# Tema 3. Análisis sintáctico

1. Introducción.

2. Especificación sintáctica

   2.1. Generadores: Gramáticas independientes del contexto

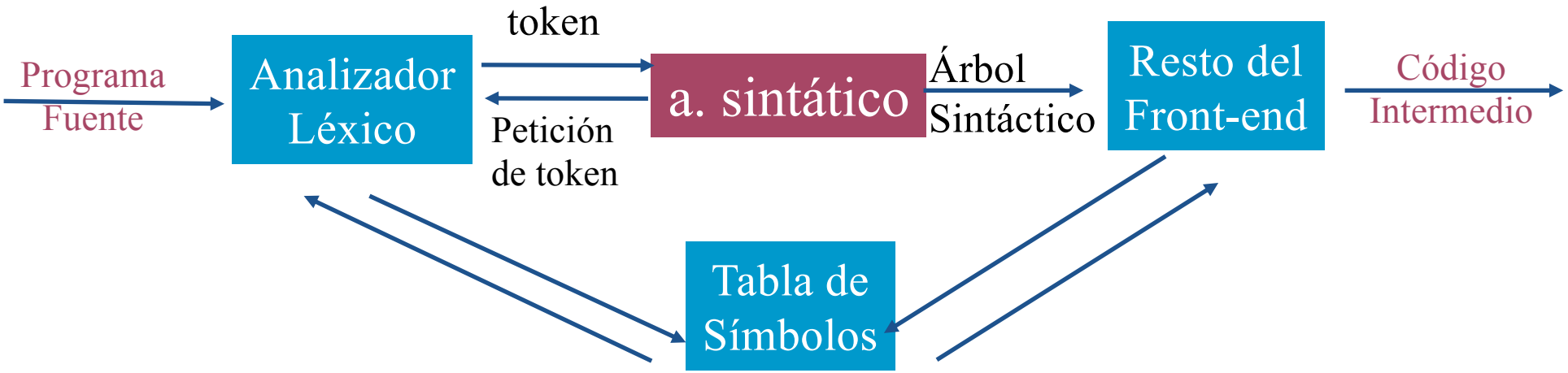
   2.2. Reconocedores: Autómatas de pila

3. Métodos de análisis sintáctico

# 1. Introducción

# Funciones del a. sintáctico

- Detecta e informa de errores sintácticos.
- Construye un árbol de análisis sintáctico.



- Se necesita una especificación formal del lenguaje
- La sintaxis de un lenguaje de programación puede describirse por una gramática independiente del contexto:
  - Un compilador construido a partir de una gramática es más sencillo de mantener y modificar.
  - Se pueden emplear herramientas de generación automática de a. sintácticos.

# 2. Especificación sintáctica

# Generadores: gramáticas

***Gramática*** *G = (N, Σ, P, S)*:

- N: Conjunto finito de símbolos no-terminales
- Σ: Conjunto finito de símbolos terminales
- P: Conjunto de producciones
- S: Símbolo inicial $S \in N$ .

**Jerarquía de Chomsky**

- **Gramática regular:** Si y solo si es lineal a derechas (a izquierdas):

  Cada producción de P es de la forma , $A \rightarrow wB$ o $A \rightarrow w$.

  donde $A, B \in N$ y $w \in \Sigma$
- **Gramática independiente del contexto:** Cada producción en P es de la forma $A \rightarrow \alpha$ , donde $A \in N$ y $\alpha \in (N \cup \Sigma)^*$
- **Gramática sensible al contexto:** Cada producción en P es de la forma $\alpha \rightarrow \beta$ donde $|\alpha| \leq |\beta|$
- **Gramática inrestringida:** Cada producción en P es de la forma $\alpha \rightarrow \beta$ donde $\alpha \neq \varepsilon$

# Lenguaje generado por una gramática

Derivación directa ($\Rightarrow$):

Proceso de reescritura: Cómo se deriva una cadena a partir de otra

$$\delta\, A\, \gamma \Rightarrow \delta\, \beta\, \gamma \quad \text{si y solo si (ssi)}$$
$$\exists\, (A \rightarrow \beta) \in P \;;\; \delta, \gamma \in (N \cup \Sigma)^*$$

Derivación ($\Rightarrow^*$): $\alpha \Rightarrow^* \beta$ ssi $\exists\ \alpha_0, \ldots, \alpha_m \in (N \cup \Sigma)^*$

$$\alpha = \alpha_0 \Rightarrow \alpha_1 \Rightarrow \ldots \alpha_{m-1} \Rightarrow \alpha_m = \beta$$

Forma sentencial: $\alpha \in (N \cup \Sigma)^*$ ssi $\exists\ S \Rightarrow^* \alpha$

Sentencia: $\alpha \in \Sigma^*$ sii $\exists\ S \Rightarrow^* \alpha$

Lenguaje generado por G:

$$L(G) = \{x \,/\, x \in \Sigma^* \,:\, S \Rightarrow^* x\}$$

# Árbol de a. sintáctico

**Derivación más a izquierdas ( derechas ):**

- En cada paso de derivación se reescribe el no-terminal más a la izquierda ( derecha ).

**Árbol de análisis sintáctico:**

- Representación gráfica que muestra como se deriva una cadena del lenguaje a partir del símbolo inicial de la gramática.

