7U-5

プロトコルにおけるフォーマット変換の形式的記述法

大竹和雄

日本電気(株) ソフトウェア生産技術研究所

1. はじめに

コンピュータネットワーク技術の進歩につれプロトコルも高度化し、日常語で記述することが困難になってきた。この解決をはかるためにプロトコルの形式的記述法の研究が盛んになっている。一方で高度化したプロトコルにのっとったソフトウェアの作成を支援するよりどころとしても形式的記述法は重要である。ところでプロトコルは手順の規約とフォーマットの規約から成る。下位層では手順が重要であるが上位層では手順は縮退し、受け渡すメッセージが大きく複雑なフォーマットを持つようになる。そこでフォーマットのための形式的な記述法が求められる。また下位層では実際に使われるプロトコルは限定された個数におちつくと思われるが、上位層でははるかに多くのプロトコルが用いられると考えられる。以上の理由からフォーマットの形式的記述法とそれに基づくソフトウェア作成支援の研究が重要である。

フォーマットの形式的記述法としてはＢＮＦに基づくもの[1]や代数的記述などが知られているがそれらをソフトウェア生産技術に用いた例は見かけない。本報告で提案する記述法はフォーマットの記述だけでなくそのフォーマットを持つデータに対して施される処理のアウトラインの記述を加えることにより、ソフトウェア作成に結びつけられるようにしている。

2. フォーマット変換記述法

図１で示すような構造を持つmail_aとmail_bを例に用いる。本記述法でmail_aからmail_bへの変換を記述したものが図２である。入力するデータmail_aのフォーマットを原構文、mail_bのフォーマットと入力との対応を記述した部分を目的構文と呼ぶ。

2.1 対応の基本的表現

本記述法はＢＮＦ記法を拡張したものになっている。原構文の記述はＢＮＦそのものである。

目的構文は各要素が原構文の要素をパラメータとして持つ。

③はmail_aのheaderとbodyにそれぞれmail_bのenvelopeとcontentがおおまかに対応していることを表して

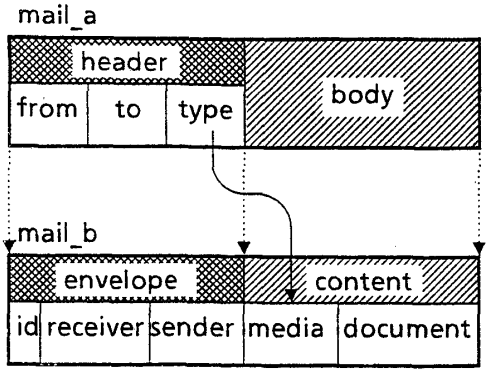

図1.

```
mail_a : header body ;                                    ①
header : from to type ;                                   ②

mail_b(mail_a) : envelope(header) contnet(body) ;         ③
envelope(header) : id() receiver(to) sender(from) ;       ④
content(body) : media(.header.type) document(body) ;      ⑤

body : characters
     | page-size characters ;                             ⑥
document(body) : (characters) free(characters)
     | (page-size, characters)
               standard(page-size, characters) ;          ⑦
```

図２.

Formal Description Technique for Protocol Format Transformation

Kazuo OHTAKE

NEC Corporation

いる。④と⑤はそのひとつ下の階層の対応を記述している。envelopeのidは毎回一意に識別できる番号を付加するものでmail_aのデータには依存しないのでパラメータを持たない。図1の矢印で示しているようにtypeとmediaは①のおおまかな対応を外れている。⑤をみるとmediaのパラメータは"."で始っているがこれは"."で区切って原構文の要素名を並べることによって階層の最上位（ここではmail_a）からのたどり方を指示している。このようにしてmediaのパラメータがbodyの下位の要素ではないことを表現している。

目的構文においてmail_bの構造は③で定義され、envelope、contentはそれぞれ④、⑤で定義されているが、その他の下線の引いてあるid、receiver、sender等はその構造は記述中では定義されない。これらの終端名はユーザがコーディングする処理プログラムとのインターフェースとなる。例えば入力の宛先toから出力の宛先receiverを作る処理はreceiverという関数名を持つプログラムによって行なわれる。

2．2　選択

入力データの形式によって出力データの形式の異なるものが選択される場合の記述を図2の⑥、⑦を用いて説明する。

入力は文字列のみの場合と頭にA4、B5等のサイズが指定してある場合があり、それぞれが出力ではfreeとstandardに対応するものとする。これを記述する場合目的構文の各選択肢は頭にガードを持つ。ガードとは入力構文中の名前の列で、入力データ中にガードの要素がすべて現れたとき、その選択肢が選ばれる。（複数のガードが条件を満たした場合はより下に書かれたものが用いられる。）

2．3　その他の機構

簡潔な記述を達成するために導入した機構と記述能力を示す例を説明する。

(1)　本記述法では名前を用いて対応関係を表現しているのですべての要素が互いに区別がつくように名前が付けられていなければならない。次の例のように区別がつかない場合に簡便に名前をつける方法として構文を埋めこむことができる。

```
a : INTEGER INTEGER ;
        ↓
a: <p1: INTEGER> <p2: INTEGER> ;
```

これは次のように書いたのと同じである。

```
a : p1 p2 ;
p1 : INTEGER ;
p2 : INTEGER ;
```

(2)　省略可能な要素を表現するために"[]"を用いる。

```
a : p1 [p2] ;
```

これは目的構文でも用いることができ、次の例では入力にp2が現れたときのみr2を作ることを表す。

```
d(a) : r1(p1) [r2(p2)] ;
```

(3)　最後によく起こる例として原構文のある要素を目的構文に対応させることが不可能な場合の表現法を示す。以下の原構文中でpというオプションが対応不可な場合

```
src : a b [o] [p] [q] ;
```

以下のように原構文からpを落とすとオプションpを持つような入力を受け付けず拒否することを表す。

```
src : a b [o] [q] ;
dst(src) : e(a) f(b) [s(o)] [t(q)] ;
```

また、入力は受け付けるがそのオプションの情報は無視してしまう場合は

```
src : a b [o] [p] [q] ;
dst(src) :  e(a) f(b) [s(o)] [t(q)] ;
```

という記述になる。

3．　おわりに

本記述法を用いてＣＣＩＴＴのメールシステムMHS[1]と当社の標準プロトコルであるＯＩＡの対応を記述し、本記述法が十分な表現能力を持つことを確かめた。現在、本記述からデータを入力してフォーマットを変換するプログラムを生成する処理系の試作を進めている。さらに処理系の試作を通して記述法の充実、他の領域への応用を考えていきたい。

参考文献

[1]　ＣＣＩＴＴ勧告　X.400、X.401、X.408、X.409、X.410、X.411、X.420、X.430