Context Free Grammars

G = (V, T, P, S)

V = *(finite) set of* ***variables*** *(or* ***non-terminal*** *symbols)*

T = *(finite) set of* ***terminal*** *symbols*

P = *finite subset of* $V \times (V \cup T)^*$ *called* ***productions***

$S \in V$ = *the* ***start*** *symbol*

Example

$G = (\{S\}, \{0,1\} , P, S)$, *where*

$\quad\quad V \quad\quad T$

$P : S \rightarrow 0S1 \mid e$ *short hand notation for* $\{ (S,0S1), (S,e) \} \subseteq V \times (V \cup T)^*$

$S \Rightarrow_G 0S1 \Rightarrow 0(0S1)1 = 0^2 S 1^2 \Rightarrow 0^2 e 1^2 = 0^2 1^2$

$S \rightarrow 0 S 1 \quad\quad S \rightarrow 0 S 1 \quad\quad S \rightarrow e$

$S \Rightarrow^3 0^2 1^2$ *: a 3-step derivation*

Derivations

Let $\boldsymbol{\alpha} A \boldsymbol{\beta} \in (V \cup T)^*$ *, with* $\boldsymbol{\alpha}, \boldsymbol{\beta} \in (V \cup T)^*$ *and* $\boldsymbol{A} \in \boldsymbol{V}$ *and let*

$A \rightarrow \gamma$ *be a production of a CFG* **G** *then :*

$\boldsymbol{\alpha} A \boldsymbol{\beta} \Rightarrow_G \boldsymbol{\alpha} \boldsymbol{\gamma} \boldsymbol{\beta}$ *is called a (one step) derivation in* **G** **;**

in a similar manner we have :

$\boldsymbol{W} \Rightarrow_G^n \boldsymbol{U}$ *and* $\boldsymbol{W} \Rightarrow_G^* \boldsymbol{U}$ *are n-step and finite step derivations in* **G** *where each step conforms to the rule for the one step derivation above.*

Definition

- *The language* $\boldsymbol{L_G(A)}$ *generated by the variable* $\boldsymbol{A}$ *of a grammar* $\boldsymbol{G}$ *is given by :* $\boldsymbol{L_G(A) := \{ v \in T^* \mid A \Rightarrow^*_G v \}}$

- *the language* $\boldsymbol{L_G}$ *generated by* $\boldsymbol{G}$ *is* $\boldsymbol{L_G := L_G(S)}$ *, where* $\boldsymbol{T}$ *and* $\boldsymbol{S}$ *denote the set of terminal symbols and the start symbol of* $\boldsymbol{G}$ *respectively.*

For the previous example $\boldsymbol{L_G = \{0^n 1^n, n \geq 0\}}$

3 Examples of CFGs

(1) Regular Expressions over Σ where $\Sigma = \{\sigma_1, \sigma_2, \ldots \sigma_n\}$

$$E \rightarrow \sigma_1 \mid \sigma_2 \ \ldots \mid \sigma_n \mid e \mid \varnothing \mid E+E \mid E.E \mid E^* \mid (E)$$

There are ***n+6*** *productions with* $\boldsymbol{n} = |\Sigma|$ *where :*

$V = \{ E \}$,

$T = \Sigma \cup \{e, \varnothing, +, \ . \ , \ *, (,) \}$

P **=** *the* ***n+6*** *productions above*

$S = E$

Example of a regular expression derivation with $\Sigma = \{0,1\}$

$E \Rightarrow^7 0.(1+0)^*$

$E \Rightarrow E.E \Rightarrow 0.\ E \Rightarrow 0.E^* \Rightarrow 0.(E)^* \Rightarrow 0.\ (E+E)^* \Rightarrow 0.(1+E)^* \Rightarrow 0.(1+0)^*$

$E \rightarrow E.E$ $\quad E \rightarrow 0$ $\quad E \rightarrow E^*$ $\quad E \rightarrow (E)$ $\quad E \rightarrow E+E$ $\quad E \rightarrow 1$ $\quad E \rightarrow 0$

