★ Context free Grammar – It is a set of recursive rules used to generate patterns of strings.

A parse tree is a hierarchical representation of <u>terminals</u> and <u>non-terminations</u>

★ terminal – represented using lower case alphabets / symbols (a, b, c, +, x, /). Never appear on the LHS of the production rule. Always on the RHS.

★ Non-terminals – represented using upper case alphabets. (A, B, C) They are always further expanded. Can appear on both LHS and RHS. RHS of a production rule always starts with a non-terminal.

1. All leaf nodes → terminal symbols
2. All interior nodes → non terminal symbols
3. Inorder traversal gives you the original input.

→ These symbols represent the derivation of the grammar to yield input strings.

★ Start symbol of the grammar must be used at the root of the parse tree. They appear at the beginning of the production rule.

★ Leaves of the parse tree represents terminals.

★ Epsilon (ε) – empty

★ Production Rule – Set of rules for replacing non-terminal symbols.

Production Rule

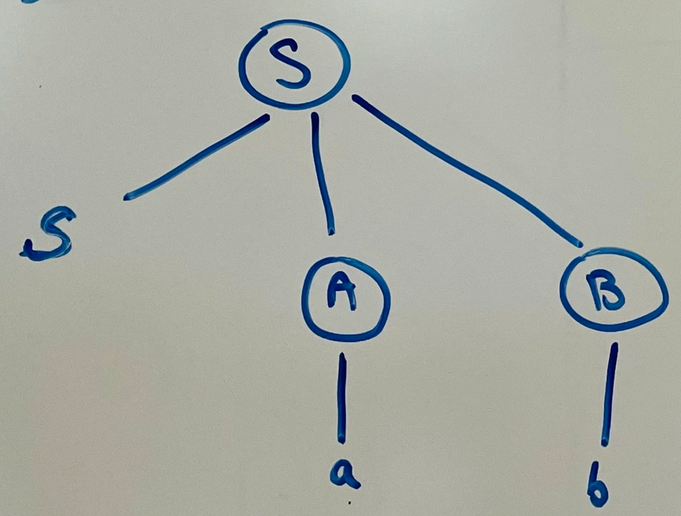
Example 1

$S \rightarrow sAB$

$A \rightarrow a$

$B \rightarrow b$

input string: "sab"



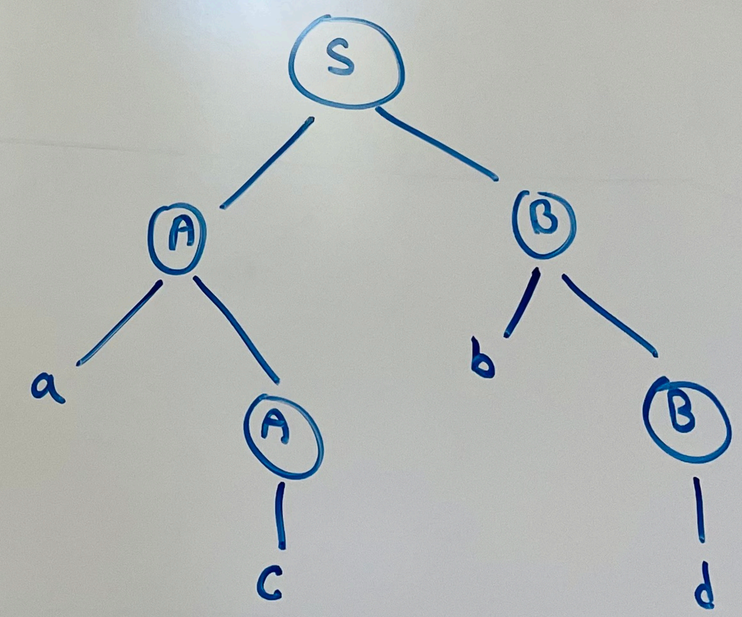
Example 2

$S \rightarrow AB$

$A \rightarrow c \mid aA$

$B \rightarrow d \mid bB$

input string: acbd

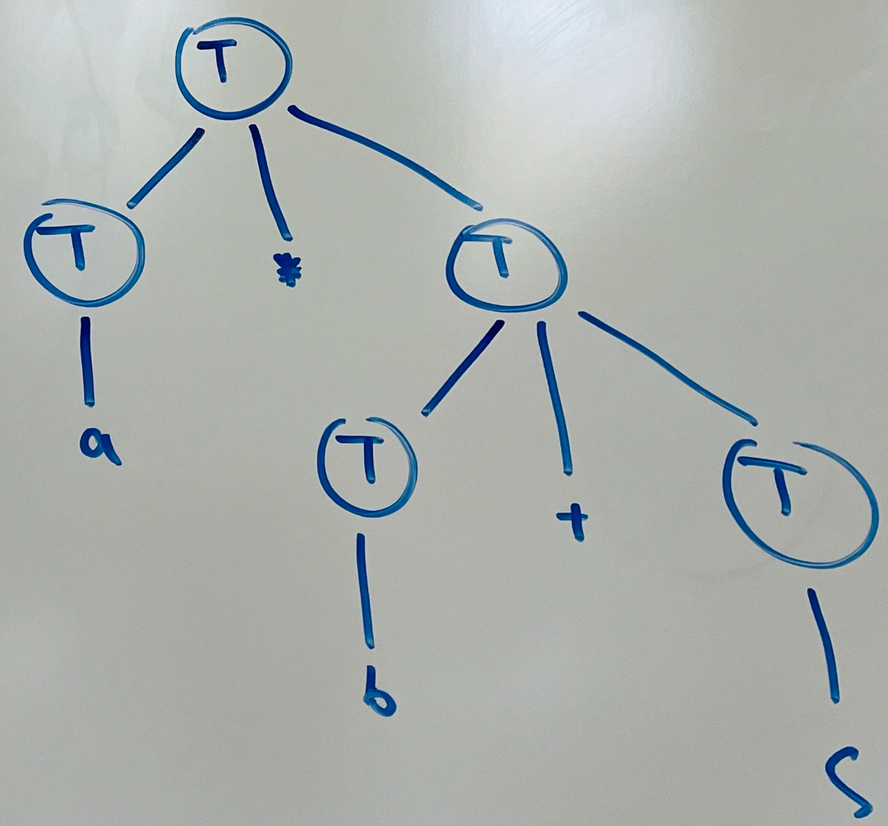


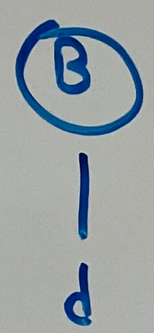
Example 3

(*) $T = T + T \mid T * T$

$T = a \mid b \mid c$

input string: a * b + c





★ Context Free Grammars is a set of recursive rules used to generate patterns of strings.

★ A parse tree is a hierarchical representation of <u>terminals</u> and <u>non-terminals</u>.

★ terminals ⟶ represented using lower case alphabets / symbols (a, b, c, d, +, *)

Never appears on the LHS of a production rule, always on the RHS.

★ non-terminals ⟶ represented using upper case alphabets. (A, B, C)

They are always further expanded. Can appear on both LHS and RHS of a production rule. The RHS of a production rule will always have a non-terminal.

① All leaf nodes ⟶ terminal symbols

② All interior nodes ⟶ non-terminal symbols

③ Inorder traversal gives you the original input.

These symbols represent the derivation of the grammar to yield input strings.

★ Starting symbol of the grammar must be used at the root of the parse tree. They are at the beginning of the production rule.

★ Epsilon (ε) - empty

★ Production rule - set of rules for replacing non-terminal symbols.

