

Diseño de Compiladores I

Análisis Sintáctico
Parsing Descendente

Análisis Sintáctico

- ▶ Agrupa los tokens del programa fuente en frases gramaticales que el compilador usará en las siguientes etapas.
- ▶ Obtiene una cadena de tokens del Analizador Léxico, y verifica que la cadena de tokens pueda generarse mediante la gramática del lenguaje.



Estrategias de Parsing

- ▶ **Parsing Ascendente (bottom up)**
 - ▶ Construye el árbol desde las hojas a la raíz

- ▶ **Parsing Descendente (top down)**
 - ▶ Construye el árbol desde la raíz a las hojas



Parsing Ascendente

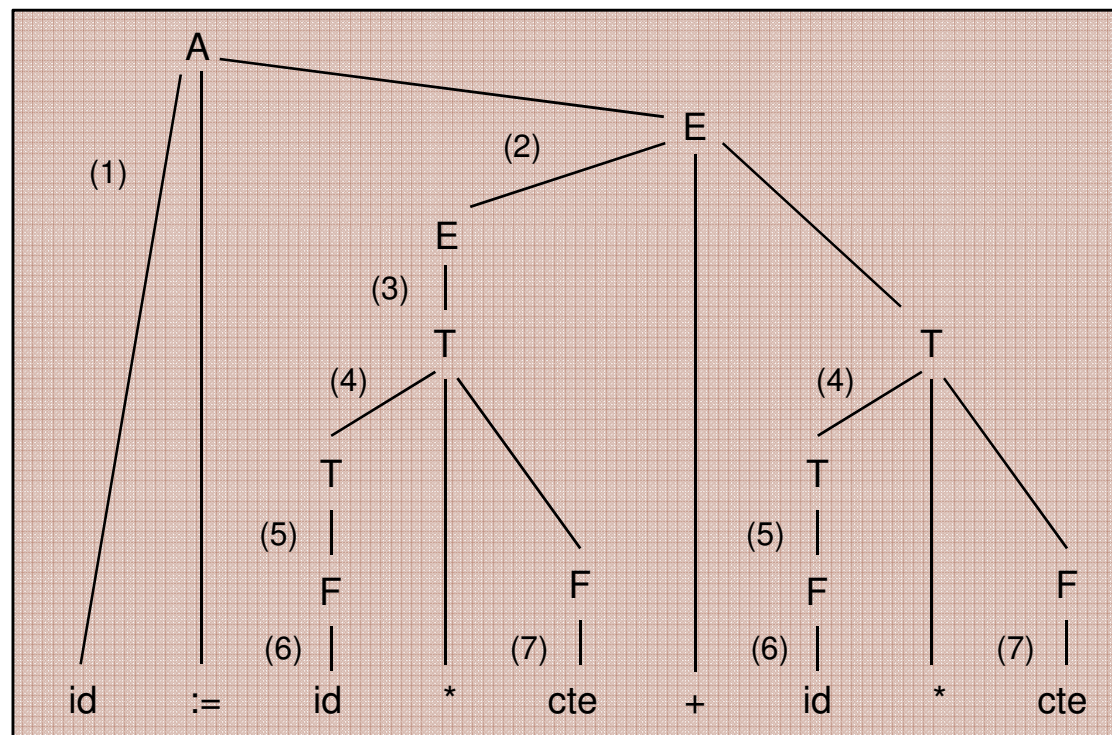
Ejemplo

precio := costo1 * 1.5 + costo2 * 1.2 → id := id * cte + id * cte

Gramática

- 1) $A \rightarrow id := E$
- 2) $E \rightarrow E + T$
- 3) $E \rightarrow T$
- 4) $T \rightarrow T * F$
- 5) $T \rightarrow F$
- 6) $F \rightarrow id$
- 7) $F \rightarrow cte$

