Theory of Automata

Lecture #22-23-24

Definition

- The language generated by CFG
  - Consists of those strings which can be produced from the start symbol S using the production Rules

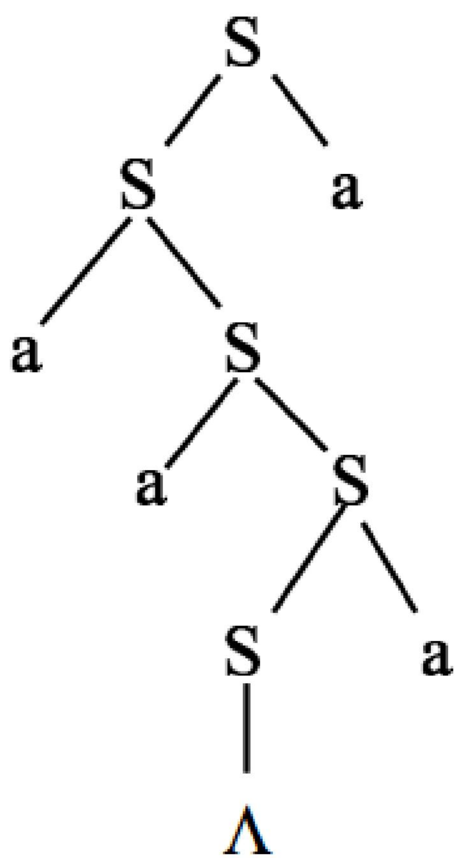
- The language generated by CFG is called Context Free Language(CFL)

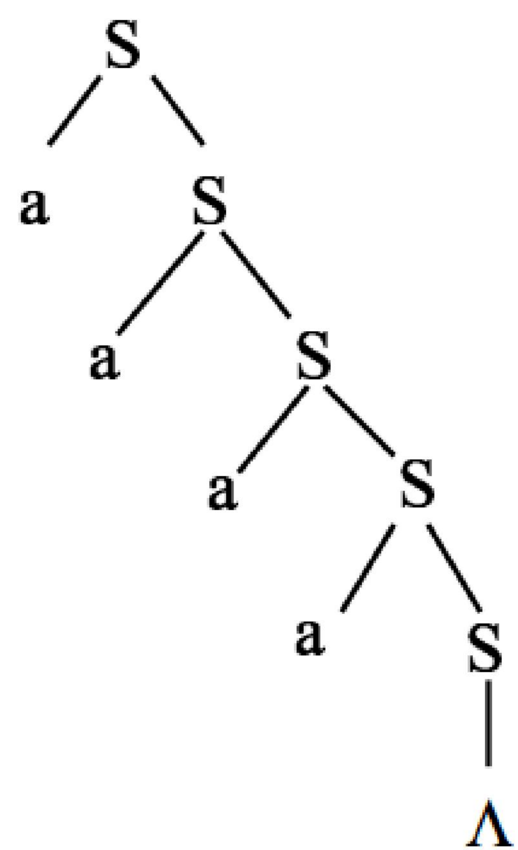
Ambiguity

- A CFG is said to be ambiguous if there is at least 1 string in L(G) having two or more distinct derivations

Ambiguity

$$S \rightarrow aS \mid Sa \mid \Lambda$$





Arithmetic Expression Example

$$S \rightarrow A$$

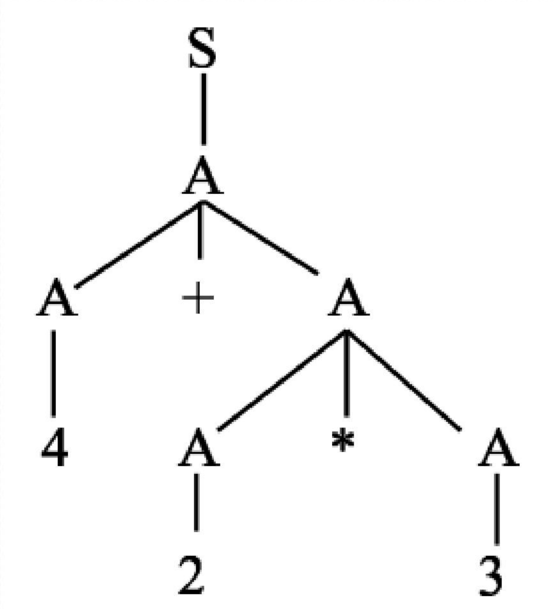
$$A \rightarrow \text{integer} \mid A + A \mid A - A \mid A * A \mid A / A \mid (A)$$

