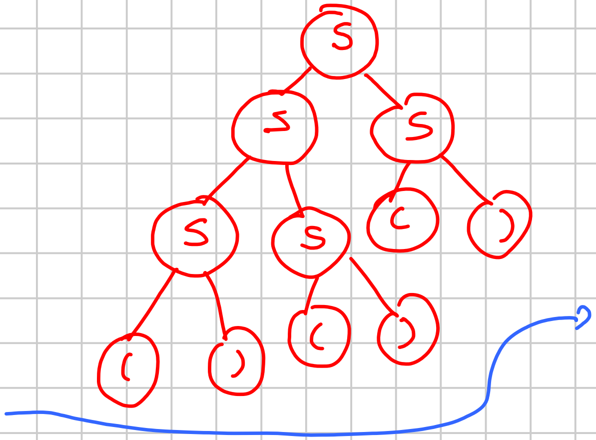
Titolo nota 28/09/2015

$$S \to () \mid (S) \mid SS$$

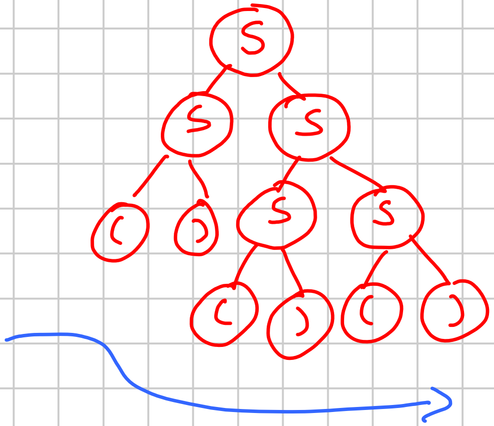
ricorsione sinistra

ricorsione destra

()()()



()()()



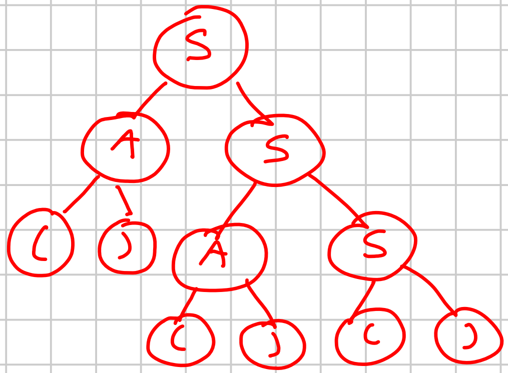
()()()

AMBIGUA

DISAMBIGUARE una grammatica significa trovarne una equivalente NON AMBIGUA

In generale non è possibile

$$S \rightarrow () \mid (S) \mid AS$$
$$A \rightarrow () \mid (S)$$



$()()()$

$$\underbrace{()}_{A}\;\underbrace{()()}_{S}$$

Disambiguazione non è un procedimento meccanico

Titolo nota 28/09/2015

Modi per definire la sintassi dei linguaggi di programmazione

Com → Ide = Exp ; | { Com_list } | if (Exp) Com else Com | while (Exp) Com

Ide → Letter | Letter Seq

Seq → Cifra | Letter | Cifra Seq | Letter Seq

Cifra → ∅ | 1 | .... | 9

Lettere → a | b | ... | z | A | B | ... | Z

Exp → Num | Exp + Exp | Exp * Exp | ...

Num → Cifra | Cifra Num

Com_list → Com | Com Com_list

x = 5+3 ;

?

$\Lambda = \{ =, ;, ∅, 1, \ldots, 9, a, b, \ldots, z, A, B, \ldots, Z, +, *, \{, \}, (,), if, else, while \}$

