então de duas partes bem distintas: A primeira consiste de um compilador que analisa uma especificação de PDU em ASN.1 e, caso esta esteja correta sob o ponto de vista sintático, gera a árvore do Protocolo correspondente. A segunda parte consiste de um Editor/Gerador que, partindo da Árvore Sintática do Protocolo e de valores fornecidos por um operador, via terminal de vídeo/teclado, gera o código, segundo a sintaxe de transferência, correspondente a uma determinada instância de PDU do protocolo em questão.

Desta forma, para cada instância de comunicação de um determinado protocolo de aplicação, basta executar-se o Editor/Gerador. A comunicação entre o usuário e o Editor/Gerador correspondente à simulação das primitivas de passagem de dados entre as Camadas de Aplicação e de Apresentação. A Figura 4 ilustra a arquitetura do projeto do tradutor.

Observando-se a figura, nota-se que quisemos reforçar aspecto que sugere que, quando um protocolo possui mais de um bloco PDU, a especificação de todos eles deve ser feita dentro de um único módulo ASN.1, começando por uma declaração de um tipo CHOICE (escolha de alternativa).

Finalizando, queremos dizer que a implementação foi efetuada em Linguagem Pascal e desenvolvida para microcomputadores da linha IBM PC-XT na configuração mínima de 640K e com dois drives.

REFERÊCIAS BIBLIOGRÁFICAS

(PERP88) - TRADUTOR AUTOMÁTICO ASN.1/X.409 - APLICAÇÃO EM SISTEMAS DE TRATAMENTO DE MENSAGENS X.400

(CCITT409)- CCITT - Red Book - Recommendation X.400 - MESSAGE HANDLING SYSTEMS - SYSTEM MODEL - SERVICE ELEMENTS - 1984

(ISO1) - ISO/DIS 8824.2 - Information Processing - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1) - 07/10/1986

(ISO2) - ISO/DIS 88254.2 - Information Processing - Open Systems Interconnection - Specification of Basic Ecoding Rules for Abstract Syntax Notation One (ASN.1) - 07/10/86.

BIBLIOGRAFIA

PRESENTATION LAYER SERVICES FOR OPEN SYSTEMS INTERCONNECTION - J. Larmouth - Salford Univesity, England - Open Systems - Data Transfer - August 85.

A Tutorial on Abstract Syntax Notation One (ASN.1) - David Chappell - Open System Data Transfer - D ecember 1986.

Um Gerador de Analisadores R*S . J. C. Rangel, S. M. Schneider - V. Simpósio Software Básico - 25 a 27 de novembro de 1985.

Prospect - A Tool for Protocol and Conformance Testing - B. Plattne, E. A. Blau, T. Walter - Congresso IFIP - WG 6.5/25 a 29 de abril de 1987.

FIGURA 1 : Exemplo de Notação Padrão.

```
Exemplo DEFINITIONS ::=
     BEGIN
          PDU ::= SEQUENCE
               { parte-fixa SEQUENCE
                    { protocolo INTEGER ,
                      chamado Endereco ,
                      chamador Endereco } ,
                 parte-variavel  CHOICE  { [0] PDUPedido,
                                           [1] PDUResposta } ,
                 info-Roteamento CHOICE {SEQUENCE OF Endereco,
                                         NULL }}
                 -- no  caso  da operacao ter sido desviada  para
                 -- outro sistema.

          PDUPedido ::= SEQUENCE
               { Operacao,
                 respostaExigida BOOLEAN ,
                 Classe DEFAULT 0 ,
                 id-Transacao PrintableString ,
                 Seguranca DEFAULT media ,
                 dados-Usuario OCTET STRING OPTIONAL ,
               -- parametros da operacao, por exemplo.
                 log BIT STRING
                    { aplicacao (0) ,
                      gerente-Rede (1) ,
                      gerente_Sistema (2) }}
               -- quais   elementos  devem  manter  registro   da
               -- transacao efetuada.

          Classe ::= INTEGER

          PDUResposta ::= SET
               { id-Transacao PrintableString ,
                 resultado INTEGER { sucesso(0), falha(1) },
                 log GeneralizedTime ,
                 dados-usuario OCTET STRING OPTIONAL }
             -- resultados da operacao , se nouver.

          Seguranca ::= [APPLICATION 0] IMPLICIT INTEGER
               { alta (0), media (1), nenhuma (3) }

          Endereco ::= [APPLICATION 1] IMPLICIT SEQUENCE
               { pais [0] IMPLICIT NumericString ,
               -- Maximo de 3 digitos segundo norma internacional
                 area [1] IMPLICIT NumericString,
               -- Maximo de 2 digitos
                 numero-rede [2] IMPLICIT NumericString ,
               -- Maximo de 10 digitos
                 entidade PrintableString }
              -- Maximo  de  5 caracteres, comencando com  uma letra

          Operacao ::= [APPLICATION 3] CHOICE { codigo INTEGER,
                                                nome PrintableString }

     END
```

