Universidad Nacional Autónoma de México
Facultad de Ciencias
Compiladores 2025-1

Huriel Osorio Escandón — Huriel117@ciencias.unam.mx — 317204652

2024

En G1/ tenemos una definición con BYACC/J de la gramatica inicial de la p 4:

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

En el archivo .y tenemos Expr' y Term' como expr1 y term1 respectivamente. En G2/ tenemos una def. con BYACC/J de la anterior gramatica, despues de haber eliminado la ambigüedad:

P = {
s -¿ expr | asig
asig -¿ VAR var = expr
expr -¿ term expr1
expr1 -¿ + term expr 1 | - term expr1 |ε
term -¿ factor term1
term1 -¿ * factor term1 | / factor term1 |ε
factor -¿ num | var | (expr) | - factor
num -¿ entero decimal
decimal -¿ . entero |ε
entero -¿ DIGITO | DIGITO entero
var -¿ LETRA pos
pos -¿ var |ε
}

En la nueva gramatica no es necesario eliminar la recursividad izquierda, pues en este caso la recursion es derecha, por ejemplo:

expr1 -¿ + term expr1, como BYACC/J esta basado en analisis ascendente (LR), puede manjera esto sin problema.

Tampoco es necesaria la factorizacion izquierda pues la estructura de expr1 y term1 esta diseñada para funcionar sin ella.

S = s
N = {
s
asig
expr
expr1
term
term1
factor
num
decimal
entero
var
pos
}
σ = {
=
+
-
*
/
(
)
.
VAR
LETRA
DIGITO
}