(2) Simple Arithmetic Expressions *(variables and binary numbers)*

*Two operations : + and * and numbers and variables that are strings in* $x0 \cup x1\{0,1\}^*$ (i.e. x_0, x_1, x_2 … ; *variables with binary indices)*

$V = \{E, I, J\}$
$T = \{0, 1, x, +, *, (,)\}$

E = *(Arithmetic) Expression*
I = *Identifier* , **J** = *Identifier trailer*

P = 11 *productions given next*
$S = E$

Productions :

$E \rightarrow I \mid E + E \mid E^*E \mid (E)$

$I \rightarrow x0 \mid x1J \mid 0 \mid 1J$; $J \rightarrow 0J \mid 1J \mid e$

*Example for + ; * arithmetic expressions*

$E \Rightarrow^{13} x1*(x0+11)$ *in ordinary notation* : $x_1 \cdot (x_0 + 3)$

$E \Rightarrow E*E \Rightarrow I*E \Rightarrow x1J*E \Rightarrow x1e*E \Rightarrow^2 x1*(E+E) \Rightarrow x1*(I+E) \Rightarrow x1*(x0+E) \Rightarrow x1*(x0+I)) \Rightarrow x1*(x0+1J) \Rightarrow^2 x1*(x0+11)$

(3) The Grammar of Balanced Parentheses

$V = \{ E \}$

$T = \{ (\ ,\ ) \}$

P = *the 3 productions below*

$S = E$

P :

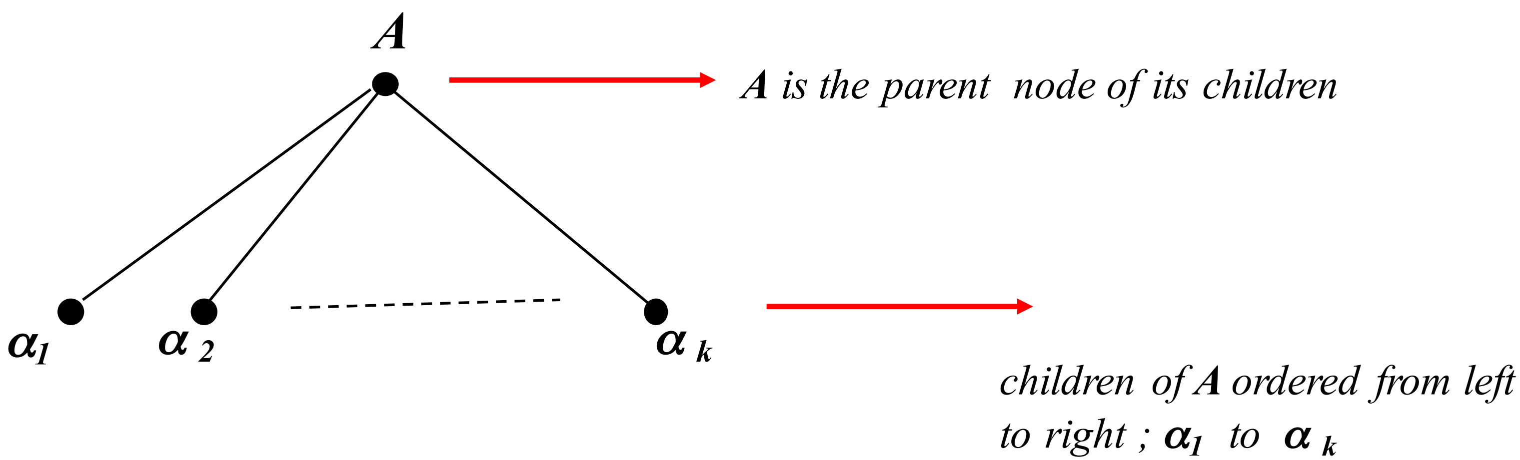
$E \rightarrow EE \mid (E) \mid e$

Example

$E \Rightarrow^* (()())()$

$E \Rightarrow EE \Rightarrow (E)E \Rightarrow (EE)E \Rightarrow ((E)E)E \Rightarrow ((e)E)E \Rightarrow (()(E))E \Rightarrow (()(e))E \Rightarrow (()())(E) \Rightarrow (()())(e) \Rightarrow (()())()$