$Exp \rightarrow Num \mid Exp + Exp \mid Exp * Exp$

$Num \rightarrow Cifra \mid Cifra\ Num$

$Cifra \rightarrow 0 \mid 1 \mid \dots \mid 9$

doppie ricorsioni

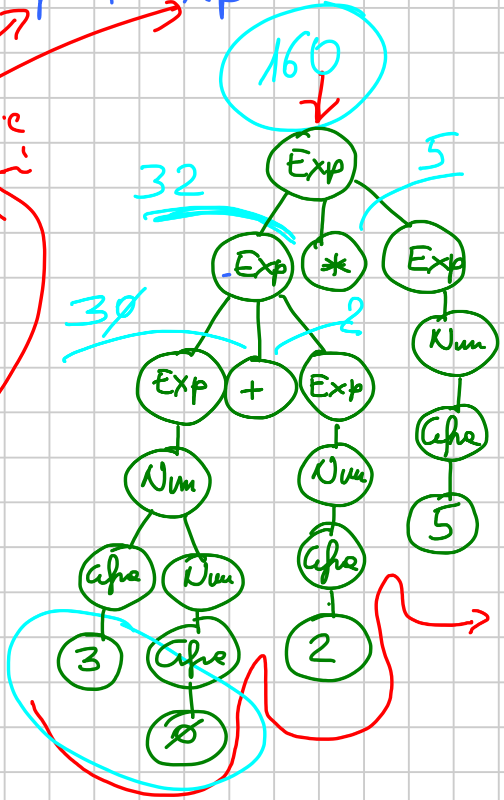
30 + 2 * 5

$V = \{ Exp, Num, Cifra \}$

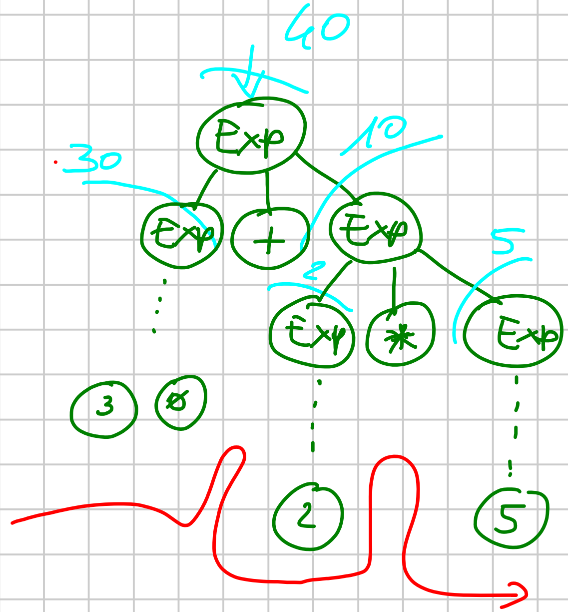
$\Lambda = \{ +, *, 0, 1, \dots, 9 \}$

$S = Exp \in V$

P



30 + 2 * 5



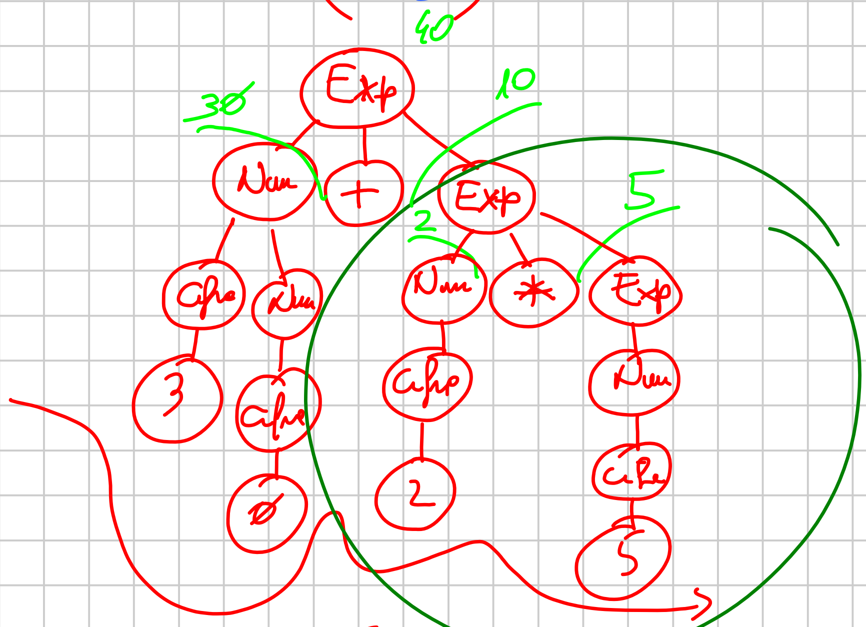
30 + 2 * 5

Exp → Num | Num + Exp | Num * Exp

Num → . . . .

