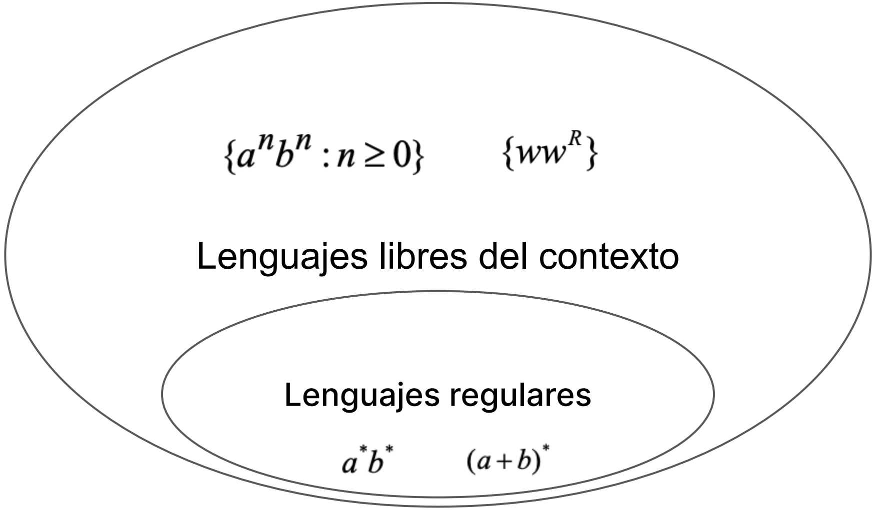
Gramáticas libres del contexto

Universidad Nacional de Quilmes

Emanuel Alvaredo, Federico Lochbaum

Overview


$\{a^n b^n : n \geq 0\}$
$\{ww^R\}$
Lenguajes libres del contexto
Lenguajes regulares
$a^* b^*$
$(a+b)^*$

¿Cómo reconocemos los LLC?

Ya vimos que existen las **Gramáticas Libres del contexto**!

Pero además, vamos a ver otra manera de reconocer LLCs

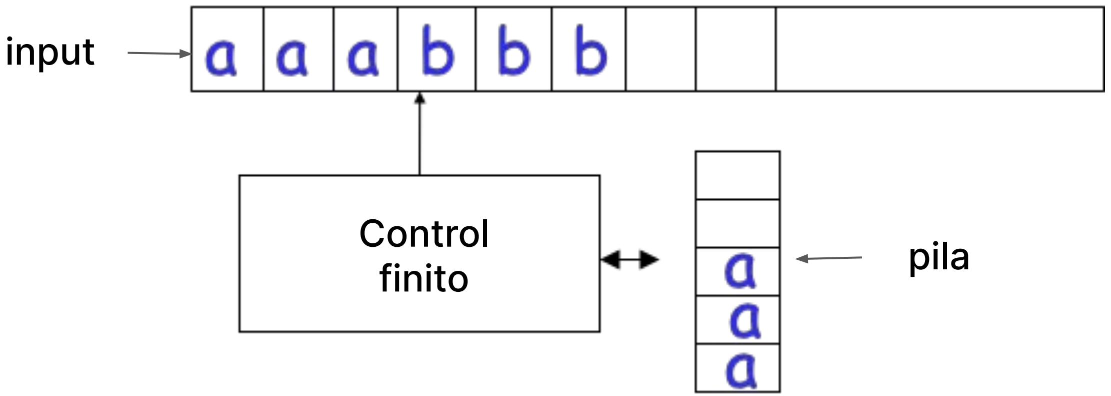
¿Cómo reconocemos los LLC?

Ya vimos que existen las **Gramáticas Libres del contexto**!

Pero además, vamos a ver otra manera de reconocer LLCs

Autómatas de pila!

Autómatas de pila



Pero… lo dejamos para la próxima clase :)

Recordamos

Una gramática es regular si es lineal a derecha o lineal a izquierda

La estructura que una gramática regular induce sobre las cadenas/strings del lenguaje **es lineal**,

es decir, es **estructuralmente** una secuencia o una lista

Ejemplo: $\boxed{\begin{array}{l} S \to abS \\ S \to a \end{array}}$ notar que la definición por inducción de los elementos de L(G) se define

$$\boxed{\begin{array}{c} w \in L(G_1) \Leftrightarrow S \overset{*}{\Rightarrow} w \\ \Leftrightarrow \\ (w = a) \vee (w = abw' \wedge S \overset{*}{\Rightarrow} w') \end{array}}$$

Definición de GLC

Notación $G=(V,\Sigma,S,P)$ donde todas las producciones P son de la forma $A \to \alpha$

No terminal **Secuencia/string de no terminales y terminales**

Ejemplo:

S → aSb | λ con $L(G)=\{a^n b^n : \ n \geq 0\}$

Lenguaje libre de contexto:

