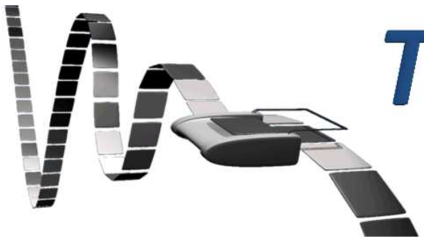
Instituto Politécnico Nacional

Escuela Superior de Cómputo



Teoría computacional

Clase 13: **Derivación de gramáticas y ambigüedad**

Solicitado: **Ejercicios 11: Derivaciones de gramáticas y ambigüedad**

M. en C. Edgardo Adrián Franco Martínez
http://computacion.cs.cinvestav.mx/~efranco
@efranco_escom
edfrancom@ipn.mx

Contenido

- Derivación
  - Ejemplo 01
  - Ejemplo 02
- Derivaciones
- Árboles de derivación
  - Ejemplo 03
  - Ejemplo 04
- Ambigüedad en gramáticas
- Problemas de la ambigüedad
- Eliminar la ambigüedad en gramáticas
- Ejercicios 11: Derivaciones de gramáticas y ambigüedad

Derivación

- **Definición:** Decimos que la cadena w_1 ***deriva en un paso*** a la cadena w_2 ($w_1 \Rightarrow_G w_2$) si y solo si existen cadenas x, $y \in \mathbf{V}^*$ tales que $w_1 = xuy$ y $w_2 = xvy$ y además existe una regla $u \rightarrow v$ en **P**.
  - Se acostumbra omitir el subíndice que indica la gramática G.

- **Definición:** una cadena $w \in \mathbf{V}^*$ es ***derivable*** a partir de la gramática G si y solo si existe una secuencia de derivación iniciando en el símbolo inicial y terminando en la cadena w: $S = w_1 \Rightarrow w_2 \Rightarrow w_3 \Rightarrow \ldots \Rightarrow w_n = w$.

  - Escribimos $\alpha \Rightarrow_G \beta$ si α deriva a β en 0 o más pasos.

- **Definición:** el ***lenguaje generado*** por una gramática *G*, *L*(*G*), es igual al **conjunto de las palabras en VT derivables a partir de *G***.

  - Una gramática describe las reglas sintácticas del lenguaje. Si una palabra no sigue las reglas, entonces no pertenecen al lenguaje generado por la gramática.

G = (VN,VT, S, P)
VN= {S, A, B}
VT = {a, b}

P:
$S \rightarrow AA$
$A \rightarrow AAA \mid bA \mid Ab \mid a \mid B$
$B \rightarrow ABAA \mid b$

Ejemplo 01

- Demostrar la pertenencia de la palabra “aabba” al L(G).

G = (VN,VT, S, P)
VN= {S, A,B}
VT = {a, b}

P:
$S \rightarrow AA$
$A \rightarrow AAA \mid bA \mid Ab \mid a \mid B$
$B \rightarrow ABAA \mid b$

Tabla de derivaciones		
Cadena	**Producción**	**Derivación**
S	$S \rightarrow AA$	AA
AA	$A \rightarrow AAA$	AAAA
AAAA	$A \rightarrow a$	aAAA
aAAA	$A \rightarrow a$	aaAA
aaAA	$A \rightarrow bA$	aabAA
aabAA	$A \rightarrow B$	aabBA
aabBA	$B \rightarrow b$	aabbA
aabbA	$A \rightarrow a$	aabba

Ejemplo 02

- **G = (VN,VT, S, P)**
- VN= {S, A}
- VT = {a, b}
- P: $S \rightarrow AA$ $A \rightarrow AAA \mid bA \mid Ab \mid a$

  - Derivar la palabra ababaa.

Derivaciones de *ababaa*

$$
\begin{array}{lll}
S \Rightarrow AA & S \Rightarrow AA & S \Rightarrow AA \\
\Rightarrow AAAA & \Rightarrow Aa & \Rightarrow aA \\
\Rightarrow aAAA & \Rightarrow AAAa & \Rightarrow aAAA \\
\Rightarrow abAAA & \Rightarrow AAbAa & \Rightarrow aAAa \\
\Rightarrow abaAA & \Rightarrow AAbaa & \Rightarrow abAAa \\
\Rightarrow ababAA & \Rightarrow AbAbaa & \Rightarrow abAbAa \\
\Rightarrow ababaA & \Rightarrow Ababaa & \Rightarrow ababAa \\
\Rightarrow ababaa & \Rightarrow ababaa & \Rightarrow ababaa
\end{array}
$$

En esta gramática existe más de una derivación de la palabra ababaa.

