Implementação de um Compilador ASN.1-C para o Ambiente SISDI-OSI

María Inés Valderrama Restovíc
Manuel de Jesus Mendes

Universidade Estadual de Campinas
FEE / DCA

Email ines@dca.fee.unicamp.br

Resumo

A crescente utilização do ASN.1 pelos protocolos de aplicação, como forma de padronizar a troca de informação durante a comunicação no ambiente OSI-ISO enfatiza, cada vez mais, a necessidade de automatizar a tarefa de produzir um código em uma linguagem de alto nível. O propósito deste artigo é descrever a implementação de um compilador ASN.1 destinado a produzir um código em C, assim como os programas codificadores e decodificadores correspondentes, dentro do ambiente SISDI-OSI[1]. O usuário fica dessa forma, dotado de uma ferramenta automática que o libera da interpretação do ASN.1 e da geração de códigos.

1 Introdução

O intercâmbio de dados em sistemas abertos de informação enfrenta uma série de obstáculos ocasionados pelas características heterogêneas dos diversos componentes destes sistemas. Aspectos como arquiteturas diferentes, códigos internos, linguagens e diferentes aplicações, exigem grandes esforços de conversões de dados entre os componentes que se comunicam entre si. Esta situação gera dois problemas: como transmitir os dados sem perda da informação e como garantir a semântica dos mesmos[Ros89].

A solução apresentada pela ISO baseia-se em dois conceitos: o de sintaxe abstrata e o de sintaxe de transferência. Por meio da sintaxe abstrata, os tipos de dados que compõem as PDU's (*Protoclo Data Unit*) a serem trocadas pelos protocolos de

[1]SIStema DIdático de protocolos OSI desenvolvido pela faculdade de Engenharia Elétrica da UNICAMP

aplicação podem ser definidos independentemente de qualquer restrição de máquina. A sintaxe de transferência por sua vez, proporciona uma representação para os valores dos tipos dados definidos em forma abstrata, a fim de que eles possam ser transmitidos pela rede.

ASN.1 (*Abstract Sintax Notation One*) é a primeira padronização, e até agora a única, de uma sintaxe abstrata e BER (*Basic Encoding Rules*) sua correspondente sintaxe de transferência.

Teoricamente, a utilização de ambas as linguagens contribuem para a redução das conversões [Tan88] necessárias, e favorece a padronização, segurança e semântica dos dados.

Na prática, é necessário fornecer uma representação real (sintaxe concreta) para a sintaxe abstrata, em alguna linguagem de alto nível como C, Pascal, etc, para que se possa utilizar as vantagens da sintaxe abstrata e de transferência.

Cada usuário deve transformar a definição de seus tipos de dados, em forma abstrata, na sintaxe concreta que ele utiliza. Como as PDU's que compõem os protocolos de aplicação cada vez mais, possuem estruturas de dados maiores e mais complexas, esta tarefa torna-se difícil de ser realizada manualmente. Outro fator a ser considerado é o volume de dados que deve ser codificado ou decodificado utilizando a sintaxe de transferência.

A construção de uma ferramenta que realize a transformação das sintaxes automaticamente e que gere codificadores e decodificadores genéricos, segundo as regras da sintaxe de transferência é descrito a seguir juntamente com uma breve revisão das sintaxes envolvidas (ASN.1 e BER).

2 Principais Tópicos das sintaxes Utilizadas

ASN.1

ASN.1 foi definido no documento ISO 8824 [ISO87a] [ISO87c] e na série de documentos X400 da CCITT nos anos 1987 e 1988. Foi a primeira padronização estabelecida para representar dados em forma abstrata.

A gramática da linguagem foi definida através de mecanismos formais e possui uma notação semelhante a BNF (*Backus Naur Form*). O princípio da linguagem é o mesmo adotado nas linguagens de alto nível: quando um valor é associado a uma variável, sua sintaxe é a do tipo definido; estes tipos são definidos do maior nível de abstração ao menor, ou seja, ASN.1 é uma linguagem *top-down*.