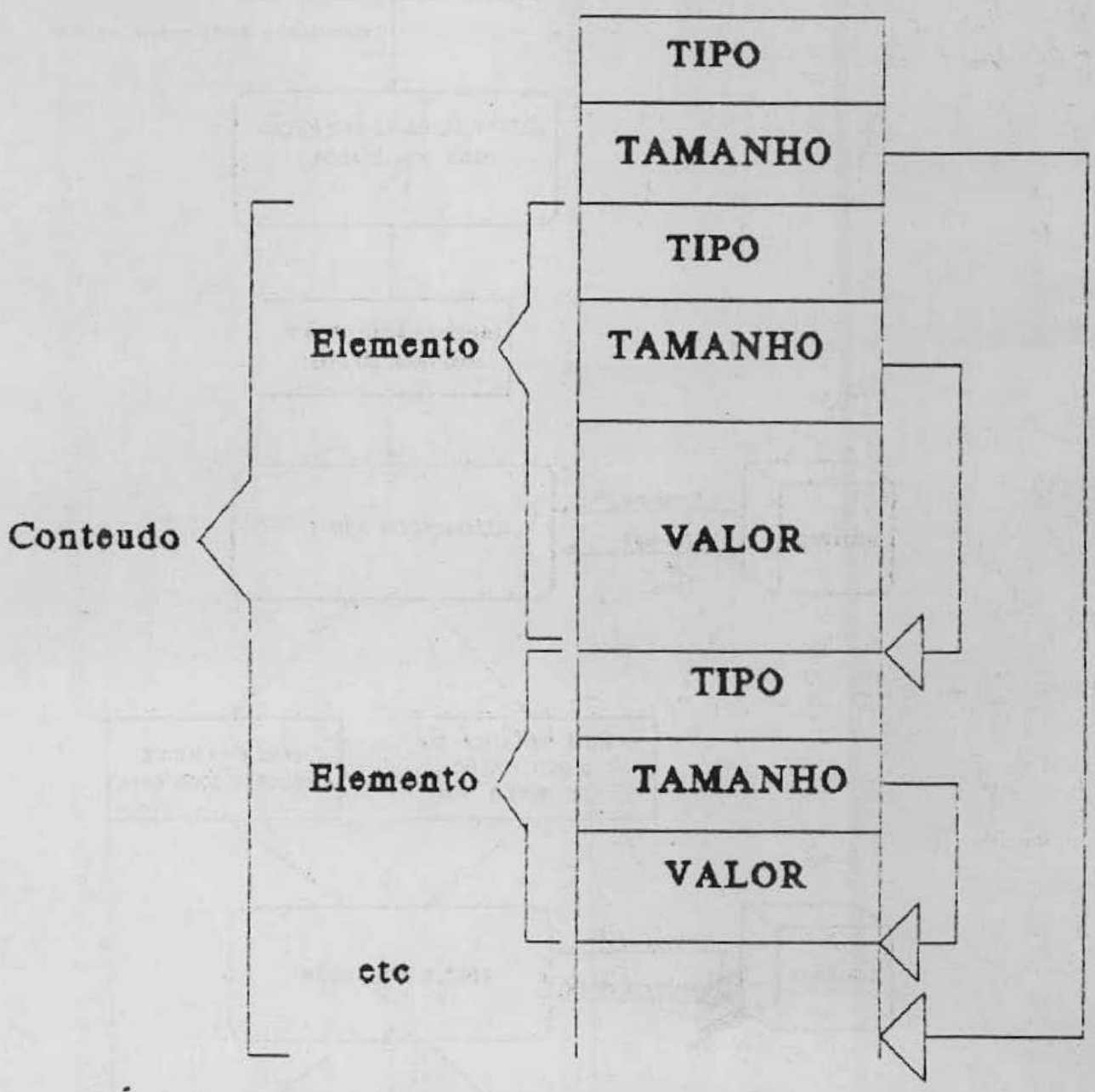

Figura 2 - Esquema de codificacao TLV

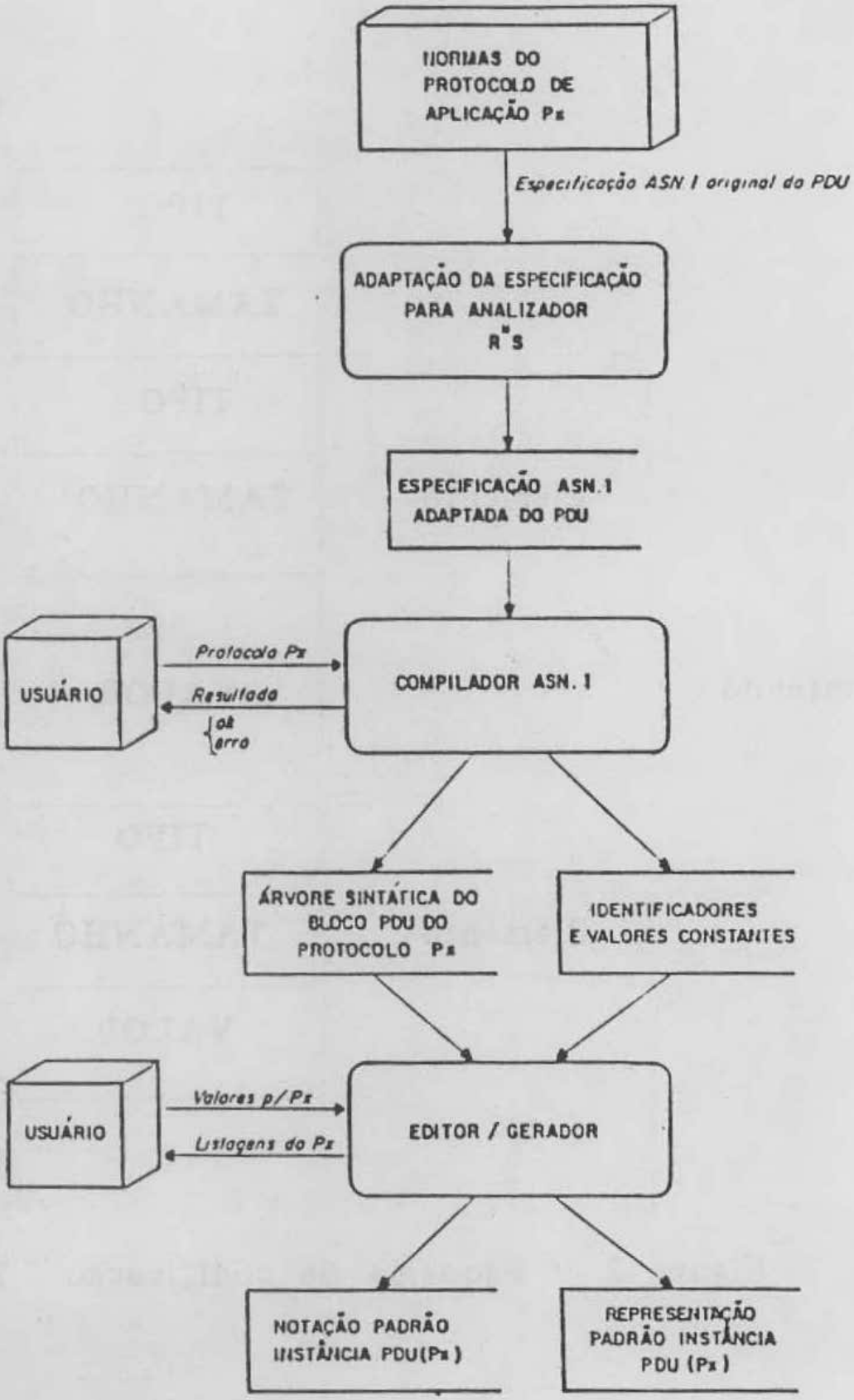

FIG. 3 - ESQUEMA GERAL DE GERAÇÃO DOS BLOCOS PDU PARA UM DETERMINADO PROTOCOLO Px

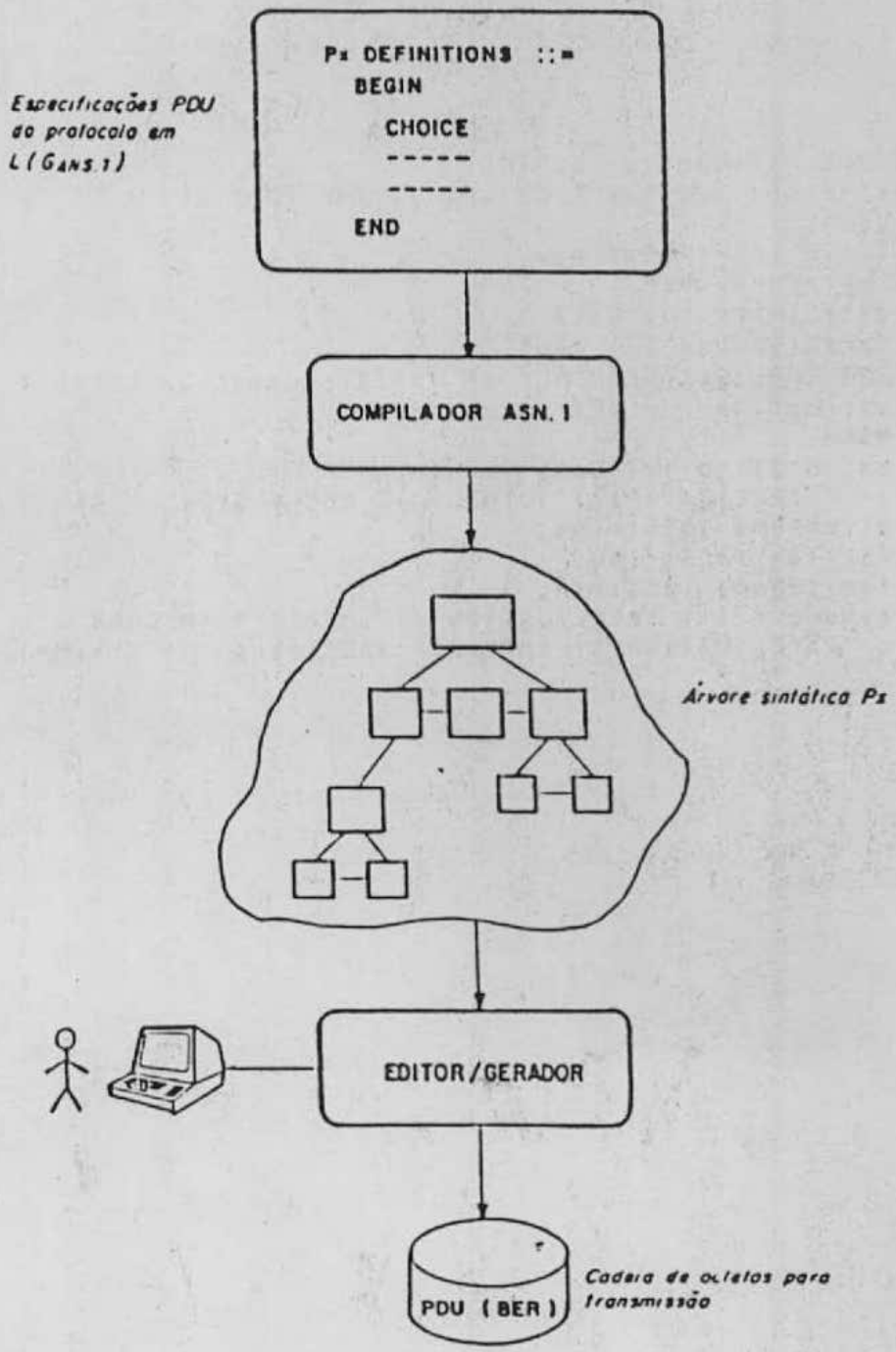

FIG. 4 - ARQUITETURA GERAL DO TRADUTOR

ANEXO - EXEMPLO 1

```
EXE409 DEFINITIONS ::= BEGIN
PersonelRecord ::= [APPLICATION 0] IMPLICIT SET {
        Name,
        title [0] IA5String,
        EmployeeNumber,
        dateOfHire [1] Date,
        nameOfSpouse [2] Name,