Dada una GIC ($G = (N, \Sigma, P, S)$): un **árbol de análisis sintáctico** cumple:

- La **raíz** está etiquetada con el símbolo inicial de G
- Cada **hoja** está etiquetada con un símbolo terminal (token), o con la cadena vacía $(\Sigma \cup \{\varepsilon\})$
- Cada **nodo interior** está etiquetado por un no-terminal (N)
- Si A es un no-terminal etiquetando un nodo interior, y $x_1,...,x_n$ son las etiquetas de sus nodos hijo (tomados de izquierda a derecha),

  entonces $A \rightarrow x_1, ..., x_n \in P$

# Gramática ambigua

**Producto** de un árbol de análisis sintáctico:

- Hojas del árbol de análisis sintáctico tomadas de izquierda a derecha.
- Es equivalente a la cadena derivada a partir del no-terminal que aparece en la raíz del árbol.

Teorema:

- Dada una GIC ($G = (N, \Sigma, P, S)$), $S \Rightarrow^* \alpha$ ( $\alpha \in (\Sigma \cup \{\varepsilon\})^*$ ) ssi hay un árbol de análisis sintáctico produciendo $\alpha$

Gramática ambigua:

- Gramática para la que se pueden construir dos o más árboles de análisis sintáctico generando el mismo producto.

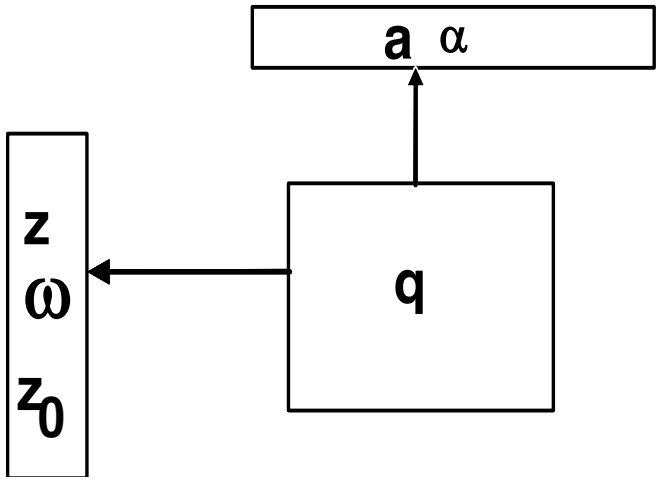
Proposición:

- El problema de determinar si una gramática es ambigua es indecidible.

# Reconocedores: Autómata a pila

$$A_p = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$$

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow P\,(Q \times \Gamma^*)$$




Configuración: (q , a α , z ω $z_0$ )

## Movimiento

$(q, a\,\alpha, z\,\omega\ \ z_0) \vdash (q', \alpha, \beta\,\omega\ \ z_0)$ **sii** $\exists\ (q', \beta) \in \delta(q, a, z)$

Con $q, q' \in Q$; $a \in \Sigma \cup \{\varepsilon\}$; $\alpha \in \Sigma^*$; $z \in \Gamma$; $\beta, \omega \in \Gamma^*$

## Lenguaje Aceptado a Pila Vacía

$L_v(A_p) = \{x / x \in \Sigma^* : (q_0, x, z_0) \vdash^* (q, \varepsilon, \varepsilon)\}$

## Lenguaje Aceptado a Estado Final

$L_F(A_p) = \{x / x \in \Sigma^* : (q_0, x, z_0) \vdash^* (q, \varepsilon, \alpha);\ q \in F\}$

# Ej. Autómata a Pila: Analizador descendente

*Gramática* → G = ({E, T, F}, {id, +, *, (, ) }, P, E)

P = { E→E + T | T ; T→T * F | F; F→(E) | id }

*Autómata* → $A_p$ = (Q, S, G, δ, q, E, ϕ)

Q = {q}  S = {id, +, *, (, ) }

G = {E, T, F} ∪ {id, +, *, (, )}

δ(q, ε, E) = { (q, E + T); (q, T) }

δ(q, ε, T) = { (q, T * F); (q, F) }

δ(q, ε, F) = { (q, '(' E ')' ); (q, id) }

δ(q, b, b) = { (q, ε) ; b ∈ S }

*Lenguaje Aceptado a Pila Vacía.*

# Ej. Autómata a Pila: Analizador descendente

***Lenguaje Aceptado a Pila Vacía.***

$Lv(A_p) = \{x \,/\, x \in \Sigma^*: \quad (q, x, E) \mid\!\!-^* (q, \varepsilon, \varepsilon)\}$

***Ej: Cadena id + id * id***

*Configuración inicial:* $(q, id + id * id, E)$

*Traza:*

$(q, id + id * id, E)$
$\vdash (q, id + id * id, T + T)$
$\vdash (q, id + id * id, id + T)$
$\vdash (q, \quad id * id, T)$

$\vdash (q, id + id * id, E + T)$
$\vdash (q, id + id * id, F + T)$
$\vdash (q, \quad + id * id, + T)$
$\vdash (q, \quad id * id, T * F) \ldots$

# 3. Métodos de análisis sintáctico

# Métodos de a. sintáctico

Universales:

- Algoritmo de backtracking general, Algoritmo de Early, y algoritmo de Cocke-Younger y Kasami.
- Válidos para cualquier GIC.
- Demasiado ineficientes para ser empleados en compiladores.

Análisis sintáctico descendente

- Construyen el árbol de análisis sintáctico desde la raíz hasta las hojas (usando una derivación más a izquierdas).
- Analizadores sintácticos descendentes recursivos, y LL.

Análisis sintáctico ascendente

- Construyen el árbol de análisis sintáctico desde las hojas hasta la raíz (usando las inversa de una derivación más a derechas).
- Análisis sintáctico por precedencia de operadores, y LR (SLR, canonical LR, LALR).