



Tecnológico de Costa Rica

Centro Académico San José

Escuela de Ingeniería en Computación

Compiladores e Intérpretes

I Semestre 2016

Apuntes Clase - 05 de Mayo del 2017

Profesor: Dr. Francisco J. Torres-Rojas

Apuntadora: Liza Chaves Carranza, 2013016573

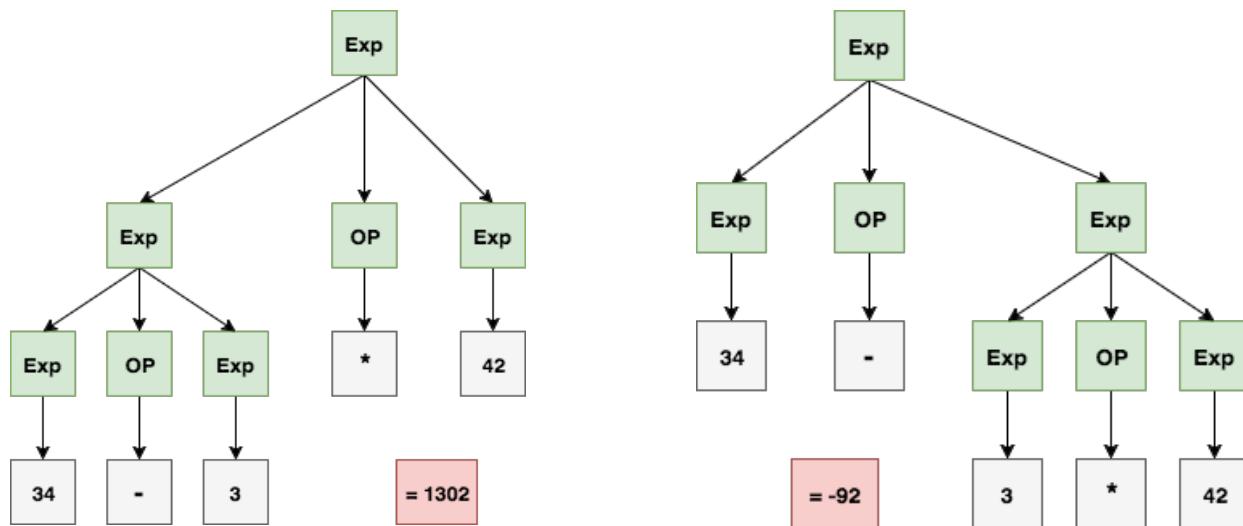
Tabla de Contenidos

Tabla de Contenidos.....	2
Repaso Sobre Árboles de Derivación	3
Ambigüedad en CFG.....	4
Precedencia.....	4
Ejemplo	4
CFG y Precedencia	5
Ejemplo	5
Ejemplo	5
Asociatividad.....	6
Ejemplo	7
CFG y Asociatividad	7
Ejemplo	7
Ejemplo	8
Parsing	9
Juego del Parsing.....	9
Dinámica.....	10
Desarrollo.....	10
¿Cómo Jugar al Parsing?	14
Análisis Sintáctico - Parsing Predictivo.....	14
Dirección del Parsing	14
Predicciones	15
Elementos del Parsing Predictivo	15
Tabla de Acciones.....	16
Algoritmo de Parsing Predictivo	16
Acción: Aceptar	16
Acción: Rechazar - Error	17
Acción: Match	17
Acción: Expandir	17
Resumen	17
Ejemplo 1.....	18
Ejemplo 2.....	19
Ejemplo 3.....	19
Ejemplo 4.....	20

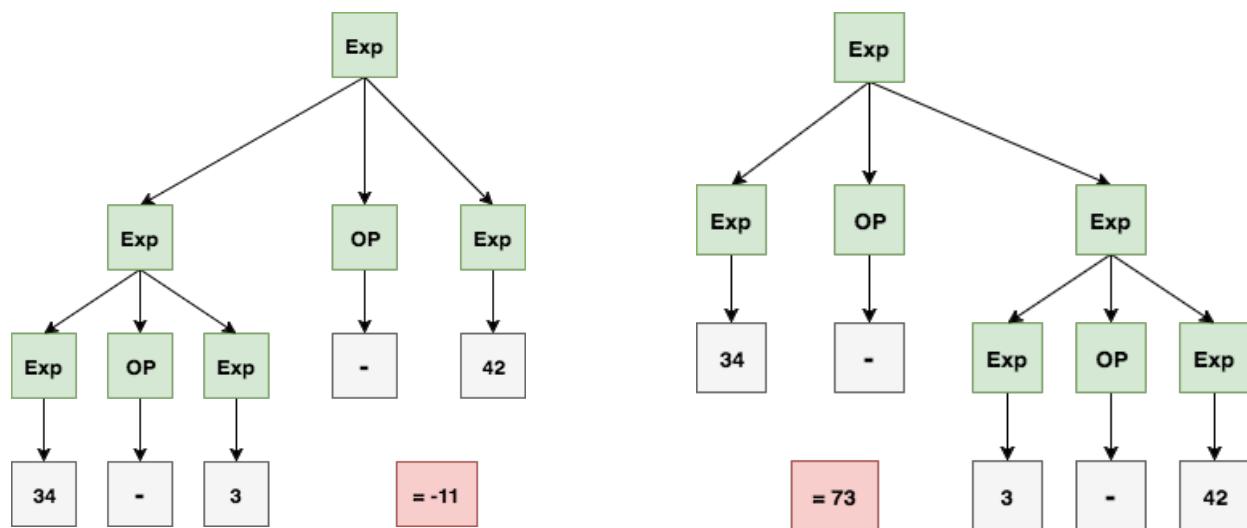
Repaso Sobre Árboles de Derivación

Una CFG es ambigua si existe al menos una hilera para la que haya al menos 2 árboles de derivación más izquierda (o al menos 2 árboles de derivación más derecha) que sean diferentes.

Y se muestra un ejemplo con la hilera “34 - 3 * 42” que si quisieramos comenzar el árbol por la derecha o la izquierda, serían diferentes árboles para una misma hilera.



También aplica para el mismo signo, puesto que existen algunas operaciones, como la resta, que no son conmutativas, por ejemplo, “34 - 3 - 42”:



Ambigüedad en CFG

- ¿Cómo se demuestra que una CFG es ambigua?
- ¿Cómo se demuestra que una CFG no es ambigua?
- ¿Si se sabe que una CFG es ambigua, se puede hacer algo?
 - Regla de desambigüación.
- No hay algoritmos generales conocidos.
- Los lenguajes de programación típicos incluyen ambigüedades pero estas pueden ser manejadas apropiadamente.

