Análisis Sintáctico (Parsing)

Compiladores e Interpretes

Introducción

Parsing…

- Muchos lenguajes formales no son regulares
- No pueden ser expresados completamente usando expresiones regulares o autómatas.
- El siguiente lenguaje el cual esta compuesto de un conjunto de paréntesis balanceados.

$$\{(^i)^i \mid i \geq 0\}$$

- Se esperan resultados tales como:

( )

(( ))

((( )))

…..

Parsing…

- Lo mismo sucede en expresiones tales como:

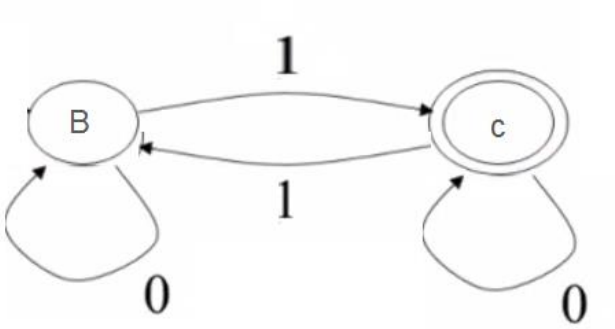
((1 + 2) * 3)

- O bien, en sentencias anidadas:

```
IF          THEN
   IF         THEN
      IF          THEN
                  ENDIF
               ENDIF
            ENDIF
```

Parsing…

- ¿Qué expresan los lenguajes regulares?



- Identifica todos los números impares basados en 1's
- No reconoce cuantas veces ha llegado al estado final
- No recuerda el largo de la hilera de entrada
- No posee un mecanismo para contar la cantidad de caracteres del string.
- Si se desea mantener el control (por ejemplo el balance de paréntesis) no se puede obtener con un conjunto finito de estados.

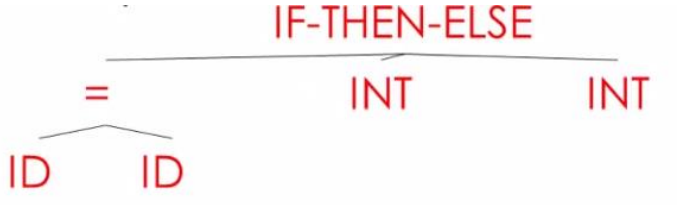
Parsing…

- ¿Qué realiza el parserador?
  - Toma la secuencia de tokens como entrada
  - Produce un árbol de parseo del programa

Entrada: if x = y then 1 else 2 fi

Entrada del Parser: IF ID = ID THEN INT ELSE INT FI

Salida del Parser:



Parsing…

- Para resumir

Fase	Entrada	Salida
Análisis Léxico	Hilera de caracteres	Hilera de tokens
Análisis Sintactico	Hilera de tokens	Árbol de parsing

- El árbol de parsing podría ser solamente implícito, osea, nunca sería construido de forma completa

Gramáticas libres de contexto

Gramáticas libre de contexto…

- No todas las hileras de tokens son programas válidos
- El parser debe distinguir entre las hileras de tokens válidas y las que no, y generar algún tipo de error.
- Es necesario un lenguaje para describir las hileras de tokens válidos
- Además, algún tipo de método que ayude a distinguir las hileras de tokens válidas de las no validas.
- Los lenguajes de programación poseen estructuras recursivas

IF **Expresion** THEN **Expresion** ELSE **Expresion**

While **Expresion** LOOP **Expresion**

Una expresión puede ser cualquier cosa de un gran número de cosas.

Gramáticas libre de contexto…

- Gramáticas libre de contexto son una notación natural para este tipo de estructura recursiva.

- Consiste en :
  - Un conjunto de terminales T
  - Un conjunto de No Terminales N
  - Un símbolo de inicio S donde (s ε N)
  - Un conjunto de producciones P

$$X \rightarrow Y_1 \ldots Y_n \quad Tq \quad X \varepsilon N \quad y \quad Y_i \varepsilon N \cup T \cup \{\varepsilon\}$$

Gramáticas libre de contexto…

- Ejemplo:

N = { S }

T = { (,) }

Dos producciones

Gramáticas libre de contexto…

- Las producciones pueden ser leídas como un tipo de reglas

$$S \rightarrow (S)$$

  - Donde sea que se utilice S podría ser reemplazado con el dato de más a la derecha, osea (S)

  - Las producciones pueden ser leídas como reglas de reemplazo de tal manera que el valor a la derecha reemplaza el valor a la izquierda.