Production Rule

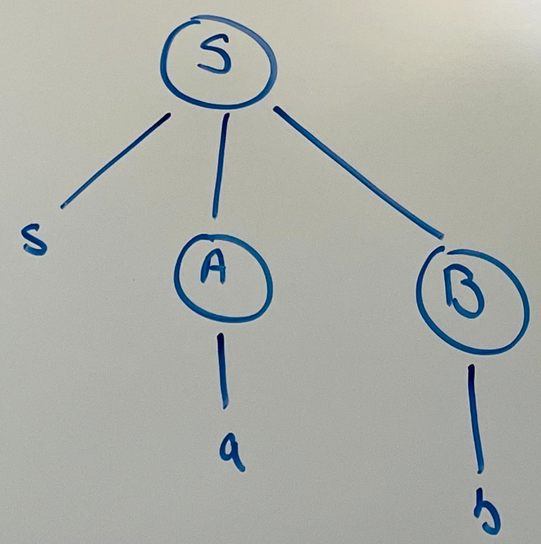
* Example 1

(*) $S \longrightarrow sAB$

$A \longrightarrow a$

$B \longrightarrow b$

input string: sab



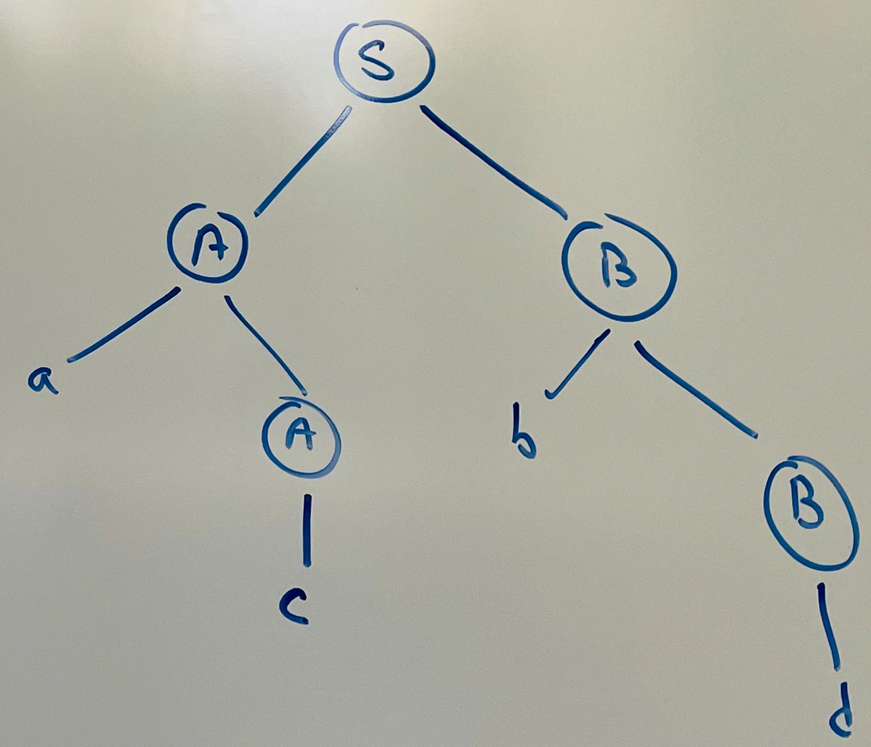
Example 2

(*) $S \longrightarrow AB$

$A \longrightarrow c \mid aA$

$B \longrightarrow d \mid bB$

input sting: acbd

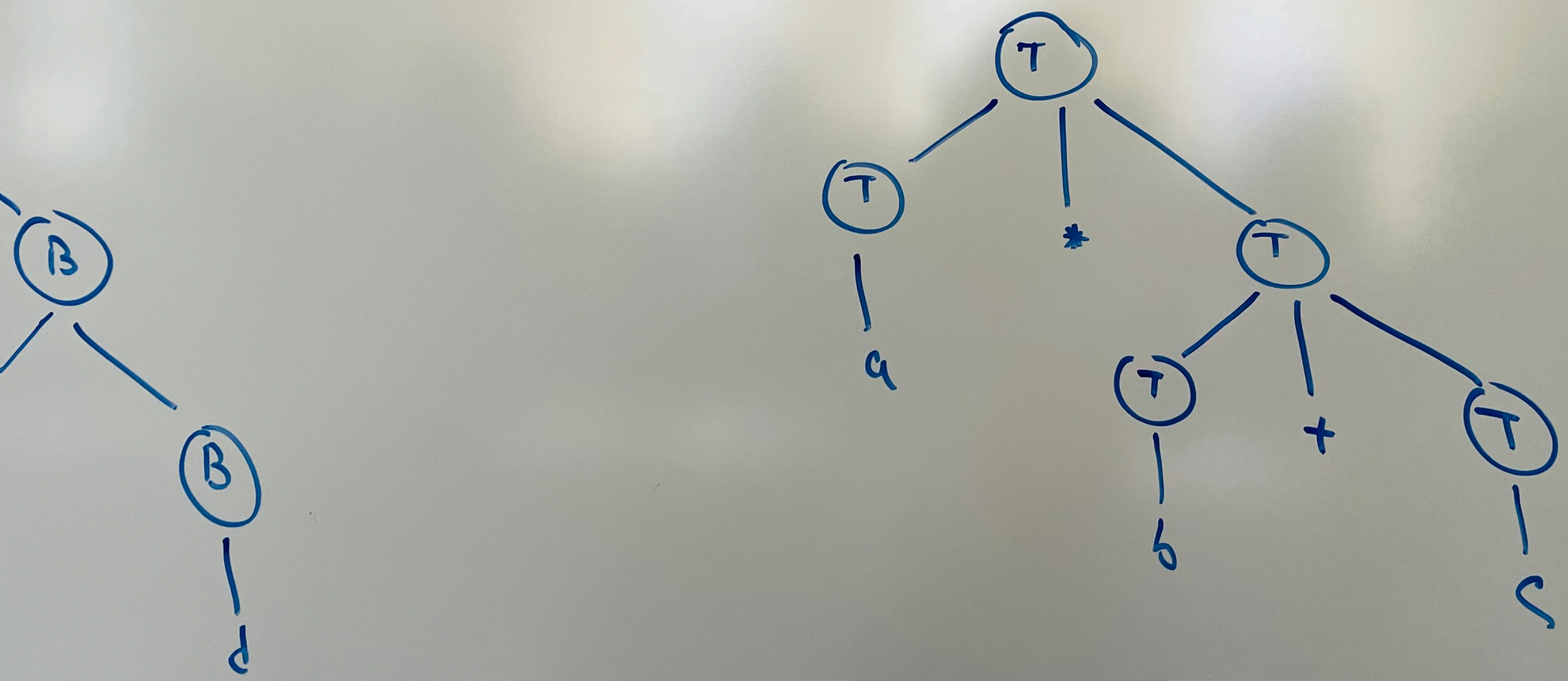


Example 3

(*) $T = T + T \mid T * T$

$T = a \mid b \mid c$

input string: a * b + c



★ Example 4

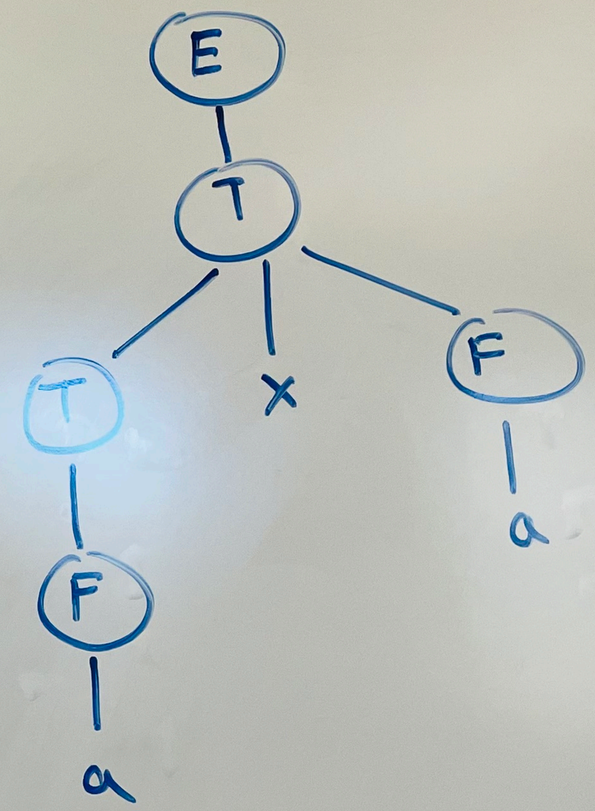
(*) $E \longrightarrow E + T \mid T$

$T \longrightarrow T \times F \mid F$

$F \longrightarrow (E) \mid a$

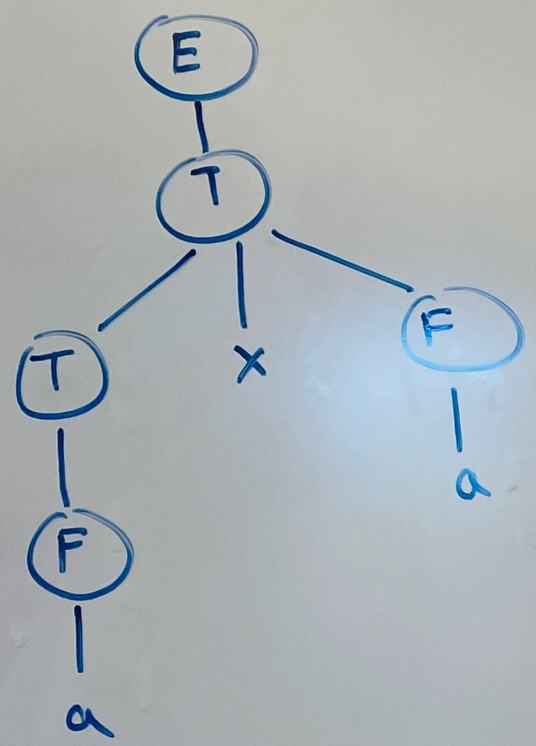
(a) a

(b) a × a

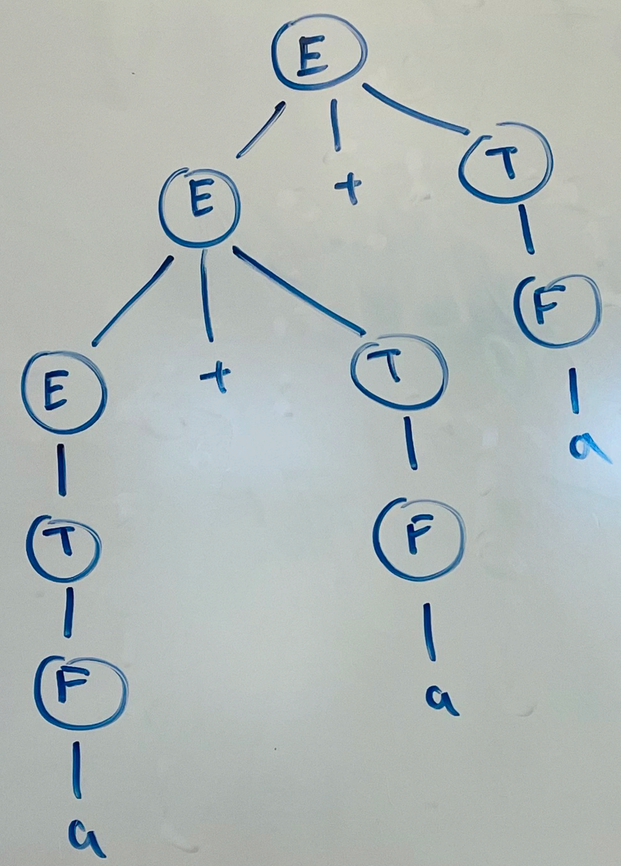


(c)