Un lenguaje L es libre de contexto si existe una gramática libre de contexto G con L(G) = L

Ejemplo:

G = S → aSa | bSb | λ L(G) = { ww^R : w ∈ { a, b}* } S ⇒ aSa ⇒ abSba ⇒ abaSaba ⇒ …

Orden de derivación y árboles de derivación

Consideremos las siguientes producciones de una gramática

$$S \Rightarrow AB \qquad A \Rightarrow aaA \mid \lambda \qquad B \Rightarrow Bb \mid \lambda$$

Vemos que la derivación *más a la izquierda* del corresponde a la cadena **aab**

$$S \overset{1}{\Rightarrow} AB \overset{2}{\Rightarrow} aaAB \overset{3}{\Rightarrow} aaB \overset{4}{\Rightarrow} aaBb \overset{5}{\Rightarrow} aab$$

Y la derivación *más a la derecha* también corresponde a la cadena **aab**

$$S \overset{1}{\Rightarrow} AB \overset{4}{\Rightarrow} ABb \overset{5}{\Rightarrow} Ab \overset{2}{\Rightarrow} aaAb \overset{3}{\Rightarrow} aab$$

Notar que la siguiente derivación no es ni más a la izquierda ni más a la derecha

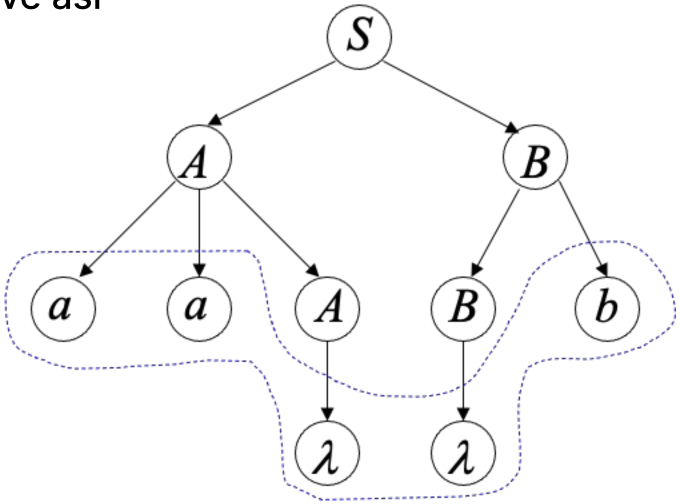
$$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaABb \Rightarrow aaBb \Rightarrow aab$$

Árbol de derivación (parse tree)

Sean las producciones $\boxed{S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda}$

El *árbol de derivación* de la siguiente derivación $S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaABb \Rightarrow aaBb \Rightarrow aab$

Se ve así



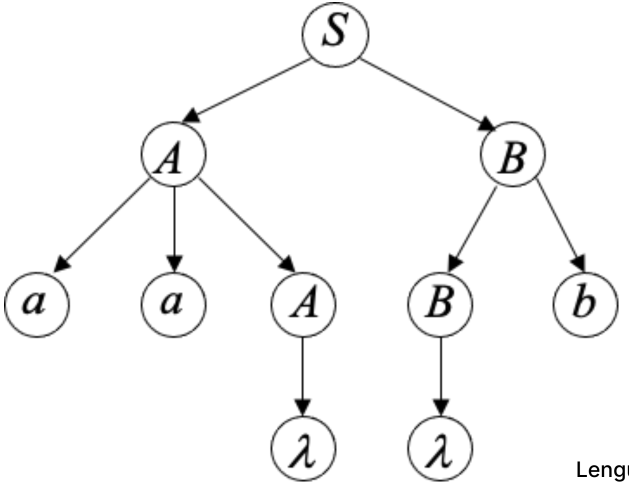
$aa\lambda\lambda b = aab$

Nota: Hay veces donde el orden no importa y la derivación a izquierda produce la misma cadena que la derivación a derecha

$$S \overset{1}{\Rightarrow} AB \overset{2}{\Rightarrow} aaAB \overset{3}{\Rightarrow} aaB \overset{4}{\Rightarrow} aaBb \overset{5}{\Rightarrow} aab$$

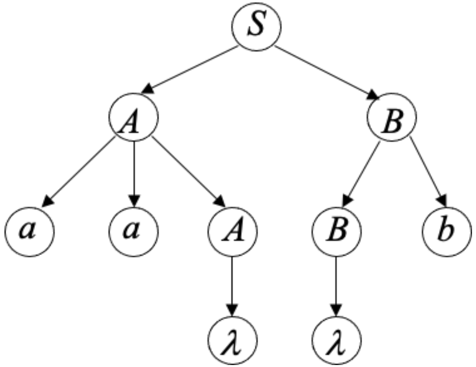
$$S \overset{1}{\Rightarrow} AB \overset{4}{\Rightarrow} ABb \overset{5}{\Rightarrow} Ab \overset{2}{\Rightarrow} aaAb \overset{3}{\Rightarrow} aab$$

