文字列書換えにもとづくＰＬ/０言語処理系記述の試み

７Ｂ－４

川上 孝仁
久留米工業大学

１．はじめに

書換え系は計算モデルとして理論的な研究が進められているが、書換えの考え方は実際のソフトウェアシステムの記述にも有効であろう。本稿では、ソフトウェア構成法としての書換え系の有効性を検証することを目的として、構文解析が比較的容易なプログラミング言語ＰＬ/０[1]の言語処理系を、文字列書換えの考え方にもとづいて記述してみたので報告する。

２．書換え系と言語[2]

ある記号列を別の記号列に置き換える規則を書換え規則という。書換え系は一組の書換え規則に従って、与えられた記号列から新しい記号列を生成する。記号には非終端記号と終端記号の２種類がある。非終端記号に属するある特別な記号である開始記号に対して、書換え規則を繰り返し適用することによって得られる、終端記号からなる記号列の集合を言語という。この場合の書換え規則を生成規則または文法という。逆に、与えられた記号列に生成規則を逆向きに繰り返し適用して開始記号に到達するかどうか判定する機構を言語の受理機構という。

３．書換え機構

本稿において書換え規則を次のように記述する。

左辺文字列 --> 右辺文字列

右辺文字列は、非終端記号ひとつ、または終端記号ひとつからなる。左辺文字列は、非終端記号、終端記号、変数、および、終端／非終端記号を区切る区切り文字の並びからなる。

プログラミング言語処理系の場合、左辺文字列の構成要素は言語処理系の構成要素とおおむね以下のように対応する。

非終端記号・・・構文要素名
終端記号　・・・予約語、定数
（if while end , . など）
変数　　・・・ユーザ定義の名前
（変数名、手続き名など）
区切り文字・・・スペース

An Experimental Implementation of PL/0 Translator based on String-Rewriting Systems
Takato Kawakami (E-mail:kawakami@cc.kurume-it.ac.jp)
Kurume Institute of Technology

書換え規則の集合、入力文字列、トップレベルの終端／非終端記号を与えられて、書換え機構は以下の手順で入力文字列の書換えを実行する。

(1)書換え規則の中で、右辺文字列がトップレベルの終端／非終端記号と一致するものを探す。

(2)その規則の左辺文字列の記号と入力文字列の記号とを比較していく。その記号同士を比較して異なるとき、次のように処理する。

(2-1)左辺文字列の終端記号が対応する入力文字列の終端記号と異なれば、同じ右辺文字列をもつほかの規則を探し、その左辺文字列との比較に移る。

(2-2)左辺文字列が非終端記号である場合は、対応する位置以降の入力文字列を、左辺文字列中の非終端記号をトップレベルの非終端記号として、この書換え手順を再帰的に実行する。

(2-3)左辺文字列が変数の場合は、入力文字列の対応する記号を変数に代入して、記号比較を続ける。

(3)規則の左辺文字列全体が入力文字列（の途中まで）と等しいとき、その入力文字列の一致した位置までを規則の右辺文字列で置き換える。

図１にＰＬ/０における式（expression）の書換え規則（部分）を示す。この規則で、+、-、*、/、(、)、0、1、. . 、9は終端記号、英大文字で始まる名前は非終端記号、$1 は変数をそれぞれ表す。非終端記号のうち名前の末尾が &、# のものは、

```
TermS&   --> Expression          (a)
Term&    --> TermS&              (b)
TermS& + Term& --> TermS&        (c)
TermS& - Term& --> TermS&        (d)
Factor   --> Term&               (e)
Term& * Factor --> Term&
Term& / Factor --> Term&         (f)
0Num#    --> Factor
1Num#    --> Factor              (g)
  :
9Num#    --> Factor
( Expression ) --> Factor        (h)
$1       --> Factor              (i)
0Num#  --> Num#
1Num#  --> Num#                  (j)
  :
9Num#  --> Num#
```