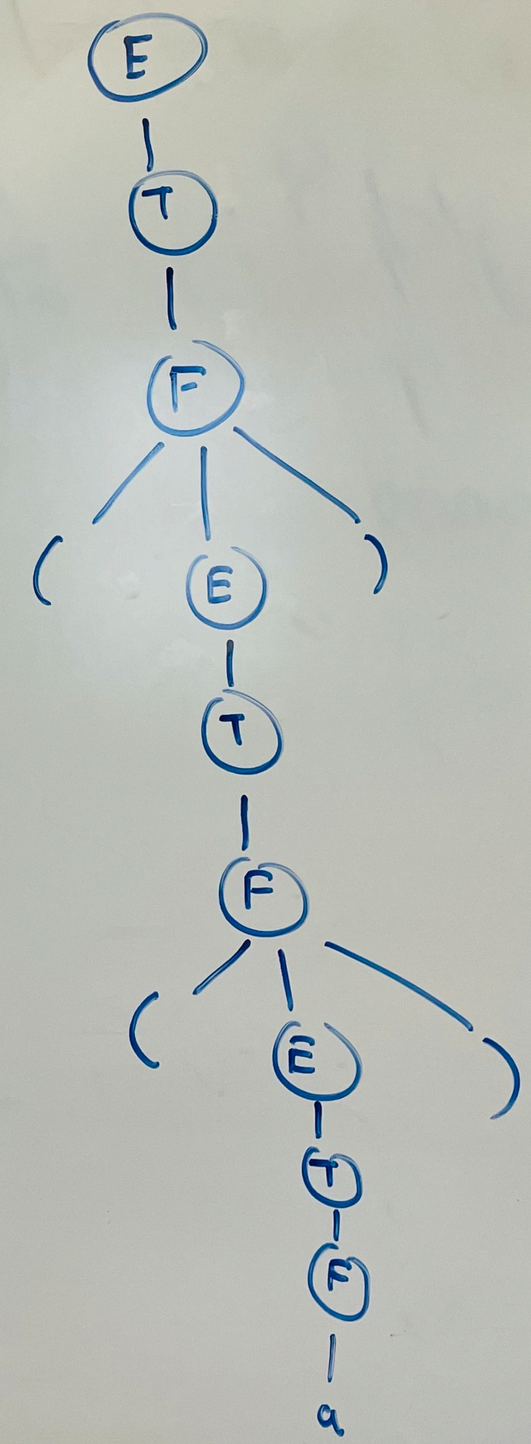
a x a



(c) a + a + a



(d) ((a))

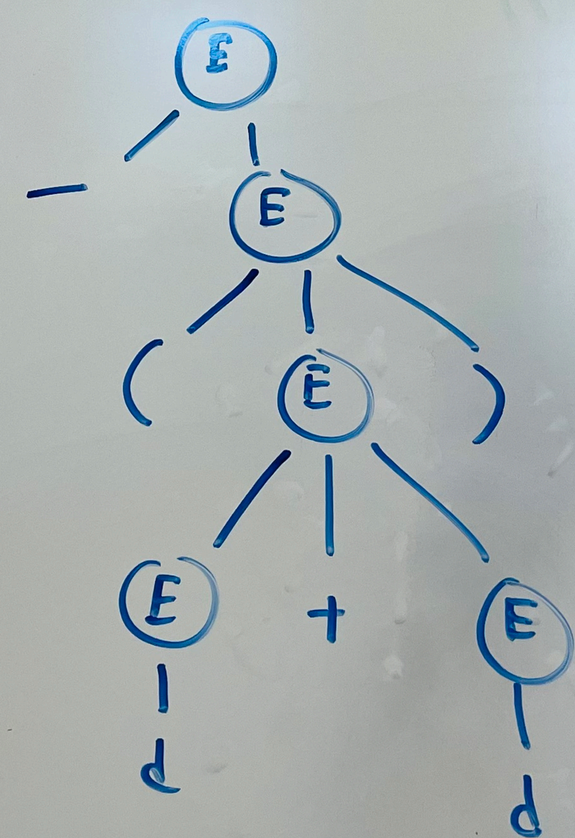


★ Example 5

Given the following grammar:

(*) $E \rightarrow E + E \mid E * E \mid (E) \mid -E \mid d$

Is the string $-(d+d)$ a sentence in the grammar?



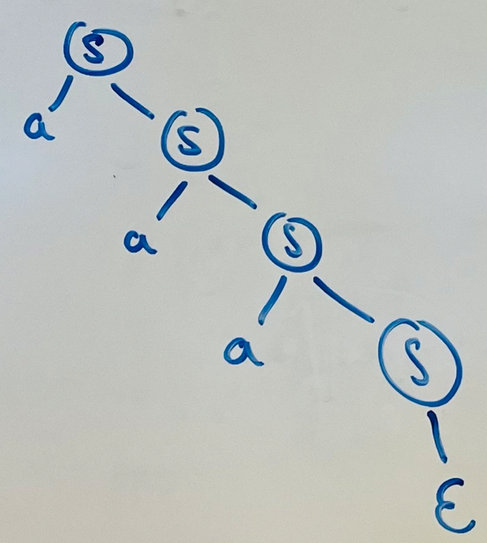
Types of Derivation

Leftmost derivation

Process of deriving input string by expanding the left-most non-terminal

$S \longrightarrow aS \mid \varepsilon$

Generate the following string: a a a

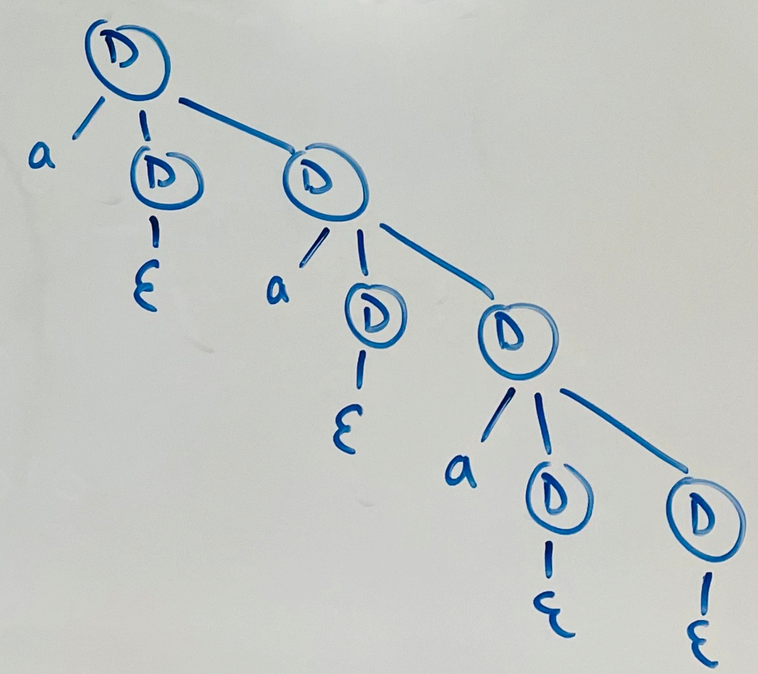


Rightmost Derivation

" right-most non-terminal.