Every production $A \rightarrow \boldsymbol{\alpha}_1 \mid \boldsymbol{\alpha}_2 \; \ldots \mid \boldsymbol{\alpha}_k$ *where each* $\boldsymbol{\alpha}_j \in V \cup T$ *corresponds to an* ***ordered tree*** *of height 1 as shown below*



Terminology on ordered trees **:** *root, order, children , siblings, parent , descendants, ancestors, leaves, internal nodes …*

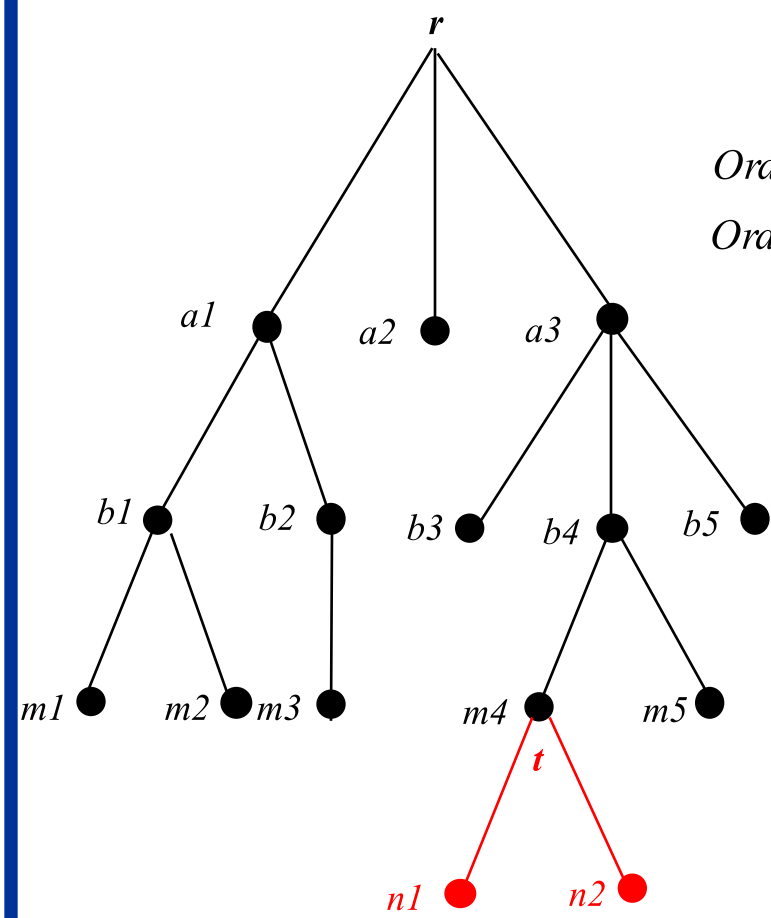
Recursive definition of ordered trees

Basis : *an ordered tree **T** of depth 1 with a root node **t** and ordered sequence of children (leaf) nodes* $(\boldsymbol{n_1}, \ldots, \boldsymbol{n_k})$ *is an ordered tree.*

Induction : *Let **S** be an ordered tree with a root node **r** and ordered sequence of leaf nodes* $(\boldsymbol{m_1}, \ldots, \boldsymbol{m_p})$ *; let **T** be an ordered tree of depth 1 with a root node **t** and children nodes* $(\boldsymbol{n_1}, \ldots, \boldsymbol{n_k})$ *then for any* $\boldsymbol{0 \le j \le p}$ *, **S'** is an ordered tree obtained from **S** by replacing the leaf node* $\boldsymbol{m_j}$ *of **S** by **T** so that the new ordered leaf nodes of **S'** are* $(\boldsymbol{m_1}, \ldots, \boldsymbol{m_{j-1}}, \boldsymbol{n_1}, \ldots, \boldsymbol{n_k}, \boldsymbol{m_{j+1}}, \ldots \boldsymbol{m_p})$ *and **t** is an internal node replacing* $\boldsymbol{m_j}$ *.*

Basis *: For a tree **T** of depth one the root node **t** is the **parent** and an **ancestor** of the children nodes* $(\boldsymbol{n_1}, \ldots, \boldsymbol{n_k})$ *and the children nodes are called **siblings** of each other and **descendants** of the root node **t.***