Gramáticas libre de contexto…

1. Inicia con un string el cual es el símbolo inicial S

2. Reemplace cualquier No Terminal X en la hilera usando el lado derecho de alguna producción X → $Y_1 \ldots Y_n$

3. Repita el paseo 2 hasta que ya no encuentre No terminales y hasta que la hilera consista en solamente terminales

- Este es un paso de la derivación libre de contexto

Gramáticas libre de contexto…

- Dada G una gramática libre de contexto con símbolo inicial S, entonces el Lenguaje L(G) de G es:

$$\{ a_1 \ldots a_n \} \mid \nexists_i \; a_i \in T \wedge S \overset{*}{\rightarrow} a_i \ldots a_n \}$$

- A los terminales se les llama así pues no tienen reglas para reemplazarlos
- Una vez generados, los terminals son permanentes
- Los terminals deben ser tokens del lenguaje

Gramáticas libre de contexto…

- Ejemplo

EXPR → if EXPR then EXPR else EXPR end

| while EXPR loop EXPR wend

| id

Gramáticas libre de contexto…

- Ejemplo

id EXPR → id

if id then id else id end

while id loop id wend

if while id loop id wend then id else id

if if id then id else id end then id else id end

Gramáticas libre de contexto…

- Otro ejemplo

E → E + E

| E * E

| (E)

| id

id

id + id

Id + id * id

(id + id) * id

.
.
.

Gramáticas libre de contexto…

- Una derivación es una secuencia de producciones

$$S \rightarrow \ldots. \rightarrow \ldots \rightarrow \ldots \rightarrow \ldots \rightarrow \ldots \rightarrow$$

Inicio Producciones

- Una derivación puede ser dibujado como un árbol
  - Símbolo inicial es la raíz del árbol
  - Para una producción $X \rightarrow Y_1 \ldots Y_n$ agregue hijos $Y_1 \ldots Y_n$ al nodo X.

Gramáticas libre de contexto…

- Ejemplo
  - Gramática

$$E \rightarrow E + E \mid E * E \mid (E) \mid ID$$

  - String

id * id + id

Se va a mostrar cómo producir una derivación para la hilera y construir el árbol

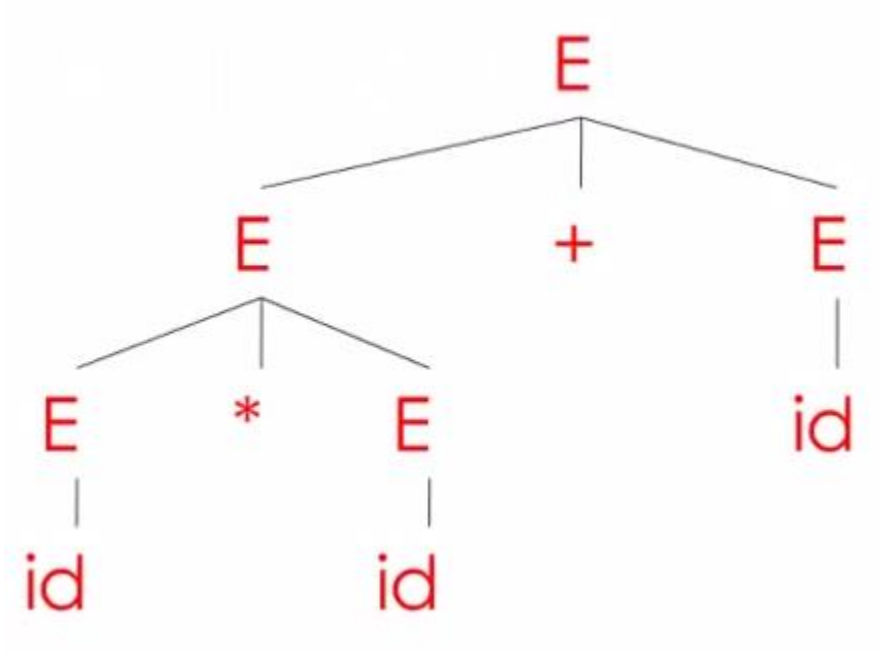
Gramáticas libre de contexto…

$$E \rightarrow E + E \mid E * E \mid (E) \mid ID$$

id * id + id

$$
\begin{aligned}
& E \\
\rightarrow\ & E + E \\
\rightarrow\ & E * E + E \\
\rightarrow\ & id * E + E \\
\rightarrow\ & id * id + E \\
\rightarrow\ & id * id + id
\end{aligned}
$$