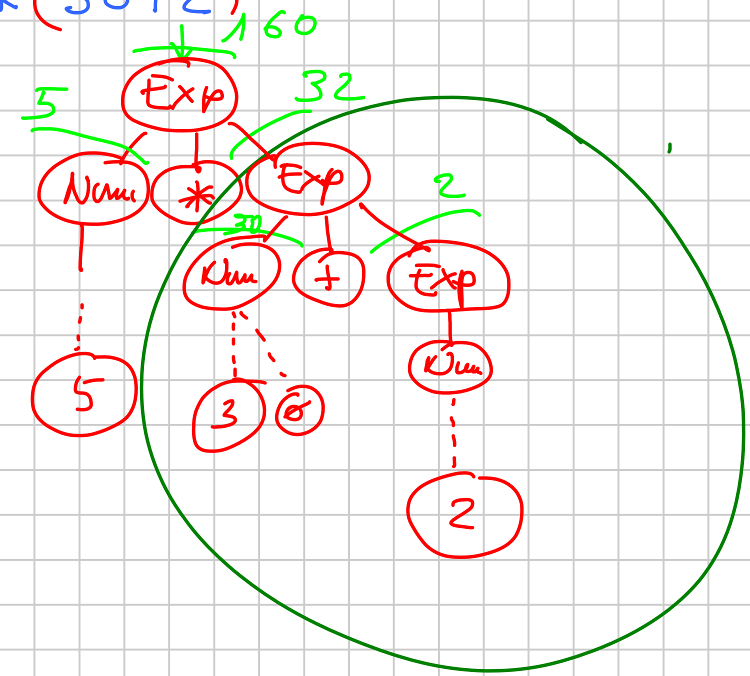
3 + 5 * 2 + 4

30 + (2 * 5)

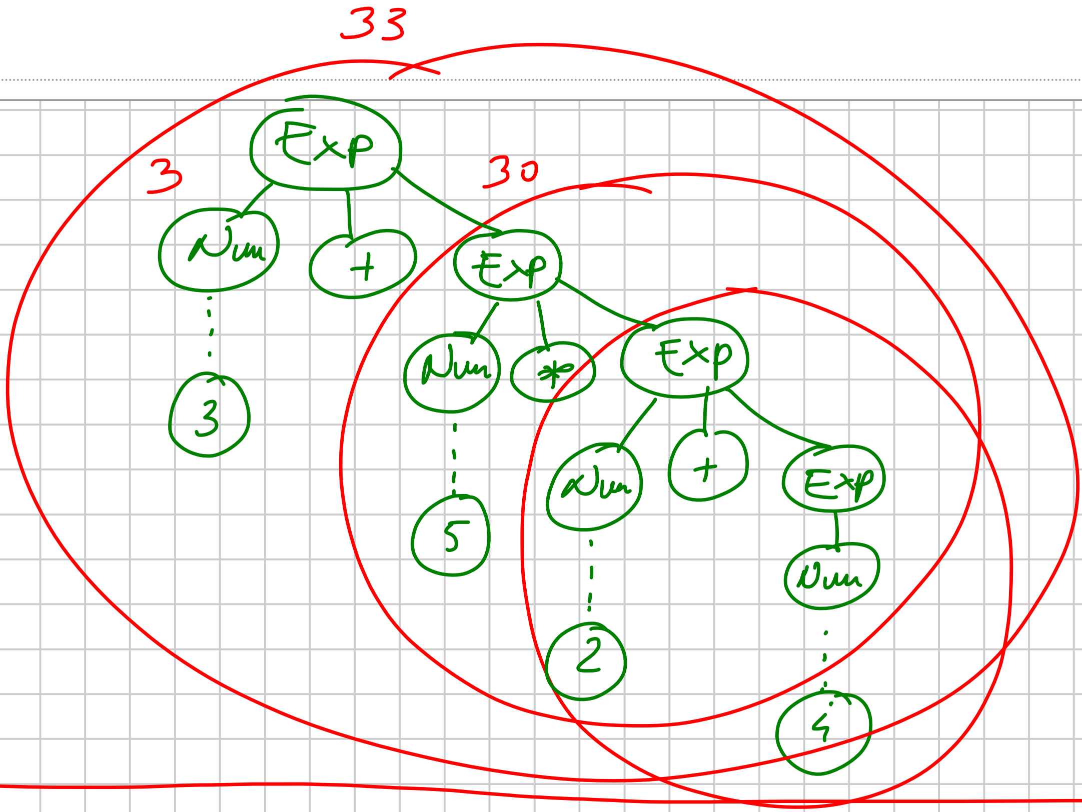


30 + 2 * 5

5 * (30 + 2)



