

# Tema 3. Análisis sintáctico

## 1. Introducción.

## 2. Especificación sintáctica

2.1. Generadores: Gramáticas independientes del contexto

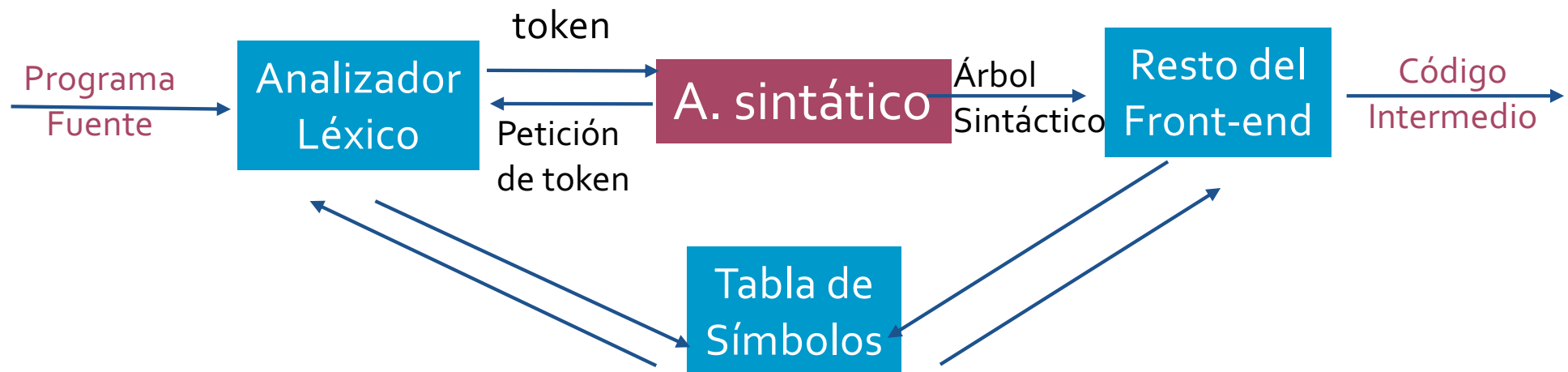
2.2. Reconocedores: Autómatas de pila

## 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**  $G = (N, \Sigma, P, S)$ :

- $N$ : Conjunto finito de símbolos **no-terminales** (o **auxiliares**).
- $\Sigma$ : 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  $A \rightarrow \omega B$  o  $A \rightarrow \omega$ .  
donde  $A, B \in N$  y  $\omega \in \Sigma$
- **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  $\alpha \neq \varepsilon$

$|\alpha|$  representa la longitud de la cadena  $\alpha$

# Lenguaje generado por una gramática

## Derivación directa ( $\Rightarrow$ ):

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

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

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

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

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

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  $x_1, \dots, x_n$  son las etiquetas de sus nodos hijo (tomados de izquierda a derecha),

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



# Ejemplo 1

Considera la GIC  $G=(\{E\}, \{+, *, (, ), id\}, \{E \rightarrow E+E, E \rightarrow E * E, E \rightarrow (E), E \rightarrow id\}, E)$ , que representaremos de forma simplificada como:

$$\begin{array}{l} E \rightarrow E+E \\ | \quad E * E \\ | \quad (E) \\ | \quad id \end{array}$$

Una posible derivación a derechas para obtener la cadena  $\omega = \text{"id+id*id"}$ , sería:

$E \rightarrow E+E \rightarrow E+E * E \rightarrow E+E * id \rightarrow E+id * id \rightarrow id+id * id$

a) Escribe una derivación a izquierdas para la misma cadena  $\omega$ . Escribe el árbol de análisis sintáctico correspondiente a la derivación a izquierdas que has realizado para la cadena  $w$

# Ejemplo 1

b) Construye un árbol de derivación “a derechas” y otro “a izquierdas” para la cadena  $\omega = \text{id+id*id}$ . ¿Qué puedes observar?

