

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

A convention on notation

Lower case a, b, c, \dots symbols in T

Upper case A, B, C, \dots symbols in V

Lower case u, w, v, z, \dots symbols in T^*

Upper case X, Y, Z, \dots symbols in $V \cup T$

Lower case Greek $\alpha, \beta, \gamma, \dots$ symbols in $(V \cup T)^*$

Example

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

$V \quad T$

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

$$S \Rightarrow_G 0S1 \Rightarrow 0(0S1)1 = 0^2S1^2 \Rightarrow 0^2e1^2 = 0^21^2$$

$S \rightarrow 0S1 \quad S \rightarrow 0S1 \quad S \rightarrow e$

hence $S \Rightarrow^3 0^21^2$ is a 3-step derivation

Derivations

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

$A \rightarrow \gamma$ be a production of a CFG \mathbf{G} then :

$\alpha A \beta \Rightarrow_{\mathbf{G}} \alpha \gamma \beta$ is called a (one step) derivation in \mathbf{G} ;

in a similar manner we have :

$W \Rightarrow_{\mathbf{G}}^n U$ and $W \Rightarrow_{\mathbf{G}}^ U$ are ***n*-step** and finite step derivations in \mathbf{G}*

where each step conforms to the rule for the one step derivation above.

Definition

- The language $L_G(A)$ generated by a nonterminal variable A of a grammar G is given by : $L_G(A) := \{v \in T^* \mid A \Rightarrow_G^* v\}$
- the language L_G generated by G is $L_G := L_G(S)$, where T and S denote the set of terminal symbols and the start symbol of G respectively.

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

3 Examples of **CFGs**

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

$$E \rightarrow \sigma_1 \mid \sigma_2 \mid \dots \mid \sigma_n \mid e \mid \emptyset \mid E+E \mid E.E \mid E^* \mid (E)$$

There are $n+6$ productions with $n = |\Sigma|$ where :

$$V = \{ E \},$$

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

$P =$ 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. $x_0, x_1, x_2 \dots$; variables with binary indices)

$$V = \{ E, I, J \}$$

$$T = \{ 0, 1, x, +, *, (,) \}$$

E = (Arithmetic) Expression

I = Identifier, J = Identifier trailer

Productions :

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

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

P = 11 productions given next

$$S = E$$

Example for arithmetic expressions

$E \Rightarrow^3 x_1 * (x_0 + 11)$ in ordinary notation : $x_1 \cdot (x_0 + 3)$

$E \Rightarrow E * E \Rightarrow I * E \Rightarrow x_1 J * E \Rightarrow x_1 e * E \Rightarrow^2 x_1 * (E + E) \Rightarrow x_1 * (I + E) \Rightarrow x_1 * (x_0 + E) \Rightarrow x_1 * (x_0 + I) \Rightarrow x_1 * (x_0 + 1J) \Rightarrow^2 x_1 * (x_0 + 11)$

