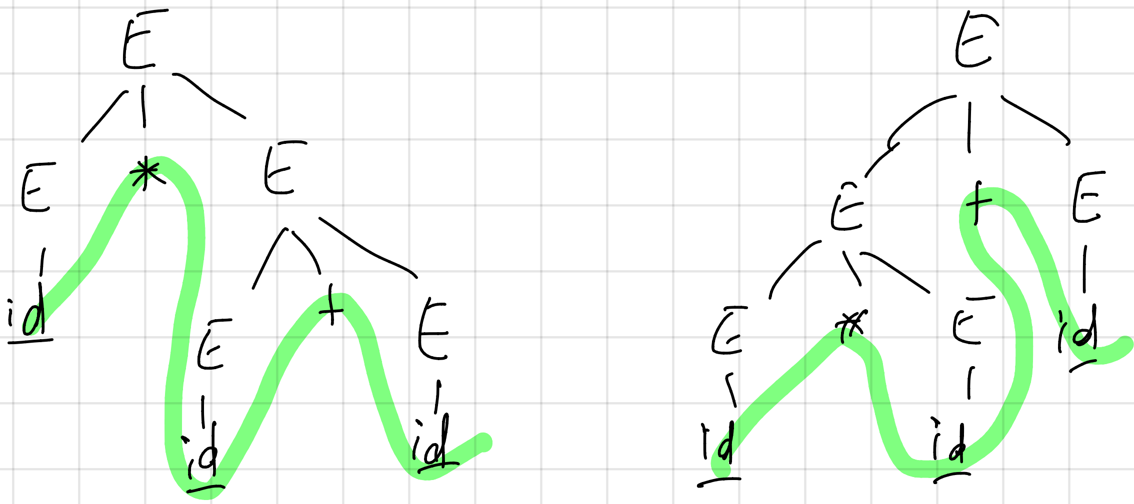
$E \to E + E \mid E * E \mid (E) \mid \underline{id} \mid \underline{num}$



TECNIQUE FOR GETTING A NON-AMBIGUOUS GRAMMAR FOR A LANGUAGE OF EXPRESSIONS USING BINARY AND/OR UNARY OPERATORS

e.g. arithmetic expressions

OPERATORS: +, -, *, / plus unary minus

OPERANDS : $\underline{id}$, $\underline{num}$ $\qquad -5$

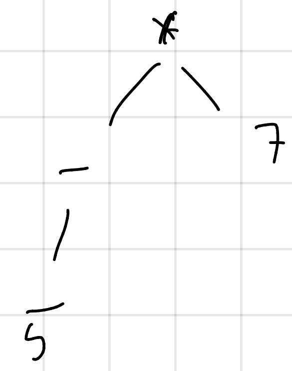
1) DECIDE PRECEDENCE among operators

+, - have the same precedence and bind less than

↳ *, / have the same precedence (greater than + and -)

unary - has more precedence of *, /

$-5 * 7 \rightsquigarrow$



2) Define associativity for BINARY OPERATORS (left or right)

$+, -, *, /$ associate to the left

3) Generate a non-terminal symbol for each level of precedence plus one for the level of the operands and arrange them in a table like this:

E	+, -	ass. LEFT
T	*, /	ass. LEFT
U	unary -	/
F	operands	<u>id</u>, <u>num</u>

$\rightsquigarrow$ CFG

INCREASING PRECEDENCE ↓

4) Generate the CFG from the table

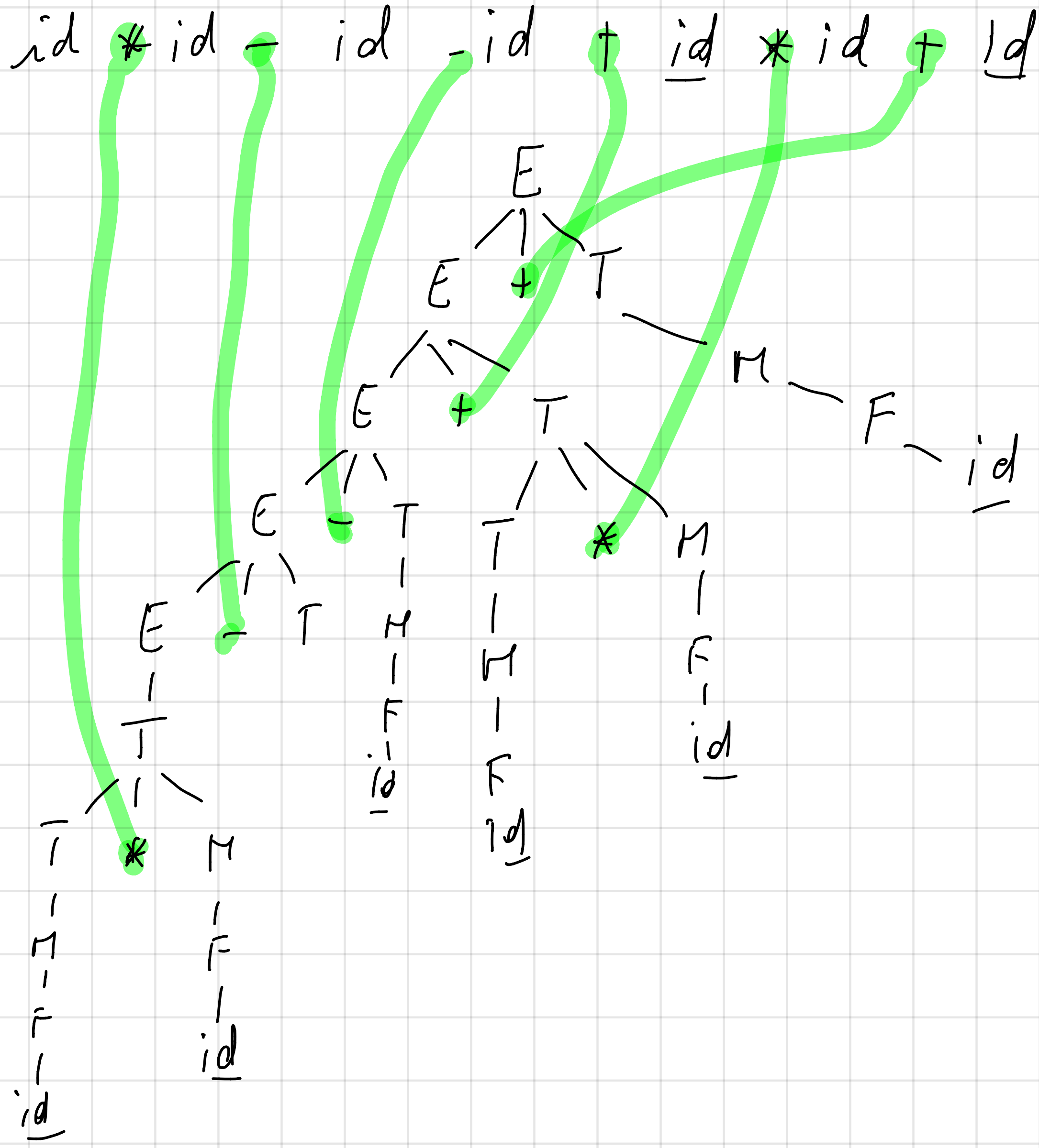
$E \rightarrow E + T \mid E - T \mid T$

$T \rightarrow T * M \mid T / M \mid M$

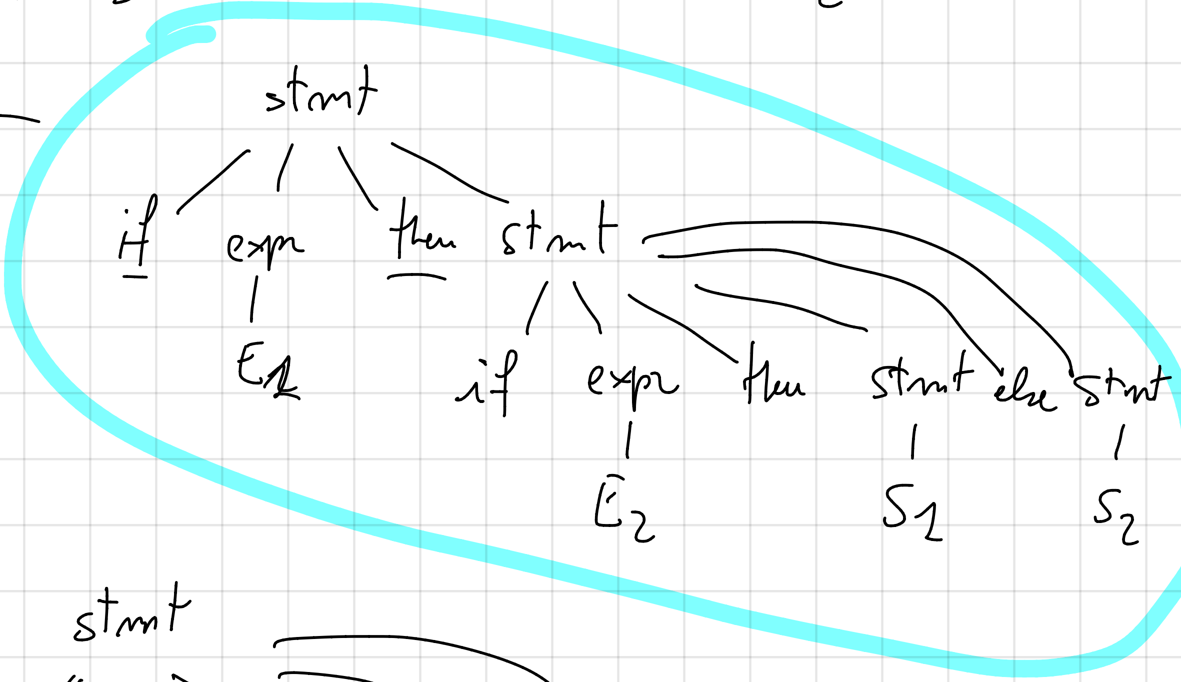