Árbol de Parsing



Lista de Reglas: 6 5 7 4 3 6 5 7 4 2 1

Parsing Descendente

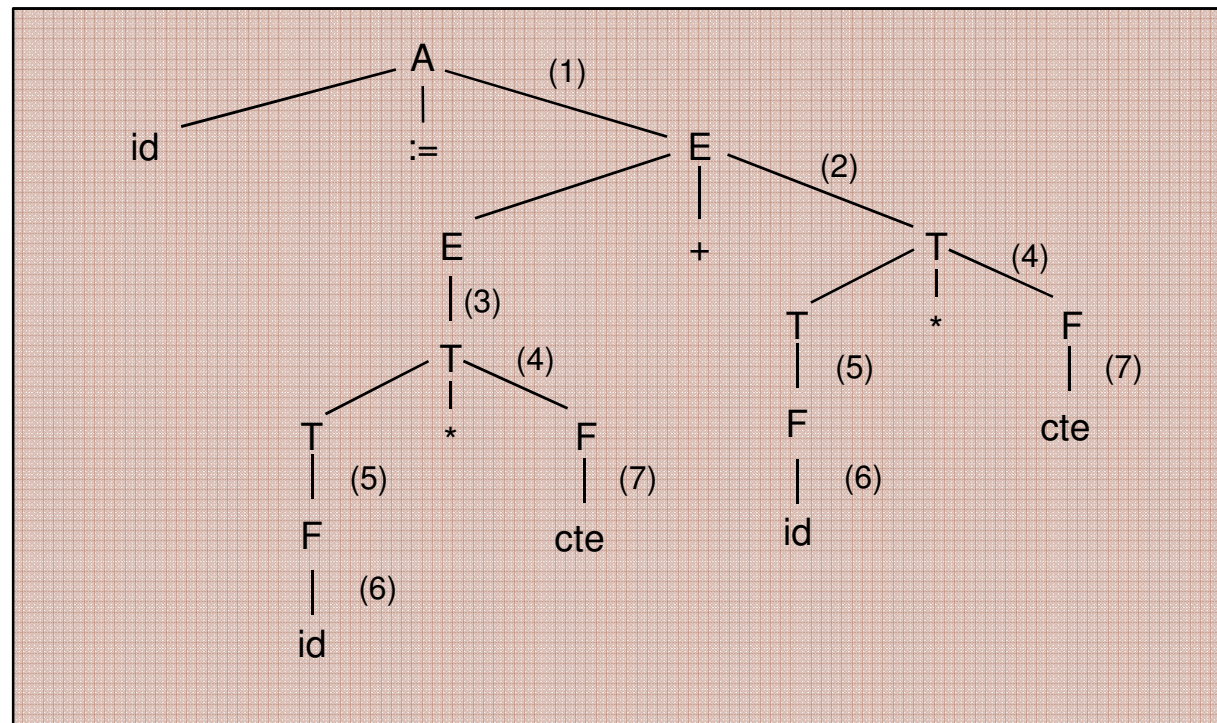
Ejemplo

precio := costo1 * 1.5 + costo2 * 1.2 → id := id * cte + id * cte

Gramática

- 1) $A \rightarrow id := E$
- 2) $E \rightarrow E + T$
- 3) $E \rightarrow T$
- 4) $T \rightarrow T * F$
- 5) $T \rightarrow F$
- 6) $F \rightarrow id$
- 7) $F \rightarrow cte$

Árbol de Parsing



Lista de Reglas: 1 2 3 4 5 6 7 4 5 6 7



Parsing Descendente

Gramáticas LL

- ▶ Va de la hipótesis al programa
- ▶ El programa se lee de izquierda a derecha (**L**).
- ▶ Las reglas se leen de izquierda a derecha (**L**):
el lado izquierdo se reemplaza por el derecho.
- ▶ Estrategia: **Expansión**



Parsing Descendente

- ▶ Predice el programa.
- ▶ Sólo puede construir el programa deseado, si se elige la secuencia correcta de reglas.
- ▶ Otras secuencias construyen otros programas.
- ▶ La dificultad está en determinar la secuencia correcta.

Parsing Descendente

Gramática

- 1) $A \rightarrow \text{id} := E$
- 2) $E \rightarrow E + T$
- 3) $E \rightarrow T$
- 4) $T \rightarrow T * F$
- 5) $T \rightarrow F$
- 6) $F \rightarrow \text{id}$
- 7) $F \rightarrow \text{cte}$



Parsing Descendente

Método lineal, directo o inductivo

$a := b * 5 + c$



$id := id * cte + id$

Parsing

- ▶ A
- ▶ $id := E$
Existen 2 opciones para E → Elijo la primera:
- ▶ $id := E + T$
Existen 2 opciones para E → Elijo la primera:
- ▶ $id := E + T + T$
Existen 2 opciones para E → Elijo la primera:
- ▶ $id := E + T + T + T$
- ▶ ...