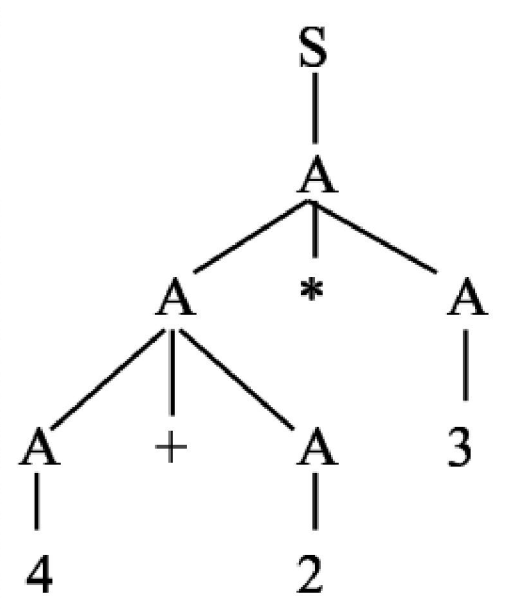
Arithmetic Expression Example

$S \to A$

$A \to \text{integer} \mid A + A \mid A - A \mid A * A \mid A / A \mid (A)$

$$4+2*3$$





So Ambiguity is

- A CFG is ambiguous if there is some word it generates which has two different parse trees

- A CFG that is not ambiguous is called unambiguous

Arithmetic Expressions

E - Expression
T - Term
F - Factor

$S \rightarrow A$

$A \rightarrow \text{integer}$

$A \rightarrow A + A$

$A \rightarrow A - A$

$A \rightarrow A * A$

$A \rightarrow A / A$

$A \rightarrow (A)$

$S \rightarrow E$

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

$T \rightarrow T * F \mid T / F \mid F$

$F \rightarrow \text{integer} \mid (E)$

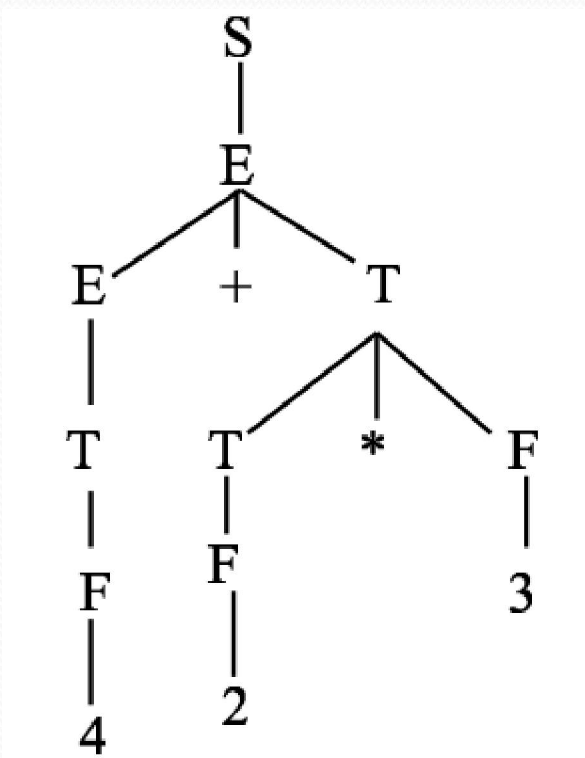
E is expression, T is term of expression and F is factor of a term

Derivation of 4+2*3 using AE

Productions

$$S \rightarrow E$$
$$E \rightarrow E + T \mid E - T \mid T$$
$$T \rightarrow T * F \mid T / F \mid F$$
$$F \rightarrow \text{integer} \mid (E)$$

Derivation



E - Expression
T - Term
F - Factor

E is expression, T is term of expression and F is factor of a term

Note

- Identify same priority
- Move right to left
  - First solve lowest priority
  - Then highest priority
- Replacement of productions

Simple Example

- Suppose CFG is

(1) $S \rightarrow S + S$

(2) $S \rightarrow 1$

(3) $S \rightarrow a$

Generate the string from the given CFG

- $1 + 1 + a$

Derivation

- $S \rightarrow$ **S+S** (rule 1 on first S)
- $\rightarrow$ **S+S**+S (rule 1 on second)
- $\rightarrow$ S+**1**+S (rule 2 on second S)
- $\rightarrow$ S+1+**a** (rule 3 on third S)
- $\rightarrow$ **1**+1+a (rule 2 on first S)