$M \rightarrow - M \mid F$

$F \rightarrow \underline{id} \mid \underline{num} \mid (E)$

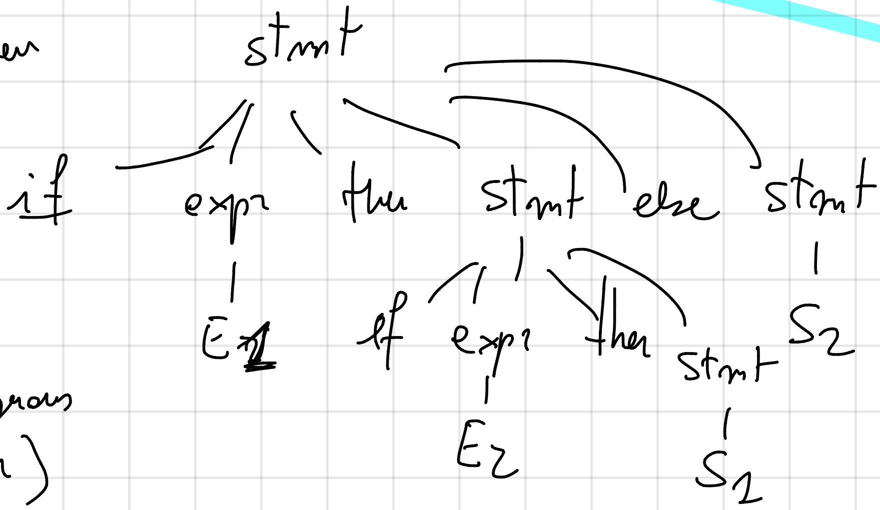
id * id - id - id + $\underline{id}$ * id + $\underline{id}$



if E_1 then if E_2 then S_1 else S_2



simpler solution always select this tree in which the "dangling" else is associated with the "closest" then



