8.3 Noção de derivação de uma palavra em GLC

Aplicando as regras de produção de uma gramática livre de contexto, existem varias seqüências de aplicação da regras que podem levar a uma mesma palavra.

Ex: seja a gramática cujas regras são as seguintes:

(1) $S \rightarrow SS$
(2) $S \rightarrow aSa$
(3) $S \rightarrow bSb$
4) $S \rightarrow \varepsilon$

A cadeia aabaab pode ser produzidas por diversas derivações diferentes, variando entre si em função da ordem de aplicação das regras. Exemplos:

CASO 1

$S \rightarrow SS \; : \; SS$

$S \rightarrow aSa \; : \; aSaS$

$S \rightarrow \varepsilon \; : \; aaS$

$S \rightarrow bSb \; : \; aabSb$

$S \rightarrow aSa \; : \; aabaSab$

$S \rightarrow \varepsilon \; : \; aabaab$

CASO 2

$S \rightarrow SS : SS$

$S \rightarrow bSb : SbSb$

$S \rightarrow aSa : SbaSab$

$S \rightarrow \varepsilon : Sbaab$

$S \rightarrow aSa : aSabaab$

$S \rightarrow \varepsilon : aabaab$

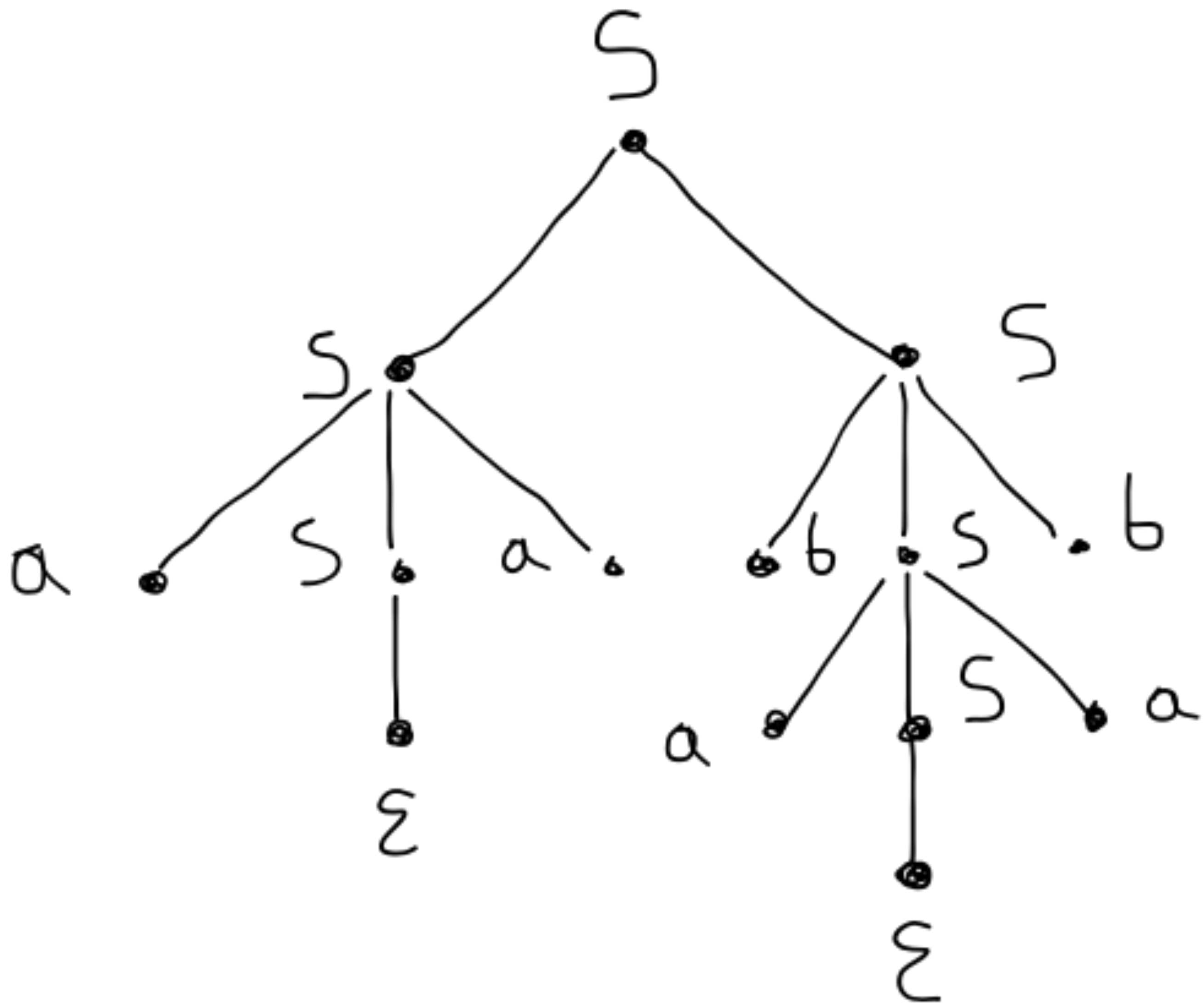
Nas gramáticas livre de contexto, a ordem de aplicação das regras de produção não tem importância, uma vez que um símbolo não terminal pode ser modificado independentemente das cadeias à direita ou à esquerda. Contudo, tal variação na ordem da aplicação produz derivações distintas para uma mesma palavra.