Types of Derivations

- **Leftmost derivation**, it is always the leftmost nonterminal
- **Rightmost derivation**, it is always the rightmost nonterminal

Left Most Derivation

• $S \rightarrow S+S$	(rule 1 on first S)
• $\rightarrow 1+S$	(rule 2 on first S)
• $\rightarrow 1+S+S$	(rule 1 on first S)
• $\rightarrow 1+1+S$	(rule 2 on first S)
• $\rightarrow 1+1+a$	(rule 3 on first S)

Left Most Derivation

```
          S
         /|\
        / | \
       /  |  \
      S  '+'  S
      |      /|\
      |     / | \
     '1'   S '+' S
           |     |
          '1'   'a'
```

Right Most Derivation

$S \rightarrow S + S$ (Rule-1)

$\rightarrow S + a$ (Rule-3)

$\rightarrow S + S + a$ (Rule-1)

$\rightarrow S + 1 + a$ (Rule-2)

$\rightarrow 1 + 1 + a$ (Rule-2)

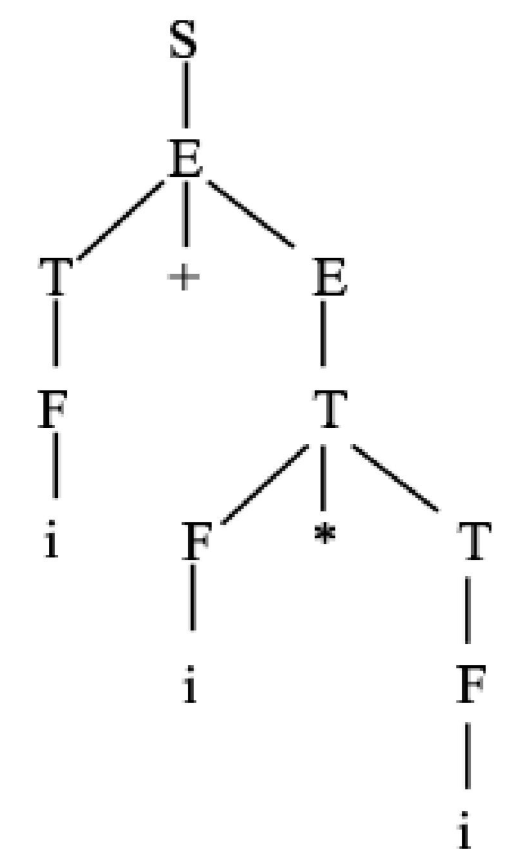
Right Most Derivation

```
            S
           /|\
          / | \
         /  |  \
        S  '+'  S
       /|\      |
      / | \     |
     S '+' S   'a'
     |     |
    '1'   '1'
```

Parse Tree from given input string

Input: i + i*i

Parse Tree



Grammar

$S \rightarrow E$

$E \rightarrow T + E$

$E \rightarrow T$

$T \rightarrow F * T$

$T \rightarrow F$

$F \rightarrow i$

E - Expression
T - Term
F - Factor

E is expression, T is term of expression and F is factor of a term

Parse Tree from given input string

Language: a^*ba^*b

Input String: **aabab**

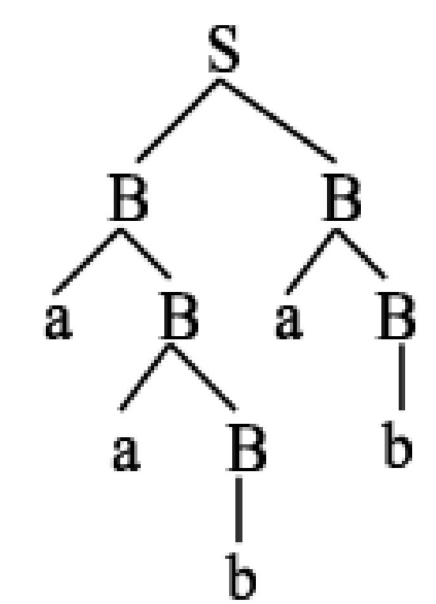
Grammar

$$S \rightarrow BB$$

$$B \rightarrow aB$$

$$B \rightarrow b$$

Parse Tree



Algebraic expressions

- Here is a context-free grammar for syntactically correct infix algebraic expressions in the variables x, y and z:

i. $S \rightarrow x$

ii. $S \rightarrow y$

iii. $S \rightarrow z$

iv. $S \rightarrow S + S$

v. $S \rightarrow S - S$

vi. $S \rightarrow S * S$

vii. $S \rightarrow S / S$

viii. $S \rightarrow (S)$

Generate Following String

- (x + y) * x - z * y / (x + x)