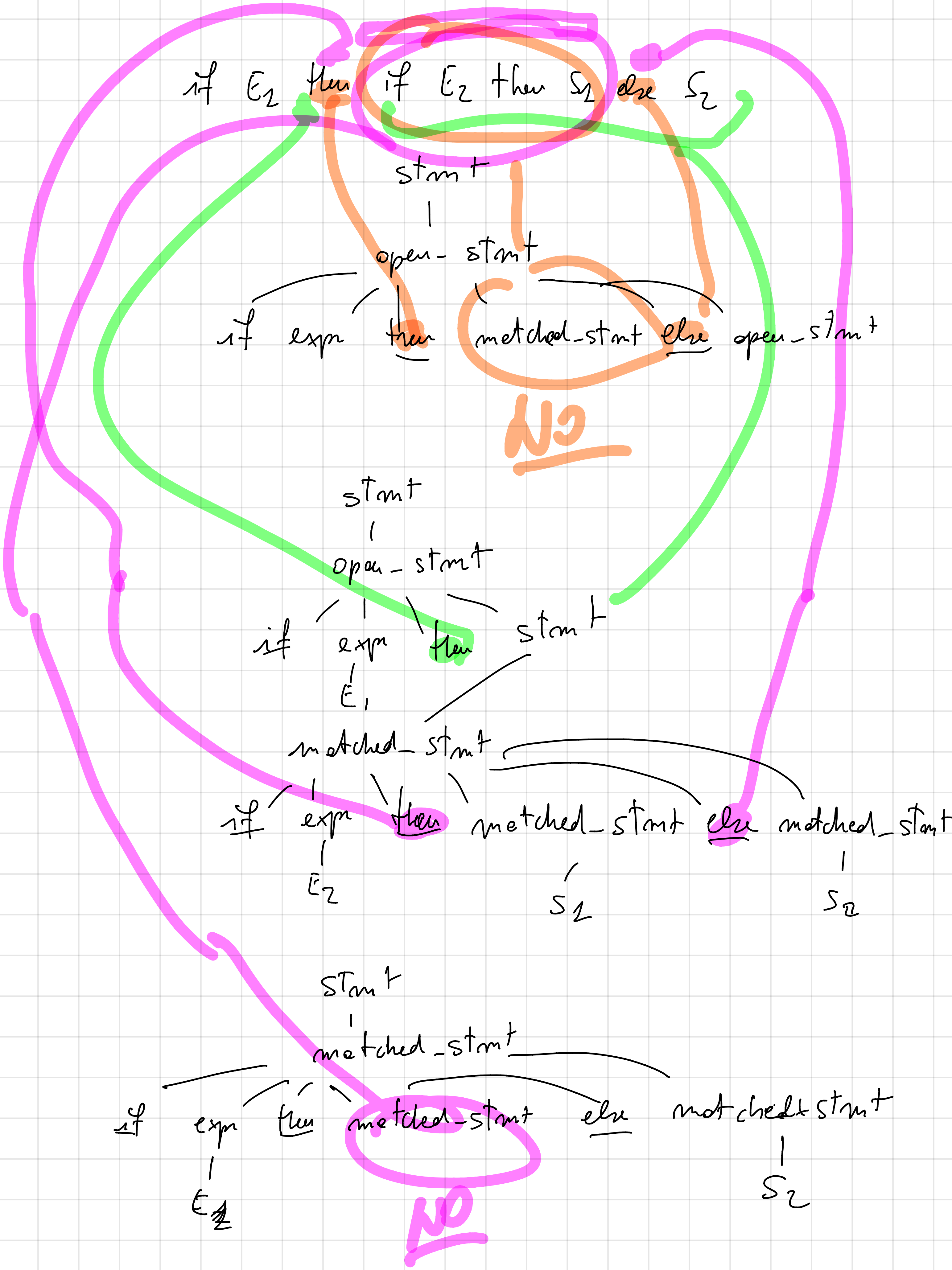
(different from the non-ambiguous grammar)