Induction : *For **S'** defined as above all nodes of **S** retain the **ancestor** and **descendant** relations in **S'** and every ancestor of the replaced node* $\boldsymbol{m_j}$ *is an ancestor of all the newly added nodes* $(\boldsymbol{n_1}, \ldots, \boldsymbol{n_k})$ *as well as the root node **t** and if* $\boldsymbol{m_j}$ *was a descendant of a node **n** in **S** then **t** and the nodes* $(\boldsymbol{n_1}, \ldots, \boldsymbol{n_k})$ *are descendants of the node **n** ; etc.*



S to S'

Ordered leaves of S → *m1 m2 m3 a2 b3 m4 m5 b5*

Ordered leaves of S' → *m1 m2 m3 a2 b3 n1 n2 m5 b5*

In ***S*** ***: a3 ancestor*** *of* ***m4*** *and* ***m4 descendant*** *of* ***a3***

In ***S' : a3 ancestor*** *of* ***t,n1,n2*** *;* ***t,n1,n2 descendants*** *of* ***a3***

A convention on notation

Lower case ***a,b,c*** *, ... symbols in* ***T***

Upper case ***A,B,C*** *, ... symbols in* ***V***

Lower case ***w,v,z*** *, ... symbols in* ***T****

Upper case ***X,Y,Z****, ... symbols in* ***V*** ∪ ***T***

Lower case Greek ***α, β, γ*** *, ... symbols in* ***(V*** ∪ ***T)****

Derivations and Parse Trees

Consider the derivation $\boldsymbol{S \Rightarrow^*_G \omega \in T^*}$; *then for each step of the derivation a production of a non-terminal is used until all symbols are terminals as in* $\boldsymbol{\omega}$.

The parse tree is obtained by replacing each non-terminal corresponding to the production used– starting from $\boldsymbol{S}$ *– by the production tree of that non-terminal.*

Order on the leaves of the parse tree is the induced order of the children in the productions (every pair of nodes have a unique common youngest ancestor whose corresponding children set the order !)

Leftmost (lm) and Rightmost (rm) derivations

Definition *A derivation is called a* ***leftmost*** *(**rightmost**)* ***derivation*** *if at each step of the derivation a production is applied to* ***the*** *nonterminal at the* ***leftmost*** *(**rightmost**) position of the string*

Theorem *For every derivation* $A \Rightarrow^* \omega$ *of a variable* ***A*** *there is a* ***leftmost*** *(**rightmost**) derivation shown* $A \Rightarrow^*_{lm} \omega$ $(A \Rightarrow^*_{rm} \omega)$ *with the same parse tree as the original derivation.*

Example

$E \Rightarrow^7_{lm} 0.(1+0)^*$

$E \Rightarrow E.E \Rightarrow 0.\ E \Rightarrow 0.E^* \Rightarrow 0.(E)^* \Rightarrow 0.\ (E+E)^* \Rightarrow 0.(1+E)^* \Rightarrow 0.(1+0)^*$

$E \rightarrow E.E$ $E \rightarrow 0$ $E \rightarrow E^*$ $E \rightarrow (E)$ $E \rightarrow E+E$ $E \rightarrow 1$ $E \rightarrow 0$

$E \Rightarrow^7_{rm} 0.(1+0)^*$

$E \Rightarrow E.E \Rightarrow E.E^* \Rightarrow E.(E)^* \Rightarrow E.(E+E)^* \Rightarrow E.(E+0)^* \Rightarrow E.(1+0)^* \Rightarrow 0.(1+0)^*$

$E \rightarrow E.E$ $E \rightarrow E^*$ $E \rightarrow (E)$ $E \rightarrow E+E$ $E \rightarrow 0$ $E \rightarrow 1$ $E \rightarrow 0$