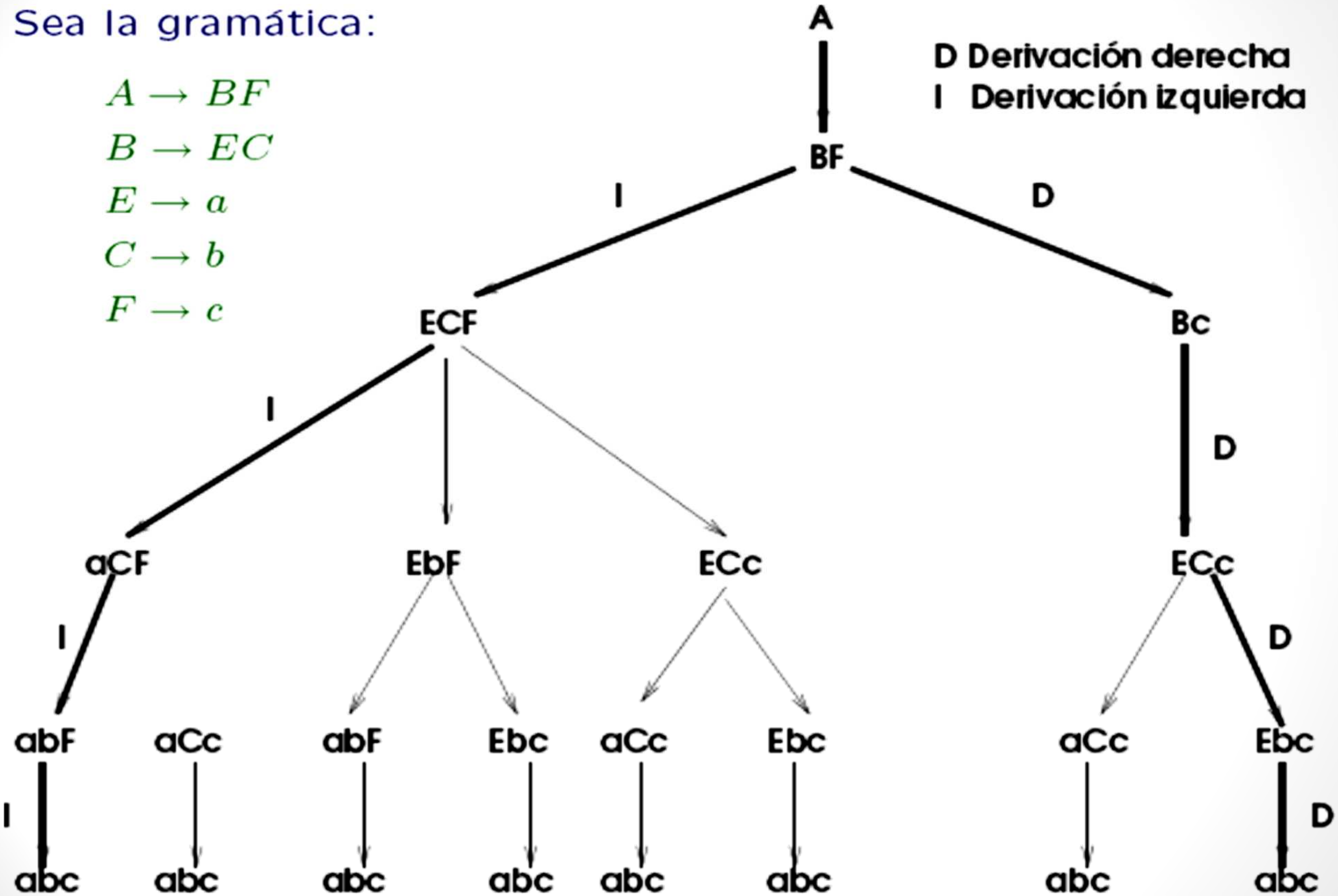
Derivaciones

- **Derivación por la izquierda**: las reglas de reemplazo son aplicadas a la primera variable de izquierda a derecha.