Os principais elementos que compõem o ASN.1 são : **tipos, valores e macros.**

Os **tipos** caracterizam um conjunto de valores e podem ser divididos em duas categorias **tipos primitivos** e **tipos construtores**. Os tipos primitivos correspondem aos mais simples e atômicos tais como, INTEGER, REAL, BOOLEAN, NULL, etc. Uma declaração de um tipo BOOLEAN em ASN.1 tem, por exemplo, a seguinte sintaxe:

EstadoCivil ::= BOOLEAN

casado EstadoCivil ::= TRUE

Os tipos construtores são tipos formados a partir de tipos primitivos. Um tipo construtor pode utilizar tipos simples e também tipos construtores para obter estruturas de tipos de complexidade e profundidade arbitrárias. Os tipos construtores são:

- SEQUENCE corresponde a uma lista ordenada de 0 ou mais elementos de qualquer tipo;
- SEQUENCE OF é uma lista ordenada de 0 ou mais elementos de tipos ASN.1 iguais;
- SET é uma lista ordenada de 0 ou vários elementos de qualquer tipo ASN.1;
- SET OF é uma lista não ordenada de 0 ou mais elementos todos do mensmo tipo ASN.1;
- CHOICE é definido como a união de um ou mais tipos — o valor do tipo corresponde a um dos tipos definidos.

Exemplo de um tipo construtor:

```
Cliente ::= SET {
                nome [0] IMPLICIT OCTETSTRING,
                numero [1] IMPLICIT INTEGER,
                endereço [2] IMPLICIT OCTETSTRING }
```

Como pode ser visto no exemplo acima, é necessário identificar corretamente e sem ambigüidade, cada elemento em um tipo construtor, a fim de resolver problemas de inconsistência em tempo de codificação dos dados. Com esse objetivo, existem em ASN.1 os tipos *tagged* — no exemplo, os números 0 e 2 diferenciam dois elementos do mesmo tipo. Existem quatro *tags* em ASN.1 : UNIVERSAL, CONTEX-SPECIFIC, APPLICATION, PRIVATE. Cada uma atinge um nível diferente dentro das declarações ASN.1.

Os **valores** em ASN.1 são as instâncias dos tipos e são eles que serão codificados e decodificados mediante as regras BER.

Uma ferramenta poderosa do ASN.1 são as **macros**. Criadas com o propósito de estender a linguagem original, uma macro é construída para satisfazer um requerimento que os tipos definidos originalmente não conseguem atingir. Formalmente, uma macro é um mecanismo para construir e referenciar novos tipos em ASN.1, especificando uma nova notação através de um conjunto de produções.

Finalmente, todo esse conjunto de tipos, valores e macros, que compõem as definições de um conjunto de PDU's de algum protocolo de aplicação, são reunidos em

módulos. Cada módulo, além das declarações de tipos, traz consigo as relações do módulo com outros módulos, como por exemplo nas cláusulas EXPORT e IMPORT que indicam os tipos de dados intercambiados pelo conjunto de módulos ASN.1.

BER

Os valores dos tipos de dados definidos em forma abstrata precisam de uma forma concreta de representação para que possam ser transmitidos. Com esse objetivo, a ISO descreve no documento ISO 8825 [ISO87b] a sintaxe de transferência BER (*Basic Encoding Rules*) correspondente ao ASN.1. BER é composta de um conjunto de regras de codificação para cada tipo de dado, a partir das quais cada valor de um tipo será transformado em uma sequência de octetos binários. As regras de codificação são independentes das representações internas dos dados nos diferentes computadores que participam na transferência dos dados. Para isso, a norma BER fixa a posição do bit mais significativo (o bit mais a direita da palavra).

A estrutura geral da codificação é simples e cada valor é codificado em três campos. Essa estrutura é válida para todos os tipos, primitivos ou contrutores, sendo a base da codificação sempre feita através dos tipos primitivos; os tipos construtores são codificados aplicando-se recursivamente, as regras de codificação até chegar à sua forma primitiva.