(3) The Grammar of Balanced Parentheses

$V = \{ E \}$

$T = \{ (,) \}$

$P =$ the 3 productions below

$S = E$

$P :$

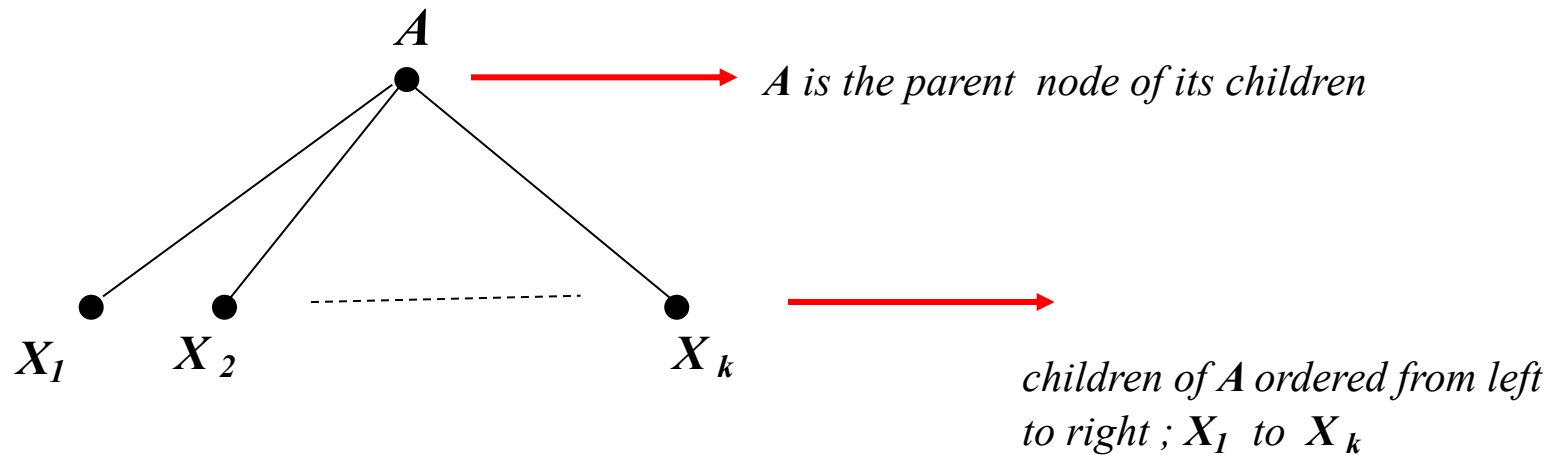
$E \rightarrow EE \mid (E) \mid e$

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 $A \rightarrow X_1 X_2 \dots X_k$ where each $X_j \in V \cup T$, 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 r and ordered sequence of children (leaf) nodes (n_1, \dots, n_k) is an **ordered tree**.*

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

*For a tree T of depth one the root node t is the **parent** and an **ancestor** of the children nodes (n_1, \dots, n_k) ; and the children nodes are called **siblings** of each other and **descendants** of the root node t .*

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

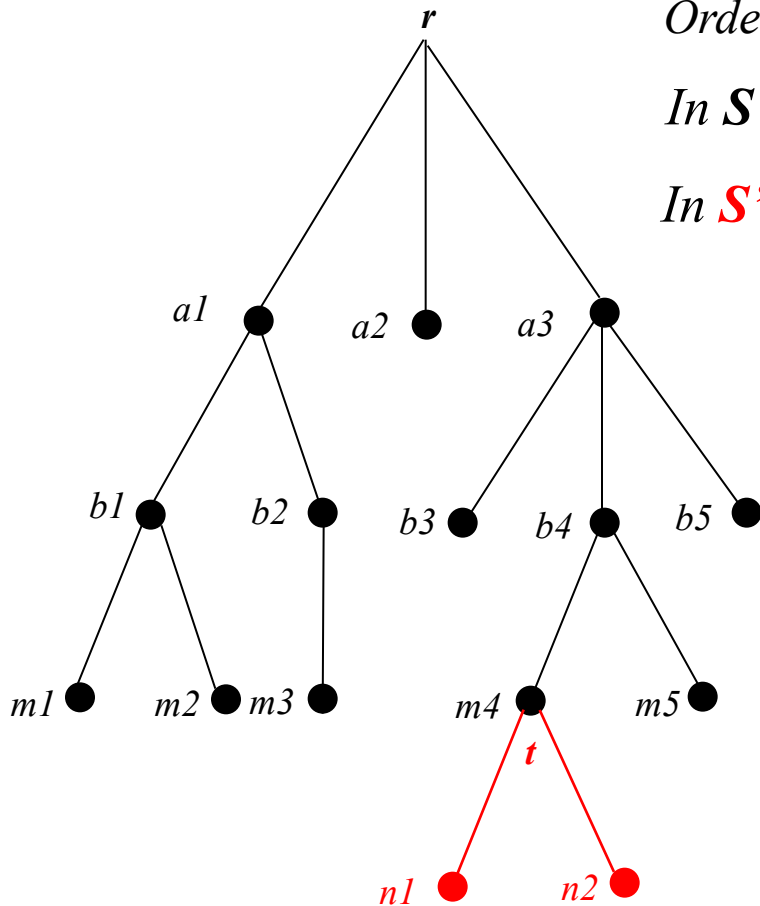
S to S'