- **Derivación por la derecha**: las reglas de reemplazo son aplicadas a la última variable de derecha a izquierda.

Sea la gramática:

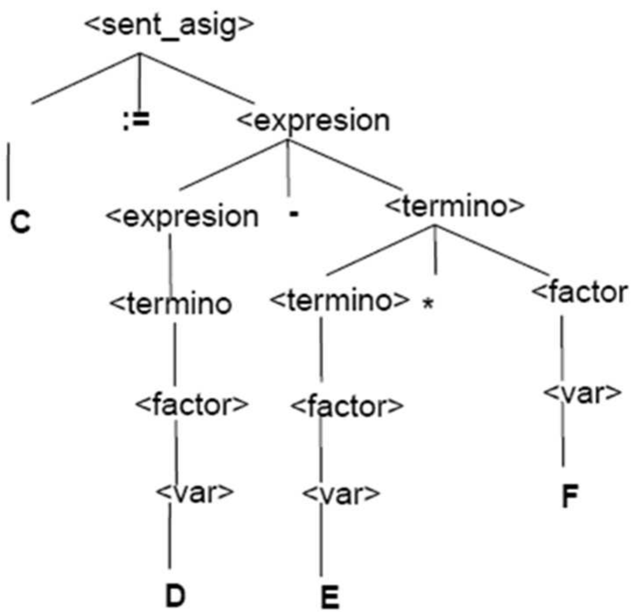
$$A \rightarrow BF$$
$$B \rightarrow EC$$
$$E \rightarrow a$$
$$C \rightarrow b$$
$$F \rightarrow c$$



Árboles de derivación

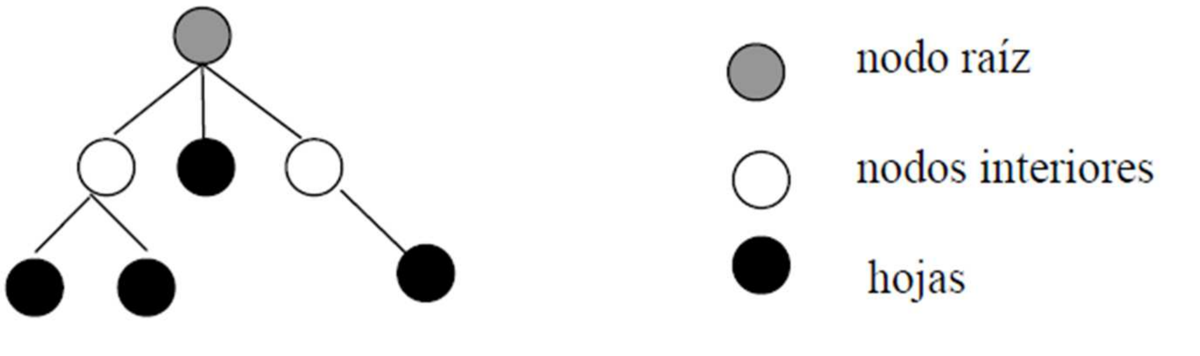
- Son una forma de representar las derivaciones, siendo utilizados, por ejemplo, en la construcción de compiladores para representar el análisis sintáctico de los programas fuente y sirviendo de base para la generación de código.

- **Sólo se pueden definir árboles de derivación para gramáticas de tipo 1 o mas restrictivas (tipos 2 y 3).**



- Un **árbol de derivación** permite mostrar gráficamente cómo se puede derivar cualquier cadena de un lenguaje a partir del símbolo inicial de una gramática que genera ese lenguaje.

- Un árbol es un conjunto de puntos, llamados **nodos**, unidos por líneas, llamadas arcos. Un arco conecta dos nodos distintos. Para ser un árbol un conjunto de nodos y arcos debe satisfacer ciertas propiedades:
  - El **nodo raíz** está rotulado con el **símbolo inicial** de la gramática.
  - Cada hoja corresponde a un **símbolo terminal**.
  - Cada nodo interior corresponde a un **símbolo no terminal**.



- Para una derivación $S \Rightarrow_G^* w$ *su* ***árbol de derivación*** se construye de la siguiente manera:

  - Inicializar el árbol con la raíz *S*.

  - Si $A \rightarrow x_1x_2...x_n$ (con $x_i \in V$) es la regla de derivación aplicada a la cadena *uAv*, entonces añadir $x_1, x_2, ..., x_n$ como los hijos de *A* en el árbol.

