Cairo University
FCI

Theory of Computation

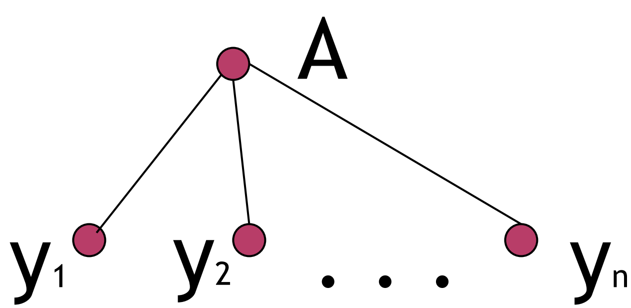
Parsing

- CFG grammar is about categorizing the statements of a language.
- Parsing using CFG means categorizing a certain statements into categories defined in the CFG.
- Parsing can be expressed using a special type of graph called Trees where no cycles exist.
- A *parse tree* is the graph representation of a derivation.

Parse tree

(1)A vertex with a label which is a Non-terminal symbol is a parse tree.

(2) If $A \rightarrow y_1\ y_2 \ldots\ y_n$ is a rule in R, then the tree



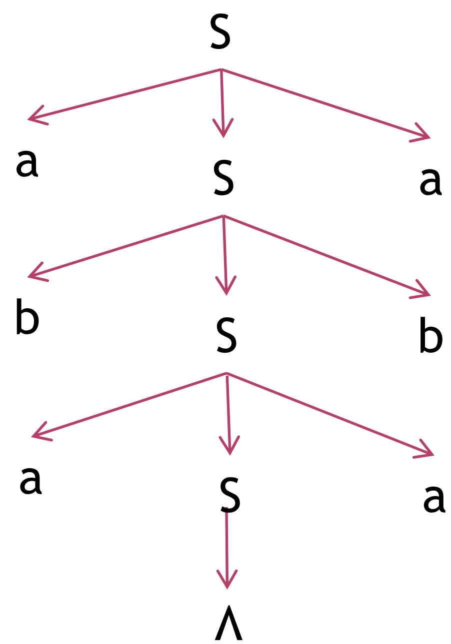
is a parse tree.

Ambiguity

- A grammar can generate the same string in different ways.
- Ambiguity occurs when a string has two or more leftmost derivations for the same CFG.
- There are ways to eliminate ambiguity such as using Chomsky Normal Form (CNF) which does n’t use Λ.
- Λ cause ambiguity.

Ex 1 Even-Plaindrome grammar

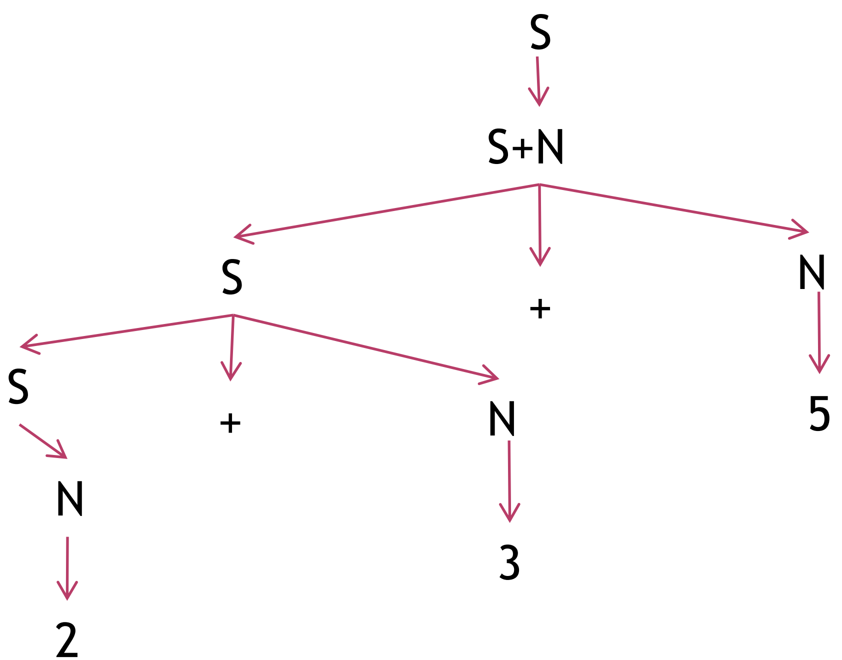
- i.e. {Λ, ab, abbaabba,… }
- S → aSa| bSb| Λ   Derive abaaba