Gramática

- 1) $A \rightarrow id := E$
- 2) $E \rightarrow E + T$
- 3) $E \rightarrow T$
- 4) $T \rightarrow T * F$
- 5) $T \rightarrow F$
- 6) $F \rightarrow id$
- 7) $F \rightarrow cte$

El algoritmo entra en un ciclo infinito

Causa: Recursividad a izquierda



Parsing Descendente

**El Parsing Descendente es incompatible
con la recursividad a izquierda**

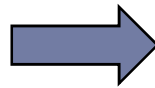


Parsing Descendente

- ▶ El Parsing Descendente es incompatible con la recursividad a izquierda.
- ▶ Se debe modificar la gramática para que sea recursiva a derecha.

Gramática

- 1) $A \rightarrow \text{id} := E$
- 2) $E \rightarrow E + T$
- 3) $E \rightarrow T$
- 4) $T \rightarrow T * F$
- 5) $T \rightarrow F$
- 6) $F \rightarrow \text{id}$
- 7) $F \rightarrow \text{cte}$



Gramática

- 1) $A \rightarrow \text{id} := E$
- 2) $E \rightarrow T + E$
- 3) $E \rightarrow T$
- 4) $T \rightarrow F * T$
- 5) $T \rightarrow F$
- 6) $F \rightarrow \text{id}$
- 7) $F \rightarrow \text{cte}$



Parsing Descendente

Eliminar la recursividad a izquierda

1) $A \rightarrow A \alpha$

2) $A \rightarrow \beta$



1) $A \rightarrow \beta A'$

2) $A' \rightarrow \alpha A'$

3) $A' \rightarrow \lambda$

1) $A \rightarrow \text{id} := E$

2) $E \rightarrow E + T$

3) $E \rightarrow T$

4) $T \rightarrow T * F$

5) $T \rightarrow F$

6) $F \rightarrow \text{id}$

7) $F \rightarrow \text{cte}$



1) $A \rightarrow \text{id} := E$

2) $E \rightarrow T E'$

3) $E' \rightarrow + T E'$

4) $E' \rightarrow \lambda$

5) $T \rightarrow F T'$

6) $T' \rightarrow * F T'$

7) $T' \rightarrow \lambda$

8) $F \rightarrow \text{id}$

9) $F \rightarrow \text{cte}$



Parsing Descendente

Eliminar la recursividad a izquierda

- 1) $A \rightarrow id := E$
- 2) $E \rightarrow T E'$
- 3) $E' \rightarrow + T E'$
- 4) $E' \rightarrow \lambda$
- 5) $T \rightarrow F T'$
- 6) $T' \rightarrow * F T'$
- 7) $T' \rightarrow \lambda$
- 8) $F \rightarrow id$
- 9) $F \rightarrow cte$



- 1) $A \rightarrow id := E$
- 2) $E \rightarrow T + E$
- 3) $E \rightarrow T$
- 4) $T \rightarrow F * T$
- 5) $T \rightarrow F$
- 6) $F \rightarrow id$
- 7) $F \rightarrow cte$

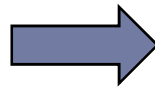


Parsing Descendente

- ▶ La recursividad a derecha modifica la asociatividad de los operadores:

Gramática

- 1) $A \rightarrow id := E$
- 2) $E \rightarrow T + E$
- 3) $E \rightarrow T - E$
- 4) $E \rightarrow T$
- 5) $T \rightarrow F * T$
- 6) $T \rightarrow F / T$
- 7) $T \rightarrow F$
- 8) $F \rightarrow id$
- 9) $F \rightarrow cte$



Ejemplo:

$$(3 + 2) + 2 = 3 + (2 + 2)$$

Pero...

$$(3 - 2) - 2 \neq 3 - (2 - 2)$$

Parsing Descendente LL(0)

Ejemplo

$a := b * 5 + c \rightarrow id := id * cte + id$

	Reconocido	Lista de reglas	Postergada
	A		
✓	<u>id := E</u>	1	
	id := T + E	1,2	3
	id := F * T + E	1,2,4	5
✓	<u>id := id * T + E</u>	1,2,4,6	7
	id := id * F * T + E	1,2,4,6,4	5 ✓
✗	<u>id := id * id * T + E</u>	1,2,4,6,4,6	7 ✓
✗	<u>id := id * cte * T + E</u>	1,2,4,6,4,7	
	id := id * F + E	1,2,4,6,5	
✗	<u>id := id * id + E</u>	1,2,4,6,5,6	7 ✓
✓	<u>id := id * cte + E</u>	1,2,4,6,5,7	