Ambas producen **el mismo árbol de derivación**



Una gramática **libre de contexto** induce sobre las cadenas una estructura **arborescente**, es decir, **estructuralmente** son árboles generales

$$w \in L(G) \Leftrightarrow S \overset{*}{\Rightarrow} w$$
$$\Leftrightarrow$$
$$w = w'w''$$
$$\wedge$$
$$A \overset{*}{\Rightarrow} w' \wedge B \overset{*}{\Rightarrow} w'')$$



La estructura que una gramática libre de contexto induce sobre las cadenas/strings generadas es única?

Es decir, cada cadena tiene exactamente un único árbol de derivación?

Ambigüedad

Gramática de las expresiones aritméticas

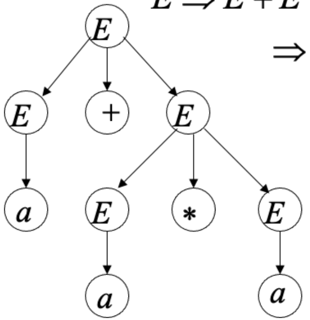
$$\boxed{E \to E+E \;\mid\; E*E \mid (E) \mid a}$$

Vemos que esta gramática acepta cadenas del estilo $(a+a)*a+(a+a*(a+a))$

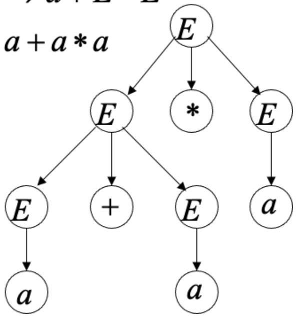
donde a denota cualquier número

Dos derivaciones a izquierda para **a + a * a**

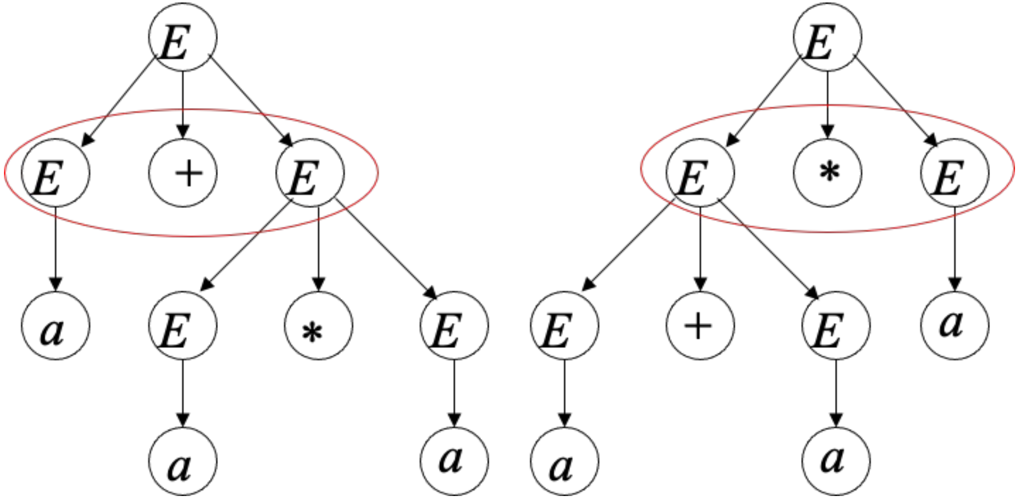
$$E \Rightarrow E+E \Rightarrow a+E \Rightarrow a+E*E$$
$$\Rightarrow a+a*E \Rightarrow a+a*a$$



$$E \Rightarrow E*E \Rightarrow E+E*E \Rightarrow a+E*E$$
$$\Rightarrow a+a*E \Rightarrow a+a*a$$



Vemos que



Tener dos árboles de derivación diferentes **puede causar problemas** si uno quiere

- Evaluar expresiones
- Hacer compiladores de lenguajes de programación

Vamos a decir que una gramática libre de contexto G es **ambigua** si existe una cadena $w \in L(G)$ tal que o bien **tiene dos árboles de derivación diferentes** o bien **dos derivaciones más a la izquierda diferentes**

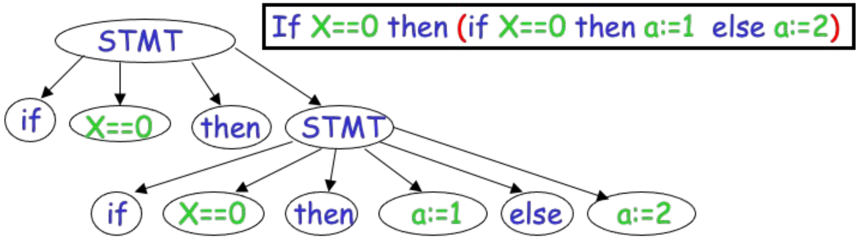
Vimos el ejemplo de las expresiones aritméticas