Parse Tree



Gramáticas libre de contexto…

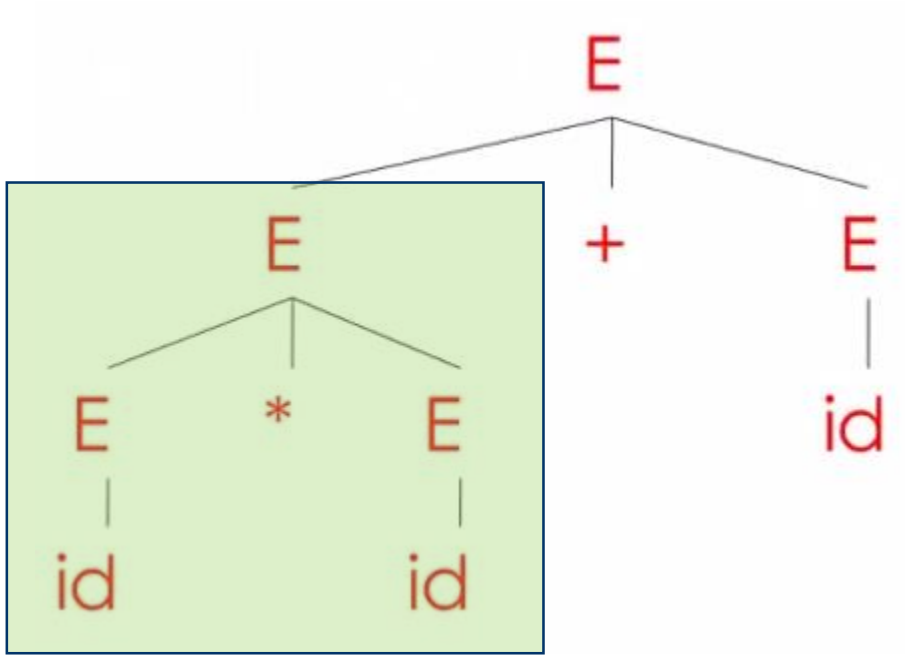
- Un árbol de parseo tiene
  - Terminales en las hojas
  - No Terminales en el interior de los nodos
- Un recorrido en el orden de las hojas es la entrada original
- El árbol de parseo muestra la asociación de las operaciones, la hilera de entrada no.

Gramáticas libre de contexto…

$$E \rightarrow E + E \mid E * E \mid (E) \mid ID$$

$$
\begin{aligned}
& E \\
\rightarrow\ & E + E \\
\rightarrow\ & E * E + E \\
\rightarrow\ & id * E + E \\
\rightarrow\ & id * id + E \\
\rightarrow\ & id * id + id
\end{aligned}
$$

Parse Tree



Subárbol.
Realiza primero la multiplicación

Gramáticas libre de contexto…

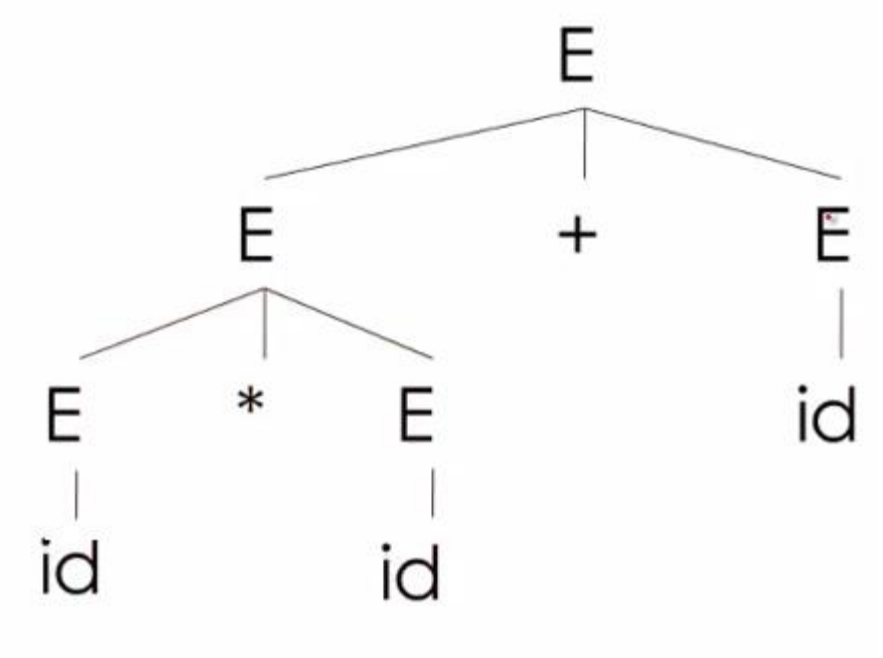
- El ejemplo anterior es una derivación “left-most” (más a la izquierda)
  - Reemplaza primero el no terminal más a la izquierda