$D \longrightarrow aDD \mid \varepsilon$

Generate the following string: a a a

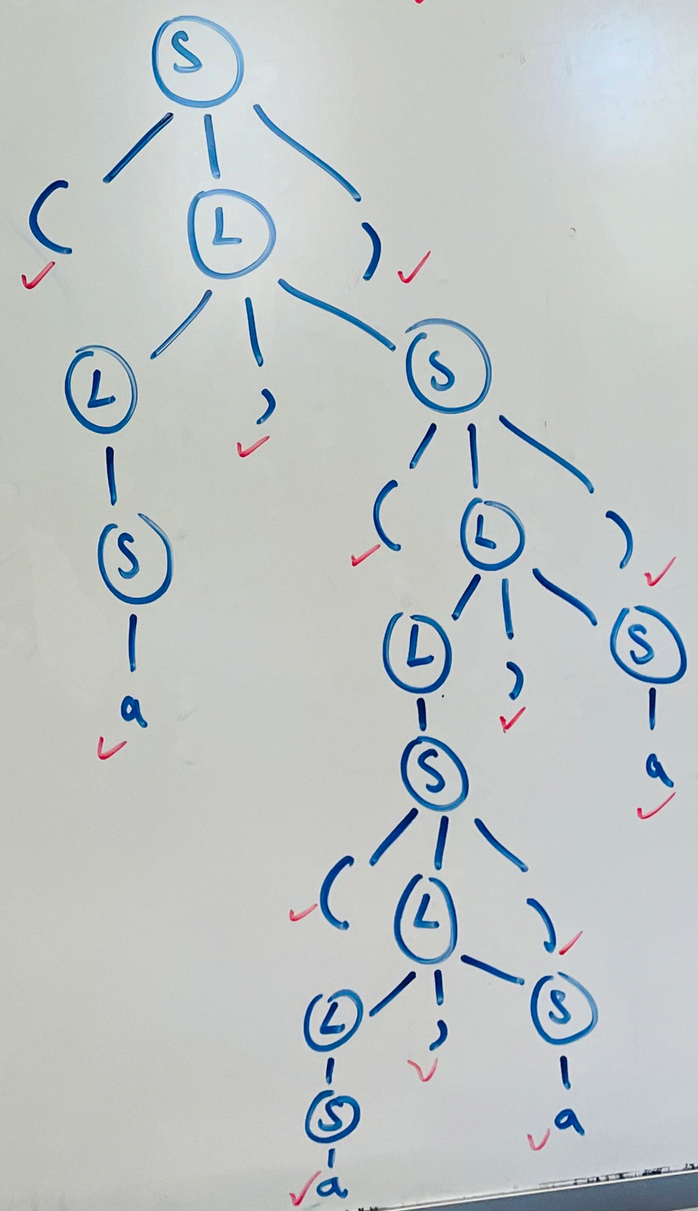


★ Example 6

(*) $S \rightarrow (L) \mid a$

$L \rightarrow L, S \mid S$

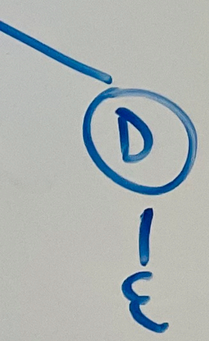
input string: $(a, ((a, a), a))$



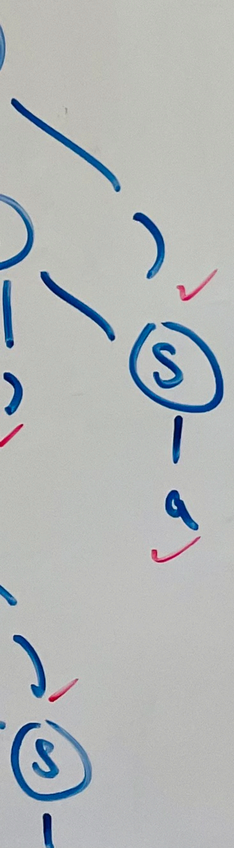
★ Example 7

$S \rightarrow \varepsilon \mid (S)$

input string: $(( )$



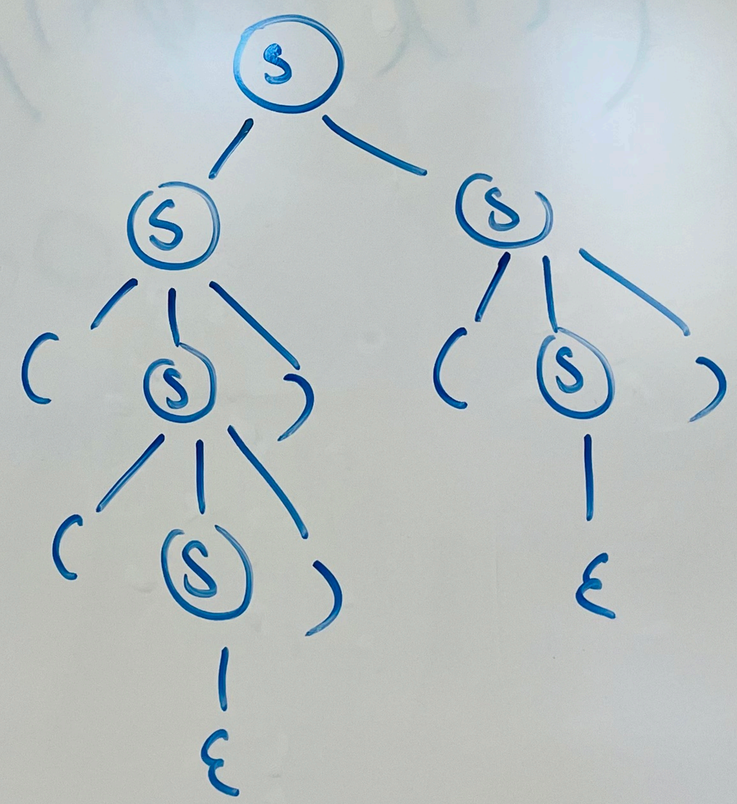
a))



★ Example 7

$$S \rightarrow \varepsilon \mid (S) \mid SS$$

input string: (( )) ( )



se Erase!
ards After
ecture

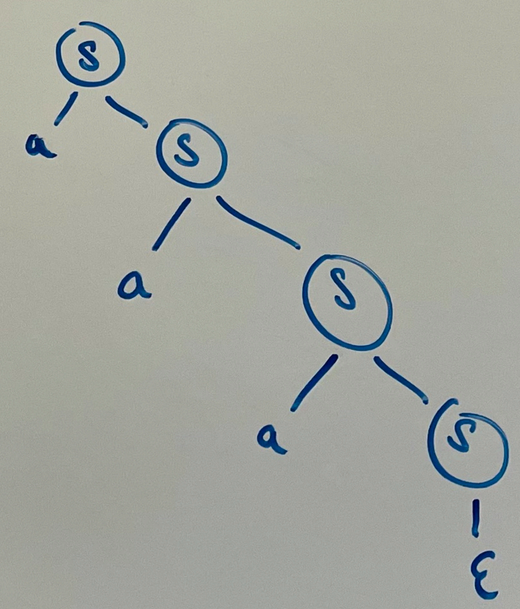
Types of Derivation

Leftmost Derivation

process of deriving input string by expanding the left-most non terminal

$S \rightarrow aS \mid \varepsilon$

input string : aaa

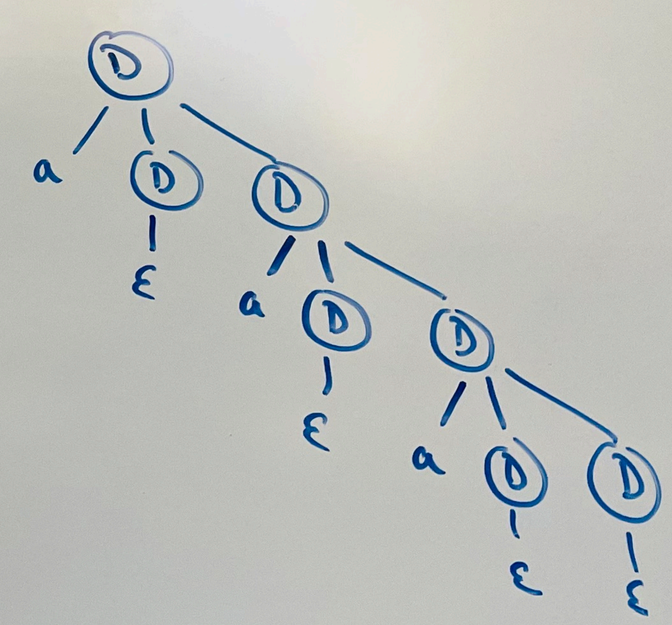


Rightmost Derivation

"
right most non terminal

$D \rightarrow aDD \mid \varepsilon$

input string : aaa



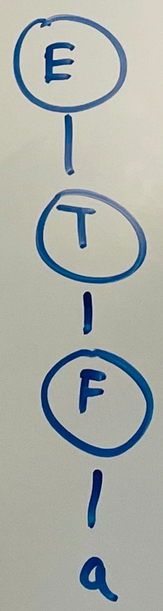
★ Example 4

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

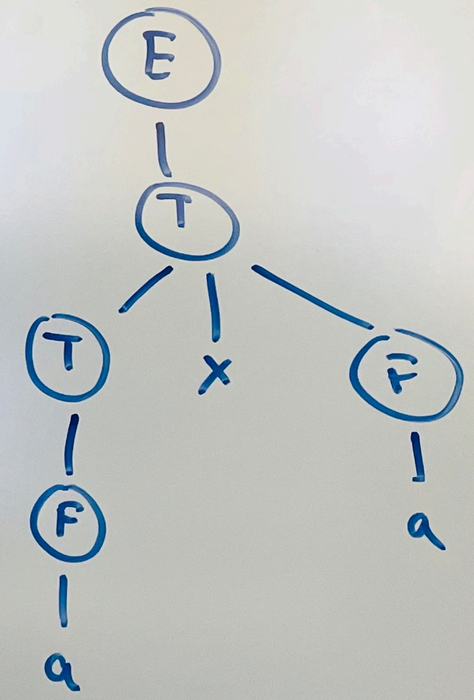
$T \rightarrow T \times F \mid F$

$F \rightarrow (E) \mid a$

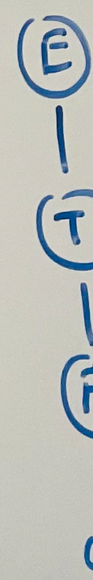
(a) a



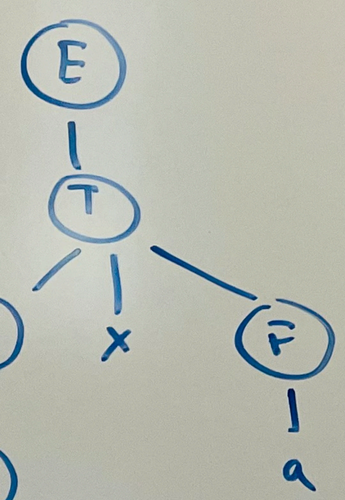
(b) a × a



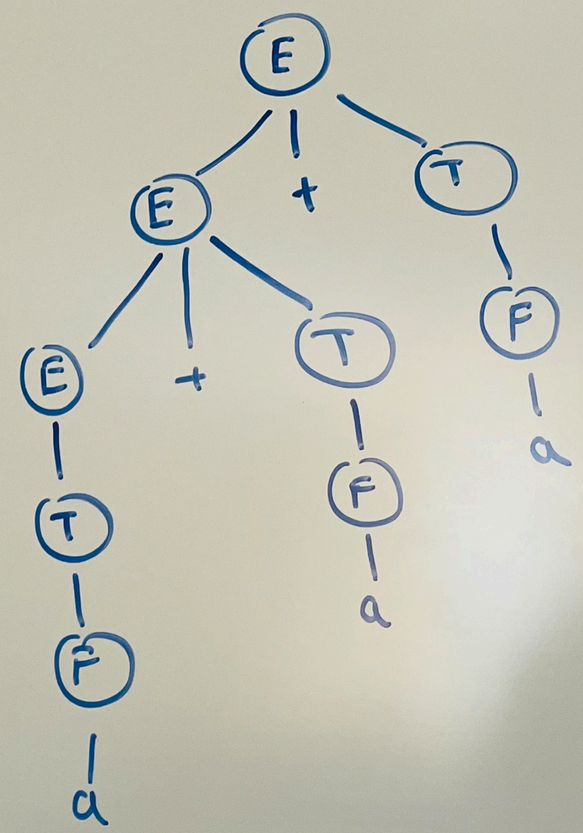
(c) a +



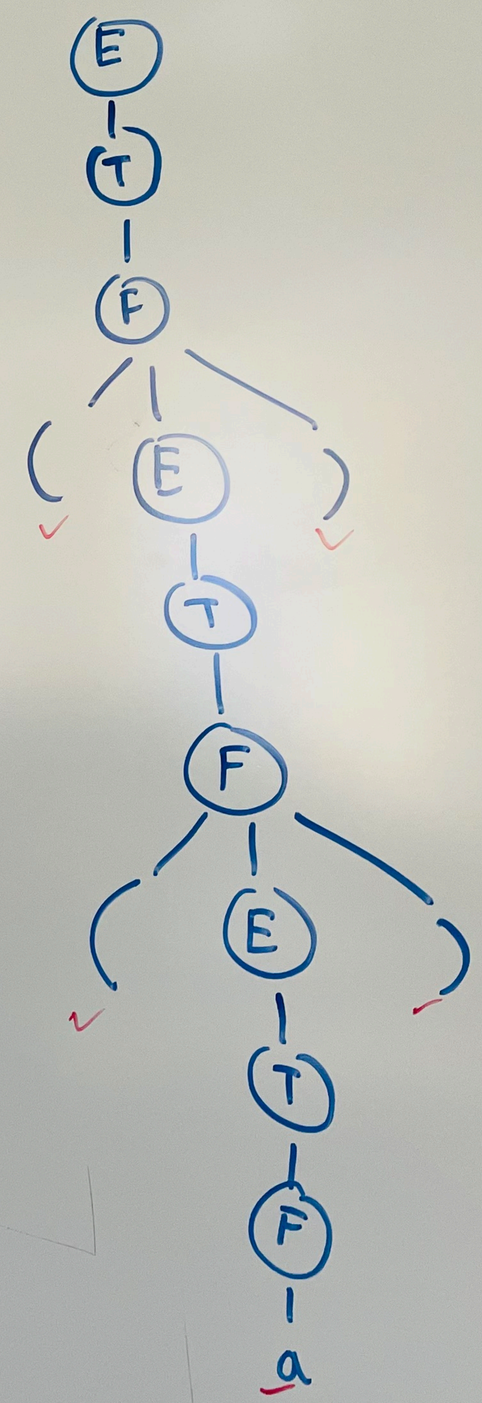
a



(c) a + a + a



(d) ((a))