Gramática

- 1) $A \rightarrow id := E$
- 2) $E \rightarrow T + E$
- 3) $E \rightarrow T$
- 4) $T \rightarrow F * T$
- 5) $T \rightarrow F$
- 6) $F \rightarrow id$
- 7) $F \rightarrow cte$

Parsing Descendente LL(0)

Ejemplo

$a := b * 5 + c \rightarrow id := id * cte + id$

	Reconocido	Lista de reglas	Postergada
✓	<u>id := id * cte + E</u>	1,2,4,6,5,7	
	id := id * cte + T + E	1,2,4,6,5,7,2	3✓
	id := id * cte + F * T + E	1,2,4,6,5,7,2,4	5✓
✗	<u>id := id * cte + id * T + E</u>	1,2,4,6,5,7,2,4,6	7✓
✗	<u>id := id * cte + cte * T + E</u>	1,2,4,6,5,7,2,4,7	
	id := id * cte + F + E	1,2,4,6,5,7,2,5	
✗	<u>id := id * cte + id + E</u>	1,2,4,6,5,7,2,5,6	7✓
✗	<u>id := id * cte + cte + E</u>	1,2,4,6,5,7,2,5,7	
	id := id * cte + T	1,2,4,6,5,7,3	

Gramática

- 1) $A \rightarrow id := E$
- 2) $E \rightarrow T + E$
- 3) $E \rightarrow T$
- 4) $T \rightarrow F * T$
- 5) $T \rightarrow F$
- 6) $F \rightarrow id$
- 7) $F \rightarrow cte$



Parsing Descendente LL(0)

Ejemplo

$a := b * 5 + c \rightarrow id := id * cte + id$

	Reconocido	Lista de reglas	Postergada
	$id := id * cte + T$	1,2,4,6,5,7,3	
	$id := id * cte + F * T$	1,2,4,6,5,7,3,4	5 ✓
✗	<u>$id := id * cte + id * T$</u>	1,2,4,6,5,7,3,4,6	7 ✓
✗	<u>$id := id * cte + cte * T$</u>	1,2,4,6,5,7,3,4,7	
✓	<u>$id := id * cte + F$</u>	1,2,4,6,5,7,3,5	
✓	<u>$id := id * cte + id$</u>	<u>1,2,4,6,5,7,3,5,6</u>	

Gramática

- 1) $A \rightarrow id := E$
- 2) $E \rightarrow T + E$
- 3) $E \rightarrow T$
- 4) $T \rightarrow F * T$
- 5) $T \rightarrow F$
- 6) $F \rightarrow id$
- 7) $F \rightarrow cte$

Lista de Reglas: 1 2 4 6 5 7 3 5 6



Parsing Descendente LL(0)

- ▶ Elige la regla sin mirar lo que viene después
- ▶ Tiene muchos retrocesos
- ▶ ¿Y si el programa está mal?

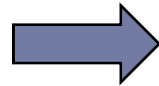


Parsing Descendente LL(1)

- ▶ Cada no Terminal debe tener una sola definición.
- ▶ Se modifica la gramática

Gramática

- 1) $A \rightarrow id := E$
- 2) $E \rightarrow T + E$
- 3) $E \rightarrow T$
- 4) $T \rightarrow F * T$
- 5) $T \rightarrow F$
- 6) $F \rightarrow id$
- 7) $F \rightarrow cte$



Gramática

- 1) $A \rightarrow id := E$
- 2) $E \rightarrow T R$
- 3) $R \rightarrow + E$
- 4) $R \rightarrow \lambda$
- 5) $T \rightarrow F Q$
- 6) $Q \rightarrow * T$
- 7) $Q \rightarrow \lambda$
- 8) $F \rightarrow id$
- 9) $F \rightarrow cte$

Ahora R y Q no tienen definición única, pero existe un terminal que permite decidir