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

Ordered leaves of *S'* \longrightarrow *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*



Derivations and Parse Trees

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

The parse tree is obtained by replacing each non-terminal corresponding to the production used— starting from 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

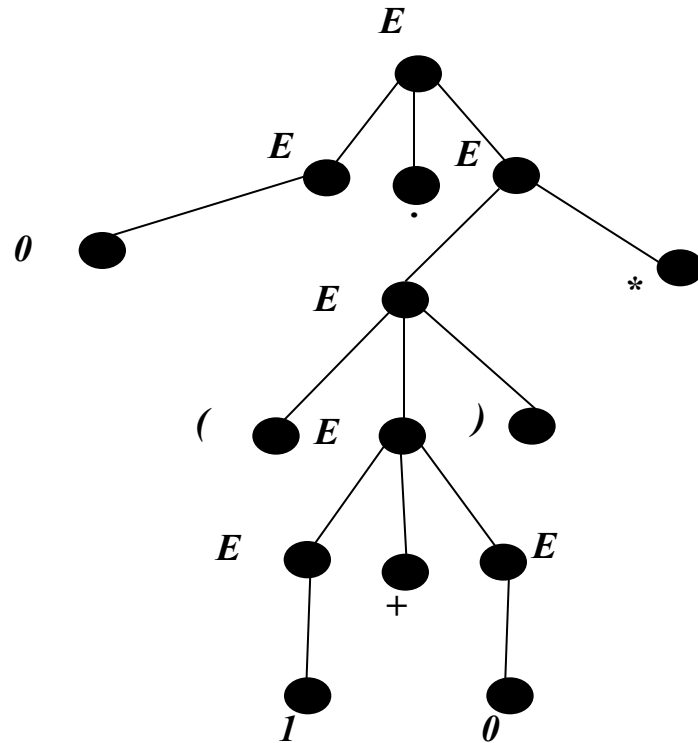
Definition A derivation is called a **leftmost** (**rightmost**) derivation if at each step of the derivation a production is applied to **the** nonterminal at the **leftmost** (**rightmost**) position of the string

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

$$E \Rightarrow_{lm}^7 0.(1+0)^* \quad \begin{array}{ccccccc} E \Rightarrow E.E \Rightarrow 0.E \Rightarrow 0.E^* \Rightarrow 0.(E)^* \Rightarrow 0.(E+E)^* \Rightarrow 0.(1+E)^* \Rightarrow 0.(1+0)^* \\ E \rightarrow E.E \quad E \rightarrow 0 \quad E \rightarrow E^* \quad E \rightarrow (E) \quad E \rightarrow E+E \quad E \rightarrow 1 \quad E \rightarrow 0 \end{array}$$

$$E \Rightarrow_{rm}^7 0.(1+0)^* \quad \begin{array}{ccccccc} E \Rightarrow E.E \Rightarrow E.E^* \Rightarrow E.(E)^* \Rightarrow E.(E+E)^* \Rightarrow E.(E+0)^* \Rightarrow E.(1+0)^* \Rightarrow 0.(1+0)^* \\ E \rightarrow E.E \quad E \rightarrow E^* \quad E \rightarrow (E) \quad E \rightarrow E+E \quad E \rightarrow 0 \quad E \rightarrow 1 \quad E \rightarrow 0 \end{array}$$



$$(1) A \Rightarrow^* w$$

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

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

(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 ($A \Rightarrow X_1 \dots X_k$)