- Existe la derivación “right-most” (más a la derecha)

$$
\begin{aligned}
& E \\
\rightarrow\ & E + E \\
\rightarrow\ & E + id \\
\rightarrow\ & E * E + id \\
\rightarrow\ & E * id + id \\
\rightarrow\ & id * id + id
\end{aligned}
$$

Gramáticas libre de contexto…

$$
\begin{aligned}
& E \\
\rightarrow\ & E + E \\
\rightarrow\ & E + id \\
\rightarrow\ & E * E + id \\
\rightarrow\ & E * id + id
\end{aligned}
$$



- Observe que es el mismo árbol de parseo que el visto anteriormente.

Gramáticas libre de contexto…

- No solamente se necesita saber que S ε L(G)
  - Es importante determinar un árbol de parseo para S

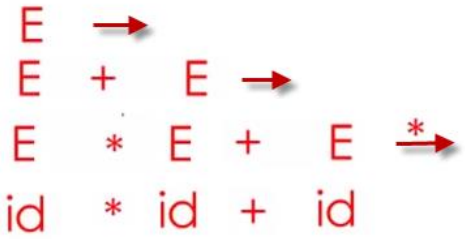
- Una derivación define un árbol de parseo
  - Pero un árbol de parseo puede tener muchas derivaciones

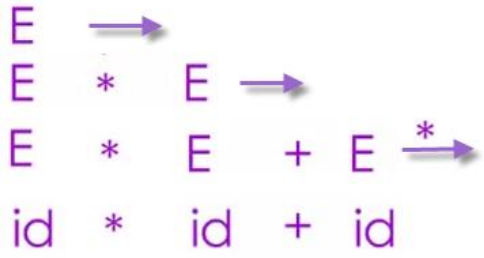
- Derivaciones “left-most” y “right-most” son importantes en la implementación del parser.

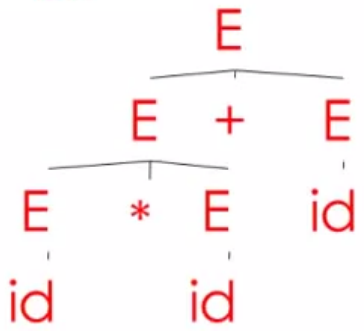
Ambiguedad

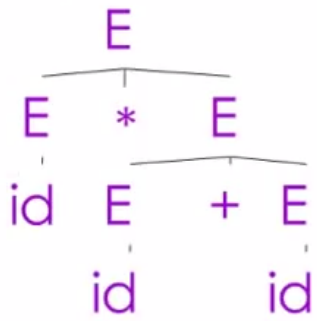
Gramáticas libre de contexto…

- Dada esta gramática: $E \rightarrow E + E \mid E * E \mid (E) \mid id$
- Dado esta hilera: $id * id + id$
- Se tienen estos dos árboles:









Gramáticas libre de contexto…

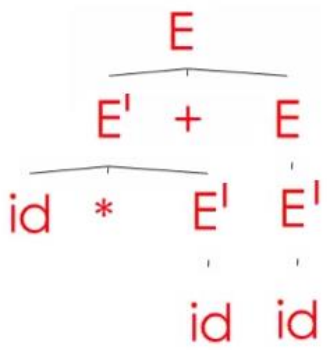
- Una gramática es ambigua si posee más de un árbol de parseo para una hilera.
  - De manera equivalente, hay más de una derivación right-most o left-most para una hilera.