Parsing Descendente LL(1) o Parsing Descendente Predictivo

A

id := E

id := T R

id := F Q R

id := id Q R

id := id * T R

id := id * F Q R

id := id * cte Q R

id := id * cte R

id := id * cte + E

id := id * cte + T R

id := id * cte + F Q R

id := id * cte + id Q R

id := id * cte + id R

id := id * cte + id

a := b * 5 + c → id := id * cte + id

Gramática

- 1) $A \rightarrow \text{id} := E$
- 2) $E \rightarrow T R$
- 3) $R \rightarrow + E$
- 4) $R \rightarrow \lambda$
- 5) $T \rightarrow F Q$
- 6) $Q \rightarrow * T$
- 7) $Q \rightarrow \lambda$
- 8) $F \rightarrow \text{id}$
- 9) $F \rightarrow \text{cte}$

→ No hay retrocesos

Parsing Descendente Predictivo

Construcción (intuitiva) de la Matriz

Gramática

- 1) $A \rightarrow id := E$
- 2) $E \rightarrow T R$
- 3) $R \rightarrow + E$
- 4) $R \rightarrow \lambda$
- 5) $T \rightarrow F Q$
- 6) $Q \rightarrow * T$
- 7) $Q \rightarrow \lambda$
- 8) $F \rightarrow id$
- 9) $F \rightarrow cte$

	Id	cte	:=	+	*	\$
A	id := E					
E						
F	id	cte				
T						
R				+ E		
Q					* T	

→ Agrego la definición de todas las reglas que empiezan con un Terminal, para la combinación (NT,T) correspondiente

Parsing Descendente LL(1)

A

id := E

id := T R

id := F Q R

id := id Q R

id := id * T R

id := id * F Q R

id := id * cte Q R

id := id * cte R

id := id * cte + E

id := id * cte + T R

id := id * cte + F Q R

id := id * cte + id Q R

id := id * cte + id R

id := id * cte + id

a := b * 5 + c → id := id * cte + id

Gramática

- 1) A → id := E
- 2) E → T R
- 3) R → + E
- 4) R → λ
- 5) T → F Q
- 6) Q → * T
- 7) Q → λ
- 8) F → id
- 9) F → cte

→ T se reemplaza por FQ cuando viene id o cte

Parsing Descendente Predictivo

Construcción de la Matriz

Gramática

- 1) $A \rightarrow id := E$
- 2) $E \rightarrow T R$
- 3) $R \rightarrow + E$
- 4) $R \rightarrow \lambda$
- 5) $T \rightarrow F Q$
- 6) $Q \rightarrow * T$
- 7) $Q \rightarrow \lambda$
- 8) $F \rightarrow id$
- 9) $F \rightarrow cte$

	Id	cte	:=	+	*	\$
A	id := E					
E						
F	id	cte				
T	FQ	FQ				
R				+ E		
Q					* T	

→ T se reemplaza por FQ cuando viene id o cte (primer Terminal de F)

▶ → Agrego FQ para las celdas (T , id) y (T , cte)

Parsing Descendente LL(1)

A

id := E

id := T R

id := F Q R

id := id Q R

id := id * T R

id := id * F Q R

id := id * cte Q R

id := id * cte R

id := id * cte + E

id := id * cte + T R

id := id * cte + F Q R

id := id * cte + id Q R

id := id * cte + id R

id := id * cte + id

a := b * 5 + c → id := id * cte + id

Gramática

1) A → id := E

2) E → T R

3) R → + E

4) R → λ

5) T → F Q

6) Q → * T

7) Q → λ

8) F → id

9) F → cte

→ E se reemplaza por TR cuando viene id o cte

Parsing Descendente Predictivo

Construcción de la Matriz

Gramática

- 1) $A \rightarrow id := E$
- 2) $E \rightarrow T R$
- 3) $R \rightarrow + E$
- 4) $R \rightarrow \lambda$
- 5) $T \rightarrow F Q$
- 6) $Q \rightarrow * T$
- 7) $Q \rightarrow \lambda$
- 8) $F \rightarrow id$
- 9) $F \rightarrow cte$

	Id	cte	:=	+	*	\$
A	id := E					
E	TR	TR				
F	id	cte				
T	FQ	FQ				
R				+ E		
Q					* T	

- E se reemplaza por TR cuando viene id o cte (primer Terminal de F, por lo tanto primer Terminal de T)
- Agrego TR para las celdas (E , id) y (E , cte)

Parsing Descendente LL(1)