Example2

- Deduce CFG of a addition and parse the following expression 2+3+5

- 1] S→S+S|N
- 2] N→1|2|3|4|5|6|7|8|9|0

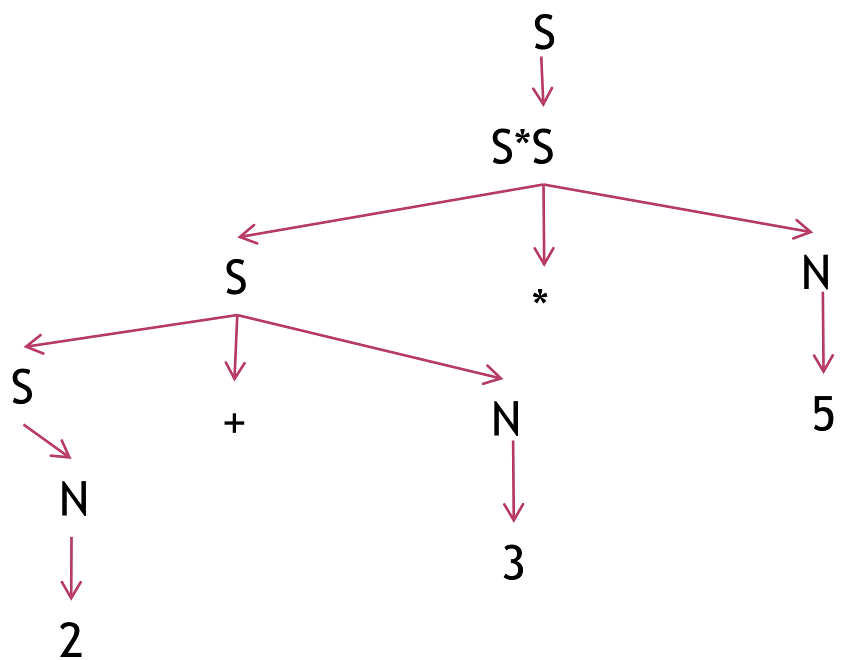


Can u make another parsing tree ?

Example3

- Deduce CFG of a addition/multiplication and parse the following expression 2+3*5

- 1] $S \rightarrow S+S \mid S*S \mid N$
- 2] $N \rightarrow 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \mid 0 \mid \; N$

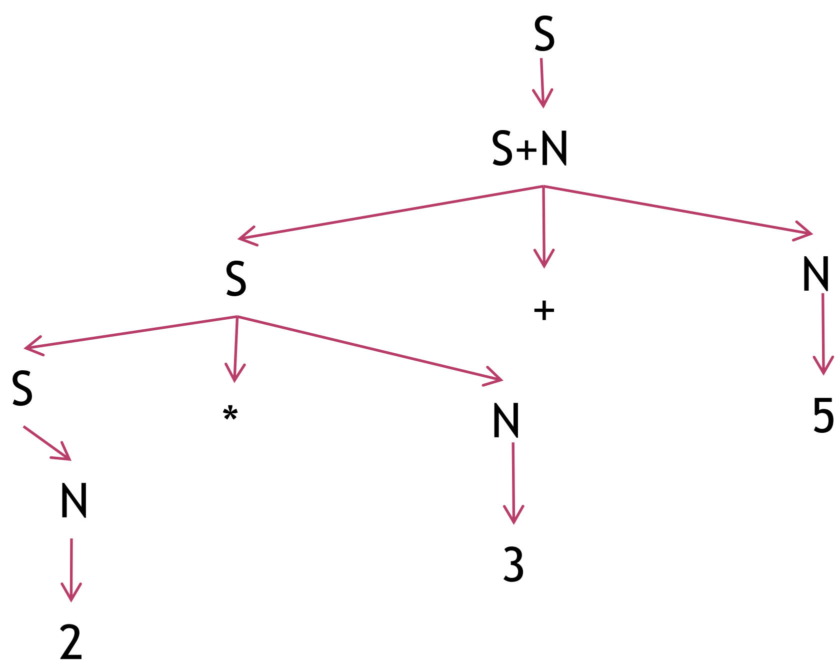


Can u make another parsing tree ?

Ex4 CFG without ambiguity

- Deduce CFG of a addition/multiplication and parse the following expression 2*3+5

1] S→ Term|Term + S
2] Term → N|N * Term
3] N→1|2|3|4|5|6|7|8|9|0



Can u make another parsing tree ?