  - Si $A \rightarrow \lambda$ es la regla de derivación aplicada a la cadena *uAv*, entonces añadir λ como hijo único de *A* en el árbol.

- Al árbol de derivación también se le llama ***árbol de análisis*** o ***árbol de análisis gramatical*** o ***árbol de análisis sintáctico***.

- El *orden* de las hojas también sigue el proceso iterativo de construcción del árbol:
  - Inicialmente sólo hay la hoja S y el orden es obvio.
  - Al utilizar la regla $A \to x_1x_2...x_n$, cada x_i se convierte en hoja y en el orden de las hojas A se reemplaza por $x_1, x_2, ..., x_n$.
  - Al utilizar la regla $A \to \lambda$, simplemente se reemplaza A por λ.
- La cadena de caracteres terminales que se obtiene al recorrer las hojas en orden se llama el *producto* del árbol.

Ejemplo 03

G=(VT,VN,E,P)

VT={a,b,c}
VN={E,C,D}

P:

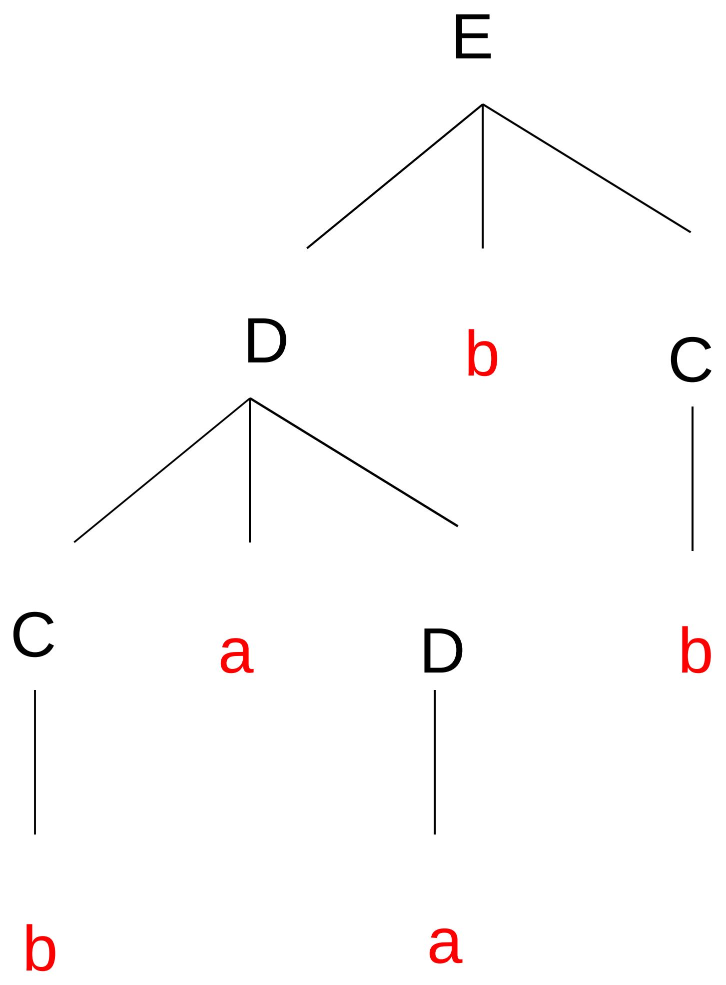
$E \rightarrow DbC$

$C \rightarrow baD$

$C \rightarrow b$

$D \rightarrow a$

$D \rightarrow CaD$



Ejemplo 04

- **P.g.** Sea la gramática G dada por:

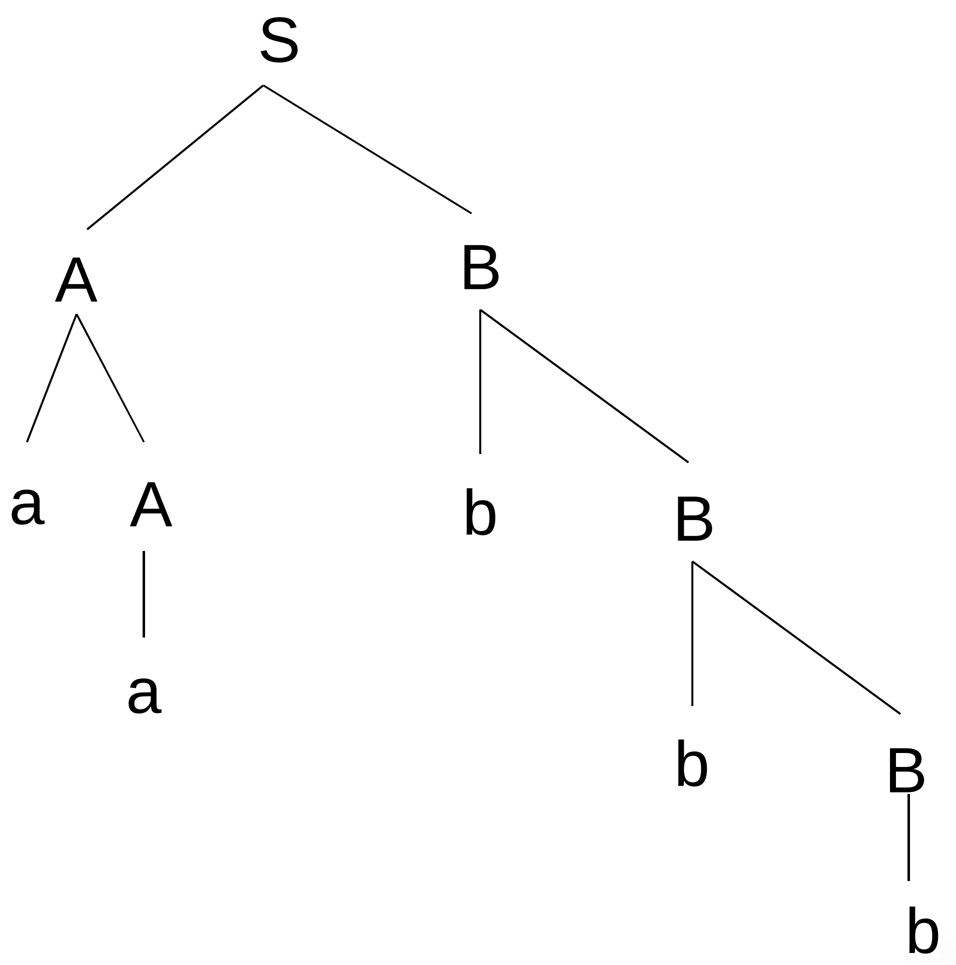
$$S \rightarrow AB$$
$$A \rightarrow aA|a$$
$$B \rightarrow bB|b$$

- La cadena *aabbb* tienen la siguiente derivación:

S => AB => AbB => AbbB => Abbb => aAbbb => aabbb

La cadena aabbb tienen la siguiente derivación:

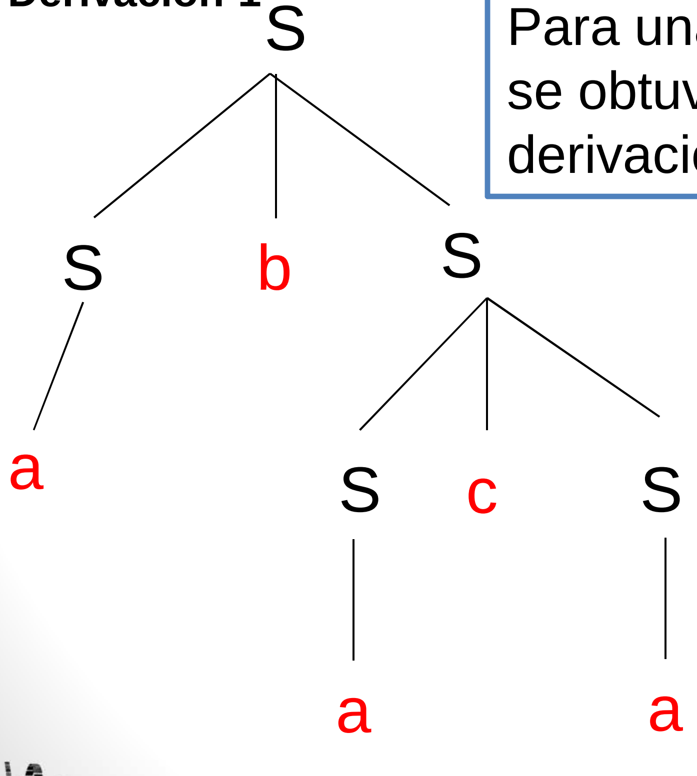
S => AB => AbB => AbbB => Abbb => aAbbb => aabbb