3 Implementação do Compilador

Antes de descrever a implementação do compilador ASN.1 [Res92], é importante revisar alguns dos aspectos principais da transformação do ASN.1 em C [2] . A idéia básica da transformação é traduzir um tipo ASN.1 em um tipo de dado em C, que permita armazenar o valor do primeiro.

Os tipos ASN.1 são abstratos e carregam muita informação adicional por meio das *tags* que os modificam, como por exemplo, tipo e número da *Tag* , condição EXPLICIT ou IMPLICIT da mesma, etc. Toda esta informação é armazenada na tabela de símbolos do compilador como será visto mais adiante, a fim de poder criar estruturas mínimas para cada tipo de dado.

Tipos Primitivos

Os tipos primitivos são a base do ASN.1 e dos processos de codificação e decodificação. Felizmente, para alguns deles, existe um tipo similar diretamente em C, como é o caso dos tipos INTEGER e REAL transformados em *int* e *float* respectivamente. A Tabela 1 mostra todos os tipos primitivos e as transformações correspondentes.

Como pode ser visto na Tabela 1, os tipos *octecstring* e *bitstring* correspondem aos tipos ASN.1 do mesmo nome e são duas estruturas similares que contêm dois campos. Um campo é uma cadeia de caracteres de tamanho finito e o outro um apontador

[2] A adoção da linguagem C no código gerado pelo compilador, deve-se basicamente a sua ampla utilização no ambiente para o qual o compilador foi desenvolvido

ASN.1	C
BOOLEAN	int
INTEGER	int
BITSTRING	bitstring
OCTETSTRING	octetstring
NULL	int
ANY	any
OBJECT IDENTIFIER	objectid
OBJECT DESCRIPTOR	octetstring
EXTERNAL	external
REAL	float
ENUMERATED	enum

Tabela 1: Tipos C para tipos primitivos

para a estrutura. Dessa forma, esta estrutura pode ser alocada dinamicamente, dependendo do tamanho do valor ASN.1, uma vez que os tipos OCTETSTRING e BITSTRING em ASN.1 não limitam o tamanho de suas cadeias.

Exemplo de uma estrutura OCTETSTRING:

```
typedef char valor[80];

struct octestring {
                    valor dado;
                    struct octetstring *next;
                    }octetstring;
```

O tipo *any* por sua vez, representa um dos tipos mais complicados, porque o tipo assumido só é conhecido no momento em que um valor é atribuído. Escolheu-se para sua representação a seguinte estrutura:

```
struct any {
                valor nome;
                void *ptr;
                } any;
```

O campo **nome** indica, no momento da codificação, o tipo do dado e é responsabilidade do usuário sua correta manipulação. O campo **ptr** será um apontador para o tipo do dado.

Outro tipo interessante de analisar é o *objectid*. Ele corresponde a uma estrutura que armazenará os identificadores númericos que compõem um OBJECT IDENTIFIER, já que são estes, os únicos elementos codificados neste tipo.

```
struct objectid {
                int num[30];
                int numtot;
                } objectid;
```

Tipos Construtores

Todos os tipos construtores do ASN.1 são representados de forma similar, mediante um tipo *struct*. Os elementos que compõem o tipo são por sua vez,transformados, em seus correspondentes tipos em C. O único fator que diferencia os tipos construtores das estruturas em C é a existência dos modificadores OPTIONAL e DEFAULT. Para tanto, foi necessário sempre que um deles ocorresse, acrescentar um campo a mais na nova estrutura. Através de valores (1 ou 0) nesse campo extra, indica-se se o modificador está presente ou não na codificação. Um exemplo desta situação :

```
Dados ::= SET{
                var1 INTEGER,
                var2 OCTETSTRING,
                var3 BITSTRING OPTIONAL
                }
```

Sua representação em C será:

```
struct Dados {
                int var1;
                octetstring var2;
                int var3op;
                bitstring var3;
                };
```

A Tabela 2 apresenta um resumo da representação dos tipos construtores em C.