Será que existe então um modelo de representação da noção de derivação mais abstrato do que as gramáticas que, independentemente da ordem em que as regras forem aplicadas, produza sempre um mesmo modelo para uma mesma palavra?

8.4 Noção de árvore de derivação (ou de análise)

Para fazer tal abstração com relação às seqüências de derivações, uma estrutura de derivação em forma de árvore pode ser utilizada.

Por exemplo, para a cadeia aabaab, a derivação da palavra baseada nas regras da gramática pode ser representada através da seguinte árvore:


S
S
S
a
S
a
b
S
b
ε
a
S
a
ε

Definição : Uma árvore de derivação de uma gramática livre de contexto $G=(V,\Sigma,R,S)$ é uma árvore tal que cada um de seus nós é rotulado por um elemento de $V \cup \varepsilon$ e que satisfaz as seguintes restrições:

- a raiz é rotulada pelo símbolo de início S;
- cada nó interno é rotulado por um símbolo não terminal e cada folha é rotulada por um símbolo terminal ou pelo símbolo ε;
- para cada nó interno, se seu rótulo é o não terminal A e seus nós descendentes imediatos n1, n2,..., nk são rotulados por X1, X2, ...,Xk, respectivamente, então a regra de produção $A \rightarrow X1X2...Xk$ deve ser uma das regras da gramática.
- se um nó é rotulado por ε, então tal nó é o único filho do nó pai correspondente (tal regra evita a introdução de instâncias inúteis de ε na árvore).

Definição: a palavra produzida por uma árvore de análise é obtida por meio da concatenação das folhas recuperadas da esquerda para a direita.

OBS: considerando tal definição, note que, no caso particular do exemplo anterior, a árvore apresentada é a única possível para analisar a palavra aabaab. Contudo, antecipe-se aqui o problema de ambiguidade de gramáticas a ser tratado a seguir: há palavras geradas por tal gramática (como " aa ", por exemplo), para as quais podem ser produzidas mais do que uma árvore de análise ...).

Teorema: considerando uma gramática livre de contexto G, uma palavra w é gerada por tal gramática se, e somente se, existir uma árvore de análise que a produz.

8.5 Ambigüidade das gramáticas

- Observa-se que uma mesma árvore de derivação pode ser gerada a partir de ordens variadas de expansão de nós, o que corresponde a variações nas ordens das derivações.

Exemplos: derivação à esquerda (sempre trocar o não terminal mais a esquerda da palavra produzida); derivação à direita; derivação à esquerda-direita; derivação ao centro ...

- Será que também existem varias árvores de derivação para uma mesma palavra?

- No caso da palavra aabaab do exemplo anterior, não. Contudo, no caso da palavra aa do mesmo exemplo, sim!