Example

$$E \Rightarrow^{7}_{lm} 0.(1+0)^*$$

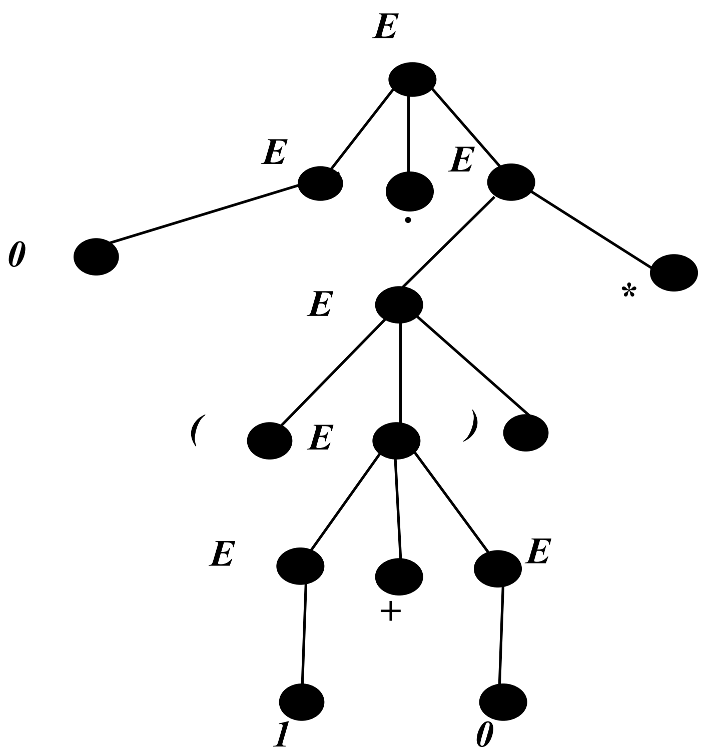
$$E \Rightarrow E.E \Rightarrow 0.E \Rightarrow 0.E^* \Rightarrow 0.(E)^* \Rightarrow 0.(E+E)^* \Rightarrow 0.(1+E)^* \Rightarrow 0.(1+0)^*$$

$E \rightarrow E.E$ $E \rightarrow 0$ $E \rightarrow E^*$ $E \rightarrow (E)$ $E \rightarrow E+E$ $E \rightarrow 1$ $E \rightarrow 0$

$$E \Rightarrow^{7}_{rm} 0.(1+0)^*$$

$$E \Rightarrow E.E \Rightarrow E.E^* \Rightarrow E.(E)^* \Rightarrow E.(E+E)^* \Rightarrow E.(E+0)^* \Rightarrow E.(1+0)^* \Rightarrow 0.(1+0)^*$$

$E \rightarrow E.E$ $E \rightarrow E^*$ $E \rightarrow (E)$ $E \rightarrow E+E$ $E \rightarrow 0$ $E \rightarrow 1$ $E \rightarrow 0$



(1) $A \Rightarrow^* w$

(2) $A \Rightarrow_{lm}^* w$

(3) $A \Rightarrow_{rm}^* w$

(4) *There is a parse tree with root* ***A*** *and yield* ***w***

$$(1) \Leftrightarrow (2) \Leftrightarrow (3) \Leftrightarrow (4)$$

How to obtain a derivation from a parse tree by using induction on depth of the parse tree. *(assume original depth of the tree is **n**)*

Step 1 *: Start from the root **A** and move to the children* X_1 *to* X_k $(A \Rightarrow X_1 \ldots X_k)$

Step 2 *: If all* X_j *are terminals done ; else note that each* X_j *is a subtree of depth at most **n-1** , hence by induction hypothesis there is a derivation* $X_j \Rightarrow^* w_j \in T^*$ *for each **j**. Use these derivations on any desired order on each* X_j *to obtain a desired derivation*

Remark *If the derivations on* X_j *are made from **left to right** (**right to left**) on* X_j *we obtain **leftmost** (**rightmost**) derivation together with the appropriate induction assumption.*

How to obtain a parse tree from a derivation by using induction on the steps of the derivation *(assume original derivation steps is **n**)*

Step 1 *: Start from variable **A** and move to the next step of the derivation where* $A \Rightarrow X_1 \ldots X_k$

Step 2 *: Set the root of the parse tree as **A** ; set each X_j as either an internal node if X_j is a variable and a leaf if X_j is a terminal. For each variable X_j the subtrees to be placed under these internal nodes follow from the induction hypothesis since their derivations have **n-1** steps or less*

