TRABAJO GRUPO 98: MEMORIA

Construcción de un Compilador

Asignatura: Procesadores de Lenguajes

Luis Bravo Collado

Inés Hernández Sánchez

Andrea Blanco Gómez

ÍNDICE

5. DISEÑO DEL ANALIZADOR SINTÁCTICO

El analizador sintáctico es el principal elemento del compilador. Recibe los tokens del analizador léxico a medida que los va tratando: trata un token, pide otro. En nuestro caso, utilizamos un analizador sintáctico descendente, que genera un análisis a izquierdas de una cadena de entrada dada.

5.1 Gramática

S' -> W
W -> SW
W -> FW
W -> eof
S -> varTid;
S -> for(D;E;O){S}S
S -> NS
S -> λ
F -> function F' id(P){S}
F' -> λ
F' -> T
T -> int
T -> bool
T -> string
P -> λ
P -> TidT'
T' -> λ
T' -> ,TidT'
D -> I
D -> λ
I -> idG
G -> &=E
G -> =E

E -> RE'
E -> (E)
E' -> && E
E' -> λ
E' -> ==E
E' -> <R
E' -> +E
R -> id
R -> numero
R -> cadena
R -> true
R -> false
O -> id=E
N -> prompt(id);
N -> print(Z);
N -> return K
N -> O
Z -> 'cadena'Z'
Z -> numeroZ'
Z -> idZ'
Z' -> +Z
Z' -> λ
K -> E
K -> λ

5.2 Demostración Condición LL de la gramática

Ya que utilizamos un analizador sintáctico descendente recursivo predictivo, debemos demostrar que nuestra gramática anterior es de tipo LL(1).

Para ello hemos utilizado el software SDGLL1. Dentro del programa tenemos que definir el conjunto de símbolos terminales, el conjunto de símbolos no terminales, el axioma y la lista de producciones. Nos ha salido un análisis satisfactorio. Luego nuestra gramática es válida.