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 $A \Rightarrow X_1 \dots X_k$

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. $Char \rightarrow a|A| \dots$
2. $Text \rightarrow e|Char\ Text$
3. $Doc \rightarrow e|Element\ Doc$
4. $Element \rightarrow Text\ |\\ Doc\ \\ |\<P>\ Doc\ |\\ List\ \\ |\dots$
5. $ListItem \rightarrow \\ Doc$
6. $List \rightarrow e\ |\ ListItem\ List$

 **Subset of CFG**

$V = (Char, Doc, Text, Element, ListItem, List, \dots) ; T = (A-z, \, \, \, \, \, \<P>)$

$\<P>$
 $\\ This\ is\ a\ warning : \$
 $\$
 $\\ Study\ hard.$
 $\\ Do\ your\ homework.$
 $\$
 $\<P>$
 $Else\ you\ will\ fail !$

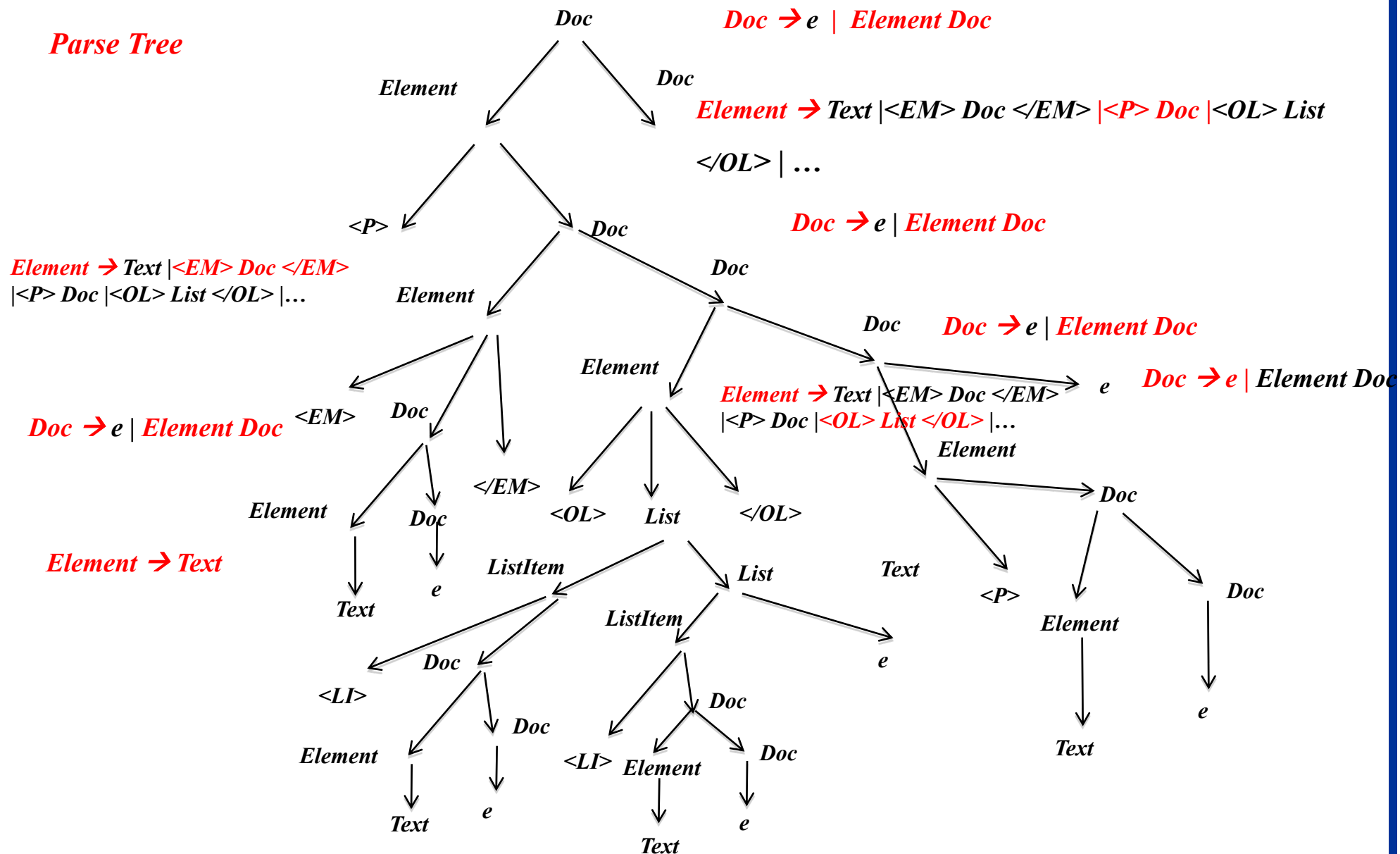
 **HTML
Program**

This is a warning :