Example 5

$S \rightarrow A \mid A\,B$

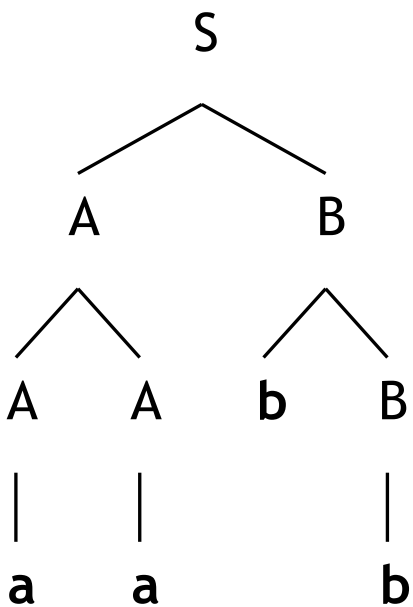
$A \rightarrow \Lambda \mid \mathbf{a} \mid A\,\mathbf{b} \mid A\,A$

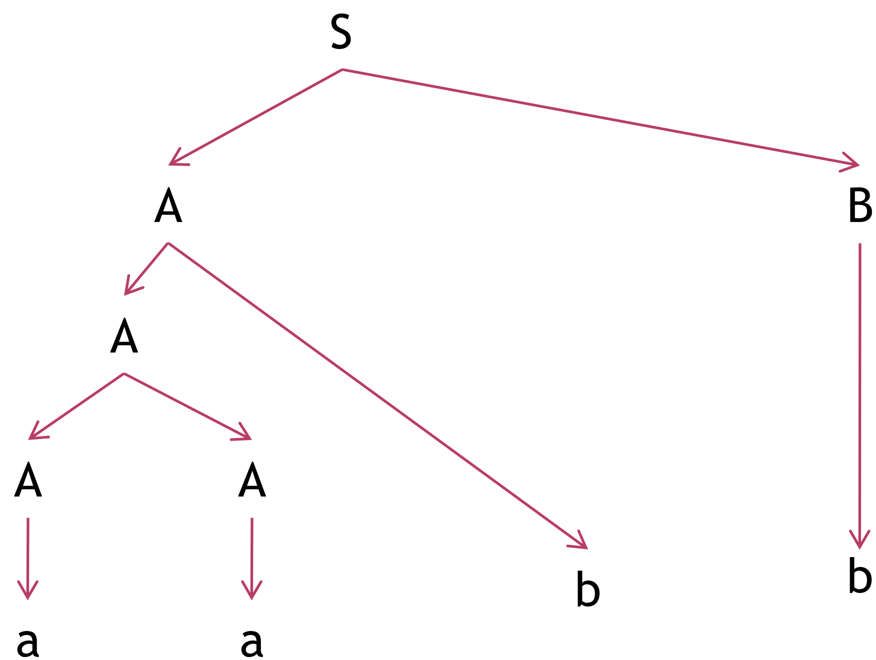
$B \rightarrow \mathbf{b} \mid \mathbf{b}\,\mathbf{c} \mid B\,\mathbf{c} \mid \mathbf{b}\,B$

Sample derivations:

$S \Rightarrow AB \Rightarrow AAB \Rightarrow$
$\mathbf{a}AB \Rightarrow \mathbf{aa}B \Rightarrow \mathbf{aab}B \Rightarrow \mathbf{aabb}$

$S \Rightarrow AB \Rightarrow A\mathbf{b}B \Rightarrow A\mathbf{bb} \Rightarrow AA\mathbf{bb}$
$\Rightarrow A\mathbf{abb} \Rightarrow \mathbf{aabb}$





