PUC

Series: Monographs in Computer Science and Computer Applications

Nº 5/69

# ON THE BROOKER - MORRIS EXPRESSION RECOGNITION ROUTINE

by

SÉRGIO EDUARDO RODRIGUES DE CARVALHO

Computer Science Department - Rio Datacenter

CENTRO TECNICO CIENTIFICO
Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 209 — ZC-20
Rio de Janeiro — Brasil

# ON THE BROOKER - MORRIS EXPRESSION RECOGNITION ROUTINE

Abstract

A program for the Expression Recognition Routine developed by Brooker and Morris was presented by Douglas K. Smith in his article "List Processing Language Slip" (Rosen, Programming Systems and Languages) (Mc Graw Hill). The purpose of this paper is to develop some considerations on certain aspects of the program, particulary those referring to the syntax that must be constructed for the Brooker and Morris routine.

## 1. Preliminary Considerations

Brooker and Morris developed a routine for the recognition of an expressionbelonging to the set of terminal strings that may be generated from a given syntax especification (1).

The basic idea consists in the construction of a syntactic tree, whose root is the class identifier of the class we suppose the terminal string belong to. This class may be composed of alternatives, each of which may be composed of at least one component. Each component may, in turn, be a class identifier, which may be defined as a sequence of alternatives; each alternative may have components, which may be class identifiers, and so on. Whenever a class identifier is composed of terminal characters, a comparison is made between these characters and the symbol of the input string being analyzed. If a match occurs, the routine backs up and examines the next component at the next higher level, and the process continues.

If the comparison fails, the routine backs up and tries the next alternative. If there are no more alternatives to try, the routine backs up and tries the next alternative at the next higher level.
There are several algorithms which can efficiently execute a top-down analysis of this kind.[(2),(3),(4)]. In a great number of cases, however, extensive syntax tables must ne constructed in the computer memory, and often they are not very easy to understand by the beginner.
The mechanism of analysis becomes very clear however if we use list-processing languages. In fact, the recursive features of such the analysis becomes an easy task.
An example of such an analysis is the SLIP program developed by D.K.Smith [(5)] for the Brooker Morris routine. This program reads the syntax rules in the form of lists, each alternative being considered as a sublist. After the initialization phase, each list will contain either a set of sublist represented by terminal symbols or a set of sublists represented by references to the list corresponding to the class identifiers of the equivalent syntax rule.
The analysis is performed by a set of statements which may be called recursively. The output of this program consists of the analysis record, which is a trace through the syntactic tree constructed. This output was slightly modified in the program that follows to permit a more convenient way of going through the analysis ; as each symbol is recognized, it is printed, together wuth the name of the class and the number of the alternative it belongs to.
The analysis record can also optionally be printed.

## 2. Some Considerations on the Syntax

The Syntax of the languages was also modified to allow the recognition of an arithmetic expression. If Ref.(5), the syntax used was, in the notation of Ref. 1.

1.[L]= A,B,C,D,...,Z　　　　2.[D]= 1,2,3,...,0

3.[L,D]=[L],[D]　　　　(2)

4. [N] = [D][N], [D]
5. [L,D*] = [L,D][L,D*], [L,D]
6. [V] = [L][L,D*], [L] (1)

These six rules establishes the way a variable may be generated: it may be a letter ( L ) followed by any number of letters pt digits ( L,D* ), or it may be a letter only (as a matter of fact, rule 4. is unnecessary). An interesting fact about the construction of these rules concerns the recursive definitions of rules 4. and 5.. The usual way in which such rules are written, in an algol-like language, is:

4. A N = D , D N
5. A L,D* = L,D , L,D L,D * (2)

or, in words, concerning definition 4.A: a number (N) may be a digit (D), or it may be a digit followed by anything already defined as a number. Suppose now that we have the number 258 that must be recognized (it is clear that this can be achieved through syntax (1)). Remembering that the analysis is done symbol by symbol, we will have first to recognize the digit 2,belonging to class N. If we had the definition 4.A in place of definition 4., we would read: a number may be a digit; a digit may be a "1" (first alternative); as this is a terminal symbol, a comparison now succeeds. As the comparison succeeds, we have now recognized a digit, and if a number is a digit, we also recognized a number. In conclusion, the digits 5 and 8 would have been bypassed and we would certainly have trouble. For this reason, all the recursive definitions were written in the form specified by definitions 4. and 5..
For the sake of simplicity, all the terminal symbols were defined as separate classes. The following are the syntax rules in the program of section 3.