Macros

As Macros do ASN.1 apresentam um dos maiores problemas de transformação e de compilação, já que é possível estender a gramática com novas regras, o que dificulta a compilação e a geração de um código. Para simplificar essa tarefa e considerando que as macros mais utilizadas são as do protocolo ROSE (*Remote Operation Service Element*, ISO 9072), somente foram criados os tipos correspondentes as quatro macros deste protocolo : BIND, UNBIND, OPERATION, ERROR [ISO89].

ASN.1	C
SEQUENCE SEQUENCE OF SET SET OF	struct { . . . }
CHOICE	union { . . . }

Tabela 2: Tipos C para tipos construtores

Arquitetura do Sistema

O sistema é composto de um compilador de dois passos para ASN.1, um codificador para a sintaxe BER e um decodificador de BER para C. Cada um destes programas utiliza bibliotecas auxiliares de rotinas básicas programadas em C, que contêm as funções de codificação, decodificação e os tipos C para ASN.1.

O compilador ASN.1 foi implementado em equipamento **SUN SPARC STATION 370**, sob o sistema operacional **SUN OS 4.1.1**, compatível com UNIX, utilizando as ferramentas **LEX** (*Lexical Analyzer Generator*) e **YACC** (*Yet Another Compiler Compiler*), além da linguagem C do próprio sistema operacional.

A disponibilidade destas duas ferramentas no sistema UNIX para auxílio na implementação das fases de análise do compilador, aliado à economia de tempo e à eficiência do código gerado, foram os principais fatores que motivaram sua utilização.

A seguir é feito um detalhamento de cada um dos componentes do sistema, apresentados na Figura 1.

Primeiro Passo do Compilador (Passo 1)

Foi necessário dividir o Compilador ASN.1 em dois passos devido à característica *top-down* já mencionada. Como a linguagem C do código gerado é uma linguagem *bottom-up*, o primeiro passo transforma o arquivo de entrada para a mesma especificação ASN.1 só que *bottom-up* (inversão do arquivo). Nessa fase, é realizada uma análise léxica completa e uma análise sintática reduzida, visando identificar cada uma das declarações de tipo para realizar a inversão.

Uma segunda tarefa do primeiro passo é a geração das rotinas específicas de codificação e decodificação para cada módulo ASN.1 compilado. À medida que os tipos de dados são compilados, esse passo vai montado, simultaneamente essas rotinas. As declarações de tipos geram chamadas a funções codificadoras ou decodificadoras específicas disponíveis nas bibliotecas básicas.