- La ambigüedad NO es buena…
  - El programa tiene deficiencias pues tiene más de una manera de solucionar y generar el código

Gramáticas libre de contexto…

- Existe varias formas de manejar gramáticas ambiguas
- El método más directo es reescribirla

$$E \rightarrow E' + E \mid E'$$
$$E' \rightarrow id * E' \mid id \mid (E) * E' \mid (E)$$



- Hacer cumplir la precedencia de * sobre +

E → E' + E → E' + E' + E → E' + E' + E' + E → … → E' + … E'

E' → id * E' → id * id * E' → id * id * id * E' → … → id * … * id

(E) (E) (E) ….. (E) … (E)

Gramáticas libre de contexto…

- Considere la siguiente gramática

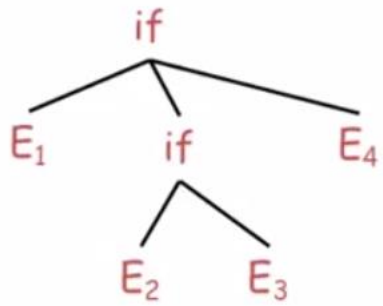
$$E \rightarrow \text{if } E \text{ then } E$$
$$\mid \text{if } E \text{ then } E \text{ else } E$$
$$\mid \text{OTHER}$$

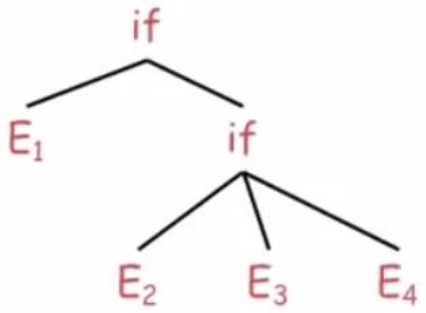
Else es opcional !!!

- La expresión

$$\text{if } E_1 \text{ then if } E_2 \text{ then } E_3 \text{ else } E_4$$

tiene dos árboles de parseo





Gramáticas libre de contexto…

- Lo que se desea es que el ELSE corresponda al THEN más cercano disponible (no asociado).

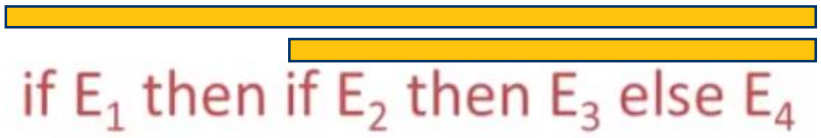
$$
\begin{aligned}
E &\rightarrow MIF \\
&\mid UIF \\
MIF &\rightarrow \text{if } E \text{ then } MIF \text{ else } MIF \\
&\mid OTHER \\
UIF &\rightarrow \text{if } E \text{ then } E \\
&\mid \text{if } E \text{ then } MIF \text{ else } UIF
\end{aligned}
$$

Gramáticas libre de contexto…

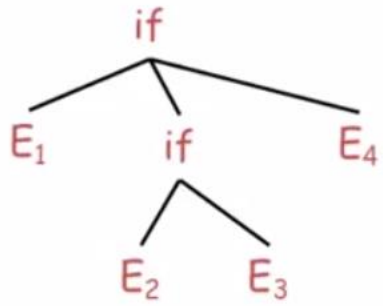
- Considerando la nueva gramática

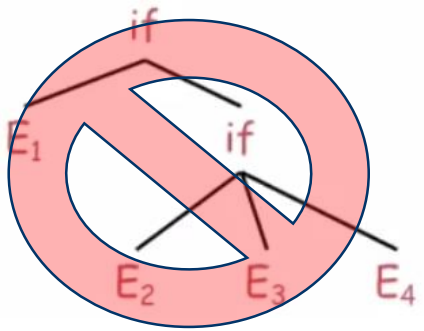
$$
\begin{aligned}
E &\rightarrow MIF \\
&\mid UIF \\
MIF &\rightarrow \text{if } E \text{ then } MIF \text{ else } MIF \\
&\mid OTHER \\
UIF &\rightarrow \text{if } E \text{ then } E \\
&\mid \text{if } E \text{ then } MIF \text{ else } UIF
\end{aligned}
$$

- La expresión



solamente queda un árbol válido





Gramáticas libre de contexto…

- No es posible determinar y “arreglar” de forma automática una gramática ambigua a una que no lo sea.

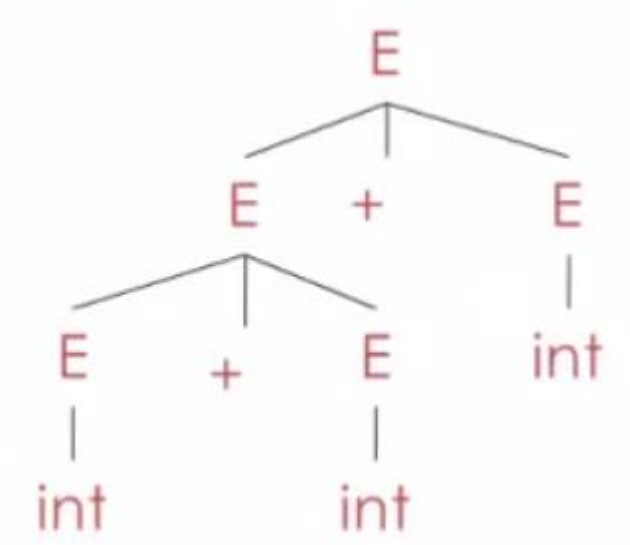
- Usada con cuidado, la ambigüedad podría simplificar la gramática
  - Podría permitir definiciones más naturales
  - Se necesitan mecanismos para trabajar con la ambigüedad y tratarla.

