Practica 4 Compiladores: Defínción de un Analizador Sintáctico en BYACC/J

Angel Sandoval Mendoza

Octubre 2024

1 Conjuntos N, Σ y Símbolo Inicial S

- **N (Conjunto de No terminales):**
  N = {S, Expr, Expr', Term, Term', Factor, Num, Decimal, Entero, Digito, Asig, Var, Pos, Letra }
- Σ **(Conjunto de Terminales):**

$$\Sigma = \{+, -, *, /, =, (,), ., , 0, 1, \ldots, 9, a, \ldots, z, A, \ldots, Z, , var\}$$

- Simbolo Inicial: S

2 Proceso de eliminación de ambigüedad

Por como esta definida la grámatica no es necesario el hacer la eliminación de ambiguedad, esto debido a que en donde se podria tener problemas de ambiguedad sería en la parte de terminos, pero por como estan definidas a traves de su presedencia, no es necesario.

3 Proceso de eliminación de Recursividad Izquierda

La grámatica parece ya haber llevado un proceso para la eliminación de la recursividad izquierda, se puede notar en las produciones Expr' y Term', ya que tienen la forma que hemos visto anteriormente para lidiar con la recusividad izquierda.

4 Proceso de factorización izquierda

a gramática está estructurada de tal manera que no se requiere factorización izquierda, ya que no tenemos producciones del tipo $A \rightarrow \alpha\beta|\alpha\gamma$, asi que no necesitaría modificación alguna.

5 Nuevos Conjuntos N y P.

Como no se realizo cambio alguno en la grámatica, serían los conjuntos originales:

- **N (Conjunto de No terminales):**

  N = {S, Expr, Expr', Term, Term', Factor, Num, Decimal, Entero, Digito, Asig, Var, Pos, Letra }

```
P = {
S → Expr | Asig
Expr → Term Expr'
Expr' → + Term Expr' | - Term Expr' | epsilon
Term → Factor Term'
Term' → * Factor Term' | / Factor Term'| epsilon
Factor → Num | Var | (Expr) | - Expr
Num → Entero Decimal
Decimal → . Entero | epsilon
Entero → Digito | Digito Entero
Digito→0|1|2|... |9
Asig → var Var = Expr
Var → Letra Pos
Pos → Var | epsilon
Letra → _|a|b|... |z|A|B|... |Z
}
```

6 Tratamiento para evitar conflictos de shift/reduce

Lo que podríamos hacer sería definir solamente la precedencía de operaciones de la siguiente manera:

```
%left '+' '-'
%left '*' '/'
```

Esto evitara errores más adelante.

7 Definición con BYACC/J

```
%%

// Declaración de los terminales
%token PLUS MINUS TIMES DIVIDE ASSIGN LPAREN RPAREN
%token INTEGER DECIMAL VAR

// Declaración de los no terminales
```

```
%type <int> Expr Term Factor Num Var

// Precedencia de los operadores
%left PLUS MINUS
%left TIMES DIVIDE

%%

// Definición de la gramática
S: Expr
  | Asig;

Expr: Term Expr'
    ;

Expr': PLUS Term Expr'
      | MINUS Term Expr'
      | /* epsilon */
      ;

Term: Factor Term'
    ;

Term': TIMES Factor Term'
      | DIVIDE Factor Term'
      | /* epsilon */
      ;

Factor: Num
      | Var
      | LPAREN Expr RPAREN
      | MINUS Expr
      ;

Num: INTEGER Decimal
   ;

Decimal: '.' INTEGER
        | /* epsilon */
        ;

Integer: Digito
        | Digito Integer
        ;

Digito: '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
```

```
      ;

Asig:  VAR Var ASSIGN Expr
    ;

Var: Letra Pos
   ;

Pos: Var
   | /* epsilon */
   ;

Letra: '_'
      | 'a'..'z'
      | 'A'..'Z'
      ;
```