1. Study hard.
 2. Do your homework.
- Else you will fail !

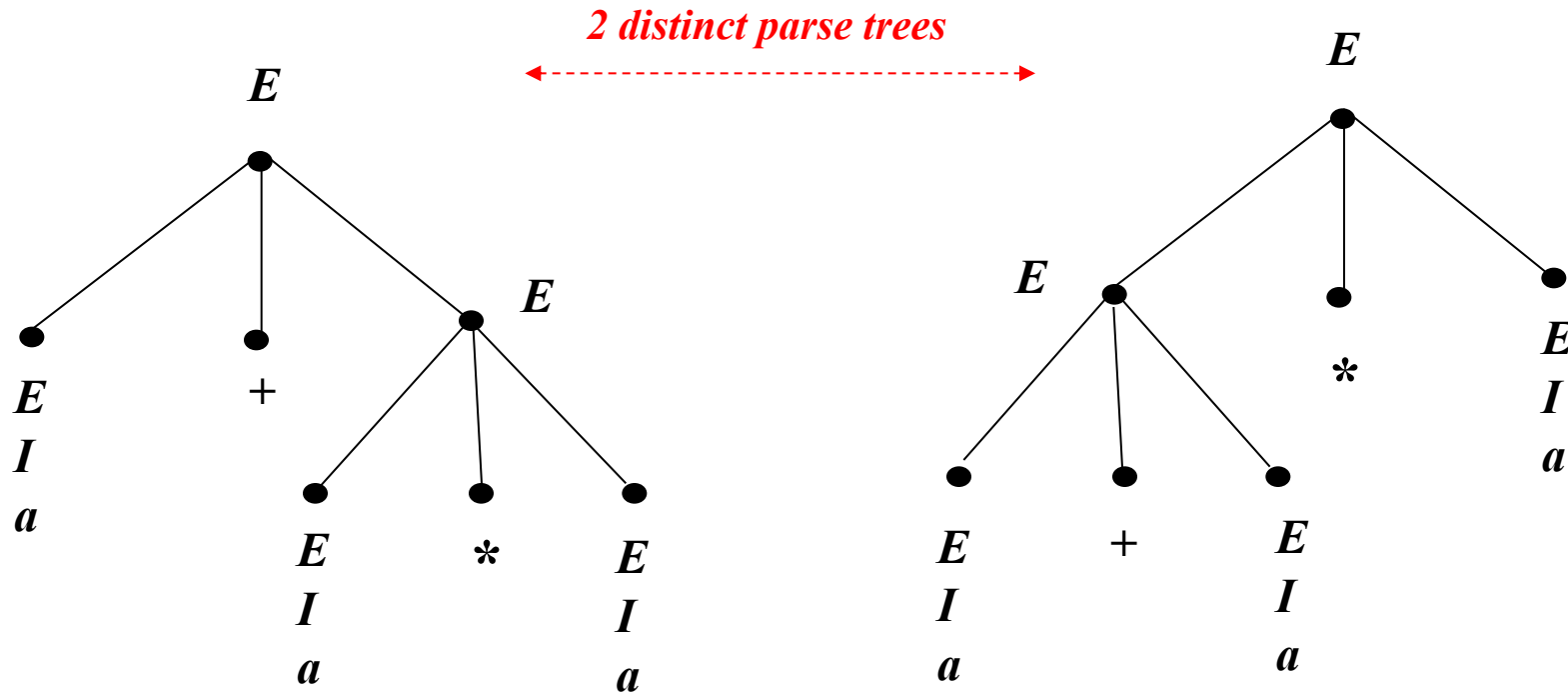
 **Output of
execution**

Parse Tree



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

Setting priority of * over +

$$E \rightarrow E+E$$

$$E \rightarrow E * E$$

$$E \rightarrow (E)$$

$$E \rightarrow I$$

$$I \rightarrow a$$

$$E+E * E ?$$

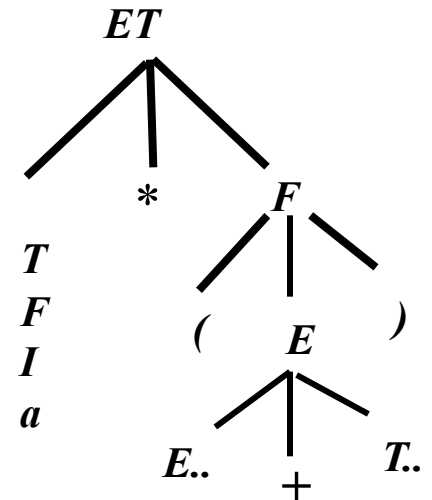
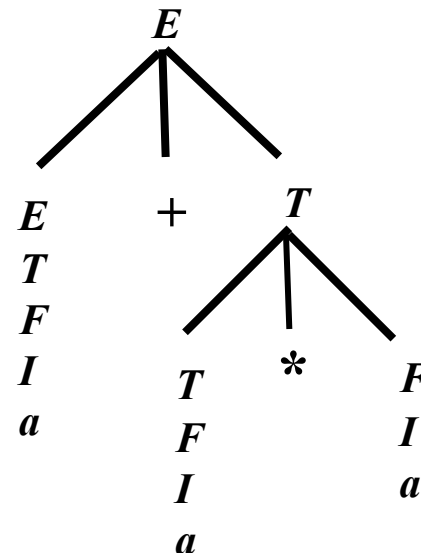
$$= a+a * a$$

$$E \rightarrow T | E+T \quad \text{-----} \rightarrow \quad E \text{ (expression) } + \text{ is protected}$$

$$T \rightarrow F | T * F \quad \text{-----} \rightarrow \quad T \text{ (term) } * \text{ of factors}$$

$$F \rightarrow I | (E) \quad \text{-----} \rightarrow \quad F \text{ (factor) is protected}$$

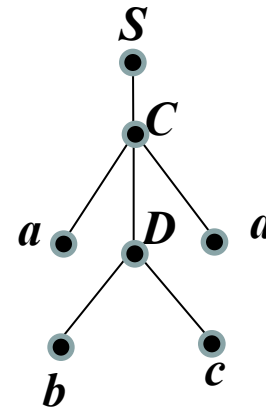
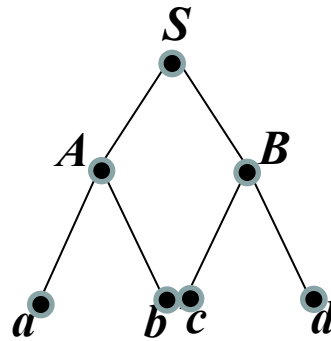
$$I \rightarrow a$$



Inherent Ambiguity (of CFLs)

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

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



*Two parse trees for **abcd***

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