<u>Note that:</u>

x, y, z are terminals
S is non-Terminal

String Generation

- $S \rightarrow S - S$ (by rule 5)
- $\rightarrow S * S - S$ (by rule 6, applied to the leftmost S)
- $\rightarrow S * S - S / S$ (by rule 7, applied to the rightmost S)
- $\rightarrow (S) * S - S / S$ (by rule 8, applied to the leftmost S)
- $\rightarrow (S) * S - S / (S)$ (by rule 8, applied to the rightmost S)
- $\rightarrow (S + S) * S - S / (S)$ (etc.)
- $\rightarrow (S + S) * S - S * S / (S)$
- $\rightarrow (S + S) * S - S * S / (S + S)$
- $\rightarrow (x + S) * S - S * S / (S + S)$
- $\rightarrow (x + y) * S - S * S / (S + S)$
- $\rightarrow (x + y) * x - S * y / (S + S)$
- $\rightarrow (x + y) * x - S * y / (x + S)$
- $\rightarrow (x + y) * x - z * y / (x + S)$
- $\rightarrow (x + y) * x - z * y / (x + x)$

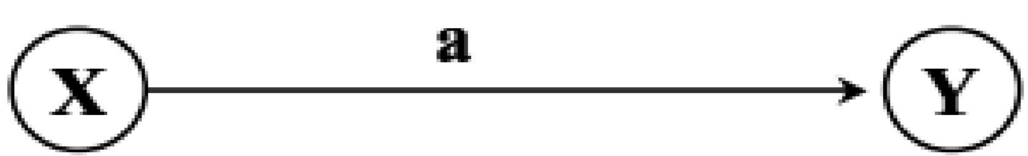
i. $S \rightarrow x$
ii. $S \rightarrow y$
iii. $S \rightarrow z$
iv. $S \rightarrow S + S$
v. $S \rightarrow S - S$
vi. $S \rightarrow S * S$
vii. $S \rightarrow S / S$
viii. $S \rightarrow (S)$

Parse Tree

```
                 S
                 |
                /|\
               S - S
             /       \
           /|\       /|\
          S * S     S / S
        /     |     |     \
      /|\     x   /|\     /|\
     ( S )       S * S   ( S )
      /          |   |       \
    /|\          z   y       /|\
   S + S                    S + S
   |   |                    |   |
   x   y                    x   x
```

NFA-Λ to CFG

1. Name all the states in the NFA-Λ by a symbol.
   – Call the Start State S.
2. For each edge



write:

$$X \rightarrow aY$$

NFA-Λ to CFG

3. For each edge

write:

$$X \to Y$$

NFA-Λ to CFG

4. For each edge

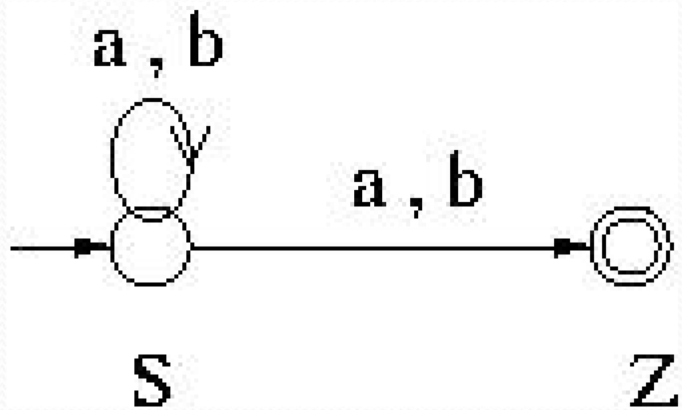
write:

$$X \rightarrow aX$$

5. For each Final State X write:

$$X \rightarrow \Lambda$$

NFA to CFG



$S \rightarrow aS$

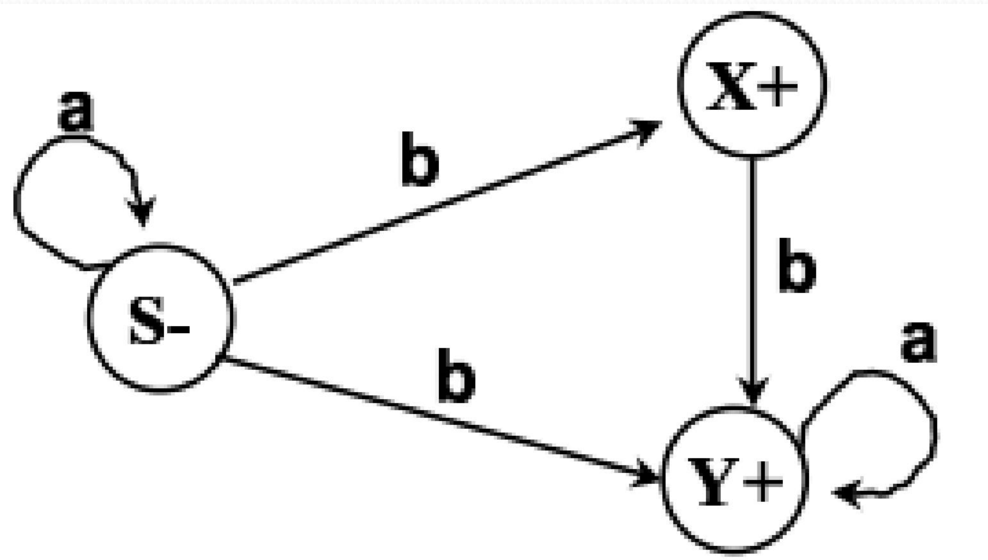
$S \rightarrow bS$

$S \rightarrow bZ$

$S \rightarrow aZ$

$Z \rightarrow \wedge$

NFA to CFG



$S \rightarrow aS$

$S \rightarrow bX$

$S \rightarrow bY$

$X \rightarrow bY$

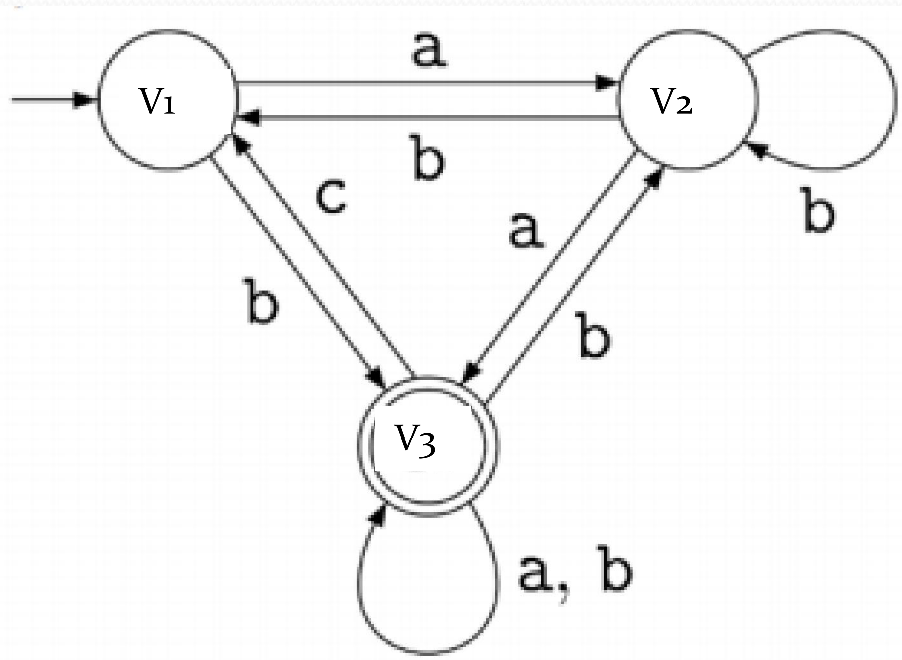
$Y \rightarrow aY$

$X \rightarrow \Lambda$

$Y \rightarrow \Lambda$

NFA to CFG

NFA



CFG

$$
\begin{aligned}
S &\rightarrow V_1 \\
V_1 &\rightarrow aV_2 \mid bV_3 \\
V_2 &\rightarrow bV_1 \mid bV_2 \mid aV_3 \\
V_3 &\rightarrow bV_2 \mid cV_1 \mid aV_3 \mid bV_3 \mid \varepsilon
\end{aligned}
$$

Here, the variables V_1, V_2, and V_3 represent the states q_1, q_2, and q_3 of the NFA.

And, since, the state q_3 is a final state, we added the rule, $V_3 \rightarrow \varepsilon$ to the grammar