        [3] IMPLICIT SEQUENCE OF ChildInformation DEFAULT {} };
ChildInformation ::= SET {
        Name,
        dateOfBirth [0] Date };
Name ::= { APPLICATION 1 } IMPLICIT SEQUENCE {
        givenName IA5String,
        initial IA5String,
        familyName IA5String };
EmployeeNumber ::= [APPLICATION 2] IMPLICIT INTEGER .
Date ::= [APPLICATION 3] IMPLICIT IA5String; -- YYYYMMDD
END
```

ANEXO - EXEMPLO 1

exe409.cod

```
60*80*
   61*80*
      16*04*4A 6F 68 6E *
      16*01*50 *
      16*05*53 6D 69 74 68 *
   00*00*
   A0*0A*
      16*08*44 69 72 65 63 74 6F 72 *
   42*01*33 *
   A1*0A*
      43*08*31 39 37 31 30 39 31 37 *
   A2*80*
      61*80*
         16*04*4D 61 72 79 *
         16*01*54 *
         16*05*53 6D 69 74 68 *
      00*00*
   00*00*
   A3*80*
      31*80*
         61*80*
            16*05*52 61 6C 70 68 *
            16*01*54 *
            16*05*53 6D 69 74 68 *
         00*00*
         A0*0A*
            43*08*31 39 35 37 31 31 31 31 *
      00*00*
      31*80*
         61*80*
            16*05*53 75 73 61 6E *
            16*01*42 *
            16*05*4A 6F 6E 65 73 *
         00*00*
         A0*0A*
            43*08*31 39 35 39 30 37 31 37 *
      00*00*
   00*00*
00*00*
```

ANEXO - EXEMPLO 1

exe409.not

```
{[givenName "John",
  initial "P",
  familyName "Smith"
 ],
 title "Director",
 51,
 dateOfHire "19710917",
 nameOfSpouse
 [givenName "Mary",
  initial "T",
  familyName "Smith"
 ],
 {{[givenName "Ralph",
    initial "T",
    familyName "Smith"
   ],
   dateOfBirth "19571111"
 },
 {[givenName "Susan",
    initial "B",
    familyName "Jones"
   ],
   dateOfBirth "19590717"
 }
}
}
```