5.3 Procedimientos

```
PROCEDURE W
  BEGIN
    IF siguiente_token='function' THEN
      F;
      W;
   ELSE IF siguiente_token='eof' THEN
      valida_token();
   ELSE IF siguiente_token ϵ {var, for, int, print, prompt, return, id} THEN
      S;
      W;
   END


PROCEDURE S
 BEGIN
    IF siguiente_token='var' THEN
      valida_palres('var');
      T;
      valida_token('id');
      valida_token('puntocoma');
    ELSE IF siguiente_token='for' THEN
      valida_palres('for');
      valida_token('parab');
      D;
      valida_token('puntocoma');
      E;
      valida_token('puntocoma');
      O;
      valida_token('parcerr');
    ELSE IF siguiente_token ϵ {print, prompt, return, id}
THEN
      N;
      S;
 END
```

```
PROCEDURE F
  BEGIN
    IF siguiente_token='function' THEN
      valida_palres('function');
      F';
      valida_token('id');
      valida_token('parab');
      P;
      valida_token('parcerr');
      valida_token('llaveabr');
      S;
      valida_token('llavecerr');
   END
END

PROCEDURE F'
  BEGIN
    IF siguiente_token ϵ {int, bool, string} THEN
       T;
    ELSE
    END
   END


PROCEDURE T
  BEGIN
    IF siguiente_token='int' THEN
       valida_palres('int');
    ELSE IF siguiente_token='bool' THEN
       valida_palres('bool');
    ELSE IF siguiente_token='string' THEN
       valida_palres('string');
  END
END


PROCEDURE P
  BEGIN
    IF siguiente_token ϵ {int, bool, string} THEN
       T;
       valida_token('id');
       T';
    ELSE
    END
  END
```

```
PROCEDURE T'
  BEGIN
    IF siguiente_token=',' THEN
        valida_token('coma');
        T;
        valida_token('id');
        T';
    ELSE
    END
  END
END


PROCEDURE D
  BEGIN
    IF siguiente_token='id' THEN
        I;
    ELSE
    END
  END
END


PROCEDURE I
  BEGIN
    IF siguiente_token='id' THEN
        valida_token(id);
        G;
  END
END



PROCEDURE G
  BEGIN
    IF siguiente_token='='THEN
        valida_token(igual);
        E;
   ELSE IF siguiente_token='&=' THEN
        valida_token(especial);
        E;
  END
END
```

```
PROCEDURE E
  BEGIN
    IF siguiente_token ϵ {id, numero, cadena, true, false} THEN
          R;
          E';
    ELSE IF siguiente_token='(' THEN
          valida_token(parab);
          E;
          valida_token(parcerr);
  END
END


PROCEDURE E'
  BEGIN
    IF siguiente_token='&&' THEN
          valida_token(and);
          E;
    ELSE IF siguiente_token='==' THEN
          valida_token(dosigual);
          E;
    ELSE IF siguiente_token='+' THEN
          valida_token(mas);
          E;
    ELSE IF siguiente_token='<' THEN
          valida_token(menor);
          R;
    ELSE
    END
  END
END


PROCEDURE R
  BEGIN
    IF siguiente_token='id' THEN
          valida_token(id);
    ELSE IF siguiente_token='numero' THEN
          valida_token(ent);
    ELSE IF siguiente_token='cadena' THEN
          valida_token(cadena);
    ELSE IF siguiente_token='true' THEN
          valida_palres(true);
    ELSE IF siguiente_token='false' THEN
          valida_palres(false);
  END
END
```

```
PROCEDURE O
  BEGIN
    IF siguiente_token='id' THEN
          valida_token(id);
          valida_token(igual);
          E;
  END
END


PROCEDURE N
  BEGIN
    IF siguiente_token='id' THEN
          O;
    ELSE IF siguiente_token='prompt' THEN
          valida_palres(prompt);
          valida_token(parab);
          valida_token(id);
          valida_token(parcerr);
          valida_token(puntocoma);
    ELSE IF siguiente_token='print' THEN
          valida_palres(print);
          valida_token(parab);
          Z;
          valida_token(parcerr);
          valida_token(puntocoma);
    ELSE IF siguiente_token='return' THEN
          valida_palres(return);
          K;
  END
END


PROCEDURE K
  BEGIN
    IF siguiente_token ϵ {id, numero, cadena, true, false} THEN
          E;
    ELSE
    END
  END
END
```

```
PROCEDURE Z
  BEGIN
    IF siguiente_token=‘cadena’ THEN
          valida_token(cadena);
          Z’;
   ELSE IF siguiente_token=‘numero’ THEN
          valida_token(ent);
          Z’;
   ELSE IF siguiente_token=‘id’ THEN
          valida_token(id);
          Z’;
  END
END


PROCEDURE Z’
  BEGIN
    IF siguiente_token=‘+’ THEN
          valida_token(mas);
          Z;
   ELSE
   END
  END
END
```

Pruebas:

```
CODIGO:
 var int a;
a=3


TOKENS:
 <PALRES,8>
<PALRES,1>
<ID,1>
<PUNTOCOMA,->
<ID,1>
<IGUAL,->
<ENT,3>
<eof,->

PARSE      1   2   5   12   7   40   36   24   32   27   8   4

CODIGO:     function suma (int a,int b) {return 0}


TOKENS:
 <PALRES,11>
<ID,1>
<PARAB,->
<PALRES,1>
<ID,2>
<COMA,->
<PALRES,1>
<ID,3>
<PARCERR,->
<LLAVEAB,->
<PALRES,9>
<ENT,0>
<LLAVECERR,->
<eof,->

PARSE      1   3   9   10   16   12   18   12   17   7   39   46   24   32   27   8   4   4


CODIGO:     var a =3


TOKENS:
 <PALRES,8>
<ID,1>
<IGUAL,->
<ENT,3>
<eof,->
Exception in thread "main" java.lang.Exception: ERROR SINTÁCTICO: No se ha aceptado ningún token'.

CODIGO:     function int () {}


TOKENS:
 <PALRES,11>
<PALRES,1>
<PARAB,->
<PARCERR,->
<LLAVEAB,->
<LLAVECERR,->
<eof,->
Exception in thread "main" java.lang.Exception: ERROR SINTÁCTICO: No se ha aceptado ningún token'.
```