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

1 2 2 3 4 5 6 7

$G = \langle \{E\}, \{+, *, -, /, (,), \underline{id}, \underline{num}\}, E, P \rangle$

- $\{E\}$: non-terminals
- $\{+, *, -, /, (,), \underline{id}, \underline{num}\}$: terminals
- E: initial symbol
- P: Productions

$P ::$

$E \to E+E$

$E \to E*E$

$\vdots$

$E \to \underline{num}$

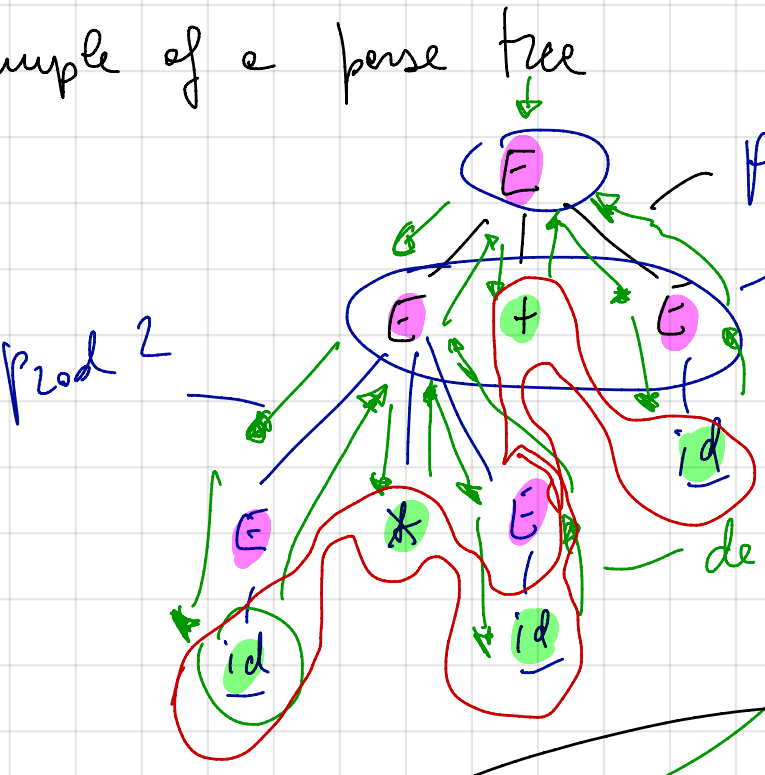
Context-free grammar

LEFT

is always only one non-terminal symbol

any string of $(V \cup T)^*$

example of a parse tree



Production 1

$E \to E+E$

prod 2

depth-first visit

this string belongs to the language generated by the grammar

this string is generated by the grammar

id * id + id

terminals printed during the visit

= frontier of the tree

$E \to E^{2} + T \mid E^{2} - T \mid T^{3}$

$T \to T^{4} * F \mid T^{5} / F \mid F^{6}$

$F \to (E)^{7} \mid id^{8} \mid num^{9}$

Non-terminals = {E, T, F}

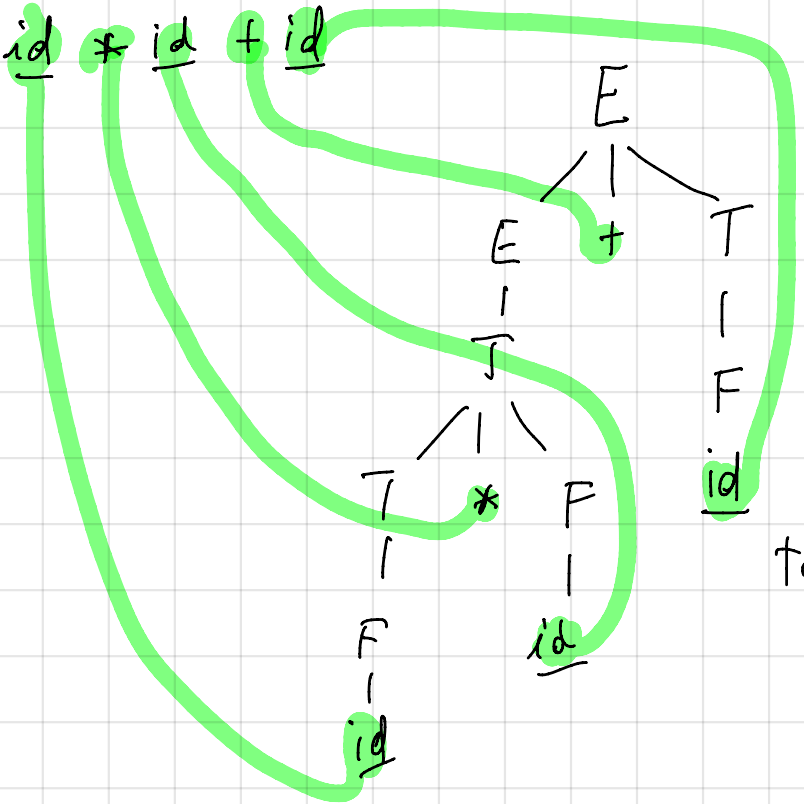
Terminals the same

Start symbol E
(it is listed first)