$3 + (5 * (2 + 4))$



$Exp \rightarrow Num \mid Exp + Num \mid Exp * Num$

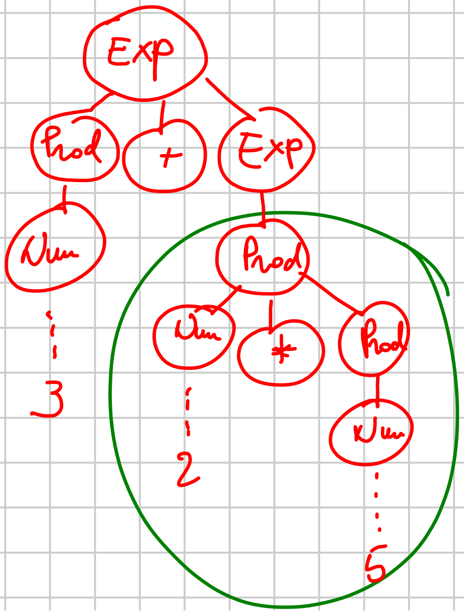
$((3+5) * 2) + 4$

Exp → Prod | Prod + Exp

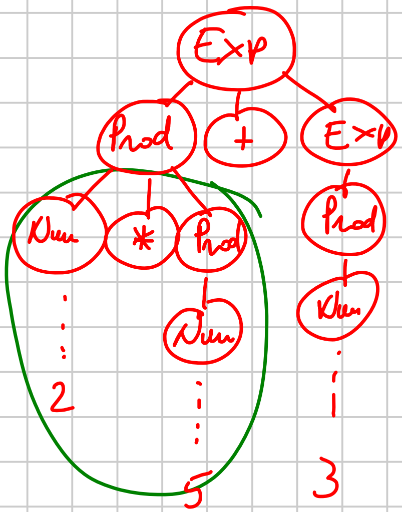
Prod → Num | Num ∗ Prod

Num → ....