A

id := E

id := T R

id := F Q R

id := id Q R

id := id * T R

id := id * F Q R

id := id * cte Q R

id := id * cte R

id := id * cte + E

id := id * cte + T R

id := id * cte + F Q R

id := id * cte + id Q R

id := id * cte + id R

id := id * cte + id

a := b * 5 + c → id := id * cte + id

Gramática

1) A → id := E

2) E → T R

3) R → + E

4) R → λ

5) T → F Q

6) Q → * T

7) Q → λ

8) F → id

9) F → cte

→ Q desaparece (se reemplaza por λ) cuando viene + o \$

Parsing Descendente Predictivo

Construcción de la Matriz

Gramática

- 1) $A \rightarrow id := E$
- 2) $E \rightarrow T R$
- 3) $R \rightarrow + E$
- 4) $R \rightarrow \lambda$
- 5) $T \rightarrow F Q$
- 6) $Q \rightarrow * T$
- 7) $Q \rightarrow \lambda$
- 8) $F \rightarrow id$
- 9) $F \rightarrow cte$

	Id	cte	:=	+	*	\$
A	id := E					
E	T R	T R				
F	id	cte				
T	F Q	F Q				
R				+ E		
Q				λ	* T	λ

- Q desaparece (se reemplaza por λ) cuando viene + (primer Terminal de R) o \$
- Agrego λ para (Q , +) y para (Q , \$)

Parsing Descendente LL(1)

A

id := E

id := T R

id := F Q R

id := id Q R

id := id * T R

id := id * F Q R

id := id * cte Q R

id := id * cte R

id := id * cte + E

id := id * cte + T R

id := id * cte + F Q R

id := id * cte + id Q R

id := id * cte + id R

id := id * cte + id

a := b * 5 + c → id := id * cte + id

Gramática

1) A → id := E

2) E → T R

3) R → + E

4) R → λ

5) T → F Q

6) Q → * T

7) Q → λ

8) F → id

9) F → cte

→ R desaparece (se reemplaza por λ) cuando viene \$

Parsing Descendente Predictivo

Construcción de la Matriz

Gramática

- 1) $A \rightarrow id := E$
- 2) $E \rightarrow T R$
- 3) $R \rightarrow + E$
- 4) $R \rightarrow \lambda$
- 5) $T \rightarrow F Q$
- 6) $Q \rightarrow * T$
- 7) $Q \rightarrow \lambda$
- 8) $F \rightarrow id$
- 9) $F \rightarrow cte$

	Id	cte	:=	+	*	\$
A	id := E					
E	T R	T R				
F	id	cte				
T	F Q	F Q				
R				+ E		λ
Q				λ	* T	λ

→ R desaparece (se reemplaza por λ) cuando viene \$

▶ → Agrego λ (R, \$)

Parsing Descendente Predictivo

Construcción de la Matriz

Gramática

- 1) $A \rightarrow \text{id} := E$
- 2) $E \rightarrow T R$
- 3) $R \rightarrow + E$
- 4) $R \rightarrow \lambda$
- 5) $T \rightarrow F Q$
- 6) $Q \rightarrow * T$
- 7) $Q \rightarrow \lambda$
- 8) $F \rightarrow \text{id}$
- 9) $F \rightarrow \text{cte}$

	Id	cte	:=	+	*	\$
A	id := E					
E	T R	T R				
F	id	cte				
T	F Q	F Q				
R				+ E		λ
Q				λ	* T	λ

→ Las celdas en blanco representan errores



Parsing Descendente Predictivo Acciones

	Id	cte	:=	+	*	\$
A	E Leer 2					
E	T Apilar R	T Apilar R				
F	Leer Desapilar	Leer Desapilar				
T	F Apilar Q	F Apilar Q				
R				E Leer		Desapilar
Q				Desapilar	T Leer	Desapilar

- ▶ **Generar una regla**
- ▶ **Leer**
- ▶ **Apilar**
- ▶ **Desapilar**

- ▶ **T + NT → Leer + Generar**
Ir a NT
- ▶ **NT1 + NT2 → Apilar NT2**
Ir a NT1
- ▶ **T → Leer + Generar**
Desapilar + Ir al NT
tope de la pila
- ▶ **λ → Desapilar + Ir al NT**
tope de la pila

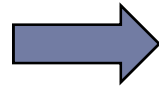


Parsing Descendente Predictivo

- ▶ Para evitar la acción Leer 2, se modifica la gramática

Gramática

- 1) $A \rightarrow id := E$
- 2) $E \rightarrow T R$
- 3) $R \rightarrow + E$
- 4) $R \rightarrow \lambda$
- 5) $T \rightarrow F Q$
- 6) $Q \rightarrow * T$
- 7) $Q \rightarrow \lambda$
- 8) $F \rightarrow id$
- 9) $F \rightarrow cte$



