

# SÍMBOLO DIRECTORES COMUNES

Leer más tokens

Ascendente

Transformar la GLC

$a \rightarrow \text{IDENT IDENT}$   
 $b \rightarrow \text{NUM}$   
 $b \rightarrow \text{IDENT}$   
 $a \rightarrow \text{IDENT } b$   
 $b \rightarrow \text{IDENT | NUM}$

# RECURSIVIDAD A IZQUIERDA

es un problema  
 GOOD DO

Ascendente

Transformar la GLC

$a \rightarrow a \dots$

$\text{expr} \rightarrow \text{expr} '+' \text{NUM}$   
 $\text{NUM}$

REGLA  
 $n \rightarrow \alpha \mid \beta \Rightarrow n \rightarrow \beta m$   
 $m \rightarrow \alpha m \mid \lambda$   
 $\text{expr} \rightarrow \text{NUM } m$   
 $m \rightarrow '+' \text{NUM } m \mid \lambda$

# GRAMÁTICAS AMBIGUAS

Transformar la GLC

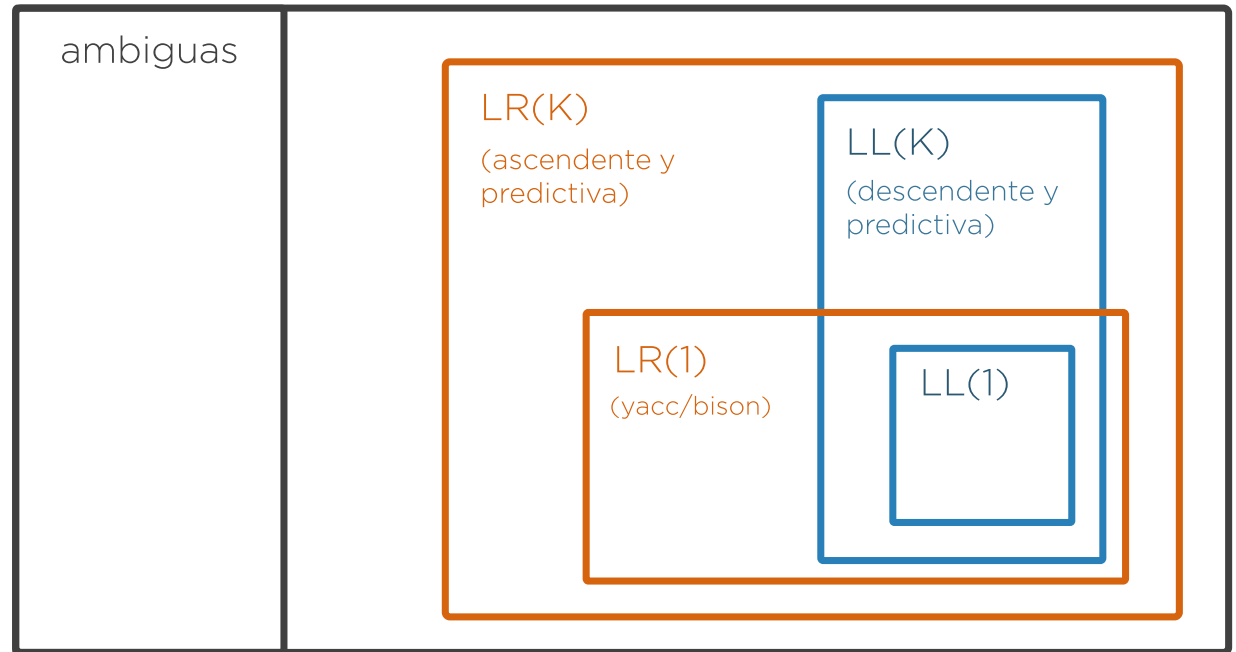
Cambiar el Lenguaje

Reglas de Selección

Llegar al mismo camino de maneras distintas. ← PROBLEMA  
 No se pueden detectar

Si una gramática no pasa únicamente el algoritmo  $LL(1)$ , no es ambigua.

En las gramáticas no existe el orden



ANTLR es una herramienta ascendente o descendente?  
 descendente.

Una gramática  $LL(1)$  puede usar recursividad a izquierda?  
 No

PREGUNTA EXAMEN  
 PREGUNTA EXAMEN

## EJERCICIO

- Operadores relacionales: '==' y '!='.
- Operadores lógicos: '||' y '&&' (OR y AND).
- Operadores aritméticos: '+', '\*', '-' y '/'.
- Agrupación de expresiones entre paréntesis.

```
print a + 2 == 3 && a / 5 != b;  
print a * (4 - 2);
```

Start: 'print' expr ';' ;

expr: IDENT

| NUM

| expr ' && ' expr

| '(' expr ')'

| expr ( '\*' | '/' ) expr

| expr ( '+' | '-' ) expr

| expr ( '==' | '!=' ) expr

| expr '||' expr

;

## EJEMPLO BNF

- Un conjunto está formado por uno o más elementos entre paréntesis separados por comas.
- Cada elemento puede ser un número u otro conjunto.
- Los números están formados por dígitos de 1 al 3.
- Los números están formados por un número impar de dígitos y son palíndromos.

En las gramáticas  
nunca importa el  
orden

(131)

(3, 222, 12321)

(12121, 2, (333), (2, 3, 111111))

(2132312, ( (1, 2), 2, 131), 1, 2), 3322233)

conjunto  $\rightarrow$  '(' elemento ')'

elementos  $\rightarrow$  elemento

| elemento ',' elementos

elemento  $\rightarrow$  num

| conjunto

num  $\rightarrow$  UNO

| DOS

| TRES

| UNO num UNO

| DOS num DOS

| TRES num TRES

} composición

SS  $\rightarrow$  sin separadores

CS  $\rightarrow$  con separadores

Patrón	BNF RI <small>recursiva izquierda</small>	BNF RD <small>recursiva derecha</small>	EBNF
1+ SS	$l \rightarrow e   le$	$l \rightarrow e   el$	$l \rightarrow e^+$
0+ SS	$l \rightarrow \lambda   le$	$l \rightarrow \lambda   el$	$l \rightarrow e^*$
1+ CS	$l \rightarrow e   l ; e$	$l \rightarrow e   e ; l$	$l \rightarrow e ( ; e )^*$
0+ CS	$l_{opt} \rightarrow \lambda$ $l \rightarrow e   l ; e$	$l_{opt} \rightarrow \lambda$ $l \rightarrow e   e ; l$	$l \rightarrow ( e ( ; e )^* ) ?$

$\left\{ \begin{array}{l} \text{class} \rightarrow \text{CLASS} \\ \text{programa} \rightarrow \text{class} | \text{programa class} \end{array} \right.$


# BNF

## Definición de variables

- Puede no haber ninguna.
- Tipos entero y real.

`int a, b; double x;`

$1 + CS$



$idents \rightarrow ID \mid idents \, ' \, ID$   
 $def \rightarrow tipo \, idents \, ' ; '$   
 $defs \rightarrow \Lambda \mid defs \, def$