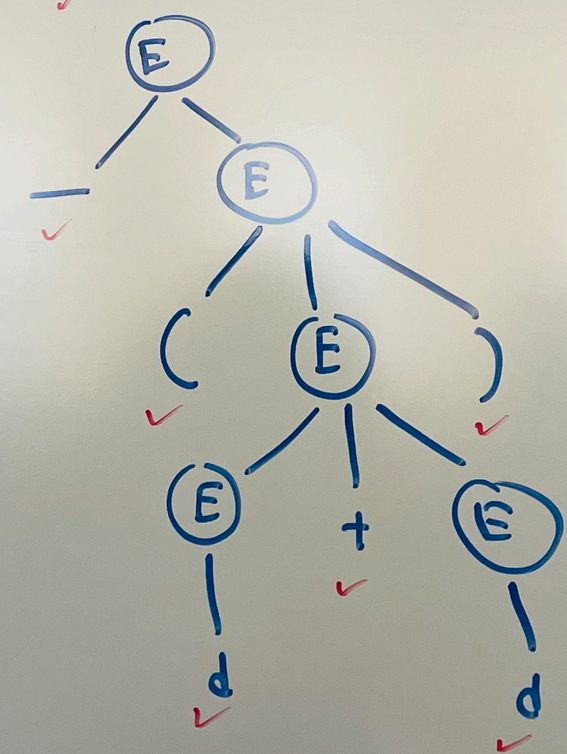


★ Example 5

Given the following grammar:

⊛ $E \rightarrow E + E \mid E * E \mid (E) \mid -E \mid d$

Is the string $-(d+d)$ a sentence in the grammar?



★ Exam

⊛ S —
L

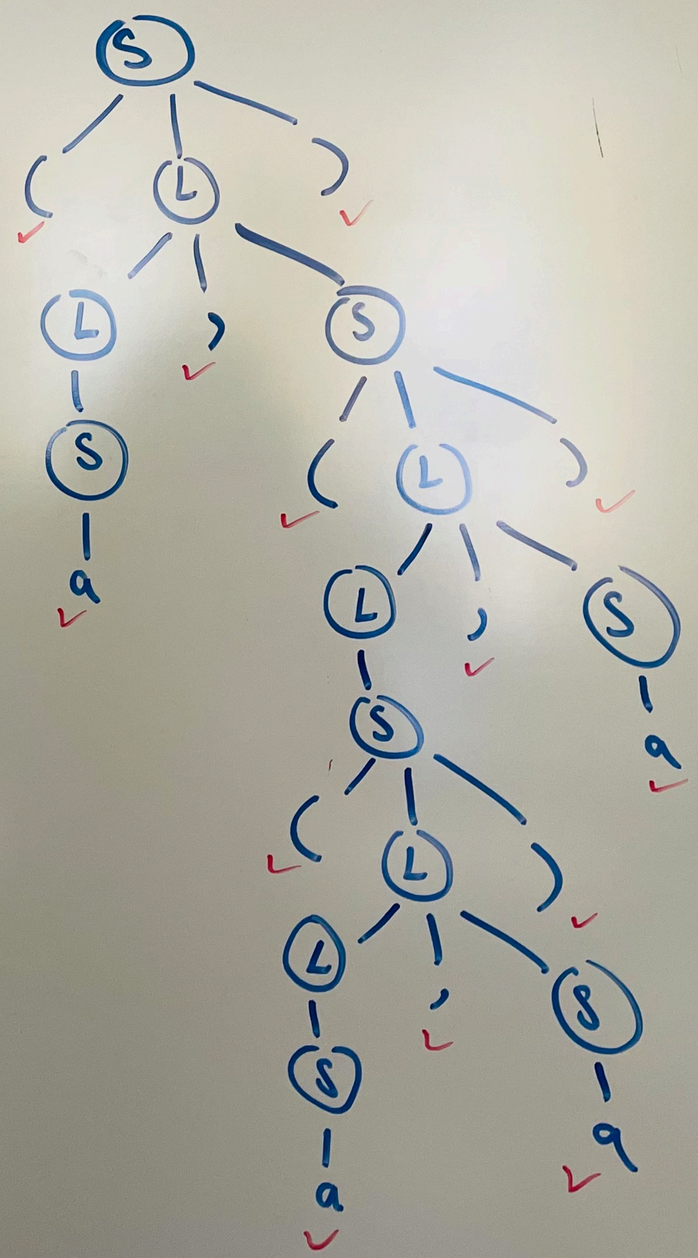
input

* Example 6

(*) $S \rightarrow (L) \mid a$

$L \rightarrow L, S \mid S$

input string: (a, ((a,a), a))

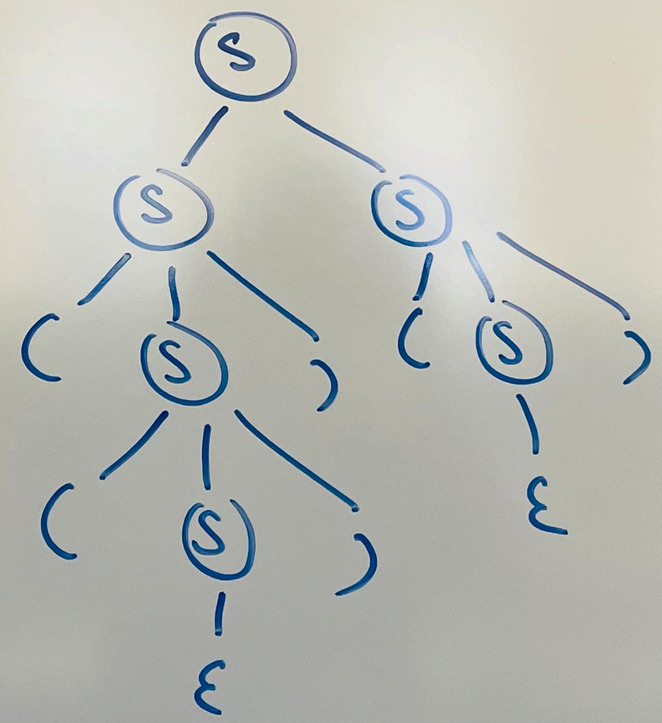


★ Example 7

$S \rightarrow \varepsilon \mid (S) \mid SS$

((( )))

input string: (( )) ( )



* Example 8

Given a context free grammar (CFG) that generates the following language :

$$A = \{ a^i b^j c^k \mid \underbrace{i = j \text{ or } j = k}_{\text{two conditions}} \text{ where } i, j, k \geq 0 \}$$

1st condition: $\left.\begin{array}{l} i = j \\ j = 3 \\ k = 1 \end{array}\right\}$ $a^3 b^3 c^1 = aaabbbc$

2nd condition: $\left.\begin{array}{l} i = 1 \\ j = k \\ k = 3 \end{array}\right\}$ $a^1 b^3 c^3 = abbbccc$

Production Set, S $\Rightarrow$

Let's start by defining tw

* $S1 \longrightarrow A$

$A \longrightarrow a$

$T \longrightarrow c$

rates the

$j = k$ where $i, j, k \geq 0 \}$

onditions

Production Set, S $\Rightarrow$ Let's start with $S1$ $(i=j)$ and $S2$ $(j=k)$

Let's start by defining two paths that S1 can take :

(*) $S1 \longrightarrow AT$

$A \longrightarrow aAb \mid \varepsilon$ $\longrightarrow$ Ensures a and b is generated at the same time. One "a" and One "b"

$T \longrightarrow cT \mid \varepsilon$ $\longrightarrow$ One "c" is generated.

(*)

