6. ANÁLISIS SINTÁCTICO

El analizador sintáctico obtiene una secuencia de componentes léxicos del analizador léxico y comprueba si la secuencia puede ser generada por la gramática del lenguaje fuente.

Los analizadores sintácticos usados en los compiladores se clasifican en descendentes o ascendentes. Como sus nombres indican, los analizadores sintácticos descendentes construyen árboles de análisis sintáctico desde la raíz hacia las hojas, mientras que los analizadores sintácticos ascendentes, desde las hojas hacia la raíz. En ambos casos, se lee la entrada al analizador sintáctico de izquierda a derecha, un símbolo a la vez.

6.1. Derivaciones por la izquierda

Una derivación por la izquierda es una derivación en la que en cada paso se reemplaza el no terminal de más a la izquierda.

Ejemplo:
$G = (\{E\}, \{+, *, (,), -, i\}, \{E \rightarrow E + E \mid E * E \mid (E) \mid - E \mid i\}, E)$

$E \Rightarrow - E \Rightarrow - (E) \Rightarrow - (E + E) \Rightarrow - (i + E) \Rightarrow - (i + i)$

6.2. Derivaciones por la derecha

Una derivación por la derecha es una derivación en la que en cada paso se reemplaza el no terminal de más a la derecha.

Ejemplo:
$G = (\{E\}, \{+, *, (,), -, i\}, \{E \rightarrow E + E \mid E * E \mid (E) \mid - E \mid i\}, E)$

$E \Rightarrow - E \Rightarrow - (E) \Rightarrow - (E + E) \Rightarrow - (E + i) \Rightarrow - (i + i)$

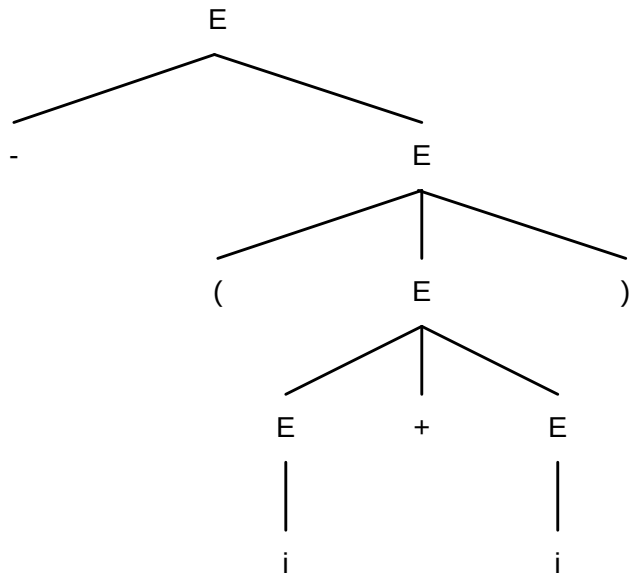
6.3. Árboles de análisis sintáctico

Un árbol de análisis sintáctico es una representación gráfica de una derivación.

Un árbol de análisis sintáctico es un árbol con las siguientes propiedades (Aho, 1990, p. 29):

1. La raíz está etiquetada con el símbolo inicial.
2. Cada hoja está etiquetada con un terminal o con ε.
3. Cada nodo interior está etiquetado con un no terminal.
4. Si A es el no terminal que etiqueta a algún nodo interior y X_1, X_2, X_3, ... , X_n son las etiquetas de los hijos de ese nodo, de izquierda a derecha, entonces $A \rightarrow X_1 X_2 X_3 ... X_n$ es una producción.

Ejemplo:
G = ({E}, {+, *, (,), -, i}, {E → E + E | E * E | (E) | – E | i}, E)