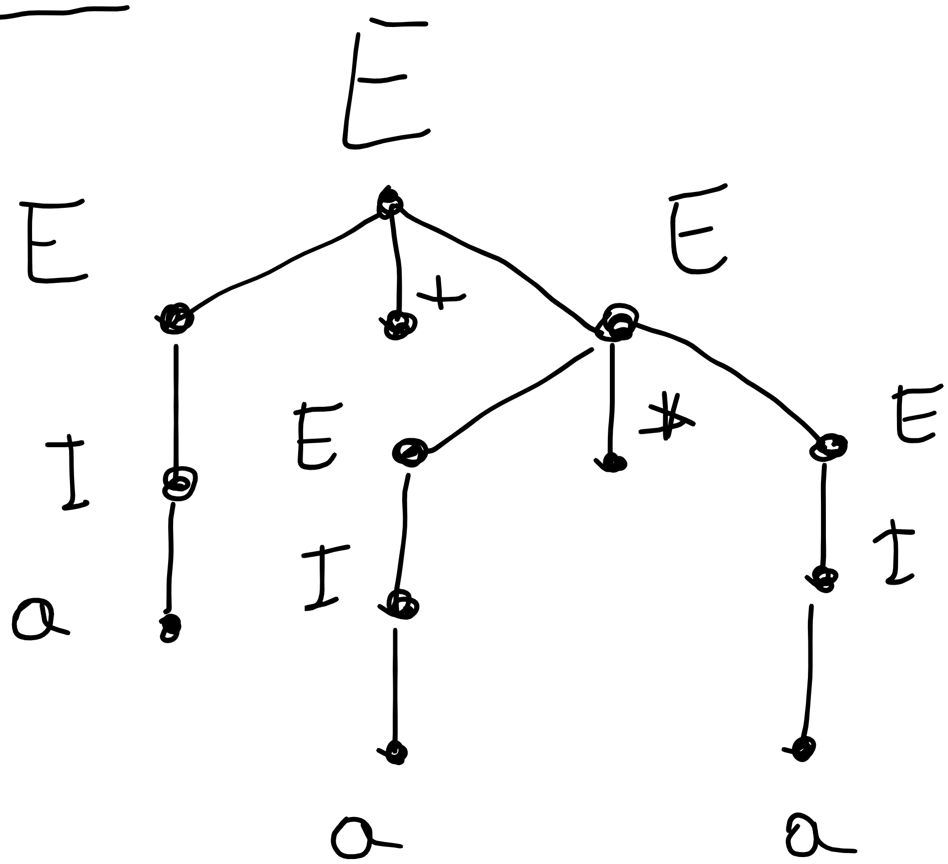
- Quando uma gramática permite a geração de árvores distintas para uma mesma palavra, ela é dita "ambígua" (como a do exemplo anterior).

Outro exemplo de gramática ambígua :

$E \rightarrow I$
$E \rightarrow E+E$
$E \rightarrow E*E$
$E \rightarrow (E)$
$I \rightarrow a$
$I \rightarrow b$

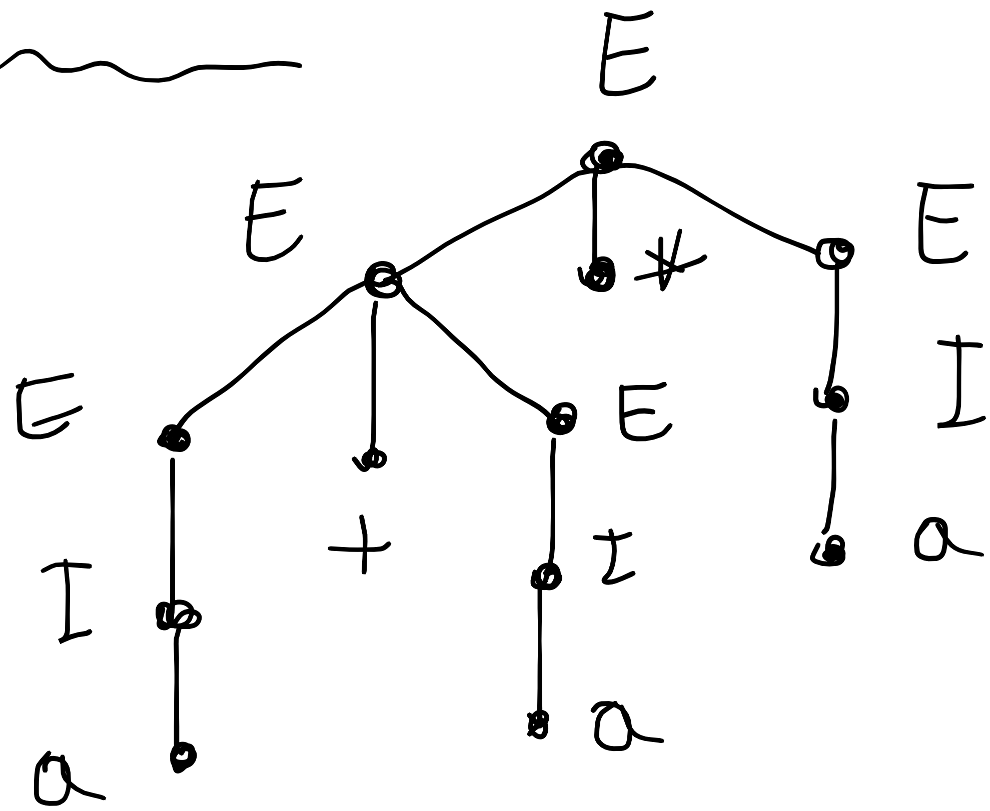
Tal gramática permite a geração de duas árvores de derivação para a palavra a+a*a , a saber:

ARVORE 1



E→I
E→E+E
E→E*E
E→(E)
I→a
I→b

ARVORE 2



E→I
E→E+E
E→E*E
E→(E)
I→a
I→b

Problema da ambigüidade: a arvore de derivação é um modelo usado em compilação.

Se as árvores são diferentes, então o significado (a semântica) da palavra será diferente também.

No caso da primeira árvore, será realizado o produto primeiro; no caso da segunda, a soma: para a mesma palavra, o resultado da fórmula (a interpretação da árvore) será diferente.

É necessário remover as eventuais ambigüidades das gramáticas sempre que possível.

Infelizmente, não existe um procedimento genérico (algoritmo) que permita decidir se uma gramática dada é ambígua ou não.

Existem também linguagens livres de contexto cujas gramáticas geradoras são TODAS ambíguas (linguagens intrinsecamente ambíguas). Nesses casos, a remoção da ambigüidade é impossível.

Na prática, no caso de gramáticas usadas para gerar expressões aritméticas, por exemplo, é possível, acrescentando símbolos não terminais à gramática ambígua, remover a ambigüidade. Para tanto, fixam-se certas prioridades entre os operadores e agrupam-se as expressões de acordo com as regras de associatividade.

Por exemplo, o símbolo * tem geralmente precedência sobre o símbolo + .

Além disso, tanto * quanto + são associativos à esquerda (as expressões conectadas por * ou + são agrupadas a partir da esquerda)

Um exemplo de gramática não ambígua que permite representar a mesma linguagem anterior é dado através das seguintes regras:

$E \to E+T$ (O não terminal E é utilizado, inicialmente, para as somas)

$E \to T$ (Depois de terminar as somas, muda-se para o não terminal T)