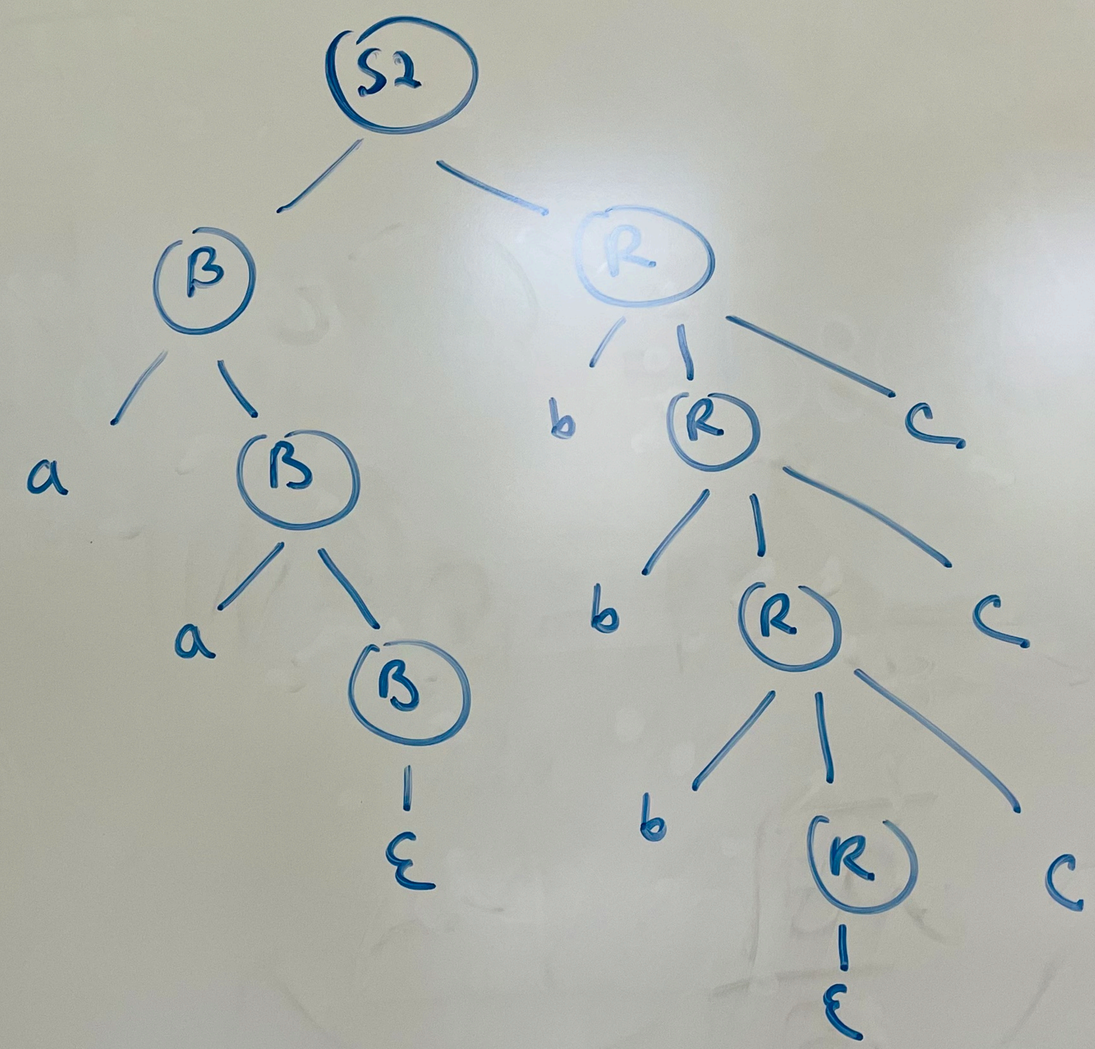
⊗ $S2 \rightarrow BR$

$B \rightarrow aB \mid \varepsilon$ → One "a" is generated

$R \rightarrow bRc \mid \varepsilon$ → One "b" and One "c" is generated.

e "b"

input string : $a^2 b^3 c^3 = aabbbccc$



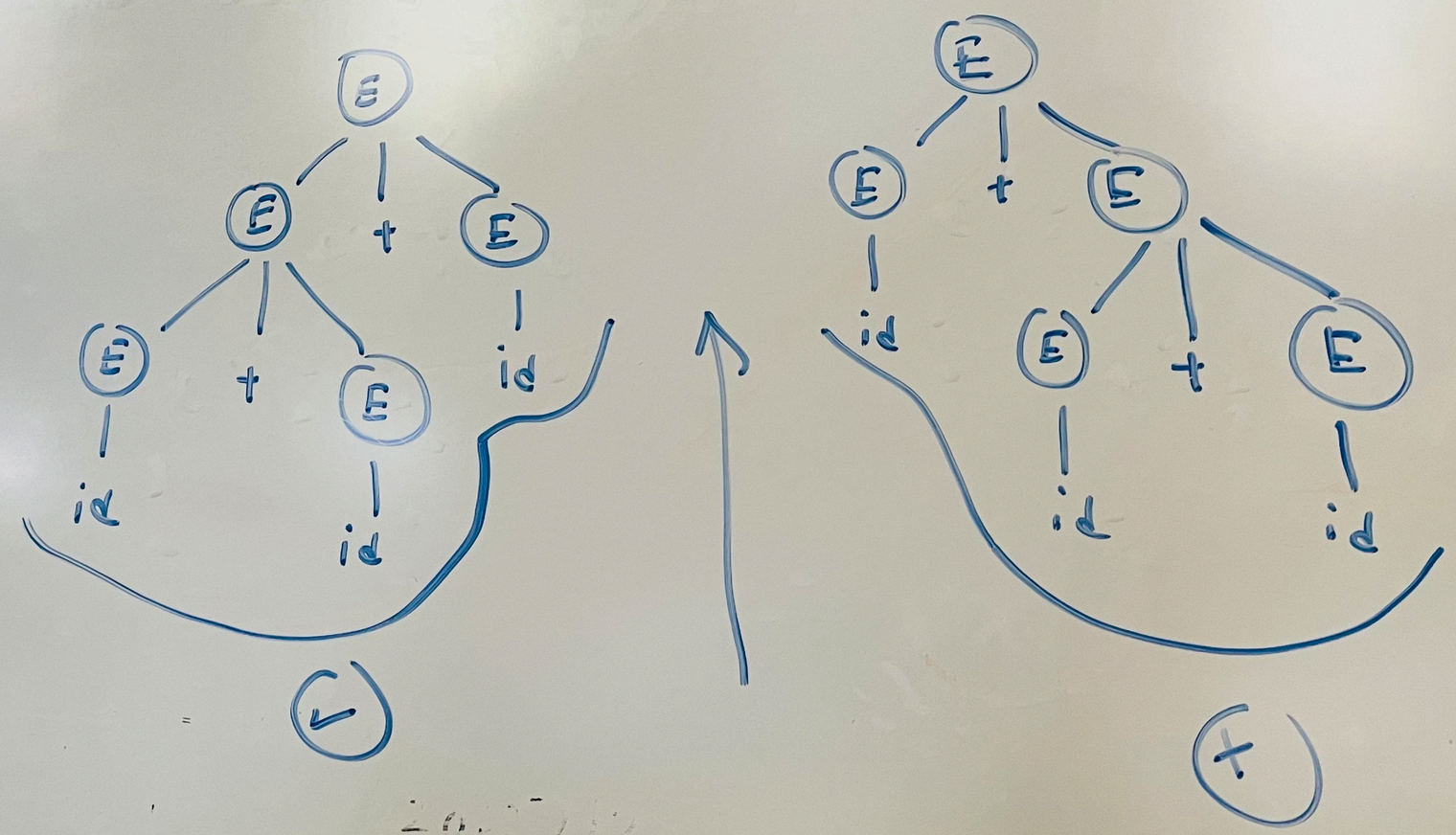
erated.