Precedencia

Se establece la precedencia entre 2 operadores o construcciones, esto significa que alguno de estos se va a ejecutar primero que el otro, y la gramática debe reflejar esta precedencia.

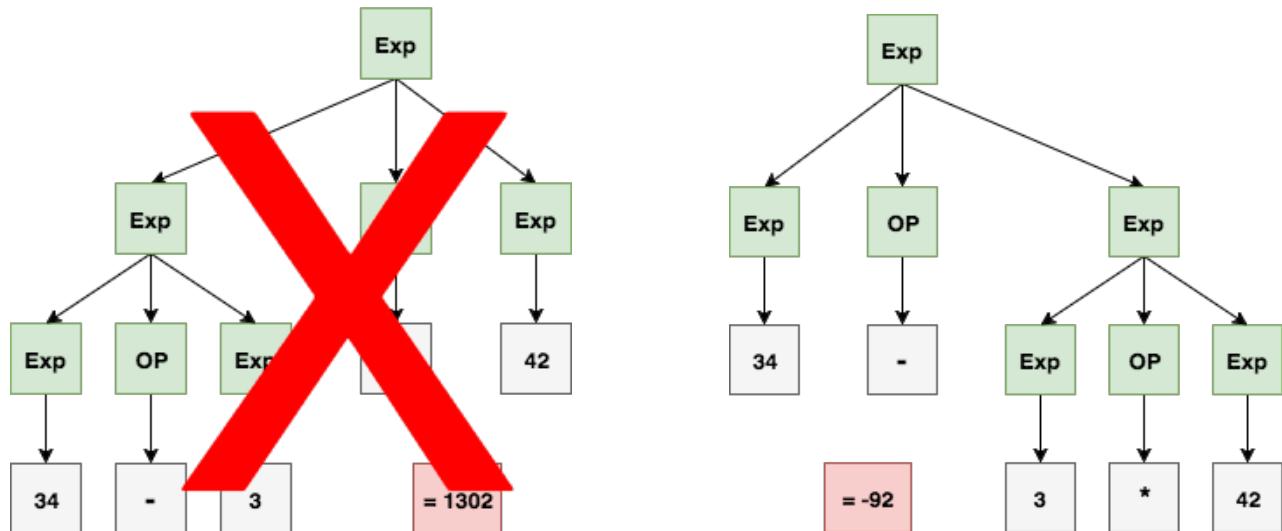
Podría haber parejas sin precedencia establecida, un ejemplo de esto, es ejecutar de izquierda a derecha, o viceversa.

Algunos ejemplos típicos en la matemática:

- Multiplicación y división sobre las sumas y las restas.
- Exponenciación sobre multiplicación.
- Paréntesis sobre cualquier otro operador.

Ejemplo

“34 - 3 * 42” La multiplicación sobre la resta:



CFG y Precedencia

Se establece que OP_1 tiene precedencia sobre OP_2 , y queremos que en el árbol de derivación:

- El subárbol de OP_1 quede más abajo.
- El subárbol de OP_2 quede más arriba.

Esto podemos reflejarlo entre las dos OPs porque las reglas más cercanas al símbolo inicial generan los operadores de menos precedencia y que se ejecutarán de último, y por ende, las de mayor precedencia irán más abajo, ejecutándose de primero.

Ejemplo: queremos que el subárbol de la multiplicación quede más abajo que el de la suma y también que estén más lejanas del símbolo inicial que las reglas de la suma.

Ejemplo

Considere la CFG:

$$\begin{aligned} \text{Exp} &\rightarrow \text{Exp OP Exp} \\ \text{Exp} &\rightarrow (\text{Exp}) \\ \text{Exp} &\rightarrow \langle \text{número} \rangle \\ \text{OP} &\rightarrow + \mid - \mid * \end{aligned}$$

Esta gramática no establece la precedencia entre multiplicación y suma/resta.

Modificación:

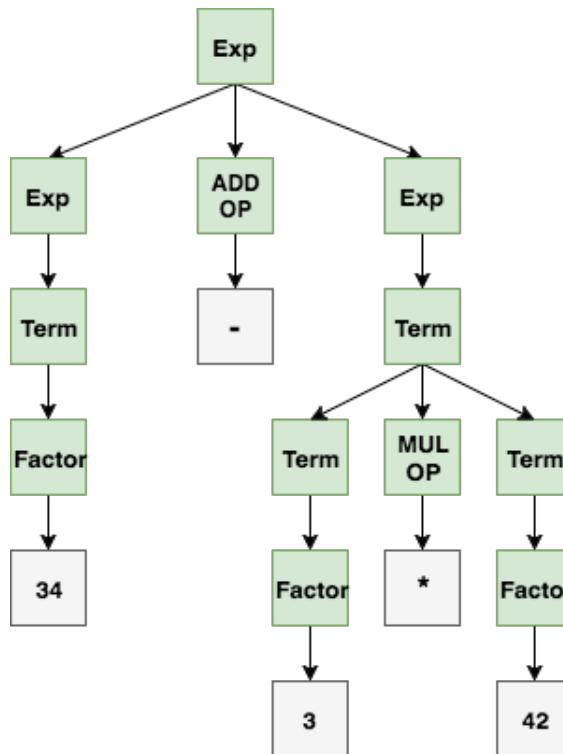
$$\begin{aligned} \text{Exp} &\rightarrow \text{Exp ADDOP Exp} \mid \text{Term} \\ \text{ADDOP} &\rightarrow + \mid - \\ \text{Term} &\rightarrow \text{Term MULOP Term} \mid \text{Factor} \\ \text{MULOP} &\rightarrow * \\ \text{Exp} &\rightarrow (\text{Exp}) \mid \langle \text{número} \rangle \end{aligned}$$

Ejemplo

Considere la CFG:

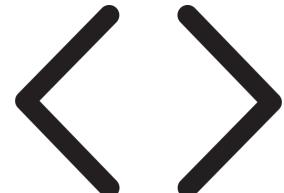
$$\begin{aligned} \text{Exp} &\rightarrow \text{Exp ADDOP Exp} \mid \text{Term} \\ \text{ADDOP} &\rightarrow + \mid - \\ \text{Term} &\rightarrow \text{Term MULOP Term} \mid \text{Factor} \\ \text{MULOP} &\rightarrow * \\ \text{Exp} &\rightarrow (\text{Exp}) \mid \langle \text{número} \rangle \end{aligned}$$

El árbol de derivación con respecto a la anterior CFG para la hilera “34 - 3 * 42” correspondería a:



Asociatividad

Dados dos operaciones o construcciones iguales (sin precedencia) se debe establecer si son **asociativos por la izquierda** o **asociativos por la derecha**.