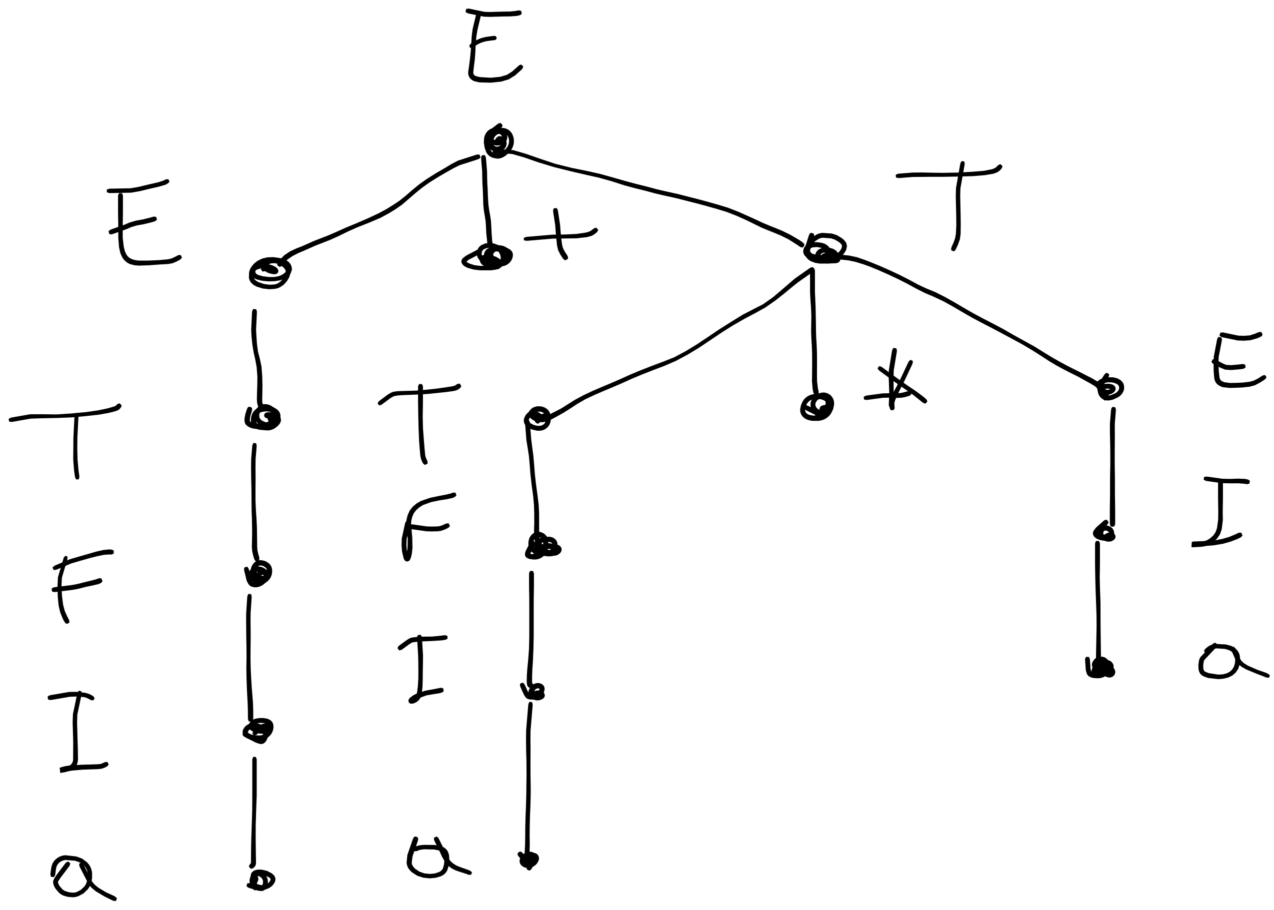
$T \to T*F$ (O não terminal T é usado para os produtos)

$T \to F$ (Concluídos os produtos, muda-se para o terminal F)

$F \to (E)$: trata as expressões parentizadas
$F \to I$

$I \to a$
$I \to b$

No caso desta gramática, somente há uma árvore de derivação possível para a+a*a:

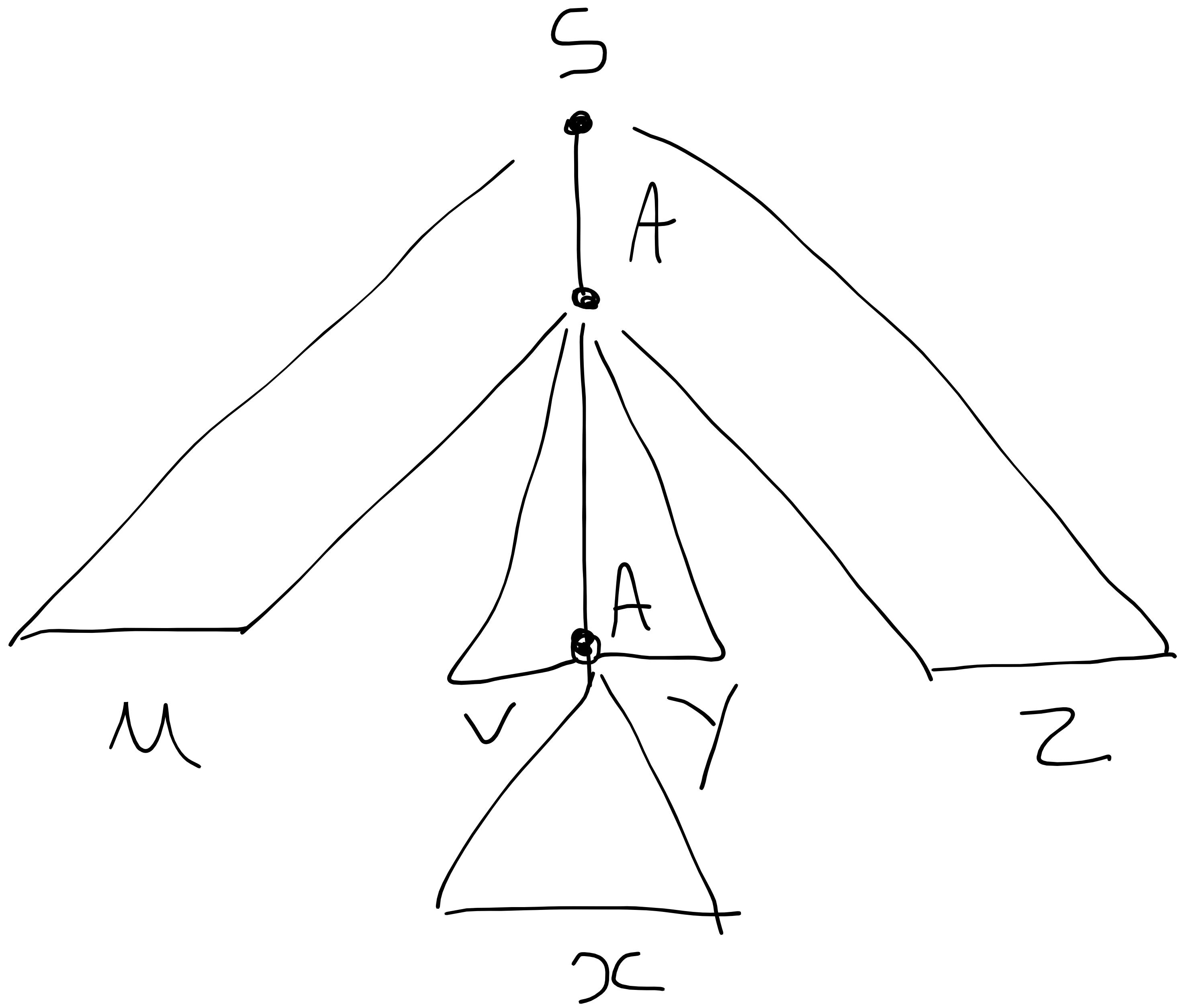


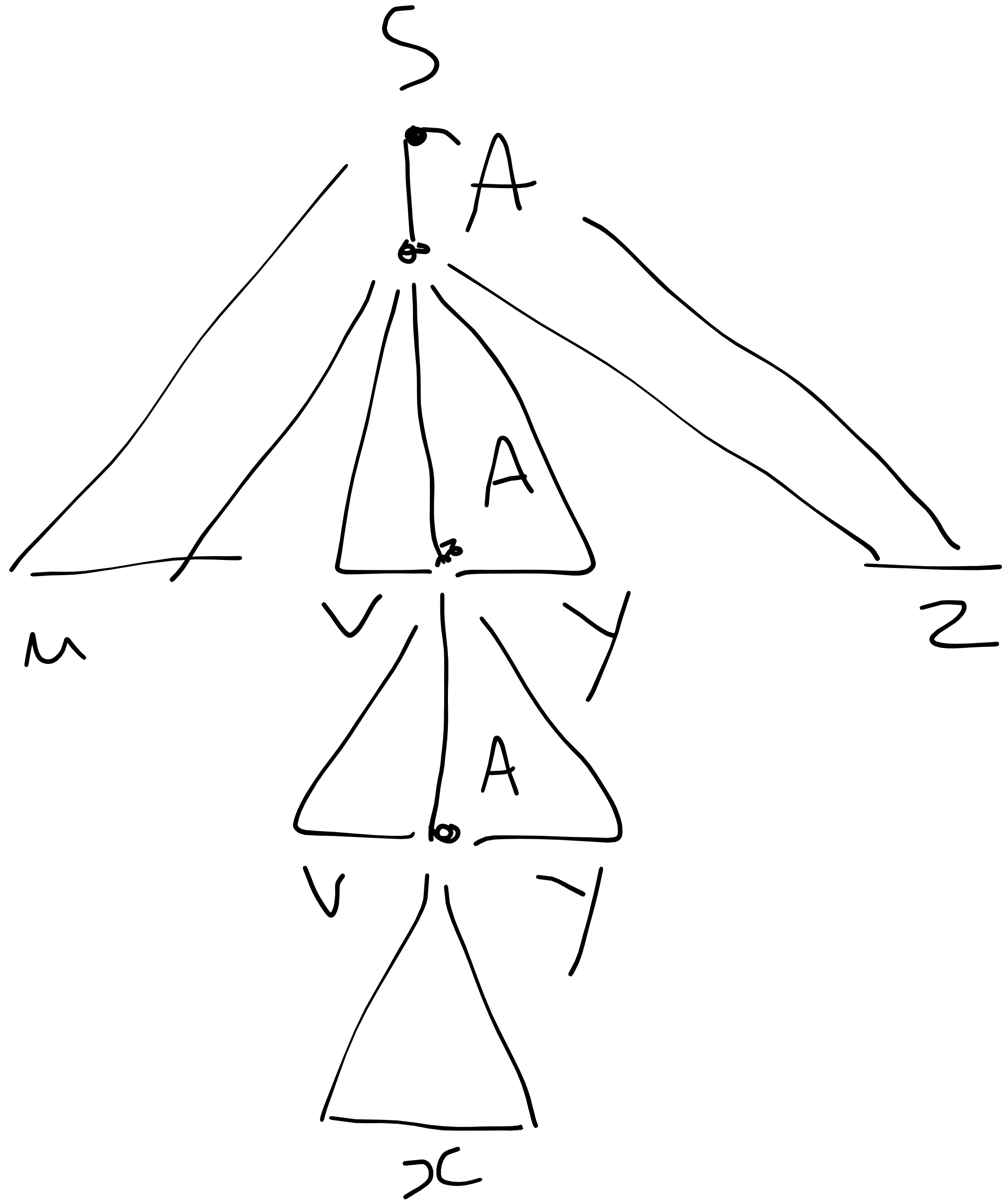
E --> E + T | T
T --> T * F | F
F --> (E) | I
I --> a | b