Gramática

- 1) $A \rightarrow id B$
- 2) $B \rightarrow := E$
- 3) $E \rightarrow T R$
- 4) $R \rightarrow + E$
- 5) $R \rightarrow \lambda$
- 6) $T \rightarrow F Q$
- 7) $Q \rightarrow * T$
- 8) $Q \rightarrow \lambda$
- 9) $F \rightarrow id$
- 10) $F \rightarrow cte$



Parsing Descendente Predictivo Acciones

	Id	cte	:=	+	*	\$
A	B Leer					
B			E Leer			
E	T Apilar R	T Apilar R				
F	Leer Desapilar	Leer Desapilar				
T	F Apilar Q	F Apilar Q				
R				E Leer		Desapilar
Q				Desapilar	T Leer	Desapilar

Gramática

- 1) $A \rightarrow id B$
- 2) $B \rightarrow := E$
- 3) $E \rightarrow T R$
- 4) $R \rightarrow + E$
- 5) $R \rightarrow \lambda$
- 6) $T \rightarrow F Q$
- 7) $Q \rightarrow * T$
- 8) $Q \rightarrow \lambda$
- 9) $F \rightarrow id$
- 10) $F \rightarrow cte$

Parsing Descendente Predictivo Ejemplo

$a := b * 5 + c \rightarrow id := id * cte + id$

Estado	Leído	Pila	Generado	Regla
A	id			
B	:=		id	1
E	id		id :=	2
T	id	R		3
F	id	QR		6
Q	*	R	id := id	9
T	cte	R	id := id *	7
F	cte	QR		6
Q	+	R	id := id * cte	10
R	+			8
E	id		id := id * cte +	4
T	id	R		3
F	id	QR		6
Q	\$	R	id := id * cte + id	9
R	\$	-		8
FIN				5

¿Cómo se detecta el FIN?

- ▶ Cuando se vacía la pila y el token leído es \$

Parsing Descendente Predictivo Ejemplo con error

$a := b * 5 c$ \rightarrow $id := id * cte id$

NT	Leído	Pila	Generado	Regla
A	id			
B	:=		id	1
E	id		id :=	2
T	id	R		3
F	id	QR		6
Q	*	R	id := id	9
T	cte	R	id := id *	7
F	cte	QR		6
Q	id	R	id := id * cte	10

ERROR: Se esperaba + o * y se leyó id



Parsing Descendente Predictivo

Construcción Formal de la Matriz



Parsing Descendente Predictivo

Construcción Formal de la Matriz

- ▶ Se completan las celdas de la matriz, considerando los Primeros y los Siguietes de los No Terminales



Parsing Descendente Predictivo

Construcción de la Matriz

PRIMEROS

1. Si x es Terminal \rightarrow $\text{PRIMEROS}(x) = \{x\}$
2. Si $X \rightarrow \lambda$ $\rightarrow \lambda \in \text{PRIMEROS}(X)$
3. Si $X \rightarrow A B C D \dots$ $\rightarrow \text{PRIMEROS}(X) =$
PRIMEROS(A)
 \cup PRIMEROS(B) $\text{Si } \lambda \in \text{PRIMEROS}(A)$
 \cup PRIMEROS(C) $\text{Si } \lambda \in \text{PRIMEROS}(A) \wedge \lambda \in \text{PRIMEROS}(B)$
 $\cup \dots$



Parsing Descendente Predictivo

Construcción de la Matriz

► Se obtienen los PRIMEROS de los NT

- | | | |
|-----|-------------------------|---|
| 1) | $A \rightarrow id B$ | |
| 2) | $B \rightarrow := E$ | $PRIMEROS(A) = \{ id \}$ |
| 3) | $E \rightarrow T R$ | $PRIMEROS(B) = \{ := \}$ |
| 4) | $R \rightarrow + E$ | $PRIMEROS(E) = PRIMEROS(T) = \{ id, cte \}$ |
| 5) | $R \rightarrow \lambda$ | $PRIMEROS(R) = \{ +, \lambda \}$ |
| 6) | $T \rightarrow F Q$ | $PRIMEROS(T) = PRIMEROS(F) = \{ id, cte \}$ |
| 7) | $Q \rightarrow * T$ | $PRIMEROS(Q) = \{ *, \lambda \}$ |
| 8) | $Q \rightarrow \lambda$ | $PRIMEROS(F) = \{ id, cte \}$ |
| 9) | $F \rightarrow id$ | |
| 10) | $F \rightarrow cte$ | |