Ambigüedad en gramáticas

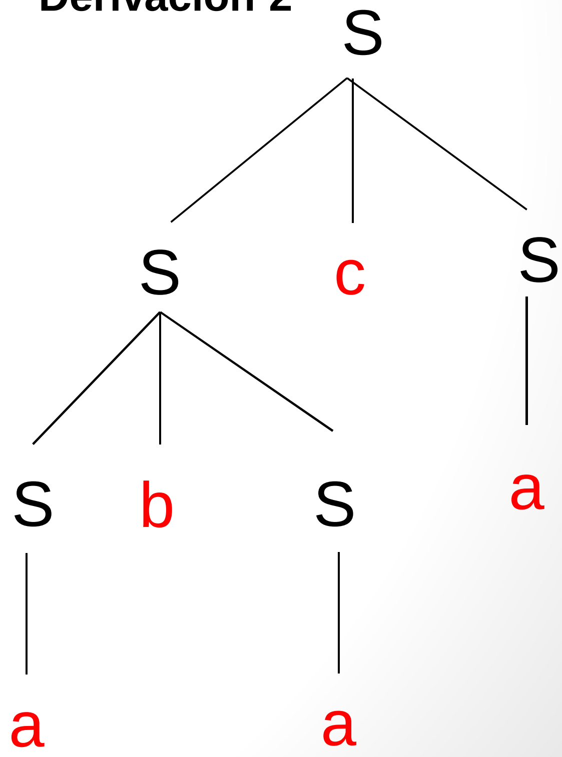
- Considérese la gramática: $S \rightarrow SbS \mid ScS \mid a$
  1. S => SbS => SbScS => SbSca => Sbaca => abaca
  2. S => ScS => SbScS => abScS => abacS => abaca

Derivación 1



Para una misma cadena, se obtuvieron árboles de derivación distintos

Derivación 2

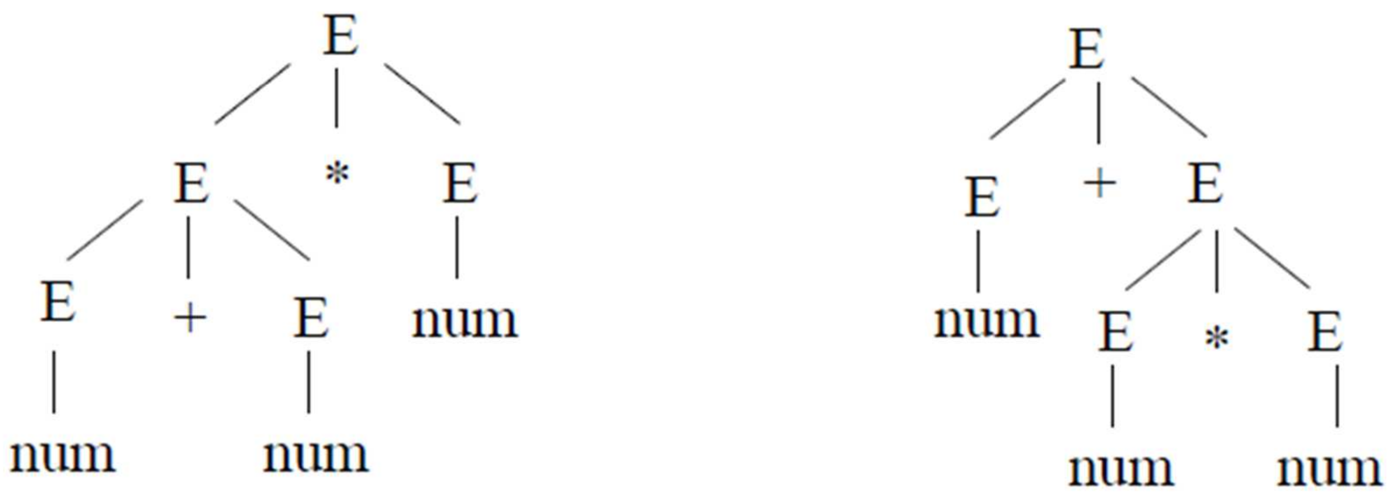


- Una **gramática es *ambigua*** cuando para una determinada cadena, se produce **más de un árbol de derivación.**