8.6 Lema do Bombeamento

Teorema: seja uma linguagem livre de contexto L. Então, existe um valor constante K tal que toda palavra $w \in L$ e que verifica $|w| \geq K$ pode ser escrito na forma $w = uvxyz$ com v ou $y \neq \varepsilon$, $|vxy| \leq K$ e $uv(n)xy(n)z \in L$ para todo $n \geq 0$

Prova: seja uma gramatica livre de contexto que produz L. Para uma palavra grande o suficiente deve existir então uma arvore de derivação que contem um caminho onde um mesmo símbolo nao terminal fica se repetindo.


S
A
A
u
v
y
z
x


S
A
A
A
u
v
y
z
v
y
x

Valor de K?

Considera o valor $p = \max \{|\alpha|, A \rightarrow \alpha \in R\}$
Cada nó da arvore de derivação tem no máximo p sucessores.

O comprimento máximo de uma palavra produzida por uma arvore de derivação de profundidade i é então igual ao número máximo de folha da arvore : $p(i)$

Se $m = |\{V- \Sigma\}|$, a profundidade máximo da arvore sem ter uma repetição de um nó não terminal será m.

Então toda arvore de derivação que produz uma palavra w tal que $|w| \geq K$ será tal que $|w| > p(m)$ e deve ter um ou mais caminhos de comprimento pelo menos igual a m+1 com um mesmo não terminal que devera aparecer no mínimo duas vezes no caminho.

Considerando o maior caminho existente na arvore e o primeiro não terminal a aparecer duas vezes no caminho invertido saindo da folha, o mesmo não terminal será atingido percorrendo no máximo m+1 arco a partir da folha.

Se o não terminal é o não terminal A da figura anterior, então a palavra vxy terá um comprimento máximo igual a $p(m+1) = K$

Para o caminho considerado as palavras v e y não são vazias as duas (se as duas estão vazia a parte da arvore entre as duas ocorrências de A podem ser eliminadas sem modificar a palavra final).

Encontrar um exemplo a partir de umas das gramáticas apresentadas...

Exemplo de aplicação do Lema:

Provar que a linguagem L = {a(n)b(n)c(n)} não é livre de contexto.

Mostrar que não é possível decompor uma palavra a(n)b(n)c(n) em 5 partes u, v, x, y e z (v ou y não vazia) tais que para todo $j \geq 0$, uv(i)xy(i)z $\in$ L.

Prova por contradição: supor a existência de uv(i)xy(i)z e mostrar uma contradição.

Casos possíveis para a construção das palavras v e y:

- v e y formadas pela repetição de um mesmo símbolo: por exemplo, $v \in a^*$ e $y \in b^*$.

Neste caso a igualdade do números de símbolos a, b e C não será preservada por causa das repetições possíveis de v e y.

-v e y formadas por símbolos diferentes.

Neste caso as palavras $uv(i)xy(i)z$ não respeitarão a forma $a^*b^*c^*$

Exercício: mostrar que a linguagem livre de contexto a(n)b(n) verifica o Lema do Bombeamento.