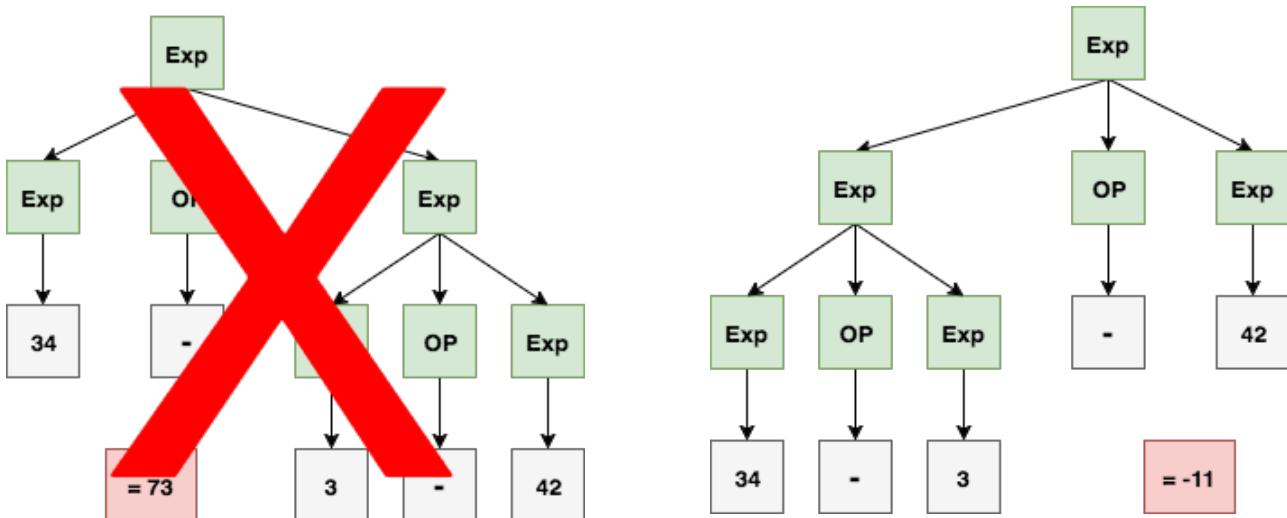
Alguno se tiene que ejecutar antes que el otro, y la gramática debe reflejar esta asociatividad.

Ejemplos típicos:

- Multiplicación, división, sumas y restas son asociativas por la izquierda.
- Asignación en el Lenguaje C es asociativa por la derecha.
- “Puntero a” en C es asociativo por la derecha.

Ejemplo

“34 - 3 - 42” la resta asociativa con respecto a ella misma:



CFG y Asociatividad

Se establece que OP_1 es asociativo por la izquierda, esto significa que si está dos veces, se hace primero el de más a la izquierda y luego el que sigue.

- El de más a la izquierda debe quedar en un subárbol más abajo en el árbol de derivación.
- El de más a la derecha debe quedar en un subárbol más arriba en el árbol de derivación.

Ejemplo: queremos que en una serie de restas, la de más a la izquierda quede en un subárbol más abajo que las de la derecha.

Ejemplo

Considere la CFG:

$$\begin{aligned}
 \text{Exp} &\rightarrow \text{Exp ADDOP Exp} \mid \text{Term} \\
 \text{ADDOP} &\rightarrow + \mid - \\
 \text{Term} &\rightarrow \text{Term MULOP Term} \mid \text{Factor} \\
 \text{MULOP} &\rightarrow * \\
 \text{Exp} &\rightarrow (\text{Exp}) \mid \langle \text{número} \rangle
 \end{aligned}$$

Esta gramática no establece la asociatividad izquierda de la multiplicación, suma y resta.

Modificación:

```

Exp → Exp ADDOP Term | Term
ADDOP → + | -
Term → Term MULOP Factor | Factor
MULOP → *
Factor → (Exp) | <número>
    
```

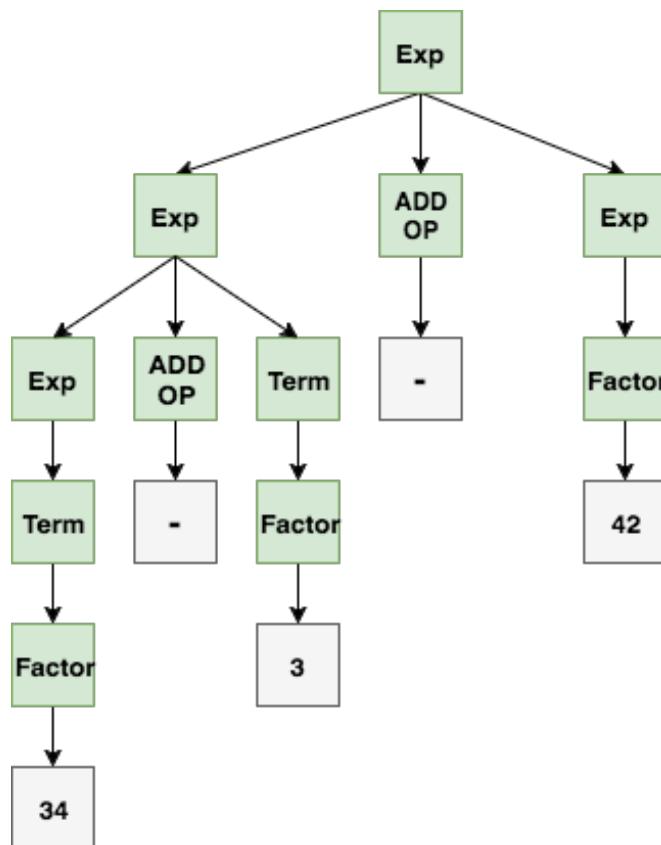
Ejemplo

Considere la CFG:

```

Exp → Exp ADDOP Term | Term
ADDOP → + | -
Term → Term MULOP Factor | Factor
MULOP → *
Factor → (Exp) | <número>
    
```

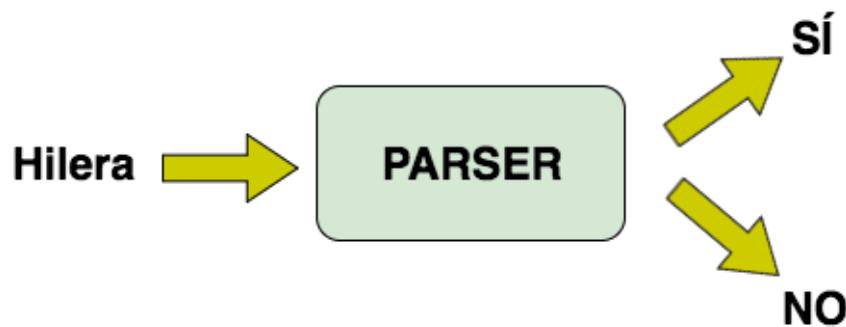
El árbol de derivación con respecto a la anterior CFG para la hilera “34 - 3 - 42” correspondería a:



Parsing

Una CFG **GENERA** un CFL, y un parser **RECONOCE** o detecta un CFL.