Removing Ambiguity

★ Example 9

$$E \rightarrow E + E \mid E * E \mid id$$

input string: id + id * id



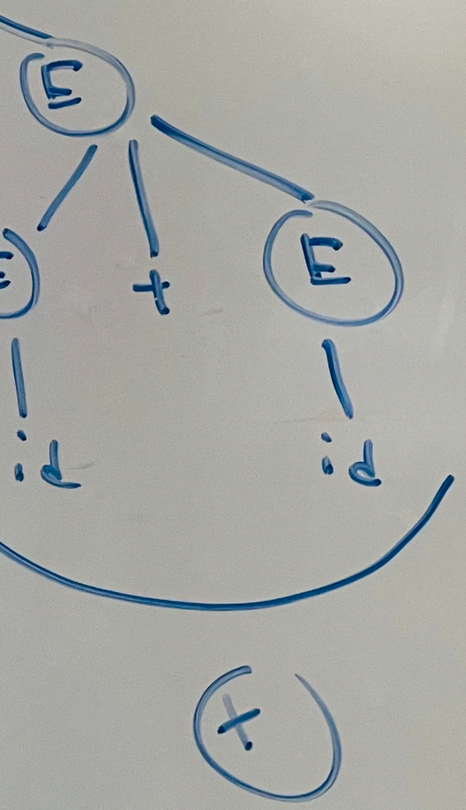
① Precedence in mathematical operation

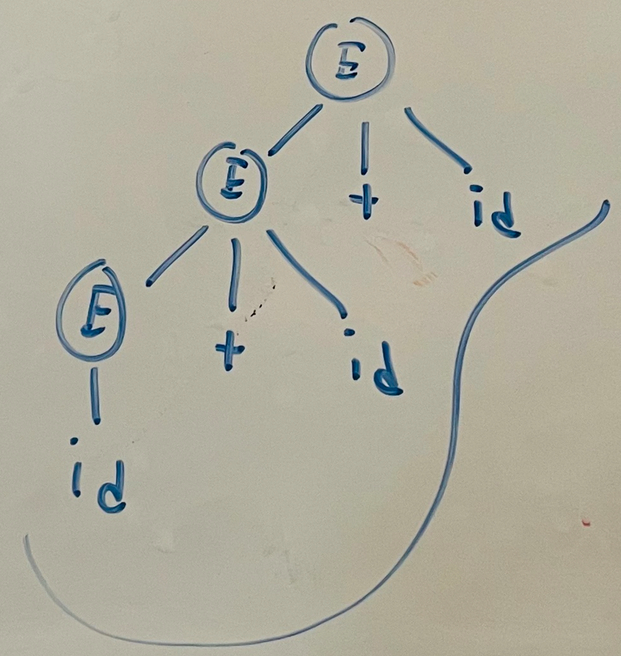
PEMDAS / BODMAS

② Bottom - Top Parsing

Restrict RH growth

$E \rightarrow E + id \mid id$





★ Exa

E →

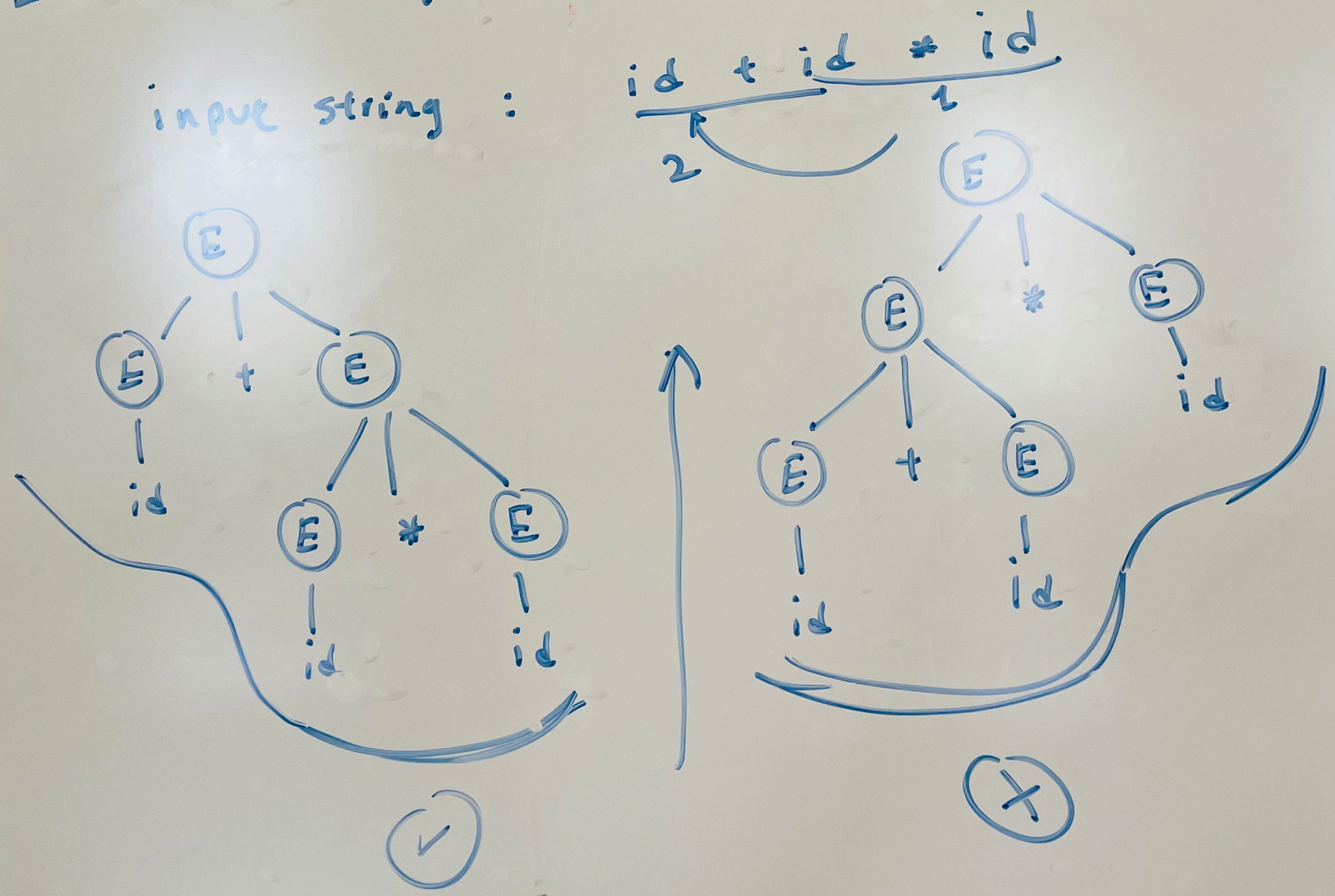
E

id

★ Example 10

$E \rightarrow E + E \mid E * E \mid id$

input string : id + id * id



E →

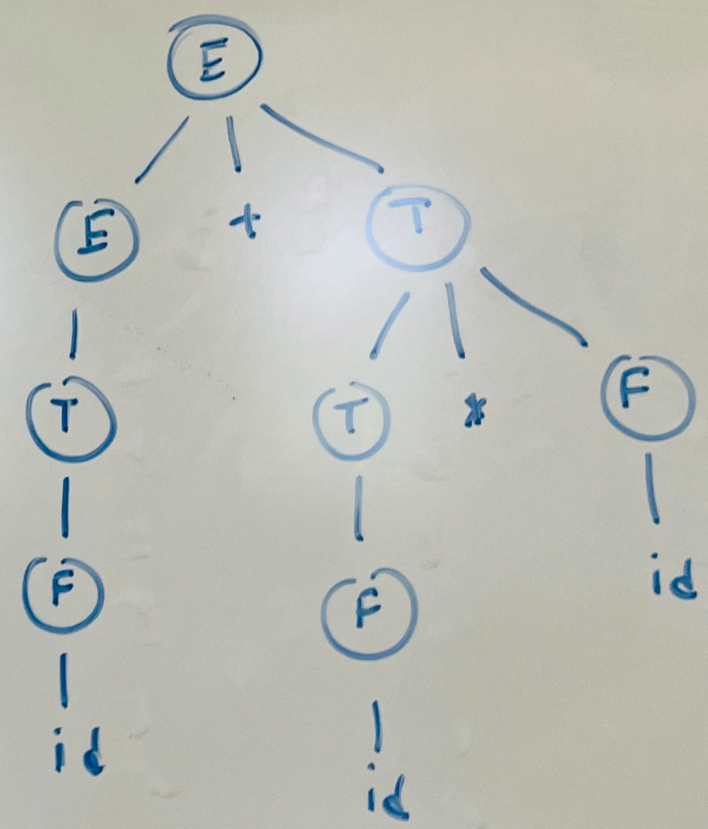
T →

F

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow id$



★ Example 8

Give a context free grammar (CFG) that generates the following language:

$$A = \{ a^i b^j c^k \mid \underbrace{i = j \text{ or } j = k}_{\text{two conditions}} \quad \text{where } i, j, k \geq 0 \}$$

1st condition:

$i = j$
$j = 3$
$k = 1$

$a^3 b^3 c^1 = aaabbbc$

2nd condition:

$i = 1$
$j = k$
$k = 3$

$a^1 b^3 c^3 = abbbccc$

Production Set, S $\Rightarrow$ Let's st

Let's start by defining two pr

(*) $S1 \longrightarrow AT$
$A \longrightarrow aAb$
$T \longrightarrow cT$

(*) $S2 \longrightarrow BR$
$B \longrightarrow aB \mid$
$R \longrightarrow bRc$

the following language:

$, j, k \geq 0\}$

Production Set, S $\Rightarrow$ Let's start S1 $(i=j)$ and S2 $(j=k)$

Let's start by defining two paths that S1 can take

(*) $S1 \longrightarrow AT$

$A \longrightarrow aAb \mid \varepsilon$ $\longrightarrow$ Generates one "a" and one "b" ensures a and b is generated at the Same time.

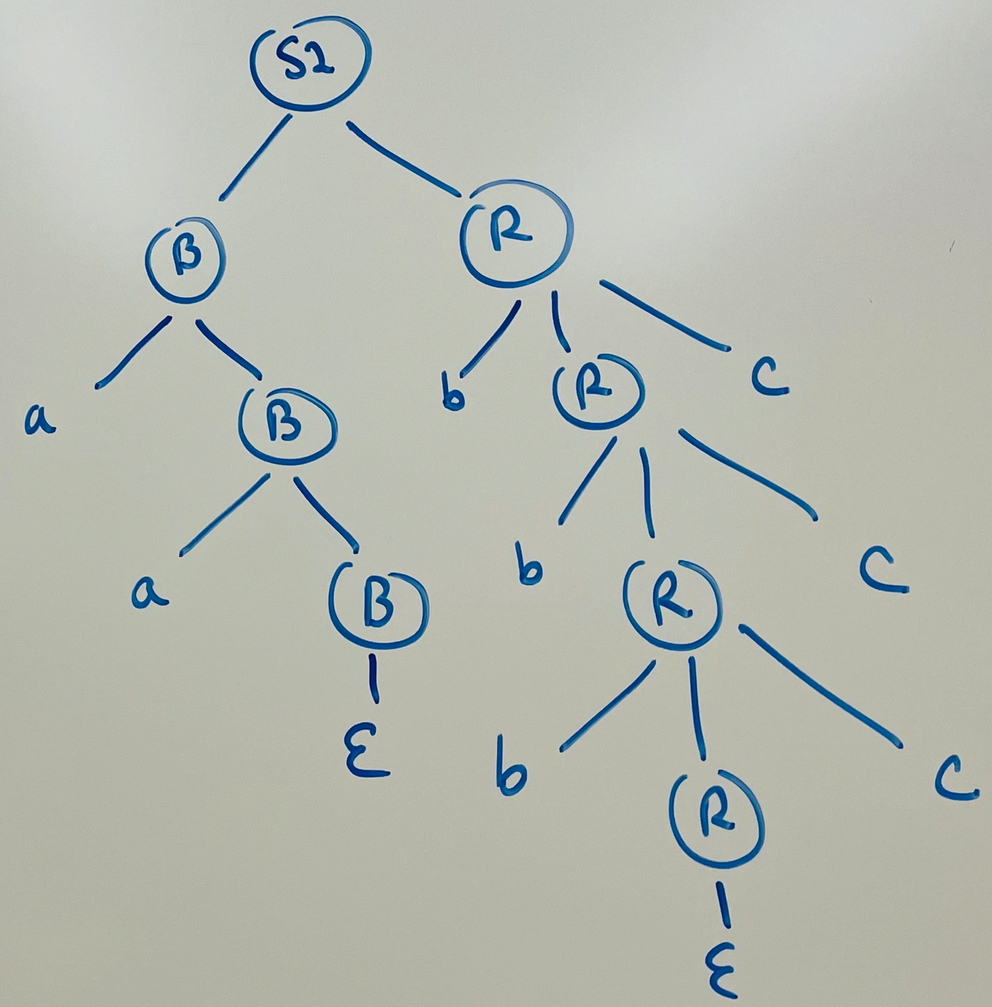
$T \longrightarrow cT \mid \varepsilon$ $\longrightarrow$ Generates one "c"

(*) $S2 \longrightarrow BR$

$B \longrightarrow aB \mid \varepsilon$ $\longrightarrow$ Generates one "a"

$R \longrightarrow bRc \mid \varepsilon$ $\longrightarrow$ Generates one "b" and one "c" Ensures b and c is generated at the Same time.

input string: $a^2 b^3 c^3$ = a a bbb ccc



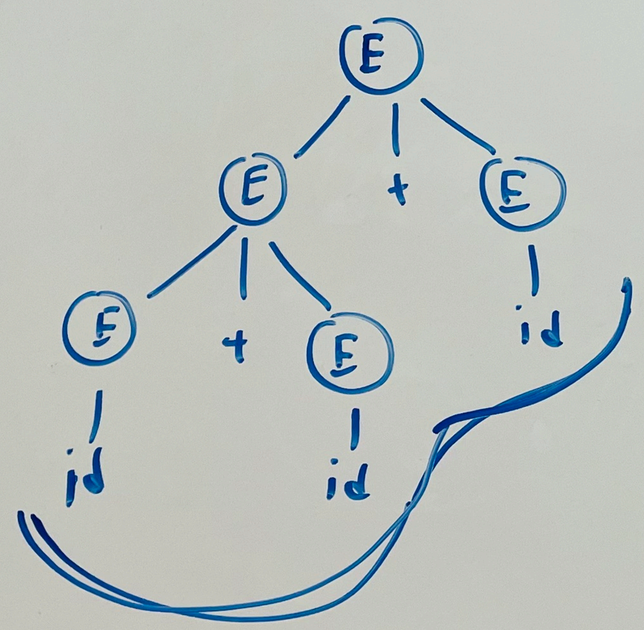
he Same time.