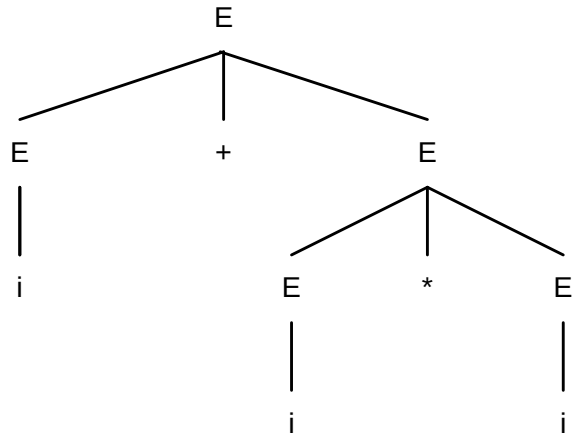
6.4. Gramáticas ambiguas

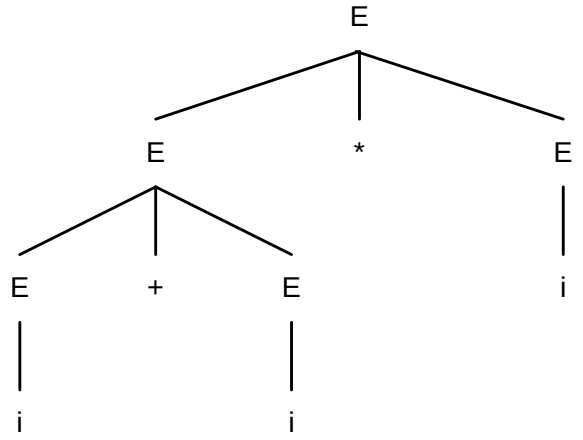
Se dice que una gramática que produce más de un árbol de análisis sintáctico para alguna palabra es ambigua. O, dicho de otro modo, una gramática ambigua es la que produce más de una derivación por la izquierda o más de una derivación por la derecha para la misma palabra.

Ejemplo:
G = ({E}, {+, *, (,), -, i}, {E $\rightarrow$ E + E | E * E | (E) | – E | i}, E)

Derivaciones por la izquierda
E $\Rightarrow$ E + E $\Rightarrow$ i + E $\Rightarrow$ i + E * E $\Rightarrow$ i + i * E $\Rightarrow$ i + i * i
E $\Rightarrow$ E * E $\Rightarrow$ E + E * E $\Rightarrow$ i + E * E $\Rightarrow$ i + i * E $\Rightarrow$ i + i * i

Árboles de análisis sintáctico





6.5. Recursividad por la izquierda

G = (N, Σ, P, S)

$$A \overset{+}{\Rightarrow} A\alpha \qquad A \in N, \ \alpha \in (N \cup \Sigma)^*$$

6.5.1. Eliminación de la recursividad por la izquierda

G = (N, Σ, P, S) ⇒ G' = (N', Σ, P', S)

$$A \to A\alpha \mid \beta \in P \Rightarrow \left.\begin{array}{l} A \to \beta A' \\ A' \to \alpha A' \mid \varepsilon \end{array}\right\} \in P'$$

$$A \to A\alpha_1 \mid A\alpha_2 \mid A\alpha_3 \mid ... \mid A\alpha_m \mid \beta_1 \mid \beta_2 \mid \beta_3 \mid ... \mid \beta_n \in P \Rightarrow \left.\begin{array}{l} A \to \beta_1 A' \mid \beta_2 A' \mid \beta_3 A' \mid ... \mid \beta_n A' \\ A' \to \alpha_1 A' \mid \alpha_2 A' \mid \alpha_3 A' \mid ... \mid \alpha_m A' \mid \varepsilon \end{array}\right\} \in P'$$

N' = N ∪ {A'}

Ejemplo:
G = ({E, T, F}, {+, *, (,), i}, P, E)
P = {
E → E + T | T
T → T * F | F
F → (E) | i
}

6.6. Factorización por la izquierda

$G = (N, \Sigma, P, S) \Rightarrow G' = (N', \Sigma, P', S)$

$$A \to \alpha\beta_1 \mid \alpha\beta_2 \mid \alpha\beta_3 \mid ... \mid \alpha\beta_m \mid \gamma_1 \mid \gamma_2 \mid \gamma_3 \mid ... \mid \gamma_n \in P \Rightarrow \left.\begin{array}{l} A \to \alpha A' \mid \gamma_1 \mid \gamma_2 \mid \gamma_3 \mid ... \mid \gamma_n \\ A' \to \beta_1 \mid \beta_2 \mid \beta_3 \mid ... \mid \beta_m \end{array}\right\} \in P'$$

$N' = N \cup \{A'\}$

Ejemplo:
G' = ({P, E}, {i, t, e, a, b}, P', P)
P' = {
P → iEtP | iEtPeP | a
E → b
}

6.7. Primero

G = (N, Σ, P, S)

$$P(\alpha) = \{\sigma \in \Sigma / \alpha \overset{*}{\Rightarrow} \sigma\beta \quad \alpha, \beta \in (N \cup \Sigma)^*\}$$

- $P(\varepsilon) = \varnothing$
- $P(\sigma\alpha) = \{\sigma\}$ $\quad \sigma \in \Sigma$, $\alpha \in (N \cup \Sigma)^*$
- $A \to \alpha_1 \mid \alpha_2 \mid \alpha_3 \mid ... \mid \alpha_n \in P \Rightarrow P(A) = P(\alpha_1) \cup P(\alpha_2) \cup P(\alpha_3) \cup ... \cup P(\alpha_n)$ $\quad A \in N$
- $\alpha = X_1 X_2 X_3 ... X_n$
  Algoritmo:
  $P(\alpha) = P(X_1)$
  i = 1
  Mientras $X_i \in N_\varepsilon$
  {
  $P(\alpha) = P(\alpha) \cup P(X_{i+1})$
  i = i + 1
  }