Se le presenta una hilera, y dice si es parte del lenguaje, o no. Si esta es parte del lenguaje, se forma un **árbol de derivación** (explícita o implícitamente).

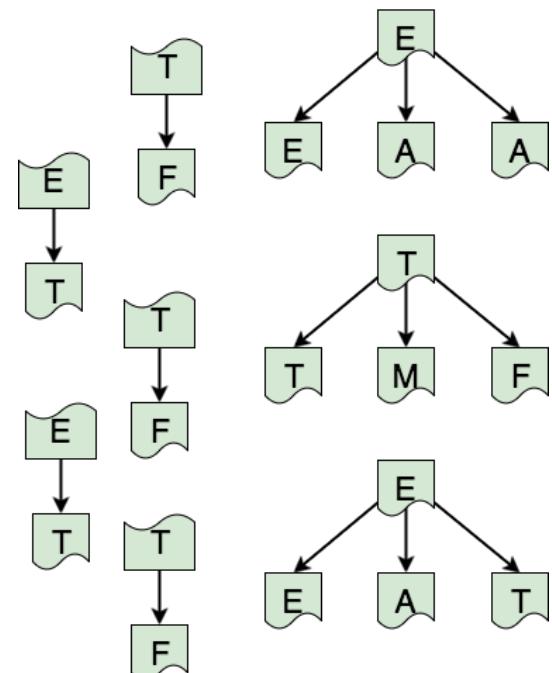


Juego del Parsing

Nos dan una CFG:

$$\begin{aligned}
 E &\rightarrow EAT \mid T \\
 A &\rightarrow + \mid - \\
 T &\rightarrow TMF \mid F \\
 M &\rightarrow * \\
 F &\rightarrow (E) \mid #
 \end{aligned}$$

- Hay cantidades infinitas de piezas.
- Corresponden a las reglas de la gramática.
- Calzan unas con otras.
- Las reparte Mauricio Avilés.



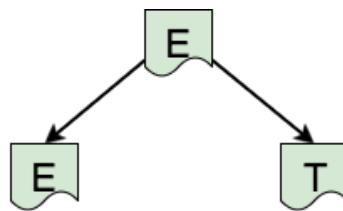
Dinámica

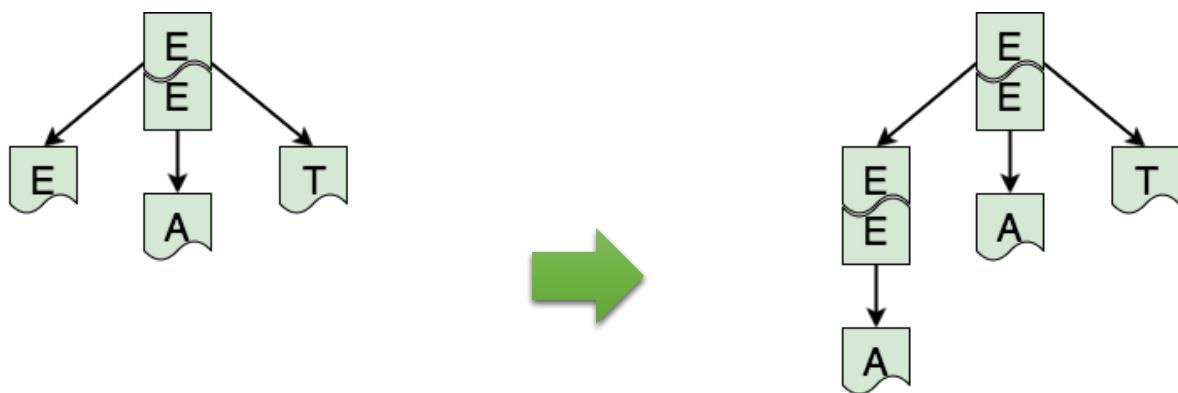
- Nos dan una hilera que hay que parsear con la gramática.
- Se pone en la parte superior de la mesa el símbolo inicial de la CFG.
- Usando piezas apropiadas, se debe conectar de manera perfecta el símbolo inicial con todos los tokens de la hilera.
- Si se puede hacer, la hilera pertenece al lenguaje.
- Si no se puede, hay **error sintáctico**.



Desarrollo

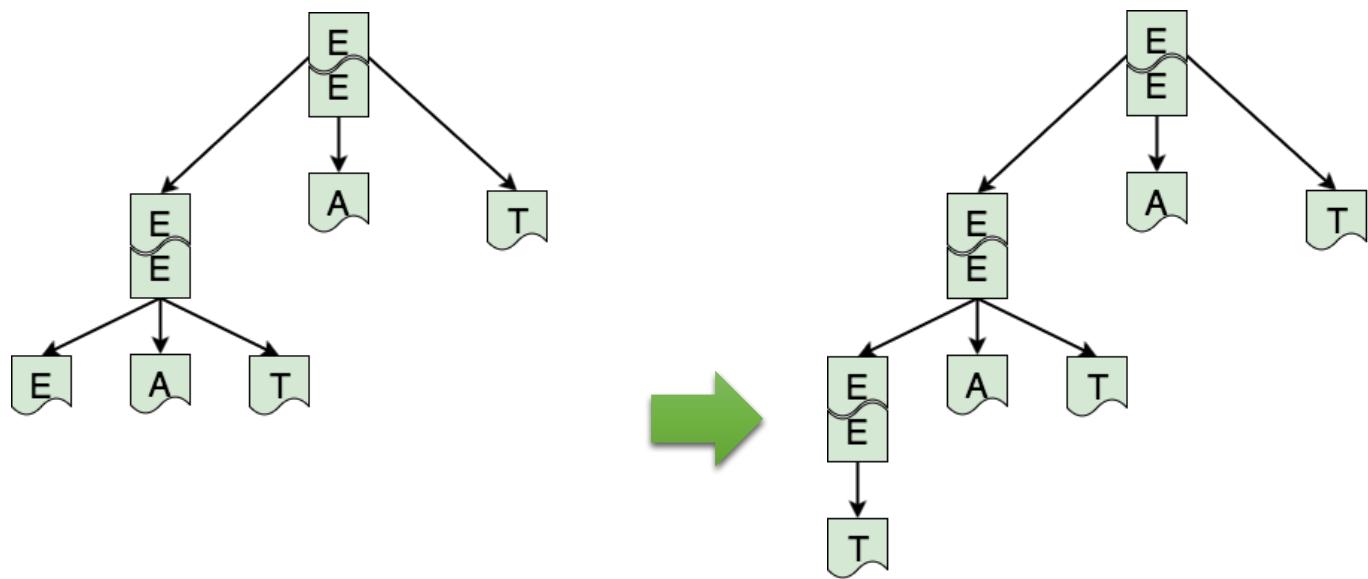
Tenemos que acomodar las piezas que nos dieron anteriormente, para formar el árbol de derivación adecuado para la hilera que nos han dado, en este caso “5 + 2 + 4”:





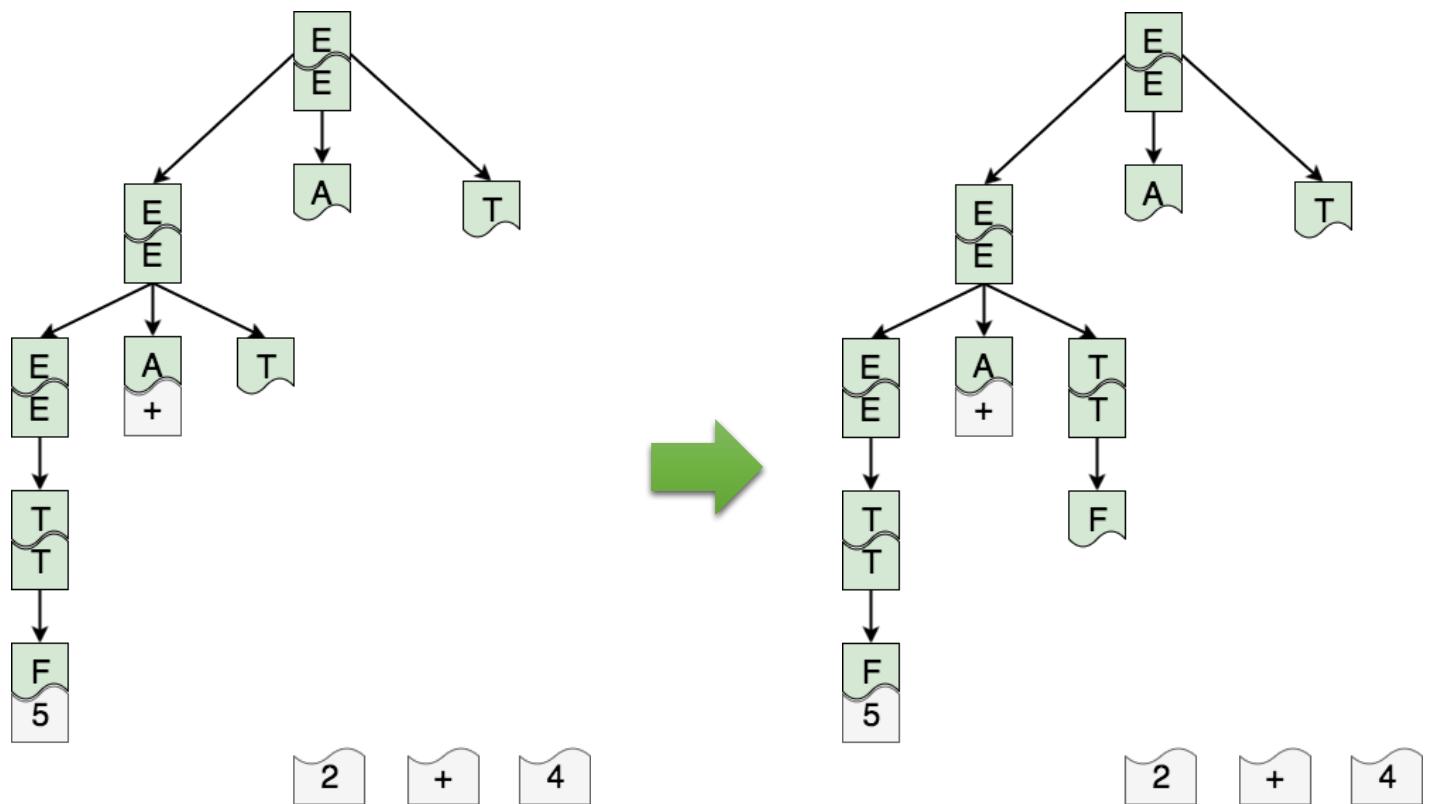
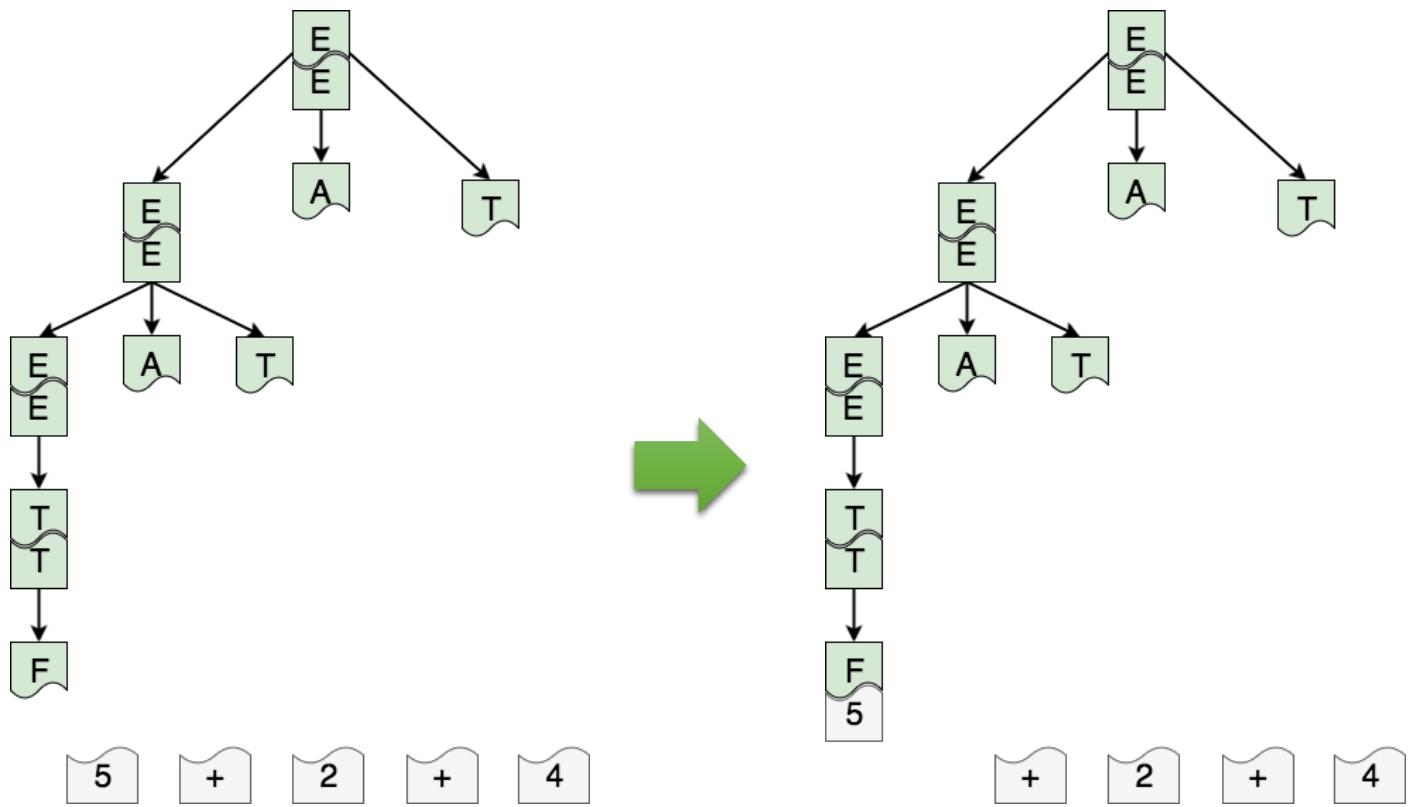
5 + 2 + 4