---

stmt → matched_stmt | open_stmt

matched_stmt → other | if expr then matched_stmt else matched_stmt

open_stmt → if expr then stmt | if expr then matched_stmt else open_stmt



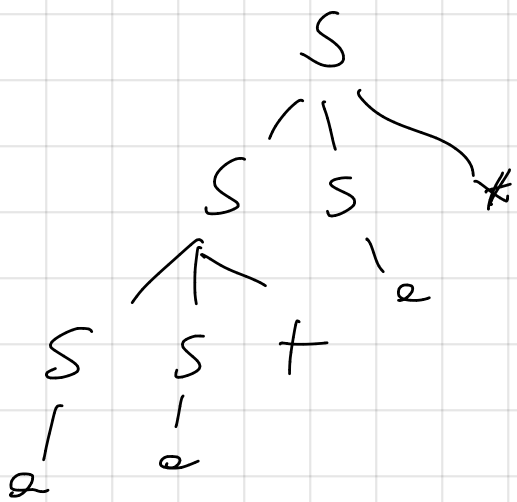
if E2 then if E2 then S1 else S2
stmt
open-stmt
if expr then matched_stmt else open_stmt
NO
stmt
open_stmt
if expr then stmt
E1
matched_stmt
if expr then matched_stmt else matched_stmt
E2
S1
S2
stmt
matched_stmt
if expr then matched_stmt else matched_stmt
E1
S2
NO

$S \to SS+ \mid SS* \mid a$ $\qquad aa+a*$

Leftmost derivation

$$S \underset{lm}{\Rightarrow} SS* \underset{lm}{\Rightarrow} SS+S* \underset{lm}{\Rightarrow} aS+S* \underset{lm}{\Rightarrow} aa+S* \underset{lm}{\Rightarrow} aa+a*$$

$$S \underset{rm}{\Rightarrow} SS* \underset{rm}{\Rightarrow} Sa* \underset{rm}{\Rightarrow} SS+a* \underset{rm}{\Rightarrow} Sa+a* \underset{rm}{\Rightarrow} aa+a*$$