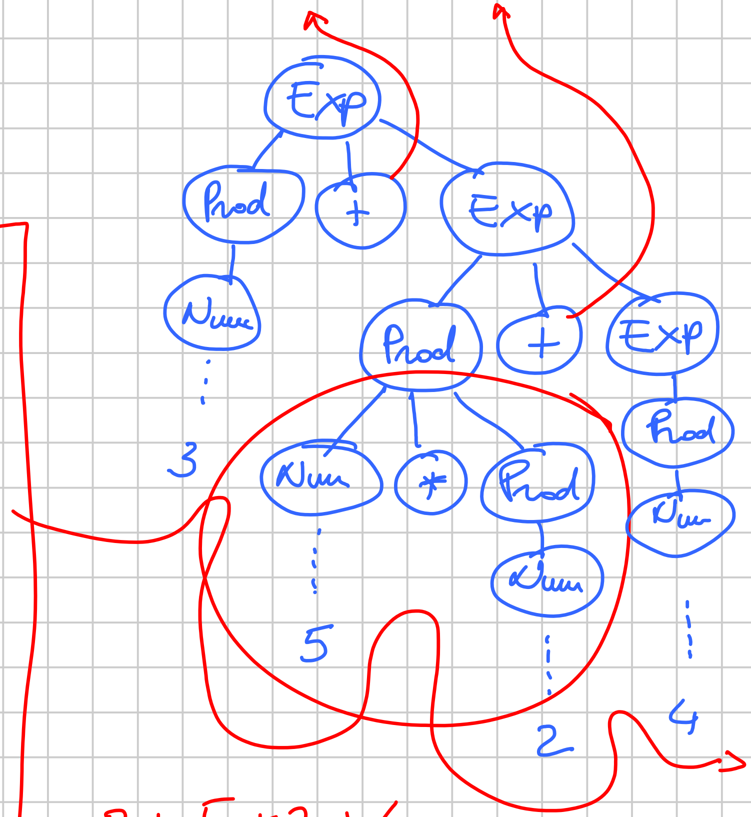
3+2∗5



2∗5+3



3+5∗2+4

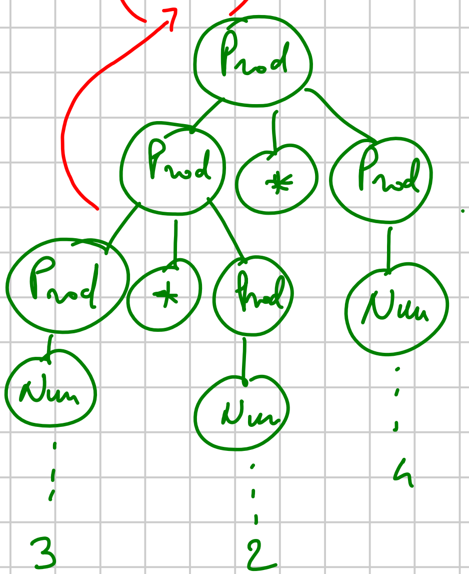


3+5∗2+4

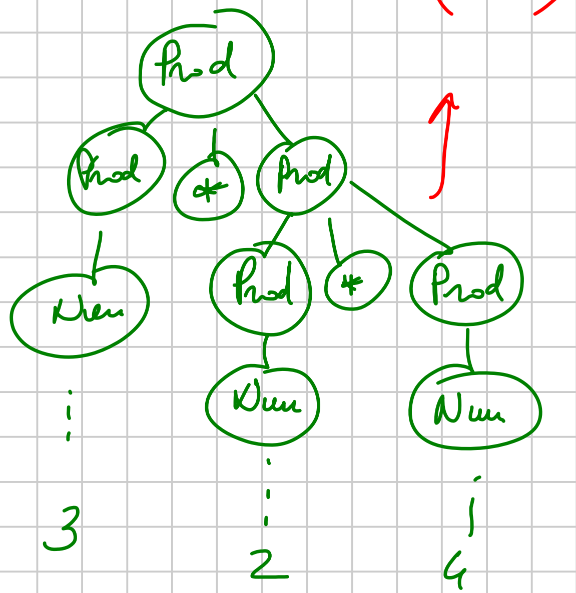
Prod → Num | Prod * Prod

Num → ....

(3 * 2) * 4

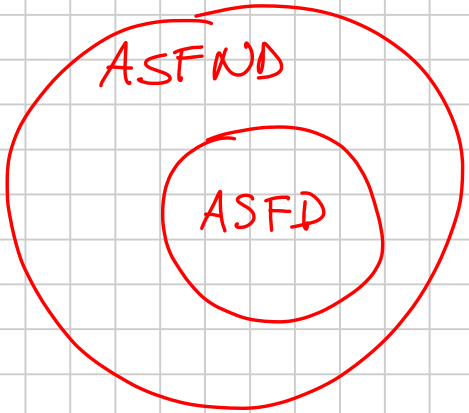


3 * (2 * 4)



ambiguità semantica comunque irrilevante

ASF



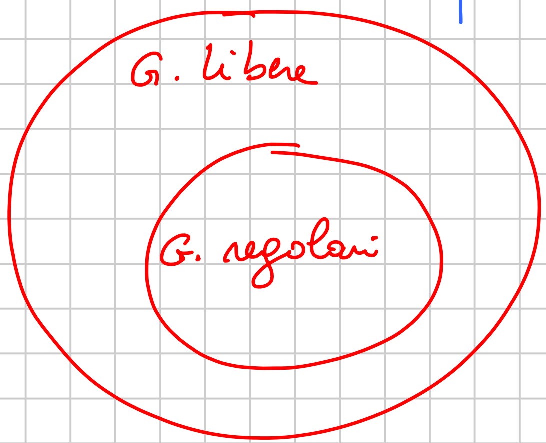
$L = \{a^n b^n \mid n > 0\}$

non è generabile da una g. regolare

ASFND $\equiv$ ASFD

riconoscono lo stesso insieme di linguaggio

Grammatiche



G. libere $\stackrel{?}{\equiv}$ G. regolari

No! Le grammatiche libere sono più potenti di quelle regolari

G. libere $A \to \alpha$ $\quad A \in V \quad \alpha \in (\Lambda \cup V)^*$

G. regolari: $A \to aB$ $\quad A, B \in V$

$A \to a$ $\quad a \in \Lambda$

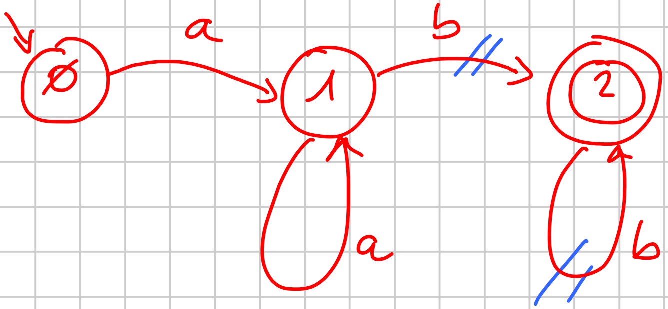
Titolo nota 28/09/2015

Grammatiche regolari hanno la stessa potenza degli ASF

Ling. generato da una g. regolare $\Rightarrow$ è possibile costruire un ASF che lo riconosce