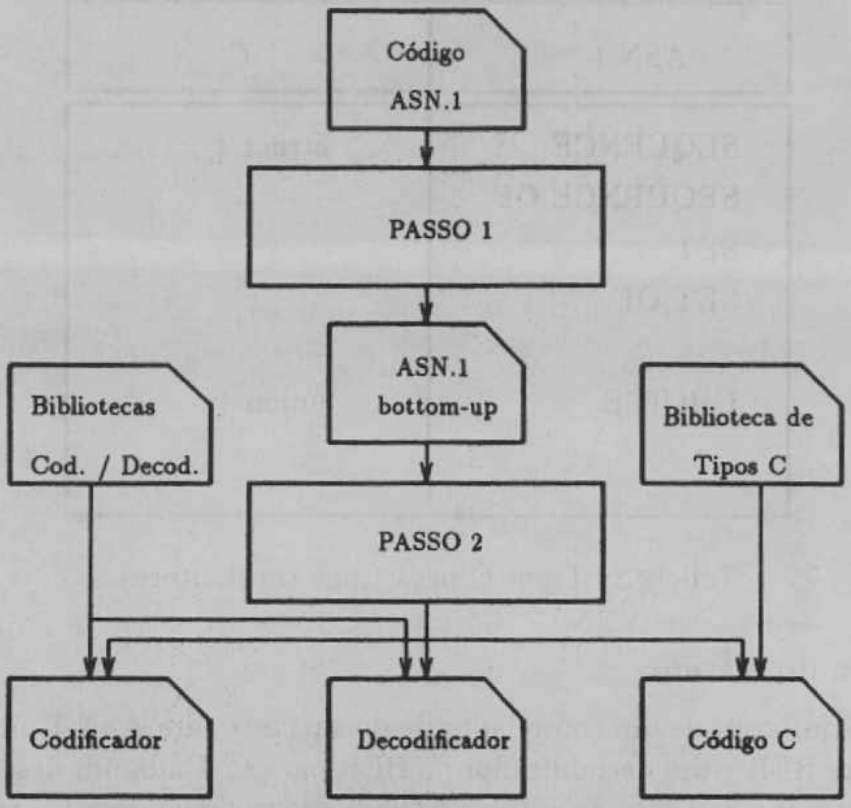

Figura 1: Arquitetura do Compilador

Segundo Passo do Compilador (Passo 2)

Durante o segundo passo do compilador é que se realiza a compilação convencional do módulo ASN.1. Em resumo, ele compreende o processo de geração do código C para as estruturas ASN.1 e a construção da tabela de símbolos do módulo, que será utilizada tanto na de compilação quanto nos processos de codificação e decodificação.

Neste passo são realizadas as análises sintática e semântica completas do arquivo invertido, produto do primeiro passo. O resultado deste passo caso a compilação tenha sucesso, é em primeiro termo o código C equivalente em função dos tipos básicos criados para traduzir os tipos ASN.1, e que se encontram em uma biblioteca que é incluída com os tipos de dados gerados neste passo.

Normalmente uma compilação com sucesso não utiliza a tabela de símbolos uma vez terminada sua tarefa. Entretanto para o compilador ASN.1 esta tabela é também um resultado essencial, uma vez que ela contém toda a informação adicional sobre os tipos compilados que não é mantida nos novos tipos de dados criados. Esta tabela será utilizada pelos programas codificadores e decodificadores para obter a informação adicional necessária, como as *tags*, cujo valor só interessa para transferência de dados.

Bibliotecas

O copilador ASN.1 conta com três bibliotecas auxiliares. A primeira biblioteca de tipos C contém as estruturas padrão do C para cada tipo ASN.1. Esta biblioteca é incluída em cada arquivo de código C gerado pelo compilador.

As rotinas de codificação e decodificação geradas pelo compilador são formadas por

