ASN.1からCへのコンパイラの評価

7M-7

長谷川 亨　　野村 真吾
国際電信電話株式会社 上福岡研究所

1. はじめに

OSIのアプリケーションプロトコル(AP)ではプロトコル要素(PDU)のデータ構造が標準記法ASN.1[1,2]により定義され、そのデータ構造から転送するバイト列への符号化規則も同様にASN.1で定められている。OSIのAPプログラムの開発には、ASN.1で定義したPDUを符号化/復号するコーダプログラム(エンコーダ/デコーダ)の作成コストが問題となる。そこで、筆者らはASN.1による仕様からC言語のデータ構造および専用のコーダプログラムを自動生成するASN.1コンパイラを作成した[3]。本稿では、ASN.1コンパイラの使い勝手(ユーザインタフェース)の良さ及び、生成したコーダプログラムの処理速度/規模を評価したので、その結果について報告する。

2. ASN.1コンパイラの機能

2.1 ASN.1の概要

ASN.1はOSIの上位2層プロトコルのPDUのデータ構造を定義するための標準記法である。PDUのデータ型はASN.1の提供する組み込み型を用いて定義される(抽象構文)。組み込み型は単純型および構造型に分類される。単純型は整数やブーリアンのように構造を持たず、直接に値を示すデータ型である。構造型はSEQUENCEのように構成要素の値を持つデータ型である。またASN.1では定義されたPDUと通信し合うシステム間で転送するバイト列の間の変換規則が符号化規則として規定される。

2.2 コンパイラの機能

コンパイラはASN.1で定義されたデータ構造をデータ型とみなし、定義されたデータ型毎に、

① 対応するCのデータ型、
② コーダとしてのCの関数、
③ 生成したCのデータ型を持つ変数を操作するのに必要なCの関数

を生成する(図1)。

APプログラムの作成者は、生成されたプログラムを用いてPDUのフォーマッティング等の動作を記述して、状態遷移を実行するプログラムを作成する。最後に、これらのプログラムをコンパイル/リンクすることにより実行プログラムを得る。

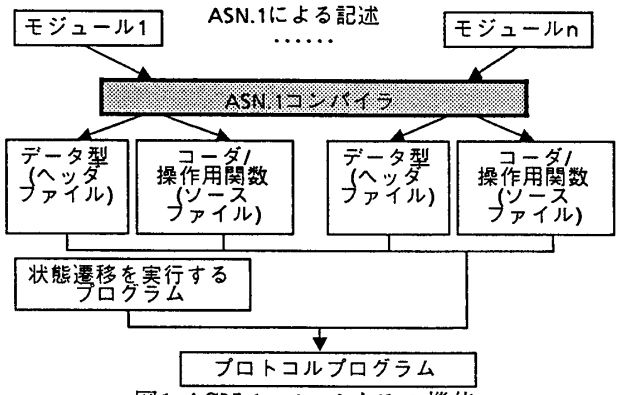

図1 ASN.1コンパイラの機能

3. 使い勝手(ユーザインタフェース)

APプログラムは、ASN.1で定義されたデータ型から変換されたCのデータ型を持つ変数に対する値の参照/設定の操作を行うプログラムになる。従って、この操作を行うプログラムの書き易さ、読み易さ等の使い勝手(ユーザインタフェース)の良さが重要である。図2に示すプログラム例を用いて本コンパイラの使い勝手の良さに対する特徴を述べる。

3.1 ASN.1の仕様中でのデータ型名による参照

本コンパイラはASN.1で定義された構造型をそれぞれ別の構造体に変換し、構造体のデータ型名およびメンバ名をASN.1の仕様中でのデータ型名および要素名を用いている。変数の宣言にASN.1の仕様上でのデータ型名を使用でき(図2の①)、構造体の要素に対する参照も構造型の要素名をメンバ名として用いることができる(図2の②)。このため、Cの変数に対する参照において元のASN.1の仕様でのデータ型の記述と対応関係が明確になり読み易く保守性の高いプログラムになる。

3.2 構造型の要素のユニフォームな参照

構造体のメンバになる構造型の要素はCの整数(int)に変換されるもの以外は全てポインタに変換する。このため構造型の入れ子になった変数の要素へのアクセスは全て"->"による一様なアクセスになり(図2の②)、プログラムが書き易くなる。

Evaluation of ASN.1 Compiler to C Programs
Toru HASEGAWA, Shingo NOMURA
KDD Kamifukuoka R & D Laboratories

```
(1) 抽象構文
Record ::= SEQUENCE {
    name IA5STRING, age INTEGER, chid Child }
Child ::= SET {name IA5STRING, age INTEGER }
(2) Cのデータ型
typedef struct {
    IA5STRING *name; int age; } Child;
typedef struct  {
    IA5STRING *name; int age;
    Child *child } Record;
(3) プログラム
Buffer *buf; /* バッファエリア */
Record *record; …①
bufferAlloc(buf); /* バッファエリアの割り付け */
record = (Record *) asnAlloc ( sizeof (Record));
record->chid = (Child *) asnAlloc ( sizeof (Child));
record->child->age = 5; …②
    …
bufferFree(buf); /* バッファエリアの解放 */
```

