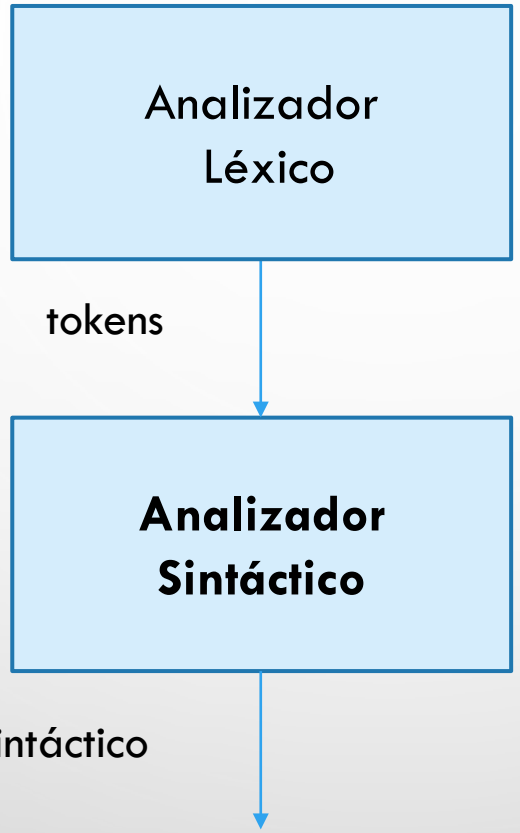
ANÁLISIS SINTÁCTICO



Comprueba si la secuencia de tokens que va recibiendo tiene una sintaxis correcta

¿Cómo?

Construyendo el árbol sintáctico, usando las reglas de la **Gramática de Contexto Libre**

```
IF x THEN a := 0
```

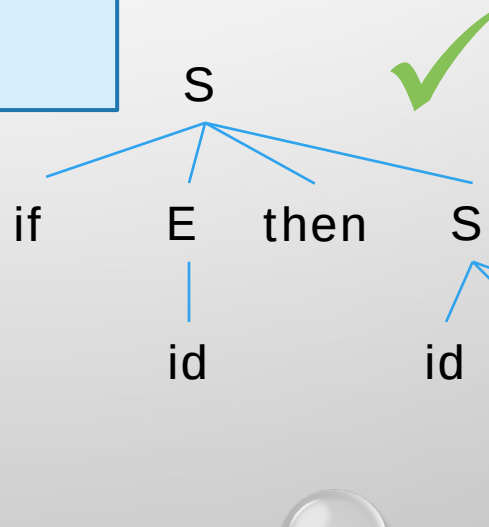
Analizador Léxico

tokens

<if, > <id, 12> <then, > <id, 3> <asig, > <ent, 0>

Analizador Sintáctico

Árbol sintáctico



GCL

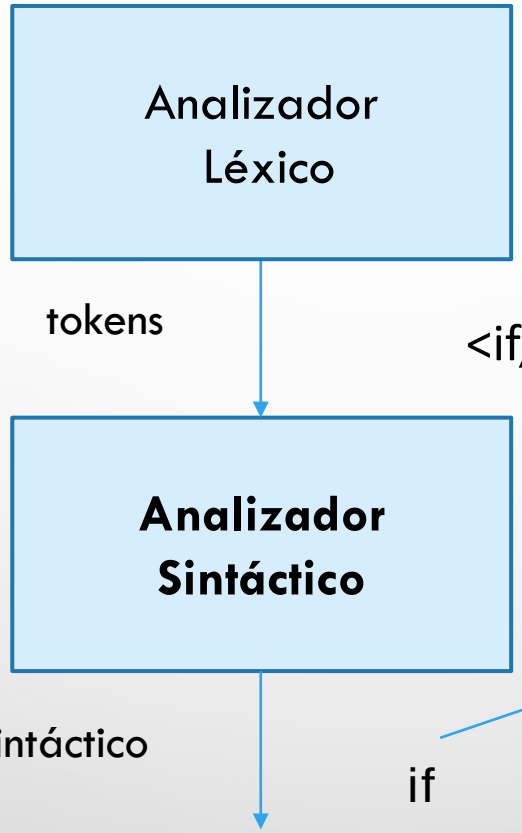
S → if E then S
S → if E then S else S
S → id := E
E → id
E → ent

Comprueba si la secuencia de tokens que va recibiendo tiene una sintaxis correcta

¿Cómo?