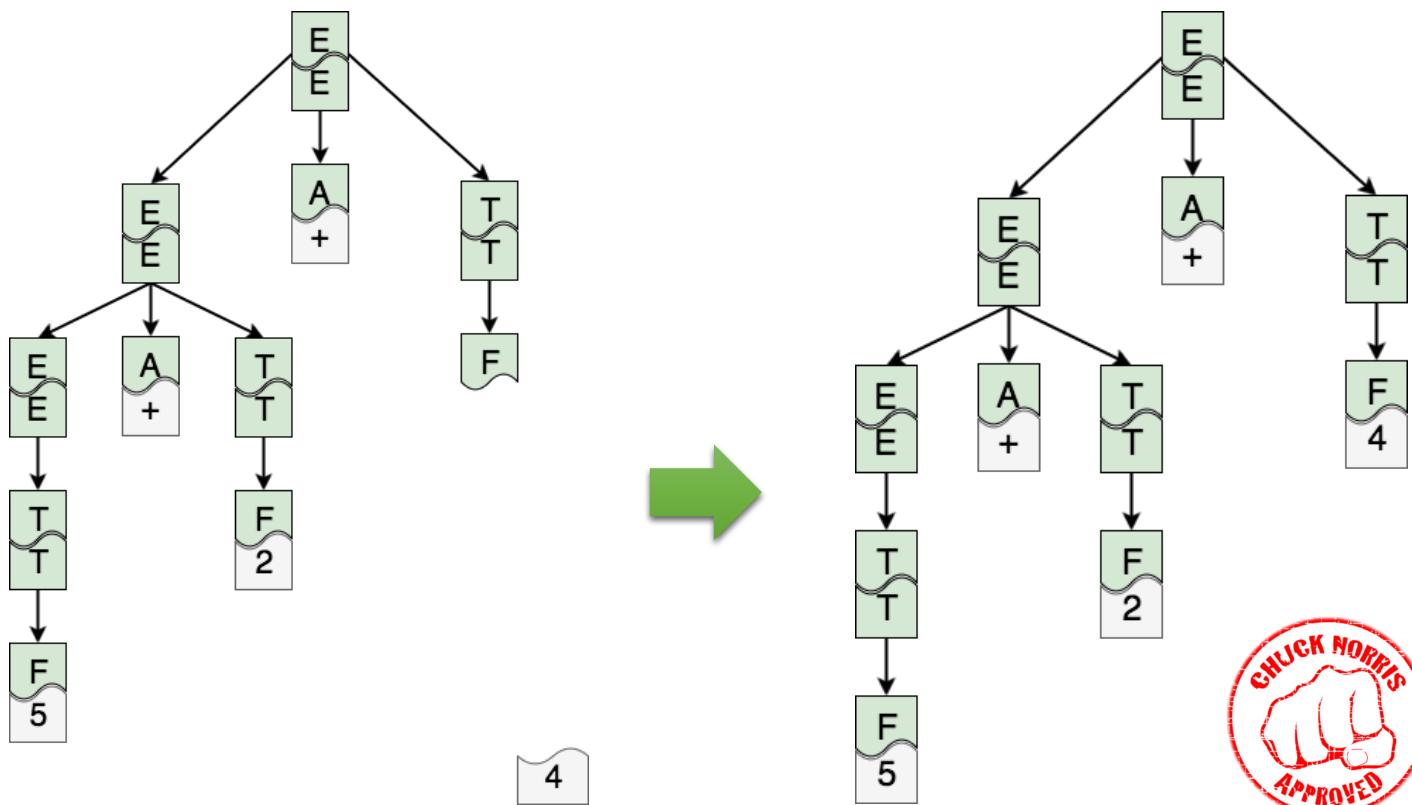
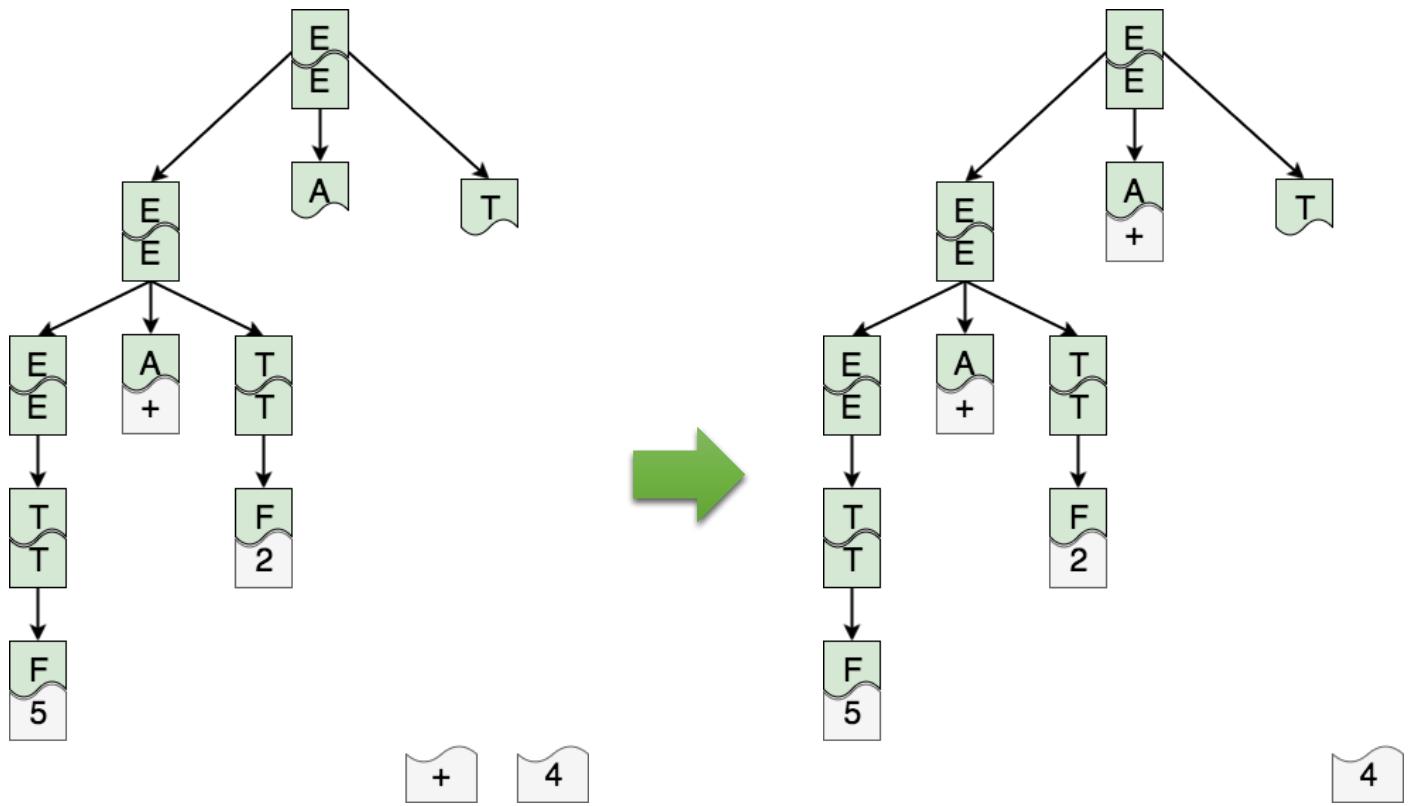
5 + 2 + 4