Remove Ambiguity

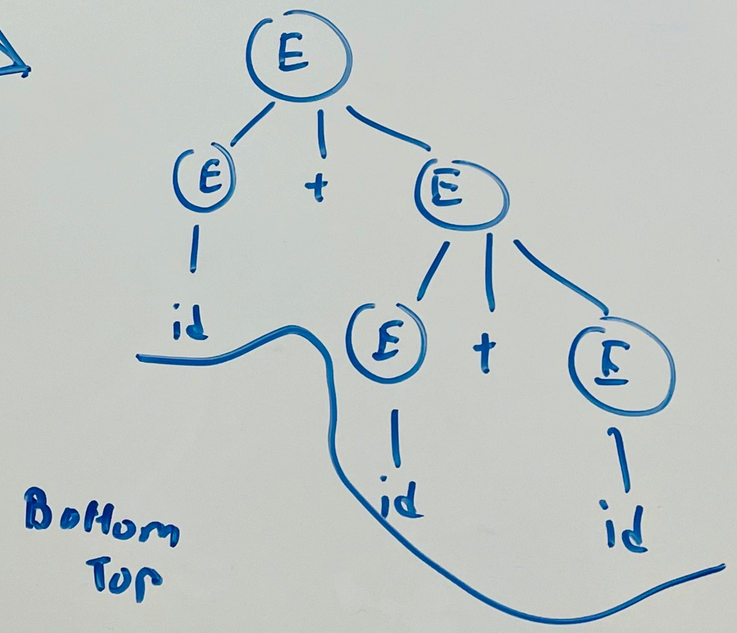
★ Example 9

$$E \longrightarrow E + E \mid E * E \mid id$$

input string: id + id + id



✓



Bottom
Top

✗

★ Prece

★ B

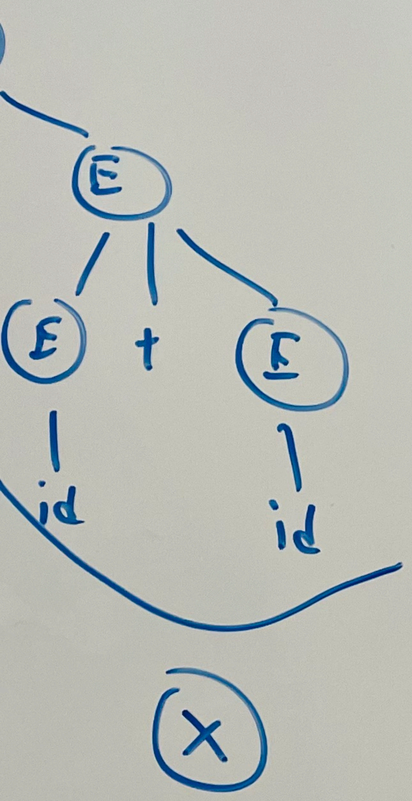
Res

1
2
3

★ Precedence in mathematical operations

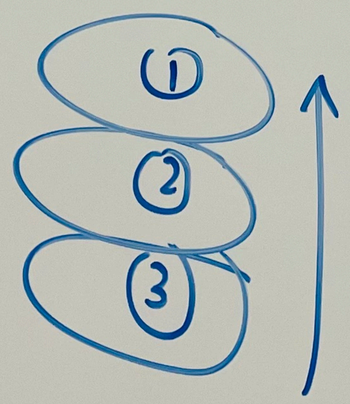
BODMAS / PEMDAS PE MD AS

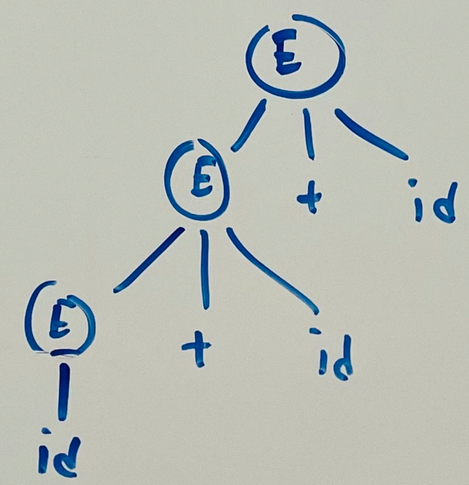
★ Bottom-Top Parsing



Restrict RH growth

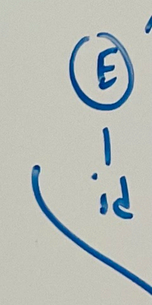
$E \longrightarrow E + id \mid id$





★ Example

E →



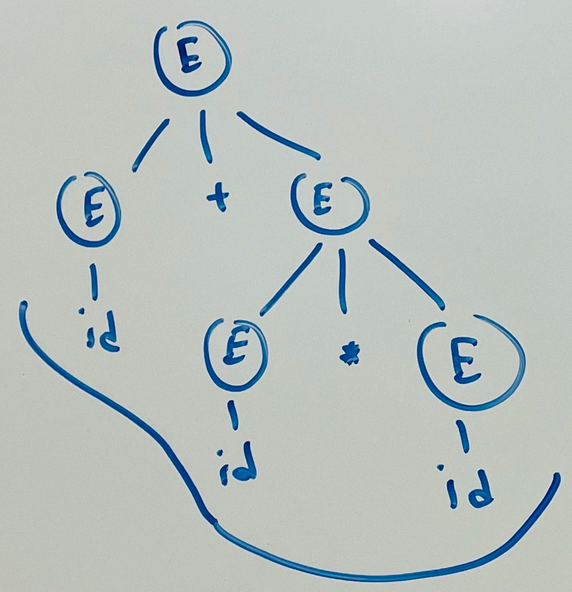
ons

E MD AS

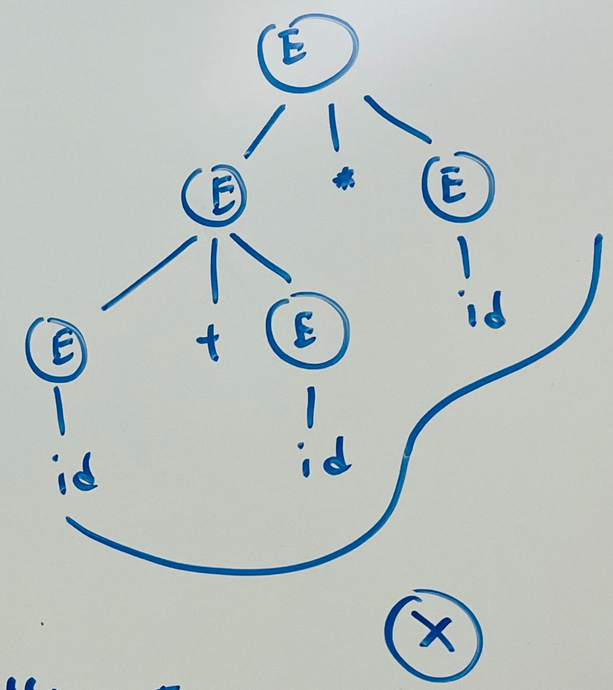
★ Example 10

$E \rightarrow E + E \mid E * E \mid id$

id + id * id



✓



✗

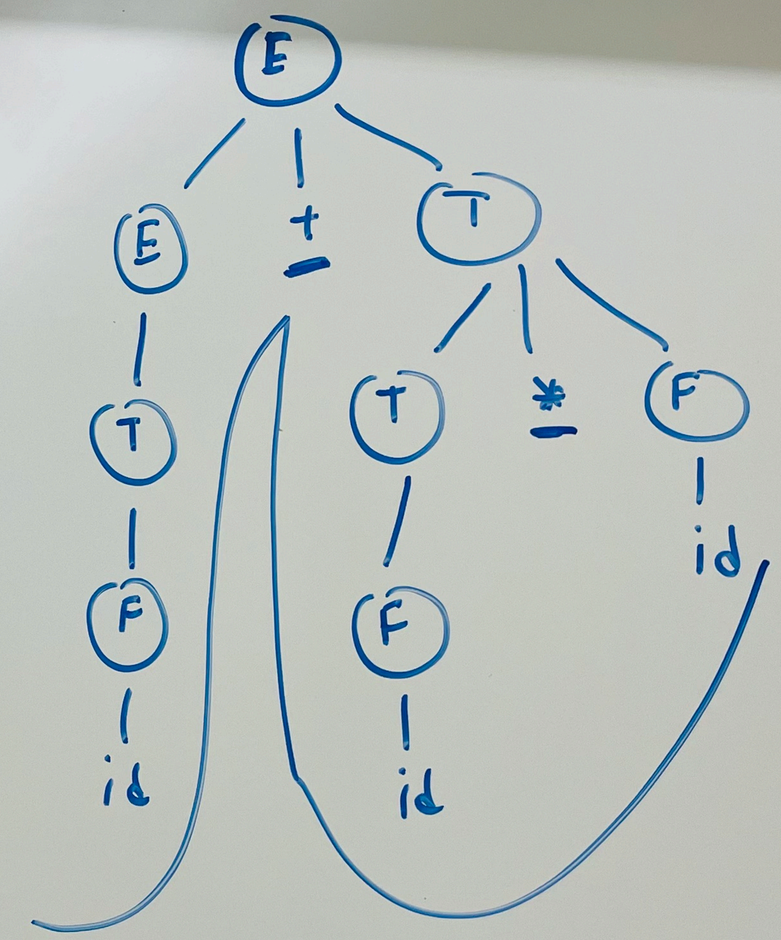
Bottom-Top parsing

E —

T —

F

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow id$