HTML Example

1. *Char → a|A| …*
2. *Text → e|Char Text*
3. *Doc → e|Element Doc*
4. *Element → Text |<EM> Doc </EM> |<P> Doc |<OL> List </OL> |…*
5. *ListItem → <LI> Doc*
6. *List → e | ListItem List*

Subset of CFG

V= (Char, Doc, Text, Element, ListItem, List ,…) ; T = (A-z, <EM>, </EM> , <LI>, <OL>, </OL> ,<P>)

<P>
<EM> This is a warning : </EM>
<OL>
<LI> Study hard.
<LI> Do your homework.
</OL>
<P>
Else you will fail !

HTML Program

This is a warning :

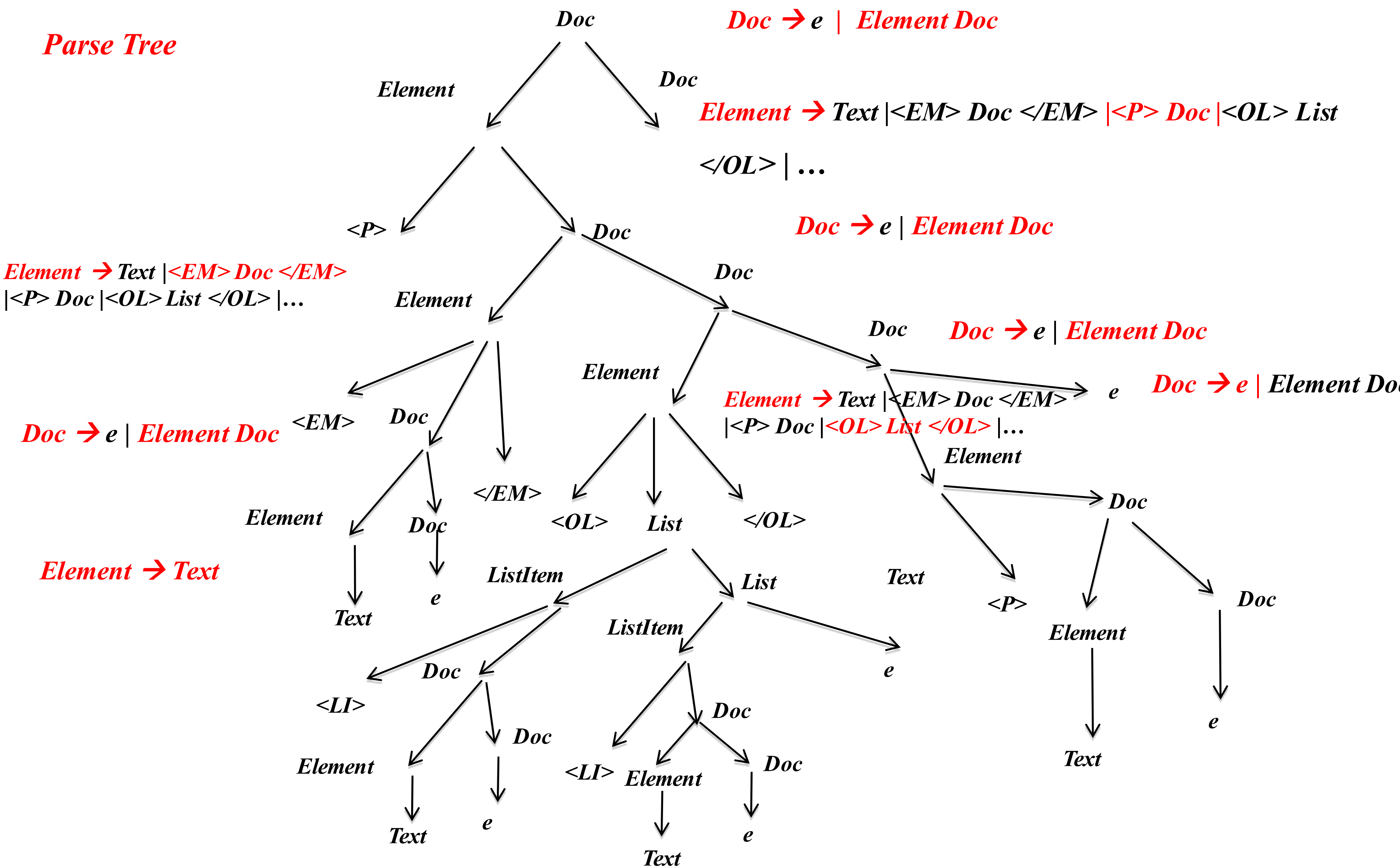
1. Study hard.