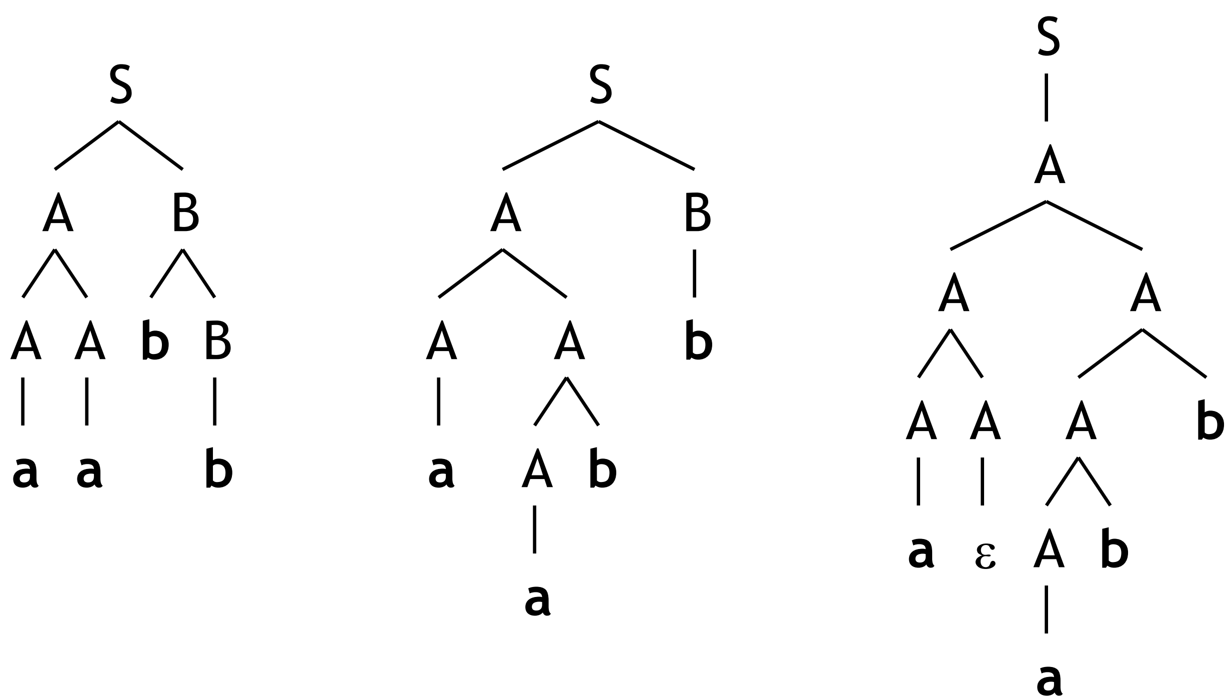
Example 6

$S \rightarrow A \mid A\ B$

$A \rightarrow \Lambda \mid \mathbf{a} \mid A\ \mathbf{b} \mid A\ A$

$B \rightarrow \mathbf{b} \mid \mathbf{b}\ \mathbf{c} \mid B\ \mathbf{c} \mid \mathbf{b}\ B$

w = **aabb**

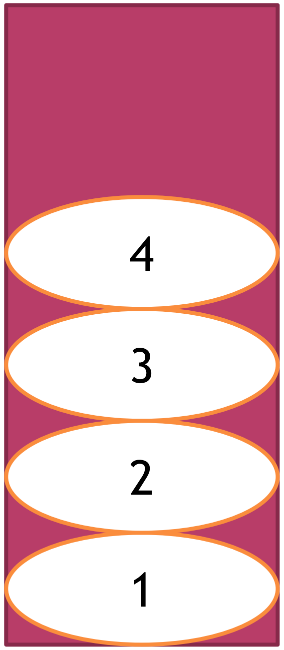


PDA

- A Pushdown Automata (PDA) is a DFA equipped with a single stack.

- A PDA is a 6-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ where
  - Q: finite set of states
  - Σ: finite input alphabet
  - Γ: finite stack alphabet
  - δ: $Q \times \Sigma_\epsilon \times \Gamma_\epsilon \longrightarrow 2^{Q \times \Gamma_\epsilon}$

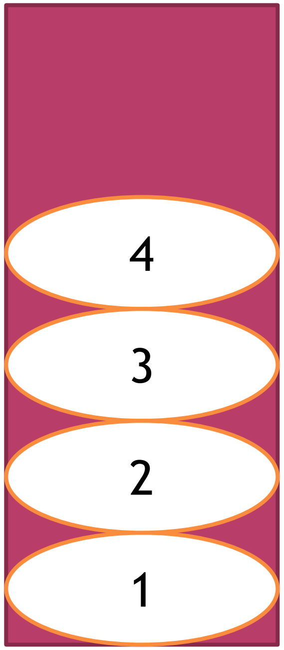
    e.g.: $\delta(q, a, Z) = \{(p_1, r_1), (p_2, r_2), \cdots, (p_n, r_n)\}$
  - $q_0 \in Q$: start state
  - $F \subseteq Q$: set of accept states



PDA as a deterministic machine

Its inputs are

1. The current state (of its "NFA"),
2. The current input symbol (or Λ), and
3. The current symbol on top of its stack.



Its actions are:

1. Change state.
2. Change stack if needed.

THANK YOU