[L] = A,B,C,D,...,Z
[N] = 1,2,...,0
[OSS] = +, -
[OMD] = *,/
[OR] = =
[OP] = .

[OE]= '  (meaning exponetiation)

[ODE]= $  (meaning left parenthesis)

[ODD]= ,  (meaning right parenthesis)

[L,N]= [L],[N]

[L,N *]=[L,N] [L,N *],[L,N]

[V]=[L] [L,N *],[L]

[N *]=[N] [N *],[N]

[K]=[N *] [OP] [N *],[OP] [N *],[N *] [OP],[N *]

[P]=[K],[V],[ODE] [SAE] [ODD]

[EXP]=[OE] [P]

[EXP *]= [EXP] [EXP *],[EXP]

[F]= [P] [EXP *],[P]

[MF]=[OMD] [F]

[MF *]=[MF] [MF *] ,[MF]

[T]= [F] [MF *],[F]

[AT]=[OSS] [T]

[AT *]=[AT] [AT *] ,[AT]

[SAE]= [OSS] [T] [AT *],[OSS] [T],[T] [AT *],[T]

[AE]= [V] [OR] [SAE]

3. Program for the recognition of arithmetic expressions

References

1. Rosen, S. "A compiler building system developed by Brooker and Morris" Comm. ACM. 403-411. July, 1964

2. Cheatham,T.E. and Sattley,K. "Syntax directed compiling. Proc. AFIPS 1964 SJCC, vol. 25, pp 31-57.

3. Feldman, I. and Gries,D. "Translator writing systems" Comm. ACM 77-113, Feb, 1968.

4. Floyd,R. "The Syntax of Programming Languages" a survey IEEE Trans. ECI 13,4, 346-353, Aug. 1964.

5. Smith,D.K. "List Processing Language Slip" "Programming Systems and Languages" Rosen,S. ed. Mc Graw Hill, 1967.

```
)7SRCASERGIO CARVALHO                          FORTRAN SOURCE LIST
 ISN        SOURCE STATEMENT

  0 $IBFTC
    C       ***************************************************************
    C       BROOKER AND MORRIS
    C
    C       EXPRESSION RECOGNITION ROUTINE
    C
    C       ***************************************************************
  1         COMMON AVSL,W(100)
  2         DIMENSION SPACE(10000),CLASS(30),ADDRS(30),XINPUT(70),OUT(200)
  3         DATA B/1H /
  4         DATA MASK/077000000000/
  5         INTEGER TOP
  6         CALL INITAS(SPACE,10000)
  7         ASSIGN 1 TO I
    C       ***************************************************************
    C       INPUT OF THE SYNTAX LISTS
    C       ***************************************************************
 10         READ 91,NCD ,IN
 13      91 FORMAT(2I5)
 14         PRINT 1000
 15    1000 FORMAT(1H=//////)
 16         PRINT 87
 17      87 FORMAT(///20X,18HSYNTAX OF LANGUAGE)
 20         DO 55 N=1,NCD
 21         READ 99,CLASS(N)
 22      99 FORMAT(A6)
 23         PRINT 86,CLASS(N)
 24      86 FORMAT(//10X,23HLIST RELATIVE TO CLASS ,A6)
 25         ADDRS(N) = RDLSTA(Z)
 26      55 CONTINUE
    C       ***************************************************************
    C       GERERATION OF THE LIST STRUCTURE
    C       ***************************************************************
 30         DO 66 N=IN,NCD
 31         CALL STRDIR(LRDROV(ADDRS(N)),R)
 32      58 X = ADVSWR(R,F)
 33         IF(F)66,60,66
 34      60 DO 62 NN=1,NCD
 35         IF(X-CLASS(NN)) 62,64,62
 36      62 CONTINUE
 40         GO TO 58
 41      64 CALL STRIND(ADDRS(NN),LPNTR(R)+1)
 42         GO TO 58
 43      66 CALL IRARDR(R)
    C       ***************************************************************
    C       INPUT OF CLASS AND EXPRESSION TO BE RECOGNIZED
    C       ***************************************************************
 45     100 J = 1
 46         II = 1
 47         K = 1
 50         READ 98,GOAL,IND,IFLAG,XINPUT
 53      98 FORMAT(A6,2I2,70A1)
 54         PRINT 1000
 55         PRINT 92,XINPUT
 56      92 FORMAT(///20X,27HEXPRESSION TO BE RECOGNIZED,5X,70A1)