$E \rightarrow E + E$   
|  $E * E$   
|  $(E)$   
|  $\text{id}$

# 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  $(G = (N, \Sigma, P, S))$ ,  $S \Rightarrow^* \alpha$  ( $\alpha \in (\Sigma \cup \{\epsilon\})^*$ ) 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.

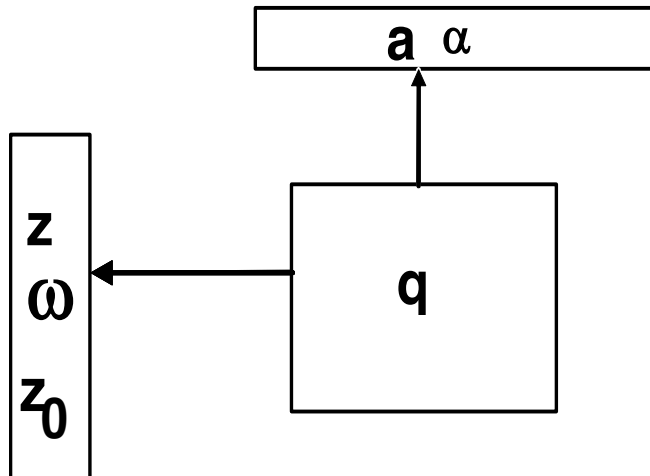
## Proposición:

- 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:  $(q, a \alpha, z \omega z_0)$

$Q$ : Conjunto de estados  
 $\Sigma$  : Alfabeto de entrada  
 $\Gamma$  : Alfabeto de pila  
 $\delta$  : Función de transición  
 $q_0$ : Estado inicial  
 $z_0$ : Símbolo inicial de pila  
 $F$ : Conjunto de estados finales

## Movimiento

$$(q, a, \alpha, z \omega z_0) \vdash (q', \alpha, \beta \omega z_0) \text{ si } \exists (q', \beta) \in \delta(q, a, z)$$

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

## Lenguaje Aceptado a Pila Vacía

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

## Lenguaje Aceptado a Estado Final

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

# Ej. Autómata a Pila: Analizador descendente

*Gramática*  $\rightarrow G = (\{E, T, F\}, \{id, +, *, (, )\}, P, E)$   
 $P = \{ E \rightarrow E + T \mid T ; T \rightarrow T * F \mid F ; F \rightarrow (E) \mid id \}$

*Autómata*  $\rightarrow A_p = (Q, S, G, \delta, q, E, \phi)$   
 $Q = \{q\} \quad S = \{id, +, *, (, )\}$   
 $G = \{E, T, F\} \cup \{id, +, *, (, )\}$

$\delta(q, \varepsilon, E) = \{ (q, E + T); (q, T) \}$

$\delta(q, \varepsilon, T) = \{ (q, T * F); (q, F) \}$

$\delta(q, \varepsilon, F) = \{ (q, '( ' E ') '); (q, id) \}$

$\delta(q, b, b) = \{ (q, \varepsilon); b \in S \}$

*Lenguaje Aceptado a Pila Vacía.*

# Ej. Autómata a Pila: Analizador descendente

*Lenguaje Aceptado a Pila Vacía.*

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

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

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

*Traza:*

$(q, id + id * id, E)$	$\mid^* (q, id + id * id, E + T)$
$\mid^* (q, id + id * id, T + T)$	$\mid^* (q, id + id * id, F + T)$
$\mid^* (q, id + id * id, id + T)$	$\mid^* (q, + id * id, + T)$
$\mid^* (q, id * id, T)$	$\mid^* (q, id * id, T * F) \dots$

### 3. Métodos de análisis sintáctico



# Métodos de a. sintáctico

## Universales:

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

## Métodos predictivos:

### 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 la inversa de una derivación más a derechas).
- Análisis sintáctico por precedencia de operadores, y LR (SLR, canonical LR, LALR).