3 productions

PARSING PROBLEM: Does a string s belong to the language generated by the grammar?

s = id * id + id

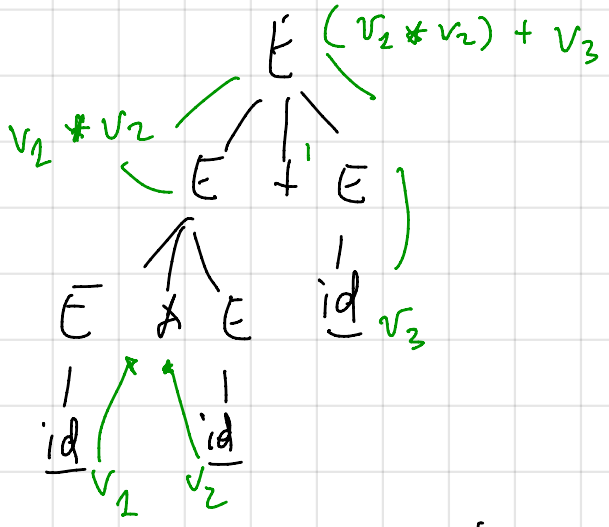


Answer in this case is yes

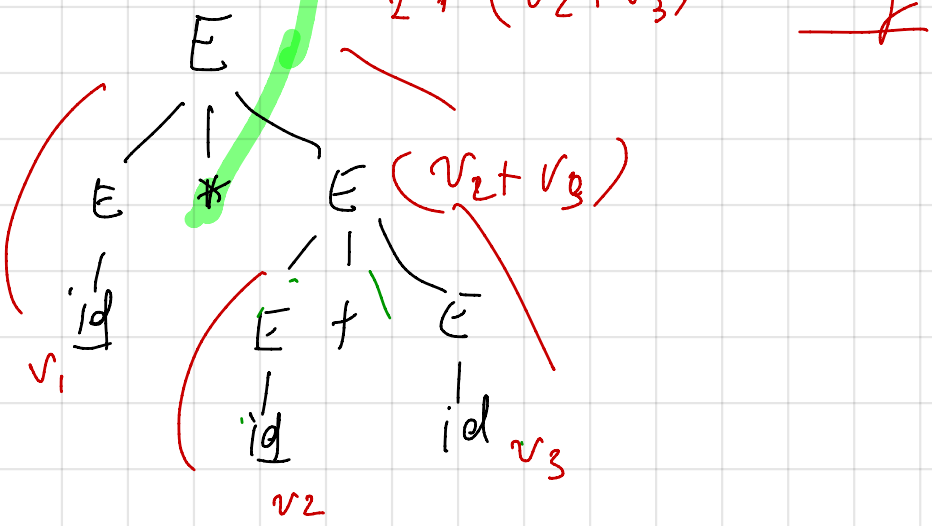
id * id + id belongs to the language of the grammar

The first grammar is AMBIGUOUS

- we generated a parse tree for id * id + id



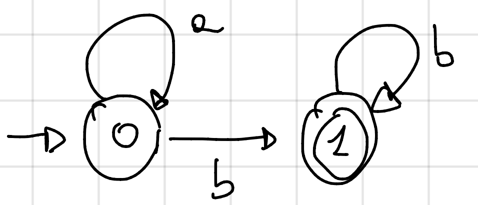
- There is ANOTHER parse tree for that string using the same productions