5 + 2 + 4

5 + 2 + 4





¿Cómo Jugar al Parsing?

¿En qué dirección nos movemos?

- Del símbolo inicial hacia la hilera: TOP DOWN.
- De la hilera hacia el símbolo inicial: BOTTOM UP.

¿Cuál pieza (regla) se debe utilizar?

- La correcta :D

¿Cómo sabemos cuál pieza usar?

- Viendo el No Terminal abierto más izquierdo.
- Viendo el siguiente símbolo de la hilera (*lookahead*).

¿Podemos decidir con precisión cuál regla usar?

- **Parsing Predictivo.**

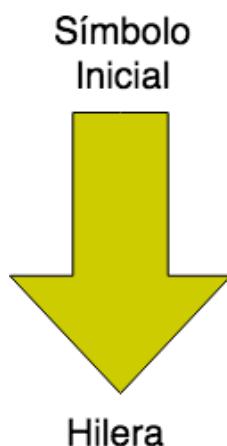
NO se puede utilizar BACKTRACKING

Análisis Sintáctico - Parsing Predictivo

Dirección del Parsing

Se comienza por el símbolo inicial y se debe llegar a la hilera analizada.

Top Down Parsing:



Predicciones

- En cada paso hay que escoger cuál regla usar, esta regla tiene que ser la correcta.
- Con *Leftmost Derivation* siempre se sabe cuál es el no terminal actual más a la izquierda, pero pueden haber muchos otros.
- Ya sabemos el lado izquierdo de la regla a usar.
- Vemos los primeros k tokens de la hilera.
 - Lookahead de k tokens.
- *Parsing Predictivo*: con k tokens de lookahead y el no terminal actual se sabe exactamente cuál regla usar o se reporta error de sintaxis.
- ¿Qué sucede con un $k = 1$?

Elementos del Parsing Predictivo

CFG

$$\begin{aligned} E &\rightarrow EA \cup T \\ A &\rightarrow + \cup - \\ T &\rightarrow TMF \cup F \\ M &\rightarrow * \\ F &\rightarrow (E) \cup \# \end{aligned}$$

Stack



Hilera (con un lookahead de 1)

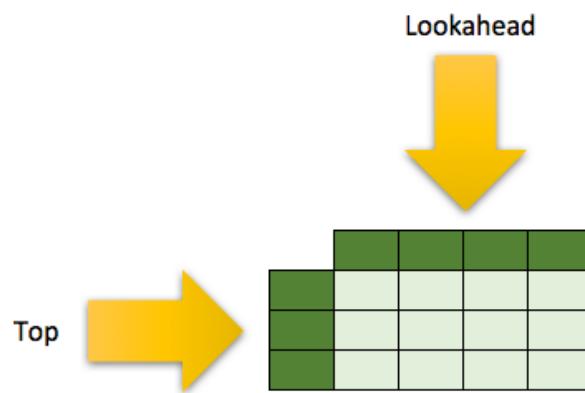
34 - 3 * 42

Tabla de Acciones

	(#)	+	-	*	\$
E	1	1					
E'			3	2	2		3
OP				4	5		
T	6	6					
T'			9	9	9	7	9
M						8	
F	10	11					

Tabla de Acciones

- Las filas de la tabla están subindexadas con el top de la pila.
- Las columnas están subindexadas con el *lookahead* actual.
- En la entrada correspondiente de la tabla se indica la acción a ejecutar.
- Hay algoritmos para generar esta tabla automáticamente.