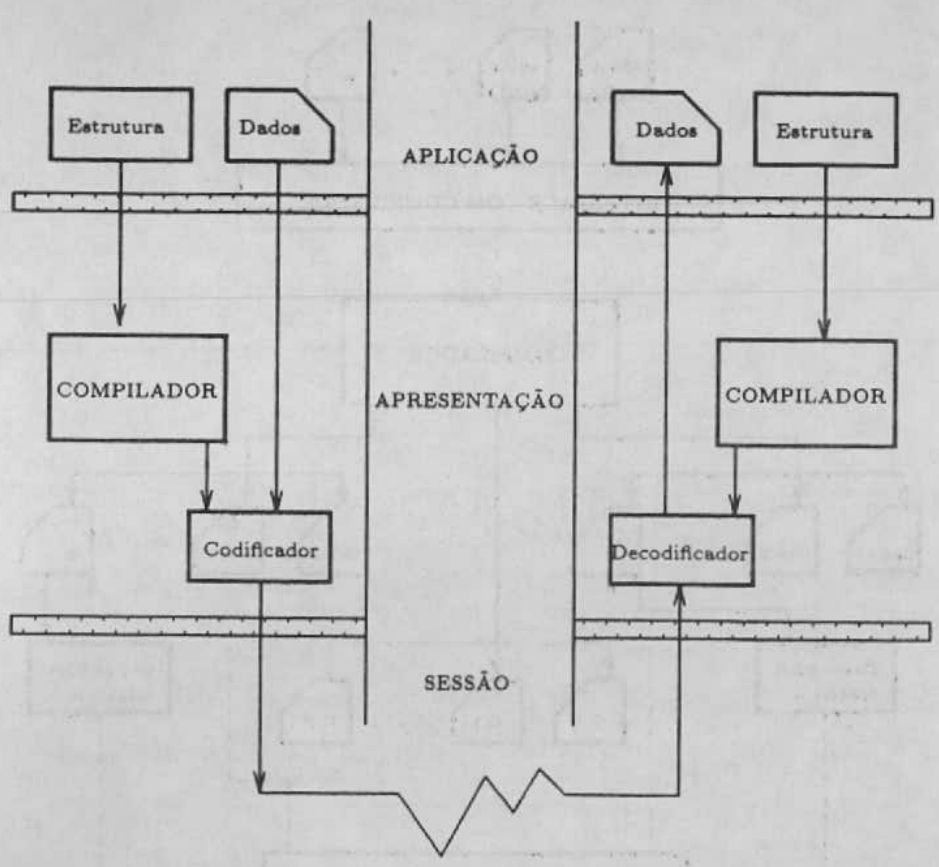

Figura 2: Camadas onde o Sistema atua

um conjunto de chamadas a funções que tratam de cada tipo básico do ASN.1. Estas funções estão reunidas em duas bibliotecas onde cada tipo ASN.1 possui um conjunto de regras na norma BER que codificam o seu valor. Além disso, existe um conjunto de regras genéricas para codificar a informação referente ao tipo e *Tags*; para tanto, existe nas bibliotecas uma função específica para codificar o decodificar cada um dos tipos e seus valores. Estas funções usam a tabela de símbolos e obtêm os seus dados de entrada segundo a especificação gerada pelo segundo passo do compilador.

Localização e Utilização do Compilador

O compilador ASN.1 deve tornar possível a utilização do ASN.1 em um ambiente de protocolos OSI, e dar uma representação ao ASN.1 numa sintaxe concreta.

Ao definir esta ferramenta, foi levado em consideração o ambiente com a qual ela interage. Basicamente, as camadas de aplicação e apresentação. Como mostra a Figura 2, o compilador está situado na camada de apresentação, que gerencia a utilização do mesmo, recebendo os módulos ASN.1 da camada de aplicação para serem compilados e os valores das PDU's que compõem o módulo quando seja necessário codificar, entregando o resultado deste processo à camada de sessão. No caso da decodificação o sentido é inverso.

A Figura 3 apresenta as etapas na utilização do sistema compilador, codificador e decodificador. É possível diferenciar claramente duas fases , uma estática e outra dinâmica.