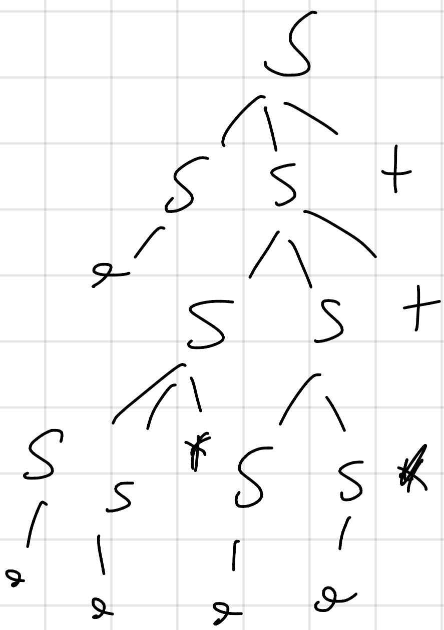
Postfix Notation

$a+a$ infix

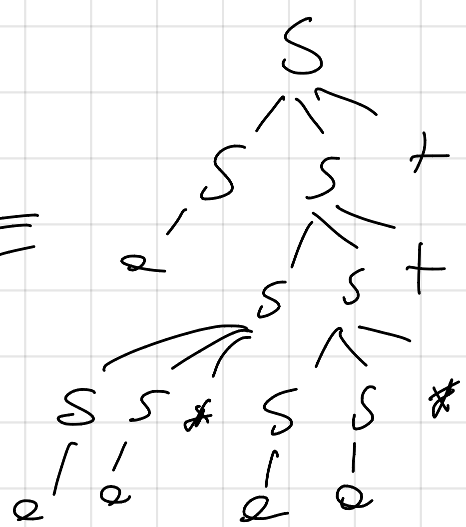
$\Downarrow$

$aa+$ postfix

$aaa*aa*++$



$\equiv$



$L = \{ w \in \{0,1\}^* \mid w \text{ is palindrome} \}$

$S \to 0S0 \mid 1S1 \mid 1 \mid 0 \mid \varepsilon$

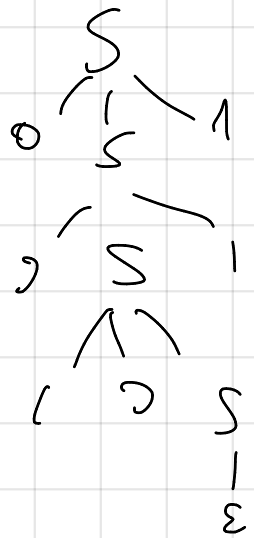
$L = \{ w \in \{0,1\}^* \mid w \text{ contains the same occurences of 1s and 0s} \}$

0110 1010 1100 101010

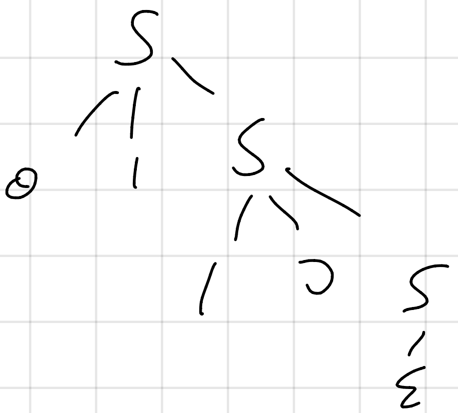
011010

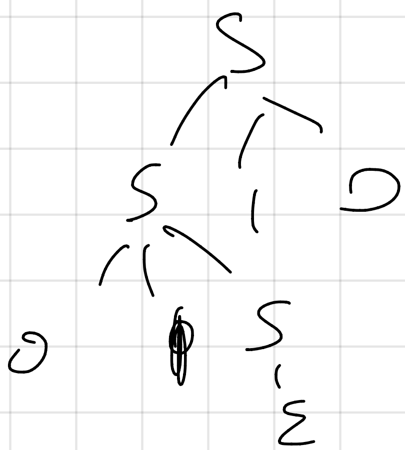
$S \to 01S \mid 10S \mid 0S1 \mid 1S0 \mid S10 \mid S01 \mid \varepsilon$