Ejemplo:
G = ({E, E', T, T', F}, {+, *, (,), i}, P, E)
P = {
E → TE'
E' → + TE' | ε
T → FT'
T' → * FT' | ε
F → (E) | i
}

6.8. Siguiente

G = (N, Σ, P, S)

$$S(A) = \{\sigma \in \Sigma / S \overset{*}{\Rightarrow} \alpha A \sigma \beta \quad A \in N \quad \alpha,\beta \in (N \cup \Sigma)^*\}$$

- $S(S) = \{\$\}$
- $A \to \alpha B \beta \Rightarrow S(B) = P(\beta) \quad A, B \in N \quad \alpha, \beta \in (N \cup \Sigma)^*$
- $A \to \alpha B \Rightarrow S(B) = S(A) \quad A, B \in N \quad \alpha \in (N \cup \Sigma)^*$
- $A \to \alpha B \beta \quad \beta \overset{*}{\Rightarrow} \varepsilon \Rightarrow S(B) = P(\beta) \cup S(A) \quad A, B \in N \quad \alpha, \beta \in (N \cup \Sigma)^*$

Ejemplo:
G = ({E, E', T, T', F}, {+, *, (,), i}, P, E)
P = {
E → TE'
E' → + TE' | ε
T → FT'
T' → * FT' | ε
F → (E) | i
}

6.9. Analizador sintáctico LL(1)

La primera L representa la lectura de la entrada de izquierda a derecha, la segunda L representa una derivación por la izquierda, y el 1 es por utilizar un símbolo de entrada de anticipación en cada paso para tomar las decisiones de la acción en el análisis sintáctico.

6.9.1. Gramáticas LL(1)

"Ninguna gramática ambigua o recursiva por la izquierda puede ser LL(1)" (Aho, 1990, p. 197).

$G = (N, \Sigma, P, S)$

1. $A \rightarrow \alpha_1 \mid \alpha_2 \mid \alpha_3 \mid ... \mid \alpha_n \in P \Rightarrow P(\alpha_i) \cap P(\alpha_j) = \varnothing \qquad \forall A \in N , i \neq j$
2. $P(A) \cap S(A) = \varnothing \qquad \forall A \in N_\varepsilon$

Ejemplo:
G = ({E, E', T, T', F}, {+, *, (,), i}, P, E)
P = {
E → TE'
E' → + TE' | ε
T → FT'
T' → * FT' | ε
F → (E) | i
}

Ejercicio:
G'' = ({P, P', E}, {i, t, e, a, b}, P'', P)
P'' = {
P → iEtPP' | a
P' → ε | eP
E → b
}

6.9.2. Tabla del analizador sintáctico LL(1)

$G = (N, \Sigma, P, S)$

Algoritmo:
$\forall\ A \rightarrow \alpha \in P$
{
  $\forall\ \sigma \in P(\alpha)$
    $M[A, \sigma] = A \rightarrow \alpha$
  Si $A \in N_\varepsilon$ entonces
    $\forall\ \sigma \in S(A)$
      $M[A, \sigma] = A \rightarrow \varepsilon$
}

Ejemplo:
G = ({E, E', T, T', F}, {+, *, (,), i}, P, E)
P = {
  E → TE'
  E' → + TE' | ε
  T → FT'
  T' → * FT' | ε
  F → (E) | i
}

ω = i + i * i

Ejercicio:
G'' = ({P, P', E}, {i, t, e, a, b}, P'', P)
P'' = {
  P → iEtPP' | a
  P' → ε | eP
  E → b
}

6.9.3. Recuperación de errores en el analizador sintáctico LL(1)

$G = (N, \Sigma, P, S)$

$M[A, \sigma] = sinc \qquad \forall A \in N , A \notin N_\varepsilon , \sigma \in S(A)$

Ejemplo:
G = ({E, E', T, T', F}, {+, *, (,), i}, P, E)
P = {
E → TE'
E' → + TE' | ε
T → FT'
T' → * FT' | ε
F → (E) | i
}

$M[A, \sigma]$ = " " ⇒ saltar σ $\qquad A \in N , \sigma \in \sum$
$M[A, \sigma] = sinc \Rightarrow$ sacar A $\qquad A \in N , \sigma \in \sum$

Si un componente léxico del tope de la pila no coincide con el símbolo de entrada, entonces se saca el componente léxico de la pila.

Ejemplo:
ω = + i * + i