Parsing Descendente Predictivo

Construcción de la Matriz

- ▶ Se obtienen los SIGUIENTES de los NT

SIGUIENTES

1. Se agrega \$ a SIGUIENTES(A)
(Siendo A la primera regla de la gramática)

2. Si $X \rightarrow \alpha A \beta$ \rightarrow
SIGUIENTES(A) =

PRIMEROS(β)
 \cup SIGUIENTES(X)

Excepto λ
Si $\beta \neq \epsilon$
 $\forall \lambda \in \text{PRIMEROS}(\beta)$



Parsing Descendente Predictivo Construcción de la Matriz

► Se obtienen los SIGUIENTES de los NT

1) $A \rightarrow id B$

2) $B \rightarrow := E$

3) $E \rightarrow T R$

4) $R \rightarrow + E$

5) $R \rightarrow \lambda$

6) $T \rightarrow F Q$

7) $Q \rightarrow * T$

8) $Q \rightarrow \lambda$

9) $F \rightarrow id$

10) $F \rightarrow cte$

$SIGUIENTES(A) = \{ \$ \}$

$SIGUIENTES(B) = SIGUIENTES(A) = \{ \$ \}$

$SIGUIENTES(E) = SIGUIENTES(B) \cup SIGUIENTES(R) = \{ \$ \}$

$SIGUIENTES(R) = SIGUIENTES(E) = \{ \$ \}$

$SIGUIENTES(T) = PRIMEROS(R) \cup SIGUIENTES(E)$

$\cup SIGUIENTES(Q) = \{ +, \$ \}$

$SIGUIENTES(Q) = SIGUIENTES(T) = \{ +, \$ \}$

$SIGUIENTES(F) = PRIMEROS(Q) \cup SIGUIENTES(T) = \{ *, +, \$ \}$



► Se completan las celdas para los PRIMEROS

- 1) $A \rightarrow id B$
- 2) $B \rightarrow := E$
- 3) $E \rightarrow T R$
- 4) $R \rightarrow + E$
- 5) $R \rightarrow \lambda$
- 6) $T \rightarrow F Q$
- 7) $Q \rightarrow * T$
- 8) $Q \rightarrow \lambda$
- 9) $F \rightarrow id$
- 10) $F \rightarrow cte$

PRIMEROS(A) = { id }

PRIMEROS(B) = { := }

PRIMEROS(E) = { id, cte }

PRIMEROS(R) = { +, λ }

PRIMEROS(T) = { id, cte }

PRIMEROS(Q) = { *, λ }

PRIMEROS(F) = { id, cte }

	Id	cte	:=	+	*	\$
A	id B					
B			:= E			
E	T R	T R				
T	F Q	F Q				
F	Id	cte				
R				+ E		
Q					* T	



- Para todos los no Terminales α tal que $\lambda \in \text{PRIMEROS}(\alpha)$, se consideran los SIGUIENTES (α)

- 1) $A \rightarrow \text{id B}$
- 2) $B \rightarrow := E$
- 3) $E \rightarrow T R$
- 4) $R \rightarrow + E$
- 5) $R \rightarrow \lambda$
- 6) $T \rightarrow F Q$
- 7) $Q \rightarrow * T$
- 8) $Q \rightarrow \lambda$
- 9) $F \rightarrow \text{id}$
- 10) $F \rightarrow \text{cte}$

$\text{PRIMEROS}(R) = \{ +, \lambda \}$

$\text{SIGUIENTES}(R) = \{ \$ \}$

$\text{PRIMEROS}(Q) = \{ *, \lambda \}$

$\text{SIGUIENTES}(Q) = \{ +, \$ \}$

	Id	cte	:=	+	*	\$
A	id B					
B			:= E			
E	T R	T R				
T	F Q	F Q				
F	Id	cte				
R				+ E		λ
Q				λ	* T	λ



Parsing Descendente Predictivo

▶ Ventajas

- ▶ No tiene retrocesos.
- ▶ Errores bien definidos.

▶ Desventajas

- ▶ No hay muchos lenguajes que se adapten a este Parsing.
- ▶ Con LL(2) y LL(3) se podrían cubrir más lenguajes, pero las matrices serían muy grandes.