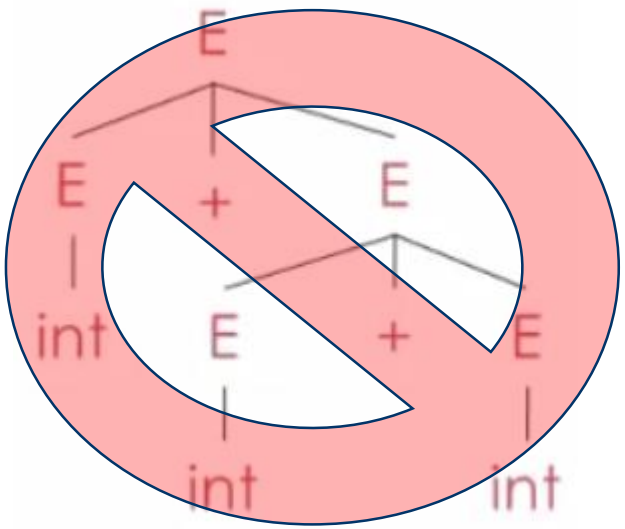
- En lugar de reescribir la gramática
  - Use la gramática más natural (ambigua)
  - En conjunto con declaraciones para tratar la ambigüedad

- La mayoría de las herramientas permiten declaraciones de precedencia y asociatividad para tratar la ambigüedad.

Gramáticas libre de contexto…

- Considere la siguiente gramática $E \rightarrow E + E \mid int$
- Ambiguedad: Dos árboles de parseo para $int + int + int$

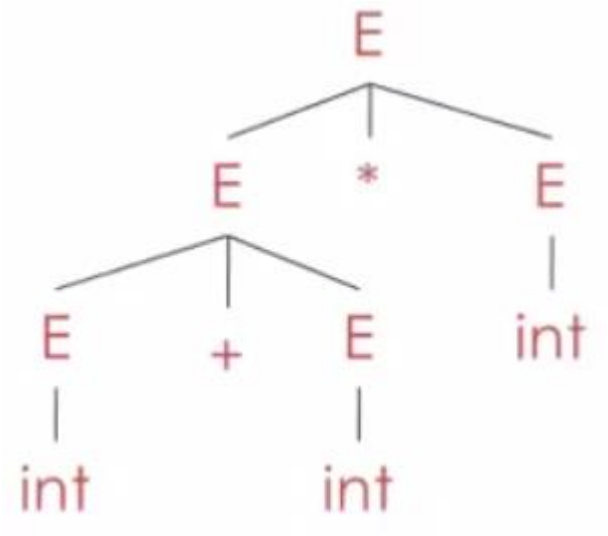


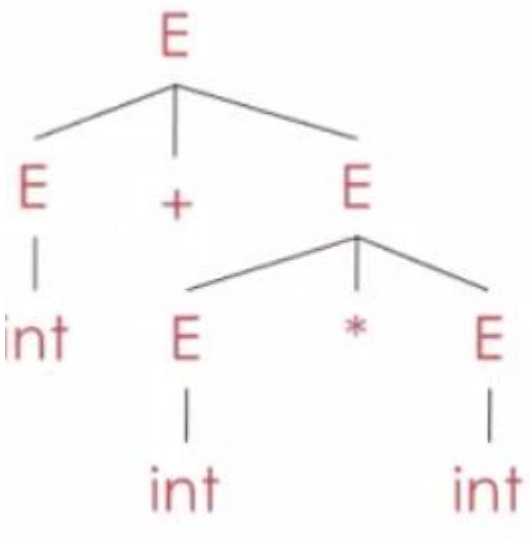


Declaración de asociatividad izquierda: `%left +` (notación usada en bison)

Gramáticas libre de contexto…

- Considere la siguiente gramática $E \rightarrow E + E \mid E * E \mid int$
- Y la hilera $int + int * int$





Declaración de precedencia: %left +

%left *

Estas reglas están fuera del árbol.