Construyendo el árbol sintáctico, usando las reglas de la **Gramática de Contexto Libre**

```
IF x ELSE a := 0
```

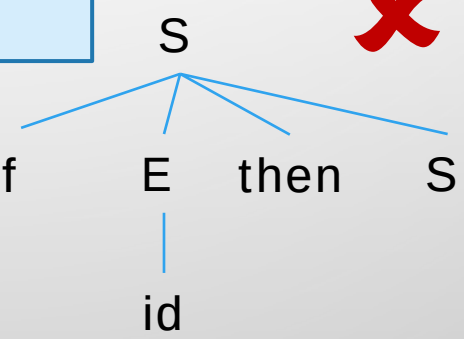


<if, > <id, 12> <else, > <id, 3> <asig, > <ent, 0>

Comprueba si la secuencia de tokens que va recibiendo tiene una sintaxis correcta

¿Cómo?

Construyendo el árbol sintáctico, usando las reglas de la **Gramática de Contexto Libre**



GCL

S → if E then S
S → if E then S else S
S → id := E
E → id
E → ent

Gramática formal

$G = (N, T, P, S)$

N: conjunto de símbolos no terminales
T: conjunto de símbolos terminales
P: conjunto de reglas de producción
S: axioma ($S \in N$)

Lenguaje Generado por una Gramática

$$L(G) = \{ \omega \mid \quad \omega \in T^* \quad S \Rightarrow^+ \omega \}$$

Formas Sentenciales

$$F(G) = \{ \sigma \mid \quad \sigma \in (N \cup T)^* \quad S \Rightarrow^+ \sigma \}$$

Gramática Tipo 2 o Gramática de Contexto Libre (GCL)

Sus reglas son de la forma: $A \rightarrow \alpha \quad A \in N \quad \alpha \in (N \cup T)^*$

Clasificación de gramáticas de Chomsky

$$GR \subseteq GCL \subseteq G\_Tipo1 \subseteq G\_Tipo0$$

Ejemplo de Gramática G

N = { D, T , L}

T = { integer, real, id }

P =

D → T L
T → integer
T → real
L → id , L
L → id

Axioma = D

Cadenas de entrada correctas

(se puede construir su árbol con G)

ω_1 = real id , id

ω_2 = real id , id , id

ω_3 = integer id

Cadenas de entrada incorrectas

(no se puede construir su árbol con G)

ω_4 = real id ,

ω_5 = real , id , id

ω_6 = integer id ;

Ejemplo de análisis sintáctico de 2 cadenas de entrada

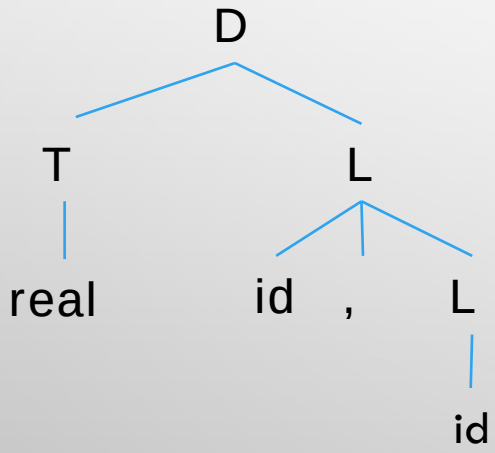
N = { D, T , L}

T = { integer, real, id }

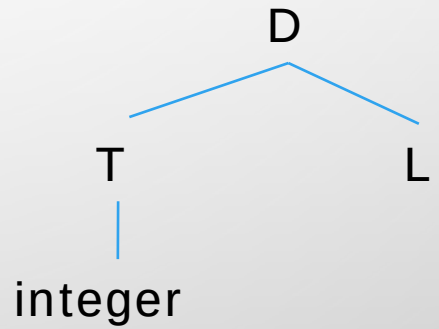
P =

D → T L
T → integer
T → real
L → id , L
L → id

Axioma = D



ω_1 = real id , id ✓



ω_7 = integer ✗

El Analizador Sintáctico ha de obtener **siempre el mismo árbol para una misma cadena de entrada**, y aplicando la misma secuencia de pasos

→ Ha de ser determinista

→ La GCL no puede ser ambigua

TIPOS DE ANALIZADORES SINTÁCTICOS:

- Descendente o Ascendente

  Descendente. Construye el árbol desde la raíz hacia las hojas

  Ascendente. Construye el árbol desde las hojas hacia la raíz

- Con retroceso o Sin retroceso

  Con retroceso. Si hay varias reglas candidatas para expandir un nodo del árbol, se elige una; si no se consigue terminar el árbol, se retrocede hasta ese nodo y se elige otra regla, y así hasta terminar el árbol o agotar las reglas candidatas → Ineficientes