001011



0110





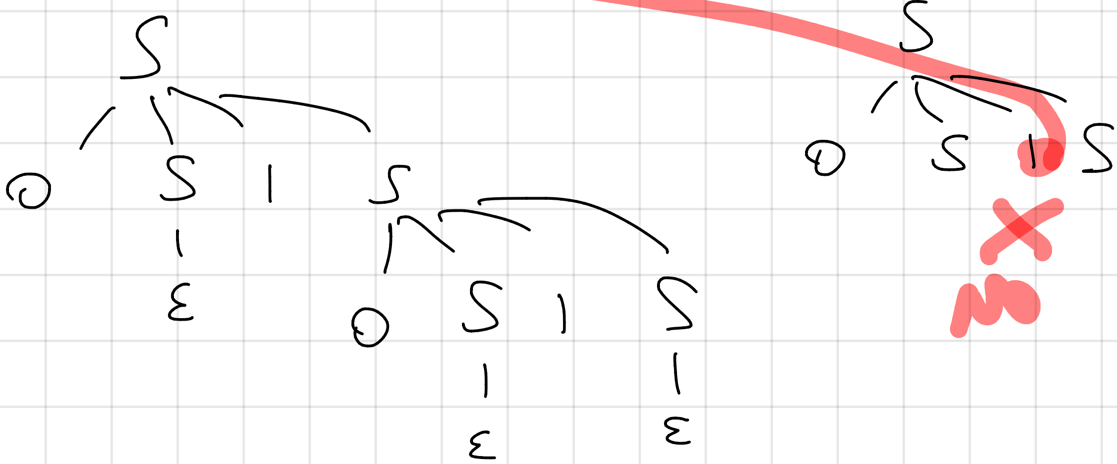
Alternative Solution

$$S \to 1S0S \mid 0S1S \mid \varepsilon$$

similar to balanced parentheses

$$S \to (S)S \mid \varepsilon$$

0110

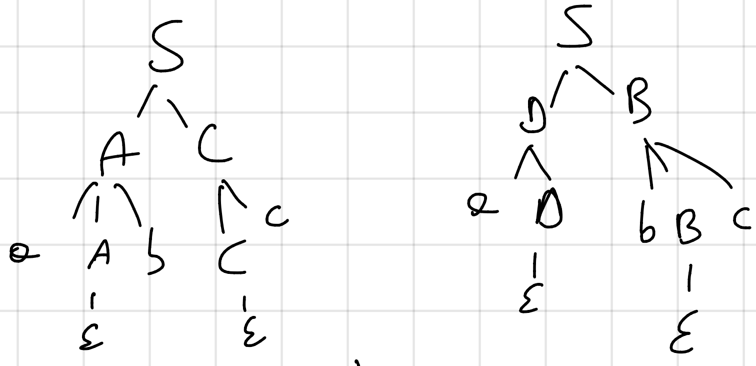


$$\{a^n b^m c^k \mid n = m \text{ or } m = k,\ n, m, k \geq 0\}$$

$$S \to AC \mid DB$$

abc

AMBIGUOUS



$$A \to aAb \mid \varepsilon$$

$$B \to bBc \mid \varepsilon$$

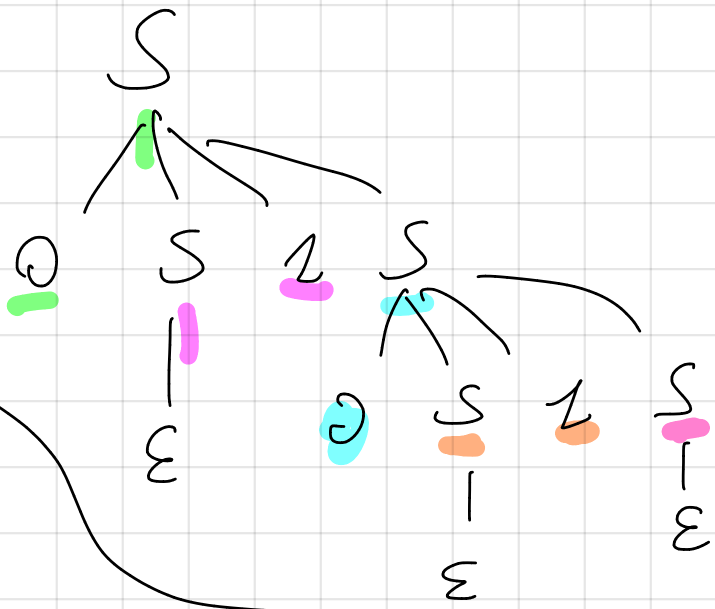
$$C \to cC \mid \varepsilon$$

$$D \to aD \mid \varepsilon$$

$S \to 0S1S \mid 1S0S \mid \varepsilon$

0101$

TOP-DOWN PARSER



LL(K)

read the input from left to right

generate left-most derivation

the number of lookahead symbols (usually 1)

FIRST(S) = {0, 1, ε}

FOLLOW(S) = {$, 1, 0}

	0	1	$
S	$S \to 0S1S$ $S \to \varepsilon$	$S \to 1S0S$ $S \to \varepsilon$	$S \to \varepsilon$

the grammar is not LL(1)