Ejercicios

Gramáticas libre de contexto…

- Ejercicio:

✓ Encuentre una gramática libre de contexto que sea completamente equivalente a la siguiente expresion regular:

$$0^*1(0+1)^*$$

✓ Encontrada la anterior gramática libre de contexto, muestre las derivaciones por la derecha y por la izquierda de los siguientes lenguajes

a. 00101
b. 1001
c. 00011

Gramáticas libre de contexto…

- Solución:

$$S \rightarrow A1B$$
$$A \rightarrow 0A \mid \varepsilon$$
$$B \rightarrow 0B \mid 1B \mid \varepsilon$$

Gramáticas libre de contexto…

- Repaso:

Gramática:

```
E → E + E
E → E * E
E → id
```

Hilera de entrada: **id + id * id**

Derivación left-most

```
E → E * E
E → E + E * E
E → id + E * E
E → id + id * E
E → id + id * id
```

Derivación right-most

```
E → E + E
E → E + E * E
E → E + E * id
E → E + id * id
E → id + id * id
```

Gramáticas libre de contexto…

- Ejercicio:

✓ Considere la siguiente gramática libre de contexto:

$$S \rightarrow aABe$$
$$A \rightarrow Abc \mid b$$
$$B \rightarrow d$$

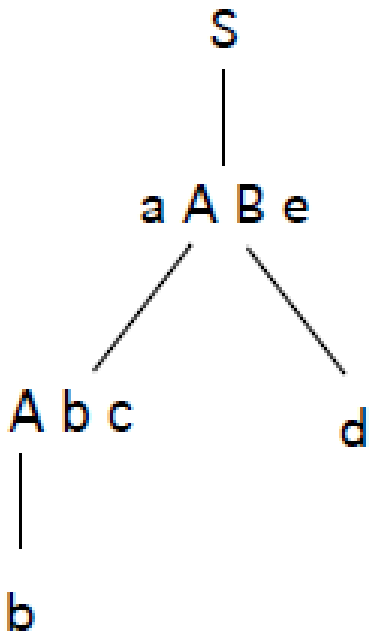
a) Parsee la hilera “abbcde” usando la derivación right-most
b) Parsee la hilera “abbcde” usando la derivación left-most
c) Dibuje el árbol de parseo

Gramáticas libre de contexto…

- Solución:

a) $S \rightarrow aABe \rightarrow aAde \rightarrow aAbcde \rightarrow abbcde$

b) $S \rightarrow aABe \rightarrow aAbcBe \rightarrow abbcBe \rightarrow abbcde$



Gramáticas libre de contexto…

- Ejercicio:

✓ Considere la siguiente gramática libre de contexto:

stmt	→	NIL | stmt ';' stmt | ifstmt | whilestmt | stmt
ifstmt	→	IF bexpr THEN stmt ELSE stmt
	→	IF bexpr THEN stmt
whilestmt	→	WHILE bexpr DO stmt

Bexpr es un NoTerminal no se coloca pero no es necesario para los objetivos del ejercicio.

Para esta gramática conteste:

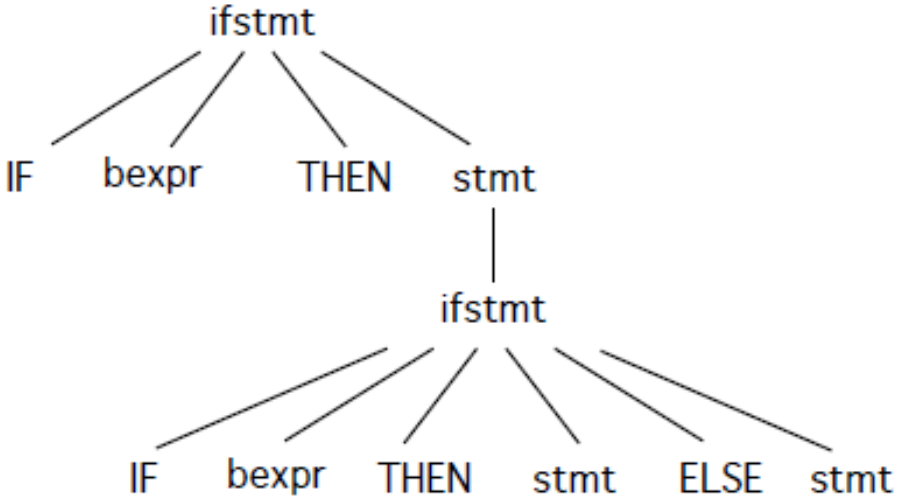
a) ¿la gramática anterior es ambigua? ¿eso genera un problema? Derive mostrando ese comportamiento con un ejemplo.