PARSING ALGORITHMS, in general, DO NOT WORK with Ambiguous grammars

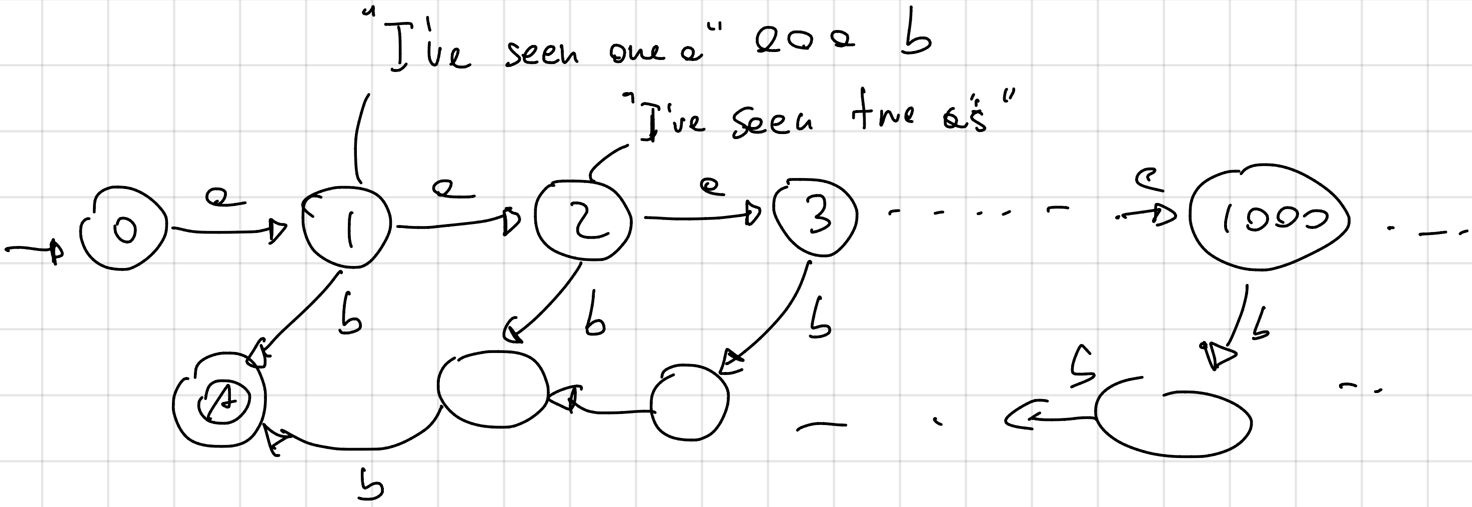
Expressive power of FSA $\Sigma = \{a, b\}$

$\mathcal{L} = \{ a^n b^n \mid n > 0 \}$



→ accepts ab, aabb, aaabbb, aaaabbbb

✗ no but also accepts a bbb



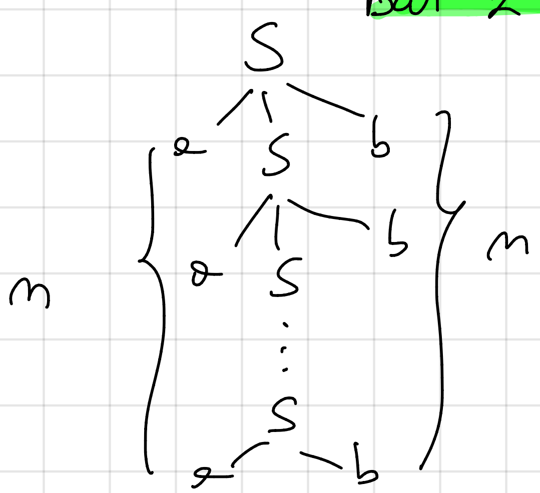
The problem is that I would need an INFINITE number of states. But we are using FINITE state automata.

⟹ $\mathcal{L}$ is NOT a REGULAR LANGUAGE

There is a CFG for $\mathcal{L}$

$S \to aSb \mid ab$ (1, 2)

But $\mathcal{L}$ is a CONTEXT FREE LANGUAGE



Push-down Automaton accepting $\mathcal{L} = \{a^n b^n \mid n > 0\}$

initial state

S_0	a	b	$\$$
Z_0	new state: S_1 / string to put on stack: A	/	/
A	/	/	/

$aaabbb\$$

initial symbol of the stock

S_1	a	b	$\$$
A	S_1, AA	S_2, ε	/
Z_0	/	/	/

A means "I've seen one a"

S_2	a	b	$\$$
A	/	S_2, ε	/
Z_0	/	/	/

$F = \emptyset$ there is not a final state, but a string is accepted if the input is finished and the stack is empty

INPUT	STATE	STACK	MOVE
aaabbb$	s_0	$\$ Z_0$	s_2, A
aabbb$	s_1	$\$A$	s_1, AA
abbb$	s_2	$\$AA$	s_2, AA
bbb$	s_2	$\$AAA$	s_2, ε
bb$	s_2	$\$AA$	s_2, ε
b$	s_2	$\$A$	s_2, ε
$	s_2	$	ACCEPT because the input is finished and the stack is empty

---

Derivation (general) $E \to \overset{1}{E+E} \mid \overset{2}{E*E} \mid \overset{3}{\underline{id}} \mid \overset{5}{\underline{num}}$

$E \overset{1}{\Rightarrow} E+E \overset{2}{\Rightarrow} E+E*E \overset{3}{\Rightarrow} E+\underline{id}*E \overset{3}{\Rightarrow}$

$E+\underline{id}*id \overset{3}{\Rightarrow} \underline{id}+\underline{id}*id$

sentential forms

STRING

STOP

Leftmost derivation : You have to select every time the non-terminal at the left (most)

$$E \overset{1}{\Rightarrow} E + E \overset{3}{\Rightarrow} id + E \overset{2}{\Rightarrow} id + E * E \overset{3}{\Rightarrow} id + id * E \overset{3}{\Rightarrow} id + id * id$$

Rightmost derivation

$$E \overset{1}{\Rightarrow} E + E \overset{2}{\Rightarrow} E + E * E \overset{3}{\Rightarrow} E + E * id \overset{3}{\Rightarrow} E + id * id \overset{3}{\Rightarrow} id + id * id$$