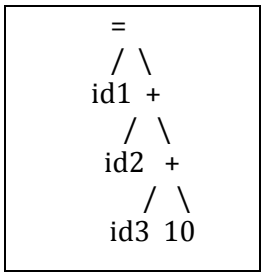
Análisis Sintáctico

Diego Armando – Cale Pilco

1. Análisis Sintáctico

El analizador sintáctico impone una estructura jerárquica a la cadena de componentes léxicos, generada por el analizador léxico, que es representada en forma de un árbol sintáctico.



2. GRAMÁTICAS

- **Tipo 0 o Sin Restricciones o Estructuradas por Fase (MT: Maquinas de Turing)**
  G = (N, T, P, S)
  N: Conjunto de Símbolos No Terminales
  T: Conjunto de Símbolos Terminales
  P: Conjunto de Reglas de Producción
  $S \in N$: Símbolo Inicial

- **Tipo 1 o Sensibles al Contexto (CSG) (ALA: Autómata Linealmente Acotado)**
  G = (N, T, P, S)
  N: Conjunto de Símbolos No Terminales
  T: Conjunto de Símbolos Terminales
  P: Conjunto de Reglas de Producción
  $S \in N$: Símbolo Inicial
  $P \subseteq (T \cup Vn)^* Vn (T \cup Vn)^* \times (T \cup Vn)^*$
  α β con $|\alpha| \leq |\beta|$

- **Tipo 2 o Libres de Contexto (CFG) (AP: Autómatas de Pila)**
  G = (N, T, P, S)
  N: Conjunto de Símbolos No Terminales
  T: Conjunto de Símbolos Terminales
  P: Conjunto de Reglas de Producción

$S \in N$: Símbolo Inicial
$P \subseteq N \times (T \cup Vn)^*$

Ejemplo:

Supóngase que utilizamos E en lugar de <expresion> para la variable de la gramática. Podemos expresar esta gramática de la manera formal como:

G = (N,T,P,S) Donde: T = { + , * , (,) , id }
N = { E }
P = { E → E + E
E → E * E
E → (E)
E → id }
S = E

- **Tipo 3 o Regulares (AF: Autómatas Finitos)**

Definen la sintaxis de los identificadores, números, cadenas y otros símbolos básicos del lenguaje.

G = (N, T, P, S)
N: Conjunto de Símbolos No Terminales
T: Conjunto de Símbolos Terminales
P: Conjunto de Reglas de Producción
$S \in N$: Símbolo Inicial

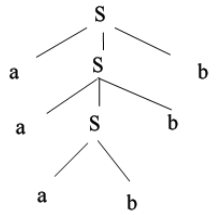
Regular a Derecha: $P \subseteq N \times (TN \cup T \cup \{\lambda\})$
$A \rightarrow a \mid aB$ (lineal por la derecha)

Regular a Izquierda: $P \subseteq N \times (NT \cup T \cup \{\lambda\})$
$A \rightarrow a \mid Ba$ (lineal por la izquierda)
Donde: $A, B \in N$, $a \in T^*$

Las gramáticas regulares guardan estrecha relación con los autómatas finitos.
Las gramáticas estudiadas en la teoría de lenguajes son la 2 y 3.

3. Ejemplo:

Sea G = {N, T, P, S} una GLC cobn P: S→ ab | aSb.
La derivación de la cadena aaabbb sera $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaabbb$ y el arbol de derivación:



i. GRAMÁTICAS NO AMBIGUAS

Sea G = (T , N , P , S) que acepta expresiones aritméticas como:

X + Y – X * Y

T = { X , Y , + , - , * , / , (,) }

N = { EXPR , TERM , FACTOR }

P = { EXPR → TERM | EXPR + TERM | EXPR – TERM
TERM → FACTOR | TERM * FACTOR | TERM / FACTOR
FACTOR → X | Y | (EXPR)