図１．ＰＬ/０の式（expression）の書換え規則（部分）

それぞれ左再帰的規則、右再帰的規則の右辺文字列であることを示す。

図２に、この規則による書換えの過程を、適用した規則とともに示す。

```
[入力文字列] ( a + b ) / 23 - 4
[規則(i)]    ( Factor + b ) / 23 - 4
[規則(e)]    ( Term& + b ) / 23 - 4
[規則(b)]    ( TermS& + b ) / 23 - 4
[規則(i)]    ( TermS& + Factor ) / 23 - 4
[規則(e)]    ( TermS& + Term& ) / 23 - 4
[規則(c)]    ( TermS& ) / 23 - 4
[規則(a)]    ( Expression ) / 23 - 4
[規則(h)]    Factor / 23 - 4
[規則(e)]    Term& / 23 - 4
[規則(j)]    Term& / 23Num# - 4
[規則(j)]    Term& / 2Num# - 4
[規則(g)]    Term& / Factor - 4
[規則(f)]    Term& - 4
[規則(b)]    TermS& - 4
[規則(j)]    TermS& - 4Num#
[規則(g)]    TermS& - Factor
[規則(e)]    TermS& - Term&
[規則(d)]    TermS&
[規則(a)]    Expression
```

図２．書換えの過程

４．スタックおよび書換え時に実行する手続き

以上の方法で構文解析を行うことができるが、ここまでででは与えられた入力文字列が文法的に正しいかどうかの判定が行えるだけである。さらにコンパイラとして構成するには、構文解析の過程でオブジェクトコードを出力する機能を組み込まなければならない。そこで、以下の機能を書換え機構に付加した。

(1)スタック ・・・ 書換えの過程における情報の記憶域として用いる。

(2)書換え時に実行する手続きを規則に付け加えて記述する（記述言語はＣ）ようにした。手続きは２種類ある。

(2-1)書換え前処理手続き ・・・ 適用規則が選択された後、下位の書換えを呼び出す前に実行される手続き。

(2-2)書換え後処理手続き ・・・ 適用規則により書換えが行われた後、実行される手続き。

ＰＬ/０コンパイラの書換え規則（式の部分）を図３に示す。出力するオブジェクトコードは参考文献[1]のインタプリタ（実行時にスタックを用いる）用中間言語である。規則に付加する処理手続きの区切り文字として @ を用いた。処理手続き中の関数として以下のものが別に定義されている。

push, pop ・・・ スタック操作
put, put1, put2 ・・・ オブジェクトコード出力
idref ・・・ 変数／定数表参照

```
"TermS&"   --> "Expression"
"Term&"          --> "TermS&"
"TermS& + Term&" --> "TermS&"    @@@ put("add");@
"TermS& - Term&" --> "TermS&"    @@@ put("sub");@
"Factor"         --> "Term&"
"Term& * Factor" --> "Term&"     @@@ put("mul");@
"Term& / Factor" --> "Term&"     @@@ put("div");@
"0Num#" --> "Factor"          @ push(0);@
                @ x=pop(); put1("lit",x);@
"1Num#" --> "Factor"          @ push(1);@
                @ x=pop(); put1("lit",x);@
   :
"9Num#" --> "Factor"          @ push(9);@
                @ x=pop(); put1("lit",x);@
"( Expression )" --> "Factor"
"$1"    --> "Factor"
 @@@ idref($1,&x,&y);
 if (x==-1) put1("lit",y); else put2("lod",lv-x,y);@
"0Num#"   --> "Num#"       @ x=pop()*10+0; push(x);@
"1Num#"   --> "Num#"       @ x=pop()*10+1; push(x);@
   :
"9Num#"   --> "Num#"       @ x=pop()*10+9; push(x);@
```

図３．ＰＬ/０コンパイラの書換え規則（式の部分）

ＰＬ/０コンパイラ全体の大きさを以下に示す。

規則数： 73 個

付加手続き（Ｃ言語）： 127 行

ただし、言語処理系において重要な、誤りの処理や回復の機能は含まれていない。

５．書換え規則をＣプログラムに変換する書換え規則

書換え規則として構成したＰＬ/０コンパイラは、その規則をＣ言語の文法の形式に変換し、書換え前／後処理手続きを組み込み、書換え機構のプログラムとともにコンパイルすることを経て実行可能なものとなる。この、規則のＣ言語への変換、手続きの組み込みの過程も書換えの考え方にもとづいて構成し、容易に作ることができた。

６．おわりに

本稿では、ＰＬ/０コンパイラを書換えの考え方にもとづいて構成し、記述量少なく実現できることから、書換え系が有効であることを示した。さらに高機能なプログラミング言語も、同様に書換えの方法を用いて効率的に言語処理系を記述できるであろう。

参考文献

[1] N. Wirth:"アルゴリズム＋データ構造＝プログラム", pp. 320-399, 日本コンピュータ協会, 1979.

[2] A. Salomaa:"形式言語とベキ級数". In J. van Leeuwen, editor,"コンピュータ基礎理論ハンドブックII", 第３章, pp. 101-129, 丸善, 1994.