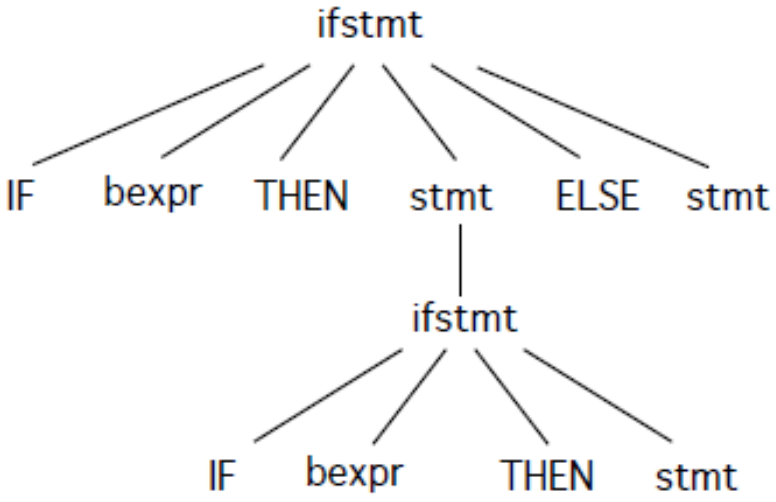
b) Diseñe una gramática equivalente no ambigua

Gramáticas libre de contexto…

- Si es ambigua. Ejemplo:

IF (a < 0) THEN IF (b > 1) THEN ... ELSE ...

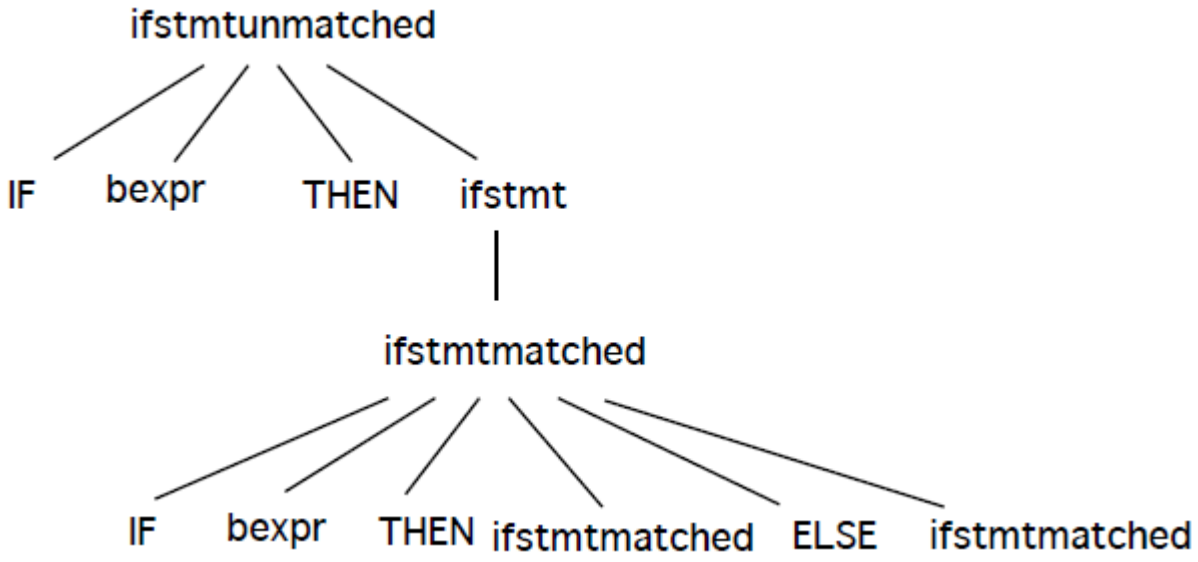




Gramáticas libre de contexto…

- Propuesta del ejercicio:

ifstmt → ifstmtmatched | ifstmtunmatched
ifstmtmatched → IF bexpr THEN ifstmtmatched ELSE ifstmtmatched
→ …
ifstmtunmatched → IF bexpr THEN ifstmt
→ IF bexpr THEN ifstmtmatched ELSE ifstmtunmatched



Análisis Sintáctico (Parsing)

Compiladores e Interpretes