S = {EXPR}

G no es ambigua, porque tiene un solo árbol de derivación

- **Derivación por la izquierda**

Se realiza el reemplazo de cada N que esta más a la izquierda

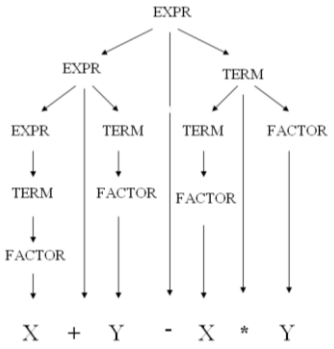
EXPR → EXPR – TERM
→ EXPR + TERM – TERM
→ TERM + TERM – TERM
→ FACTOR + TERM – TERM
→ X + TERM – TERM
→ X + FACTOR – TERM
→ X + Y – TERM
→ X + Y – TERM * FACTOR
→ X + Y – FACTOR * FACTOR
→ X + Y – X * FACTOR
→ X + Y – X * Y

- **Derivación por la derecha**

Se realiza el reemplazo de cada N que esta más a la derecha

EXPR → EXPR – TERM
→ EXPR - TERM * FACTOR
→ EXPR - TERM * Y
→ EXPR - FACTOR * Y
→ EXPR - X * Y
→ EXPR + TERM - X * Y
→ EXPR + FACTOR - X * Y
→ EXPR + Y - X * Y
→ TERM + Y - X * Y
→ FACTOR + Y - X * Y
→ X + Y – X * Y

Arbol de derivacion:



ii. GRAMÁTICAS AMBIGUAS

Sea G = (T , N , P , S) que acepta expresiones aritméticas como:

X + Y – X * Y

T = { X , Y , + , - , * , / , (,) }
N = { EXPR , OP }
P = { EXPR → EXPR OP EXPR | (EXPR) | X | Y
OP → + | - | * | /
S = {EXPR}

G es ambigua, porque tiene más de un árbol de derivación

- **Derivación por la izquierda**

Se realiza el reemplazo de cada N que esta más a la izquierda

EXPR → (EXPR)
→ (EXPR OP EXPR)
→ (X OP EXPR)
→ (X + EXPR)
→ (X + (EXPR))
→ (X + (EXPR OP EXPR))
→ (X + (Y OP EXPR))
→ (X + (Y - EXPR))
→ (X + (Y – (EXPR OP EXPR)))
→ (X + (Y – (X OP EXPR)))
→ (X + (Y – (X * EXPR)))
→ (X + (Y – (X * Y)))

- **Derivación por la derecha**

Se realiza el reemplazo de cada N que esta más a la derecha

EXPR → EXPR OP EXPR
→ EXPR OP Y
→ EXPR * Y
→ (EXPR) * Y
→ ((EXPR)) * Y
→ ((EXPR OP EXPR)) * Y
→ ((EXPR OP (EXPR))) * Y
→ ((EXPR OP (EXPR OP EXPR))) * Y
→ ((EXPR OP (EXPR OP X))) * Y
→ ((EXPR OP (EXPR - X))) * Y
→ ((EXPR OP (Y - X))) * Y
→ ((EXPR + (Y - X))) * Y

→((X+(Y - X))*Y

4. Bibliografía

1. Viñuela, P. I., Viñuela, P. I., MillÔan, D. B., FernÔandez, P. M., Martínez, P., Borrajo, D., ... & Millán, D. B. (1997). Lenguajes, gramáticas y autómatas: un enfoque práctico. Pearson Educación.
2. Aho, A. V., Sethi, R., & Ullman, J. D. (1998). Compiladores: principios, técnicas y herramientas. Pearson Educación.
3. Gutiérrez, J. J., Escalona, M. J., Villadiego, D., & Mejías, M. (2005). Comparativa de herramientas para la enseñanza de lenguajes relacionales. Actas de las XI Jornadas de Enseñanza Universitaria de la Informática.
4. Sanchis Llorca, F. J., & Pascual, C. G. (1986). Compiladores: teoría y construcción. Paraninfo.