2. Do your homework.

Else you will fail !

Output of execution

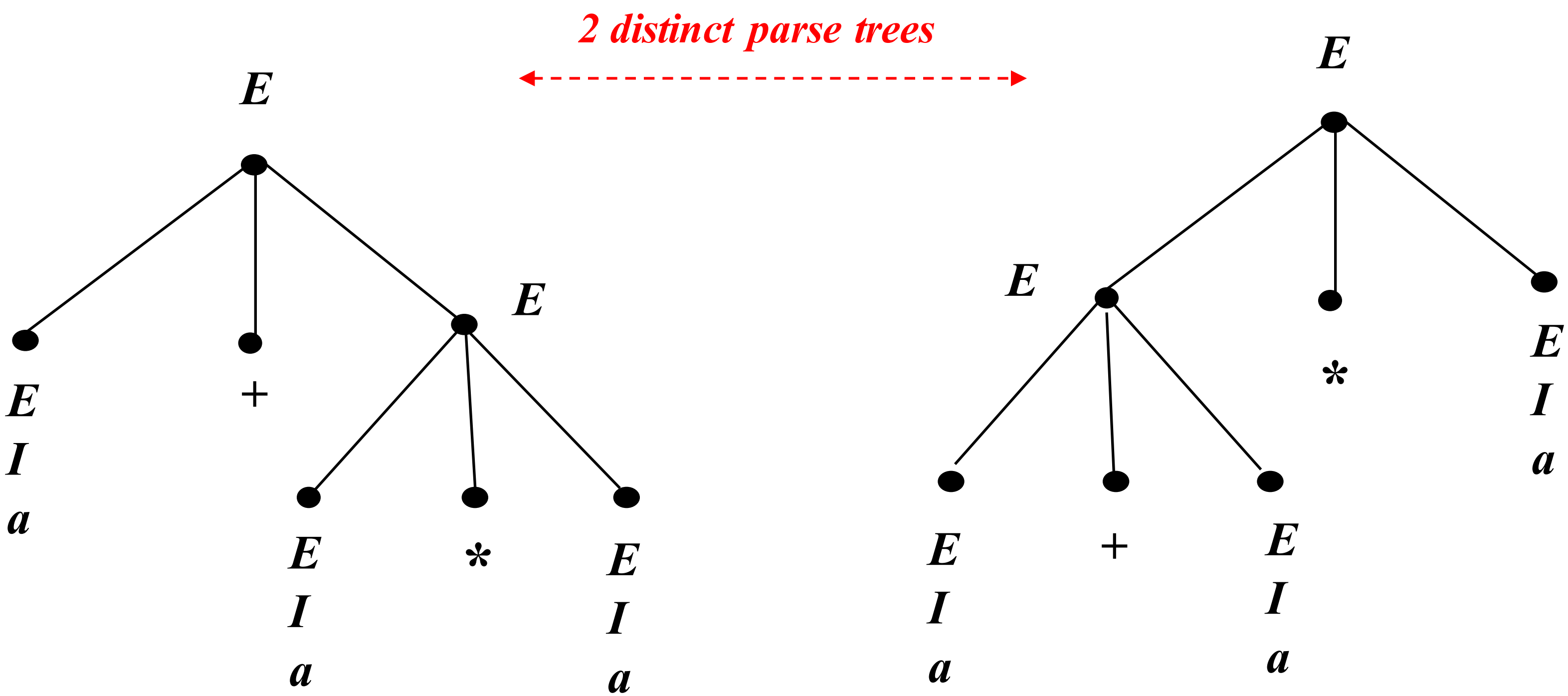
Parse Tree



Doc → e | Element Doc
Element → Text |<EM> Doc </EM> |<P> Doc |<OL> List </OL> | …
Doc → e | Element Doc
Element → Text |<EM> Doc </EM> |<P> Doc |<OL> List </OL> |…
Doc → e | Element Doc
Doc → e | Element Doc
Element → Text |<EM> Doc </EM> |<P> Doc |<OL> List </OL> |…
Doc → e | Element Doc
Element → Text
Doc
Element
Doc
<P>
Doc
Element
Doc
<EM>
Doc
</EM>
Element
Doc
<OL>
List
</OL>
Text
e
ListItem
List
<LI>
Doc
ListItem
e
Element
Doc
<LI>
Element
Doc
Text
e
Text
e
Doc
e
Element
Doc
Text
<P>
Element
Doc
Text
e

Ambiguity in Grammars and Languages

Example : a + a * a



Definition *A Grammar G is called unambiguous if for every $w \in L_G$ there corresponds a unique parse tree. Else it is called ambiguous .*

