6L−7

ASN.1 から LOTOS ADT への変換

藤尾 光彦[1] 五ノ井 敏行[2] 高橋 薫[3]
[1]シャープ(株) [2]富士通(株) [3]東北大学

1 まえがき

本稿では ASN.1 によるデータ型定義を LOTOS の ADT(抽象データ型) にいかに変換するかについて検討し，同時に ASN.1 符号化規則の LOTOS 化について考察する．前者により準形式的な言語 ASN.1 が形式的に取り扱い可能となる．また後者によりプレゼンテーション層の LOTOS を介した形式化，自動化が期待される．

2 ASN.1 の LOTOS による解釈

ASN.1 から LOTOS への変換 (解釈) においては図 1 に示すようにライブラリ方式をとる．

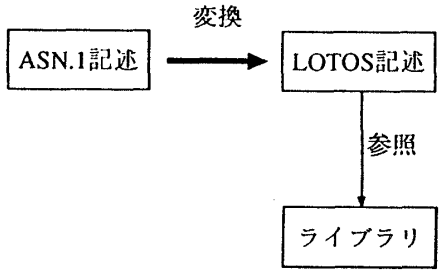

図 1: 変換方式

つまり，予め有用な LOTOS データ型を ASN.1 ライブラリとして用意し，それを基本として各 ASN.1 型を表現する．ライブラリ方式の採用により変換の柔軟性や簡潔性を利点として得ることができる．ASN.1 ライブラリでは論理型，整数型，実数型，ビット型などを定義している．

また，各型の対応するインスタンスの間の比較を可能とするため，特別なデータ型 comp を用意している．これは自然数と同型な型であり，各型 T のインスタンスを comp に埋め込む演算 h の導入とともに，比較の統一的な定義を可能にしている．つまり，

```
h: T -> comp
```

を用いて，

```
opns  _==_  :  T,T -> Bool
eqns  forall x,y:T
        ofsort Bool
      x==y  =  h(x)==h(y)  ;
```

によって定義する．

紙面の都合上，以下では論理型と順序型に絞り，例を主体に変換法を与える．

(1) 論理型 (Boolean)

```
Married ::= BOOLEAN
```

は，ASN1_Boolean を用いて，

```
type Marriage is
  ASN_Boolean renamedby
      sortnames  Married  for  Bool
endtype
```

に変換される．ASN1_Boolean では LOTOS の標準ライブラリの Boolean 型および前記の comp 型を利用し，論理値 true, false の定義並びに h による比較 == の定義を行なっている．

(2) 順序列型 (Sequence)

```
Seq ::= SEQUENCE {ok BOOLEAN, id INTEGER}
```

は ASN.1 ライブラリ中の comp, ASN1_Boolean, ASN1_Integer を用いて，

```
type Seq is comp,ASN1_Boolean,ASN1_Integer
  sorts  Seq
  opns   Seq   :  Bool, Integer -> Seq
         ok    :  Seq -> Bool
         id    :  Seq -> Integer
         h     :  Seq -> comp
         _==_  :  Seq,Seq -> Bool
  eqns   forall x,y:Seq,x1,:Boolean,x2:Integer
         ofsort Bool
         ok(Seq(x1,x2))  =  x1  ;
         ofsort Integer
         id(Seq(x1,x2))  =  x2  ;
         ofsort comp
         h(x)  =  k(h(ok(x)),h(id(x)))  ;
         ofsort Bool
         x==y  =  h(x)==h(y)  ;
endtype
```

に変換される．つまり，演算 Seq により型が構成され，その構成要素は演算 ok と id により示される．また，h を介して比較が定義されている．k は型 comp 中の演算

```
k: comp,comp -> comp
```

Transformation of ASN.1 Data Types into LOTOS ADTs
Mitsuhiko FUJIO, Toshiyuki GONOI, Kaoru TAKAHASHI
[1]SHARP Corp. [2]FUJITSU Ltd. [3]Tohoku Univ.

であり，1対1の作用素である．

3 符号化規則のLOTOS表現

符号化規則のLOTOS表現とは，ASN.1で規定されている抽象構文と転送構文の相互作用をLOTOSで行なおうというものである．(図2参照)．

図 2: 相互変換の位置付け

具体的には，前節のLOTOSで定義した抽象構文とLOTOS標準ライブラリのOctetString間の相互変換手法を与える．

ASN.1において，抽象構文と転送構文の間には一般的に1対nの関係がある．すなわち，ある抽象構文に対応する転送構文は複数存在し，どの転送構文を使用するかはインプリメンタに任されている．

本稿においては，インプリメンタに任されている部分に何ら制約を設けることなく，抽象構文と転送構文の変換を行なうことを可能にした．具体的にはLOTOSプロセスにおける非決定性を利用し，抽象構文から転送構文の変換を非決定的に行なっている．そのため符号化規則のLOTOS化はADT部のみでは対応できず，ADT部とプロセス部の共同作業により相互作用を行なうことになる．

また，それぞれの型で共通的な部分(ADT定義のみ)を符号化ライブラリとし，記述のコンパクト化も図っている．

(1) 論理型 (Boolean)

論理型の定義と使用例を以下に与える．

```
eqns  forall os:OctetString
        ofsort Bool
      DecBool(os)  =  GetCon(os) ne 0  ;
                    ⋮
(* b を os に符号化する *)
choice os:OctetString [b eq DecBool(os)][]
...
```

図 3: Boolean の符号化

図3で，eqnsから「⋮」以前の部分がADT部である(実際はライブラリを参照している．例えば，GetConはライブラリ中で定義された，内容オクテットを切り出す演算である)．後半がプロセス部である．後者でchoice文を使用することにより，Bool型からOctetString型に変換される．ASN.1ではブール値が真の時の値は0以外の任意の値としか規定されていない．しかし，この手法によると0以外の任意の値を与えることが可能となる．

(2) 順序列型 (Sequence)

前節で扱った順序型の相互変換を与える．

```
opns  DecSeq  :  OctetString -> Seq
eqns  forall os:OctetString
        ofsort Seq
      DecSeq(os)  =  Seq(
       DecBool(GetCar(GetCon(os))),
       DecInteger(GetCdr(GetCon(os))) );
                    ⋮
(* seq を os に符号化する *)
choice os:OctetString [seq eq DecSeq(os)][]
...
```

図 4: Sequence の符号化

DecSeqを用いることによりOctetString型からSeq型への変換が，choice文で振らせることによりSeq型からOctetString型への変換が，それぞれ可能となる．GetCar, GetCdrはともにライブラリ中で定義された演算で，それぞれ最初の成分，その次の成分を切り出す．

具体的に値を代入してみる．

```
os = 30 06 01 01 FF 02 01 05
```

の時に，

```
GetCon(os) = 01 01 FF 02 01 05,
GetCar(GetCon(os)) = 01 01 FF,
GetCdr(GetCdr(os)) = 02 01 05,
GetCon(GetCar(GetCon(os))) = FF,
GetCon(GetCdr(GetCdr(os))) = 05
```

より，

```
DecBool(GetCar(GetCon(os))) = true,
DecInteger(GetCdr(GetCdr(os))) = 5.
```

従って，

```
DecSeq(os) = Seq(true, 5)
```

となる．

符号化はchoice文により行なわれている．従って一意な結果とはならない．

4 むすび

本研究は，統ての型に対して具体的な変換法を与えている．
今後はこの変換法の機械化を研究していく．

参考文献

[1] 五ノ井他，"ASN.1からLOTOS ADTへの変換法"，情報処理学会マルチメディア通信と分散処理研究会 46-2, 1990.