図2 プログラムの記述例

3.3 メモリ管理

構造型を持つPDUの作成および受信PDUのデコードでは要素のメモリ領域を動的に割り付ける必要があり、そのPDUの変数が不要になった時点で処理時に割り付けたメモリ領域は全て解放する必要がある。そこで、処理に必要なメモリ領域を最初にまとめて割り付け(図2の関数bufferAlloc)、その領域から必要に応じてメモリを取ってゆく構成とする(図2の関数asnAlloc)。処理が終了した時点で全てのメモリ領域を解放できるため(図2の関数bufferFree)、メモリ領域の解放に関して気を使う必要が無い。

4. 生成するコーダプログラムの評価

ASN.1のコーダプログラムには専用コーダ[3]と汎用コーダ[4,5]の2通りの方式が考えられるが、処理速度の観点から、本コンパイラは専用コーダ方式のプログラムを生成する[3]。

4.1 処理速度

生成したコーダプログラムの処理速度を評価するために、0.9 MIPSの処理能力を持つμVAX VMS上で、エンコードおよびデコードの処理速度を評価した。図3に示す構造を持つ2種類のデータを用いて表1に示す結果を得た。またデータは各データの長さを示す値を持つ固定長のデータである。

表1 エンコード/デコードの処理時間

データ＼処理	エンコード	デコード
タイプ1	164 ms	102 ms
タイプ2	258 ms	180 ms

生成したコーダプログラムは、汎用コーダ方式による従来の報告[4]と比較して2~3倍の処理速度であり、実用的な処理速度を達成している。

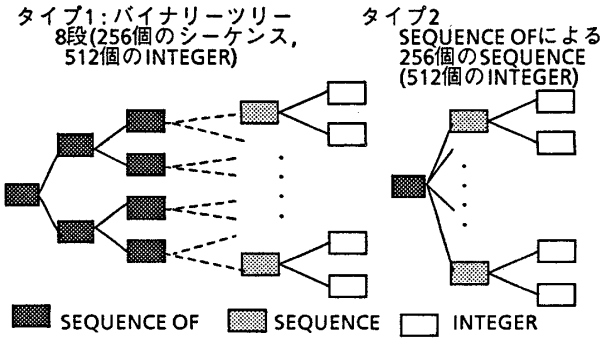

図3 評価データの構造

4.2 プログラム規模

生成するコーダプログラムの規模を評価するため、本コンパイラを実用的なプロトコル(OSI FTAM)に適用した。コンパイラは約430行のFTAMプロトコルの抽象構文に対して約9.8K行(ヘッダファイル:1.2K, ソースファイル:8.6K)の実用的な規模のプログラムを生成しており、人手によるプログラムと比較しても遜色のない大きさである。

4.3 考察

SEQUENCE OFの要素をシングルリンクの線形リストとして扱い要素のアクセスに時間がかかるため、SEQUENCE OFの要素数の多いデータ(タイプ2)に関して、コーダの処理速度が遅くなる傾向がある。そこで、この線形リストのデータ型および線形リストを処理する関数に関して書き直しを行っている。また、コンパイラではASN.1のデータ型に付加されるタグの処理等をコンパイル時に決定しているため、汎用コーダ方式に比較して速い処理速度を達成していると考えらる。

5. おわりに

本稿ではASN.1からCへのコンパイラについて評価し、実用的なプログラムを生成できることを確認した。今後、実際のOSIプログラム開発に適用し評価を進めるとともに、相互に参照しあう構造型及びマクロ機能をサポートできるように機能拡張を行う予定である。最後に日頃御指導頂くKDD上福岡研究所通信ソフトウェア研究室、小西室長、若原主任研究員、コンピュータ通信研究室加藤主査に感謝する。

参考文献

[1]:ISO,"ASN.1", ISO / IS 8824 / 8825.
[2]:CCITT, Rec. X.208 / 209 , Nov. 1987.
[3]:長谷川, 野村, 堀内 "ASN.1支援ツールの開発 - コンパイラおよびエディタ-", 情報処理学会マルチメディアと分散処理研究会, 39-4, Sept. 1988.
[4]:Nakakawaji,Katsuyama,Miyauchi, Mizuno,"Development and Evaluation of APRICOT", The second ISIIS, pp55-62, Nov. 1988.
[5]:Ohara,Suganuma,Senda, "ASN.1 tools for Semi-automatic Implementations of OSI Application Layer Protocols, pp63-70,The second ISIIS, Nov. 1988.