The problem of determining whether a given grammar G is ambiguous or not is an undecidable problem !

Disambiguation = *Removing ambiguity*

Example

$E \rightarrow E+E$

$E \rightarrow E*E$

$E \rightarrow (E)$

$E \rightarrow I$

$I \rightarrow a$

$E+E*E$?

$= a+a*a$

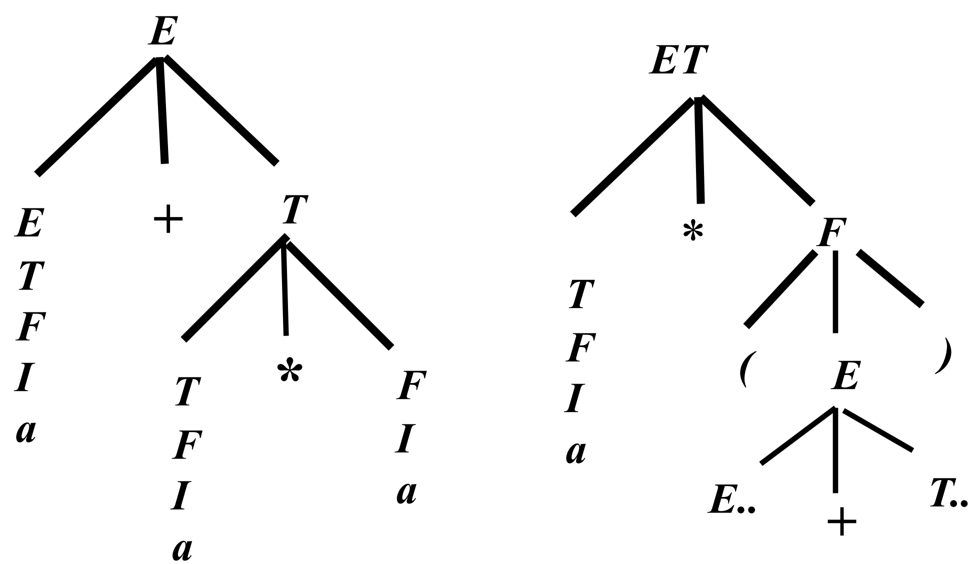
Setting priority of * over +

$E \rightarrow T \mid E+T$ ------→ *E (expression) + is protected*

$T \rightarrow F \mid T*F$ ------→ *T (term) * of factors*

$F \rightarrow I \mid (E)$ ------→ *F (factor) is protected*

$I \rightarrow a$



Inherent Ambiguity (of CFLs)

$L = \{ a^n b^n c^m d^m \mid n \geq 1,\ m \geq 1\} \cup \{ a^n b^m c^m d^n \mid n \geq 1,\ m \geq 1 \}$

$S \rightarrow AB \mid C$
$A \rightarrow aAb \mid ab$
$B \rightarrow cBd \mid cd$
$C \rightarrow aCd \mid aD\ d$
$D \rightarrow bDc \mid bc$

Intuitively a string $a^k b^k c^k d^k$ will have 2 (leftmost derivations) parse trees