Ling. riconosciuto da un ASF $\Rightarrow$ è possibile costruire una g. regolare che lo genera

ASF $\Sigma = \{a, b\} = (\Sigma, Q, S, F, \delta)$

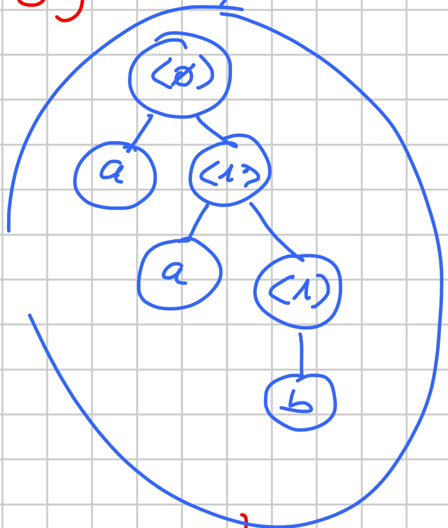


aab

$\emptyset \xrightarrow{a} 1 \xrightarrow{a} 1 \xrightarrow{b} \underline{\underline{2}}$

$L = \{ a^n b^m \mid n, m > 0 \}$

$P' = $
$\langle \emptyset \rangle \to a \langle 1 \rangle$
$\langle 1 \rangle \to b \langle 2 \rangle$
$\langle 1 \rangle \to a \langle 1 \rangle$
$\langle 2 \rangle \to b \langle 2 \rangle$
$\langle 1 \rangle \to b$
$\langle 2 \rangle \to b$



G. regolare corrispondente

$G = (\Sigma', V',$

$\Sigma' = \Sigma$ | S é la c.s. che corrisponde allo stato iniziale

V' simboli corrispondenti agli stati dell'automa $V = \{ \langle \emptyset \rangle, \langle 1 \rangle, \langle 2 \rangle \}$

$P' = \{ \langle q \rangle \to a \langle q' \rangle \mid \langle q, a, q' \rangle \in \delta \} \cup \{ \underline{\langle q \rangle \to a} \mid \langle q, a, q' \rangle \in \delta$ e $q' \in F \}$

Grammatiche libere

$L = \{ a^n b^{n+1} \mid n > 0 \}$

$L = \{ abb,\ aabbb,\ aaabbbb,\ \dots \}$

$S \to abb \mid aSb$

$L = \{ a^n b^{2n} \mid n > 0 \}$

$S \to abb \mid aSbb$

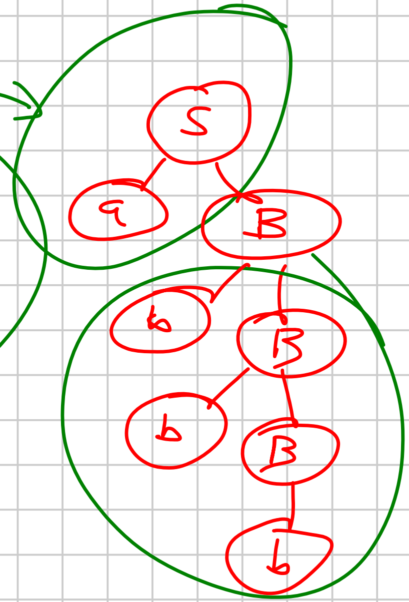
$L = \{ a^n b^n \mid n \geq 0 \}$

$S \to ab \mid aSb$

$S \to \varepsilon \mid aSb$

$S \to aS \mid aB$

$B \to b \mid bB$



$L = \{ a^n b^m \mid n, m > 0 \text{ e } n > m \}$

$S \rightarrow aaS \mid aaB$

$B \rightarrow b \mid bB$

$S \rightarrow aab \mid aSb \mid aS$