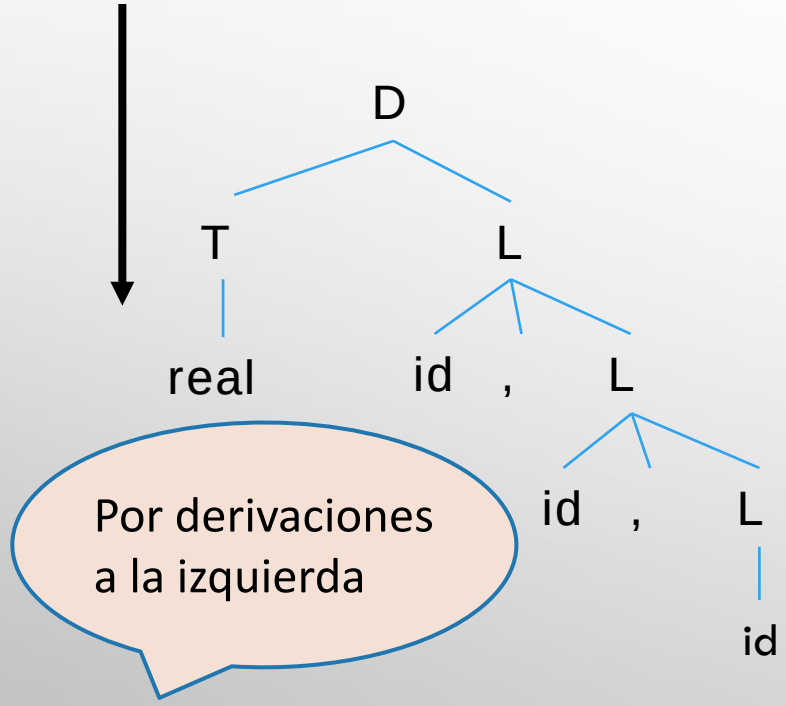
  Sin retroceso. Utiliza algún criterio para saber con certeza cuál es la siguiente regla a aplicar en cada instante → ¿Cómo?? **Gramáticas LL y LR**

Ejemplo de construcción del árbol

1. D → T L
2. T → integer
3. T → real
4. L → id , L
5. L → id

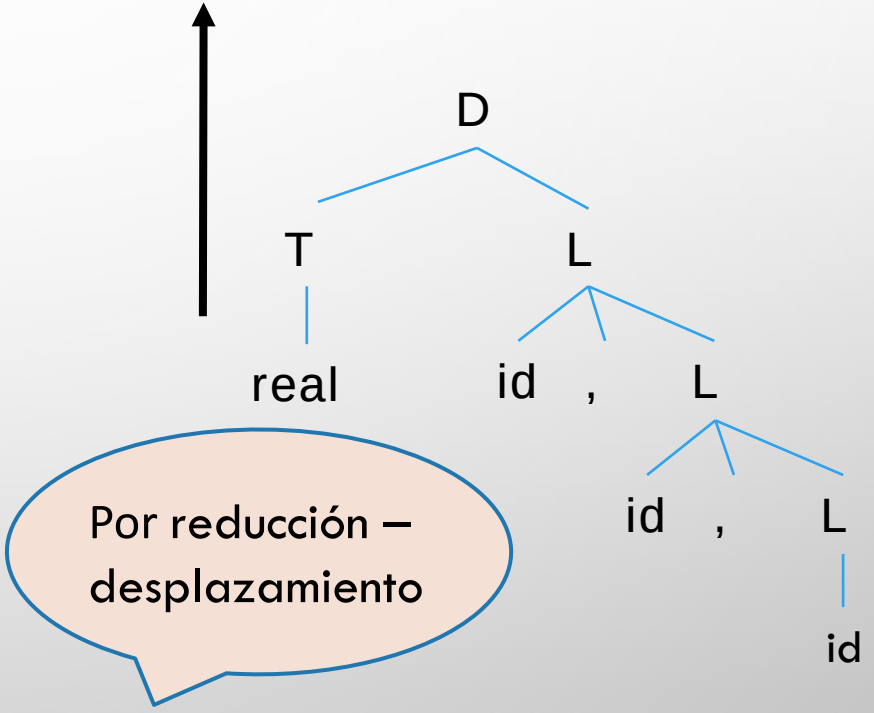
ω = `real id , id , id`

A. St. Descendente



Parse: 1 3 4 4 5

A. St. Ascendente



Parse: 3 5 4 4 1