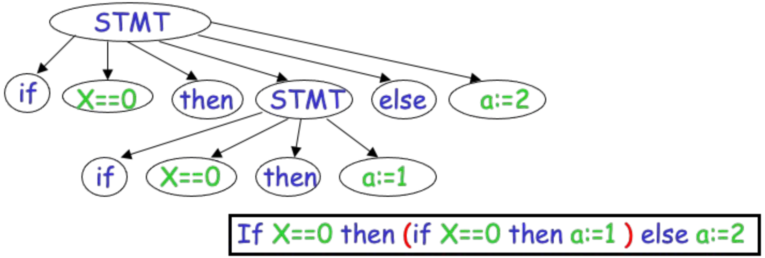
$$E \Rightarrow E+E \Rightarrow a+E \Rightarrow a+E*E$$
$$\Rightarrow a+a*E \Rightarrow a+a*a$$

$$E \Rightarrow E*E \Rightarrow E+E*E \Rightarrow a+E*E$$
$$\Rightarrow a+a*E \Rightarrow a+a*a$$

Otra gramática ambigua

STMT → if EXPR then STMT | if EXPR then STMT else STMT





En general, tener ambigüedad es malo e intentamos eliminarla

- A veces es posible encontrar una gramática no ambigua para un lenguaje
- No existe un método general para eliminar la ambigüedad

Desambiguar la gramática de expresiones aritméticas

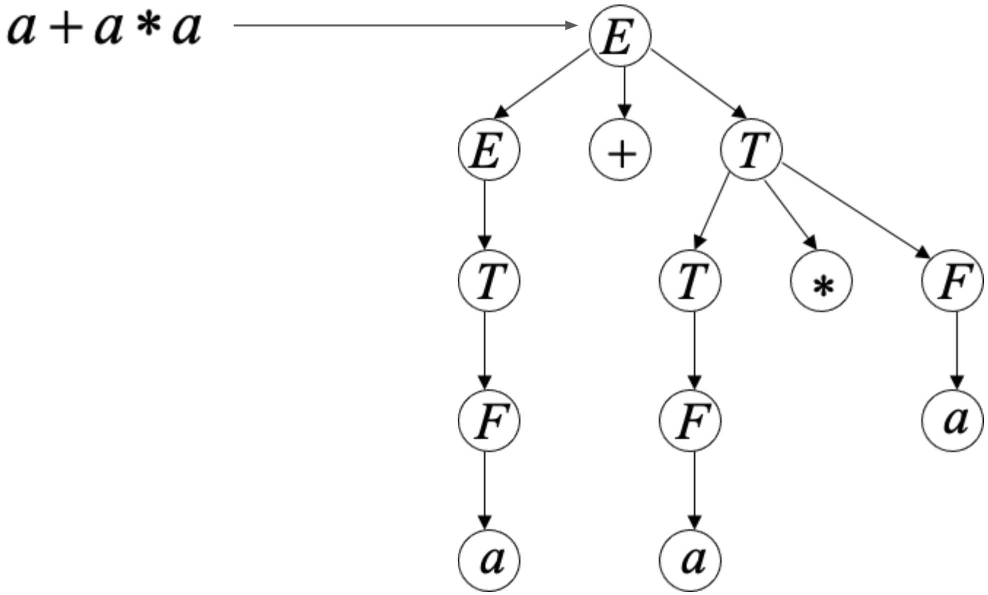
Necesitamos una forma de **precedencia** entre los operadores

$$E \to E + E$$
$$E \to E * E$$
$$E \to (E)$$
$$E \to a$$

Podemos pensar las operaciones como, una **expresión** es una suma de **términos** y un **término** un producto de **factores** y permitimos los **paréntesis** para alterar la precedencia

La nueva gramática es equivalente y no es ambigua, esta construye nueva categorías que permiten eliminar la ambigüedad forzando las alturas de las derivaciones

$$\boxed{\begin{aligned} E &\rightarrow E+T \mid T \\ T &\rightarrow T*F \mid F \\ F &\rightarrow (E) \mid a \end{aligned}}$$



Sin embargo, hay casos inherentemente ambiguos como L = { a^n b^n c^m } U { a^n b^m c^m } n, m >=0

Es decir, **toda gramática que genere este lenguaje es ambigua**

$$L = \{a^n b^n c^m\} \cup \{a^n b^m c^m\}$$

$$S \to S_1 \mid S_2 \qquad S_1 \to S_1 c \mid A \qquad S_2 \to aS_2 \mid B$$

$$A \to aAb \mid \lambda \qquad B \to bBc \mid \lambda$$