$E \rightarrow \mathbf{id} \mid \mathbf{num} \mid E + E \mid E * E \mid (E) \mid - E$

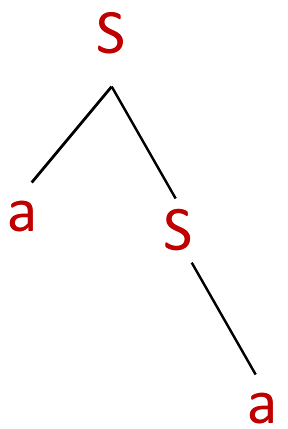
Supongamos la sentencia `id + id * id`

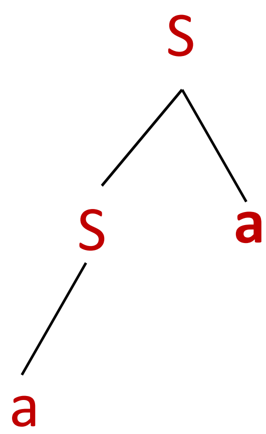
Entrada: num + num * num



- Una **gramática es *ambigua*** si existe una cadena ***w*** $\in$ **L(G)** que tiene **más de una derivación por la izquierda** o **más de una derivación por la derecha** o **si tiene dos o más árboles de derivación**.

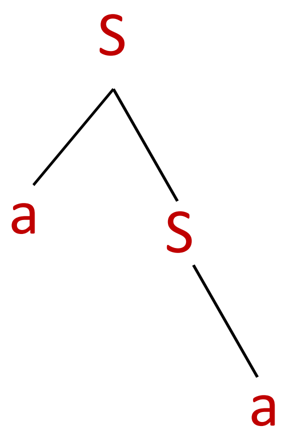
- En caso de que **toda cadena *w*** $\in$ **L(G)** tenga un **único árbol de derivación**, la **gramática es no ambigua.**

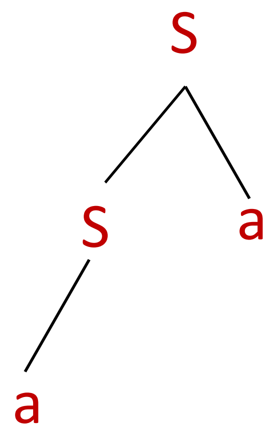




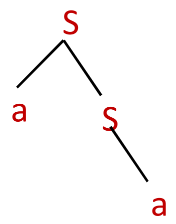
- **P.g**: la gramática $S \rightarrow aS \mid Sa \mid a$ es ambigua porque aa tiene dos derivaciones por la izquierda

$$S \Rightarrow aS \Rightarrow aa \qquad\qquad S \Rightarrow Sa \Rightarrow aa$$





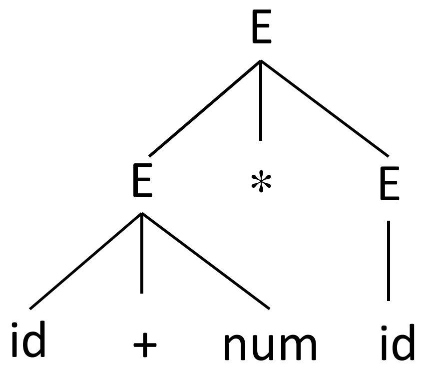
- Esta gramática genera el lenguaje a^+ que también es el lenguaje generado por la gramática no ambigua $S \rightarrow aS \mid a$.

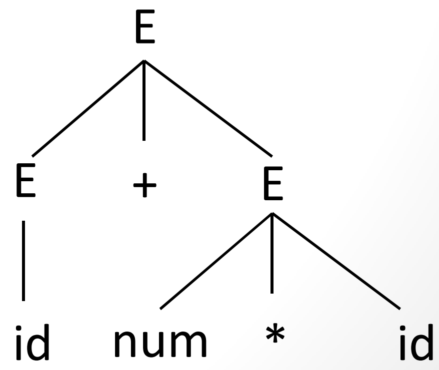


Problemas de la ambigüedad

- En la gramática para expresiones aritméticas sobre *id* y *num*:
  - $E \rightarrow E + E$
  - $E \rightarrow E * E$
  - $E \rightarrow id$
  - $E \rightarrow num$
  - $E \rightarrow -E$
  - $E \rightarrow (E)$
    - Es ambigua porque la cadena ***id + num* * *id*** **tiene dos árboles** de derivación:
    - Árbol izquierdo representa $(id + num) * id$
    - Árbol de la derecha representa $id + (\text{num} * id)$.