Algoritmo de Parsing Predictivo

1. Agregar “\$” al final de la hilera a ser parseada.
2. Hacer un *push* de “\$” en la pila vacía.
3. Hacer un *push* del símbolo inicial de la gramática.
4. Establecer el lookahead actual.
5. Las entradas de la tabla de acciones contienen acciones, valga la redundancia.
6. Mientras la hilera no sea aceptada o rechazada:
 - a. Realice la acción indicada en Tabla[Top][lookahead]

Acción: Aceptar

- **Aceptar** la hilera completa.
- El top de la pila es \$.
- Lookahead es \$.
- Significa que la hilera es derivable con la CFG.

Top	Lookahead			
	\$			\$
\$				Accept

Acción: Rechazar - Error

- **Rechazar** hilera completa.
- Combinación de top de la pila y *lookahead* es inaceptable.
- A veces es una entrada vacía.
- Significa que la hilera no es derivable con la CFG.

		Lookahead	
		+	
Top	E		Error

Acción: Match

- Hay un terminal en el top de la pila.
- *Lookahead* es igual a top de la pila.
- Efectos:
 - Avanzar en la entrada (siguiente Token)
 - POP de top de la pila
- A veces no se pone del todo (match implícito)

		Lookahead	
		(
Top	(Match

Acción: Expandir

- Hay un no terminal en el top de la pila.
- Entrada contiene una regla o número de regla.
- Efectos:
 - POP de top de la pila.
 - PUSH del lado derecho de la regla, de tal manera que el símbolo más a la izquierda quede en el top de la pila (podría ser ϵ)

		Lookahead	
		*	
Top	S		4

Resumen

- Accept
- Error
- Match (Pop del top, avanza siguiente token)
- Expand (Pop del top, push del lado derecho de regla)



Ejemplo 1

Se tiene la gramática:

1. $S \rightarrow (S)S$
2. $S \rightarrow \epsilon$

Hilera: “()()”

Y la tabla de acciones:

	()	\$
S	1	2	2
(match	error	error
)	error	match	error
\$	error	error	accept

Pasos a seguir:

\$ <u>S</u>	<u>L</u> ()()\$	→	Expandir regla 1
\$S)S <u>L</u>	<u>L</u> ()()\$	→	Match
\$S) <u>S</u>	(<u>L</u>)()\$	→	Expandir regla 1
\$S)S)S(<u>L</u>	(<u>L</u>)()\$	→	Match
\$S)S)S <u>L</u>	((<u>L</u>)())\$	→	Expandir regla 2
\$S)S) <u>L</u>	((<u>L</u>)())\$	→	Match
\$S) <u>S</u>	((<u>L</u>)())\$	→	Expandir regla 1
\$S)S)S) <u>L</u>	((<u>L</u>)())\$	→	Match
\$S)S) <u>S</u>	((<u>L</u>)())\$	→	Expandir regla 2
\$S)S) <u>L</u>	((<u>L</u>)())\$	→	Match
\$S) <u>S</u>	((<u>L</u>)())\$	→	Expandir regla 2
\$S) <u>L</u>	((<u>L</u>)())\$	→	Match
\$ <u>S</u>	((<u>L</u>)())\$	→	Expandir regla 2
\$	((<u>L</u>)())\$	→	Aceptar



Ejemplo 2

Se tiene la gramática:

1. $S \rightarrow (S)S$
2. $S \rightarrow \epsilon$

Hilera: "() ()"

Y la tabla de acciones:

	()	\$
S	1	2	2
(match	error	error
)	error	match	error
\$	error	error	accept

Pasos a seguir:

\$ <u>S</u>	<u>L</u>)(<u>)\$</u>	→ Expandir regla 1
\$S)S <u>L</u>	<u>L</u>)(<u>)\$</u>	→ Match
\$S) <u>S</u>	(<u>L</u>)(<u>)\$</u>	→ Expandir regla 2
\$S)	(<u>L</u>)(<u>)\$</u>	→ Match
\$ <u>S</u>	(<u>L</u>)(<u>)\$</u>	→ Expandir regla 1
\$S)S <u>L</u>	(<u>L</u>)(<u>)\$</u>	→ Match
\$S) <u>S</u>	(<u>L</u>)(<u>)\$</u>	→ Expandir regla 2
\$S)	(<u>L</u>)(<u>)\$</u>	→ Match
\$ <u>S</u>	(<u>L</u>)(<u>)\$</u>	→ Expandir regla 2
<u>S</u>	(<u>L</u>)(<u>)\$</u>	→ Aceptar

**Ejemplo 3**

Se tiene la gramática:

1. $S \rightarrow (S)S$
2. $S \rightarrow \epsilon$

Hilera: ") ()"

Y la tabla de acciones:

	()	\$
S	1	2	2
(match	error	error
)	error	match	error
\$	error	error	accept

Pasos a seguir:

\$ <u>S</u>	<u>L</u> (<u>)\$</u>	→ Expandir regla 2
<u>S</u>	<u>L</u> (<u>)\$</u>	→ Error

Ejemplo 4

Se tiene la gramática:

1. $E \rightarrow TE'$
2. $E' \rightarrow OPTE'$
3. $E' \rightarrow \epsilon$
4. $OP \rightarrow +$
5. $OP \rightarrow -$
6. $T \rightarrow FT'$
7. $T' \rightarrow MFT'$
8. $M \rightarrow *$
9. $T' \rightarrow \epsilon$
10. $F \rightarrow (E)$
11. $F \rightarrow \#$

Y la tabla de acciones:

	(#)	+	-	*	\$
E	1	1					
E'			3	2	2		3
OP				4	5		
T	6	6					
T'			9	9	9	7	9
M						8	
F	10	11					

Hilera: "3 + 4 "

Pasos a seguir:

$\$E$	$3+4\$$	\rightarrow	Expandir regla 1
$\$E'T$	$3+4\$$	\rightarrow	Expandir regla 6
$\$E'TF$	$3+4\$$	\rightarrow	Expandir regla 11
$\$E'T\#$	$3+4\$$	\rightarrow	Match
$\$E'T'$	$3+4\$$	\rightarrow	Expandir regla 9
$\$E'$	$3+4\$$	\rightarrow	Expandir regla 2
$\$E'T OP$	$3+4\$$	\rightarrow	Expandir regla 4
$\$E'T +$	$3+4\$$	\rightarrow	Match
$\$E'T$	$3+4\$$	\rightarrow	Expandir regla 6
$\$E'T'F$	$3+4\$$	\rightarrow	Expandir regla 11
$\$E'T'\#$	$3+4\$$	\rightarrow	Match
$\$E'T'$	$3+4\$$	\rightarrow	Expandir regla 9
$\$E'$	$3+4\$$	\rightarrow	Expandir regla 3
$\$$	$3+4\$$	\rightarrow	Aceptar