```

```
207SRCASERGIO CARVALHO                    FORTRAN SOURCB LIST
 ISN        SOURCE STATEMENT

            PRINT 90,GCAL
         90 FORMAT(/20X 19HBELONGING TO CLASS ,5X,A6)
            DO 105 L=1,NCD
            IF(CLASS(L)-GOAL) 105,107,105
        105 CONTINUE
            PRINT 85
         85 FORMAT(/15X,20HCLASS NOT RECOGNIZED)
            GO TO 202
        207 PRINT 208
        208 FORMAT(/15X,20HINCORRECT EXPRESSION)
            GO TO 202
        204 CALL EXIT
        107 CALL VISIT(I,PARMTN(ADDRS(L),J,K,0))
C           ******************************************************************
C           OUTPUT OF THE ANALYSIS RECORD
C           ******************************************************************
        206 IF(EQUAL(XINPUT(II),B))207,205,207
        205 IF(IND-1) 202,203,202
        203 K = K-1
            PRINT 210, XINPUT
        210 FORMAT(//6X,18HANALYSIS RECORD OF,6X,70A1//
          110X,8HLOCATION,4X,8HCONTENTS)
            DO 118 L=1,K
            IF(LNKL(OUT(L))) 117,115,117
        115 PRINT 89, L,OUT(L)
         89 FORMAT(10X,I5,I13)
            GO TO 118
        117 PRINT 88, L,OUT(L)
         88 FORMAT(10X,I5,9X,3HCAT,2X,A6)
        118 CONTINUE
        202 IF(IFLAG-9) 100,204,100
C           ******************************************************************
C           START OF THE SYNTAX SCAN
C           ******************************************************************
          1 CALL NEWTOP(LRDROV(TOP(W(1))),W(5))
          2 CALL STRDIR(TOP(W(5)),R)
            CALL STRDIR(TOP(W(1)),XNAME)
            J = TOP(W(2))
            K = TOP(W(3))
            CAT = ADVSER(R,F)
            IF(F) 7,11,7
          7 J = TOP(W(2))
            K = TOP(W(3))
            CALL IRARDR(R)
            CALL TERM(-1,RESTOR(5))
         11 CALL SUBSTP(0,W(4))
            OUT(K) = CAT
            K = K+1+INTGER(SQOUT(MASK,ADVLER(R,F)))
         14 Y= ADVLWR(R,F)
            IF(F) 16,18,16
         16 CALL IRARDR(R)
            CALL TERM(0,RESTOR(5))
         18 IF(NAMTST(Y)) 19,24,19
         19 IF(EQUAL(XINPUT(J),Y)) 22,20,22
```

```
ISRCASERGIO CARVALHO                          FORTRAN SOURCE LIST
ISN          SOURCE STATEMENT

     C       OUTPUT OF THE RECOGNIZED SYMBOL
     C       ***************************************************************
          20 IF(II-J)131,120,131
         120 PRINT 84, Y
          84 FORMAT(/15X,20HRECOGNIZED SYMBOL IS,4X,A1)
             DO 200 LL=   1,NCD
             IF(EQUAL(XNAME,ADDRS(LL))) 200,201,200
         200 CONTINUE
         201 CATEG = CONT(LPNTR(R)-3)
             PRINT 83, CLASS(LL),CATEG
          83 FORMAT(/15X,9HCLASS    ,A6,15H   CATEGORY    ,A6)
             IF(EQUAL(XINPUT(J+1),B)) 31,205,31
          31 II = II+1
         131 J = J+1
             GO TO 14
          22 Q = ADVLWR(LVLRV1(R),F)
             IF(F) 7,2,7
          24 M = 1+ TOP(W(4))
             CALL SUBSTP(M,W(4))
             M = M + TOP(W(3))
             CALL STRDIR(K,CUT(M))
             Z = VISIT(I,PARMTN(Y,J,K,0))
             CALL STRDIR(TOP(W(5)),R)
             IF(Z) 22,14,14
             END
```

```
7SRCASERGIO CARVALHO                     FORTRAN   PROGRAM
ER FUNCTION SUBPROGRAM REFERENCES

    CONT        NAMTST      ADVSWR      INTGER      LVLRV1      LPNTR       LN
    LRDROV      TOP         ADVLER      RDLSTA      ADVLWR      SQOUT
```