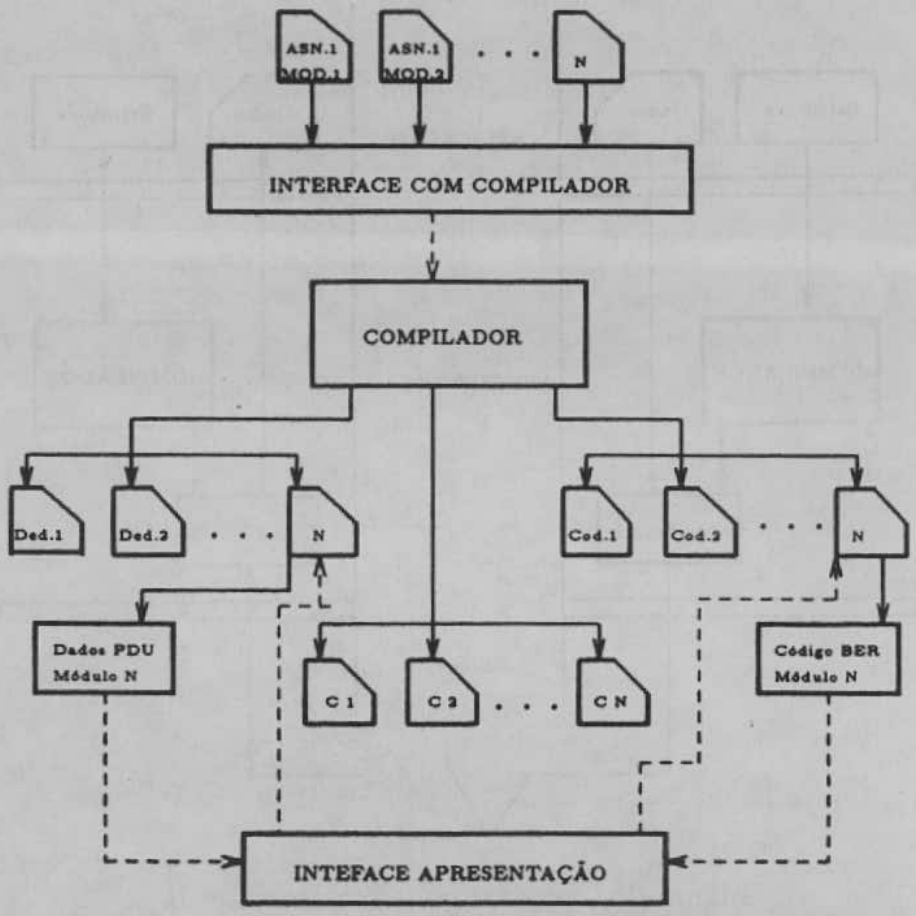

Figura 3: Operação do Sistema

A fase estática refere-se à compilação *off-line* de cada módulo ASN.1, é à geração, de uma só vez, das rotinas codificadoras e decodificadoras.

A segunda etapa, a dinâmica, refere-se à requisição por parte do usuário dos produtos da compilação do módulo ASN.1. As estruturas de dados, agora em C, são incluídas diretamente, como as declarações das PDU's, nas implementações de seus protocolos. Isso é fundamental, porque deve existir uma absoluta concordância entre os tipos de dados usados para definir os valores que serão entregues aos codificadores, já que um codificador específico está baseado nestas estruturas.

As rotinas geradas na compilação serão utilizadas pela camada de apresentação, cada vez que um serviço de codificação ou decodificação seja necessário, sendo responsabilidade da camada verificar se o módulo ASN.1 que contém as especificações das PDU's, já foi compilado. Em caso contrário, deve-se voltar à primeira etapa do sistema e compilar o módulo.

4 Conclusão

Como foi dito na introdução deste artigo, transformar manualmente as especificações das PDU's em estruturas C, torna-se um trabalho difícil a medida que as PDU's crescem em complexidade. O compilador ASN.1 que foi implementado realiza esta

tarefa para qualquer conjunto de PDU's de qualquer protocolo de aplicação, contribuindo para a padronização das estruturas de dados no ambiente em que atua. O fato dele estar programado utilizando as ferramentas **LEX** e **YACC** facilita as alterações, modificações e inclusões de novas regras gramaticais que eventualmente possam aparecer em futuros adendos da norma ASN.1. Uma outra contribuição desta ferramenta é : a geração das rotinas de codificação e decodificação, que serão utilizadas pela camada de aplicação no momento de realizar as conversões, tornando-a independente do tipo dos dados e dos valores das mesmas.

Referências

[ISO87a] *Information processing systems - Opem Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1).* International Standar 8824, 1987.

[ISO87b] *Information processing systems - Opem Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).* International Standar 8825, 1987.

[ISO87c] *Information processing systems - Opem Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1).* Extensions to ASN.1. Draft Addendum 8824/DAD1.1897

[ISO89] *Information processing systems - Text communications Remote Operartions - Part 1, Model, Notation and service definitions.* International Standar 9072-1, 1989.

[Res92] Restovíc M. I. *Compilador ASN.1 e Codificador/Decodificador para BER.*Tese de mestrado. FEE, UNICAMP, 1992.

[Ros89] Rose M.T. *The Open Book, A Practical Perspective on OSI.* Prentice Hall, Englewood Cliffs, New Jersey 1988.

[Tan88] Tannenbaum A.S. *Computer Networks.* Prentice Hall, Englewood Cliffs, New Jersey, 1988.