La ambigüedad puede producir serios problemas en lenguajes cuyo significado depende, en parte, de su estructura, como es el caso de los lenguajes naturales y los de programación.





Eliminar la ambigüedad en gramáticas

- Eliminar la ambigüedad en una gramática requiere de un **proceso de análisis propio de cada gramática** que verifique que para ninguna palabra que esta genera se puedan generar más de un árbol de derivación.

- En algunos casos, la ambigüedad de una gramática se puede eliminar **utilizando nuevas variables** que eliminen los árboles de derivación no deseados.

- También **existen los lenguajes *inherentemente ambiguos*** para los que no existe una gramática no ambigua equivalente.

- En el ejemplo de los operadores aritméticos, además de la variable **E**, que representa **expresiones**, también se utilizan las variables **F** para **factores** y **T** para **términos** y se tienen las siguientes reglas:

  - $E \rightarrow E + T$ $\quad$ $E \rightarrow T$
  - $T \rightarrow T * F$ $\quad$ $T \rightarrow F$
  - $F \rightarrow (E)$ $\quad$ $T \rightarrow -F$
  - $F \rightarrow id$ $\quad$ $F \rightarrow num$

Ejercicios 11 “Derivación de gramáticas y ambigüedad”

1. Dadas las siguientes gramáticas, de la tabla de derivaciones para comprobar la pertenecía de las cadenas dadas.

1.-

G = (VN,VT, exp, P)
VN= {exp, opsuma, term, factor,numero}
VT = {+, -,*,(,),0,1,2,3,4,5,6,7,8,9}

P:
exp $\rightarrow$ exp opsuma term |term
opsuma $\rightarrow$ + | -
term $\rightarrow$ term opmul factor | factor
opmult $\rightarrow$ *
factor $\rightarrow$ (exp) | numero
numero $\rightarrow$0|1|2|3|4|5|6|7|8|9

a) 3+4*5-6
b) 3*(4-5+6)
c) 3-(4+5*6)

2.-

G = (VN,VT, S, P)
VN= {S, A, B, C}
VT = {a, b}

P:
S $\rightarrow$ BAa
A $\rightarrow$ bBC | a
B $\rightarrow$ bB | b | λ
C $\rightarrow$ aB | aa

a) bbbaa
b) baa
c) aa

2. Dadas las siguientes gramáticas, tipo 3 encuentre la expresión regular que reconocen

1.-	2.-	3.-
$S \rightarrow aS$ $S \rightarrow \lambda$	$S \rightarrow aS \mid bA \mid \lambda$ $A \rightarrow bA \mid b \mid \lambda$	$S \rightarrow AB$ $A \rightarrow aA \mid a$ $B \rightarrow bB \mid b$

3. Dadas las gramáticas siguientes dibuje los árboles de derivación para las cadenas dadas

1.-

sentencia → sent-if | otro | λ
sen-if → if (exp) sentencia parte-else
parte-else → else sentencia | λ
exp → 0 | 1

a) if(0) if(1) otro else else otro
b) if(1) otro else if(0) otro else otro
c) if(0) otro

2.-

A → (A) A | λ | exp

a) ((exp) ((exp) (exp)))
b) (exp) (exp)
c) (exp) exp

- .

4. Demostrar que las siguientes gramáticas son ambiguas:

1.-
$S \rightarrow S + S$
$S \rightarrow S * S$
$S \rightarrow (S)$
$S \rightarrow 0\ S$
$S \rightarrow 1\ S$
$S \rightarrow 0$
$S \rightarrow 1$

2.-
sentencia → if (exp) sentencia | sent-igualada
sent-igualada → if (exp) sent-igualada else sentencia | otro
exp → 0 | 1

3.-
$S \rightarrow A\ C \mid B\ D$
$A \rightarrow a\ A\ b \mid a\ b$
$B \rightarrow a\ B \mid a$
$C \rightarrow c\ C \mid c$
$D \rightarrow b\ D\ c \mid b\ c$

Se entregarán antes del día **Domingo 20 de Octubre de 2013 (23:59:59 hora limite)*

**Incluir la redacción de cada ejercicio*

**Portada y encabezados de pagina*