Computer Science

Theory of Computation

Context Free Languages

Lecture No.- 2

Mallesham Devasane Sir

Recap of Previous Lecture

Topic	CFG
Topic	Types of CFGs
Topic	CFG Vs CFL

Topics to be Covered

- Topic: CFG Vs CFL
- Topic: Types of Normal Forms
- Topic: Simplification of CFGs

(31) $S \rightarrow aSa \mid bSb \mid \varepsilon$

ε ✓
aa ✓
bb ✓
aaaa ✓
abba ✓
bbbb ✓
baab ✓

L = Set of all even length palindromes

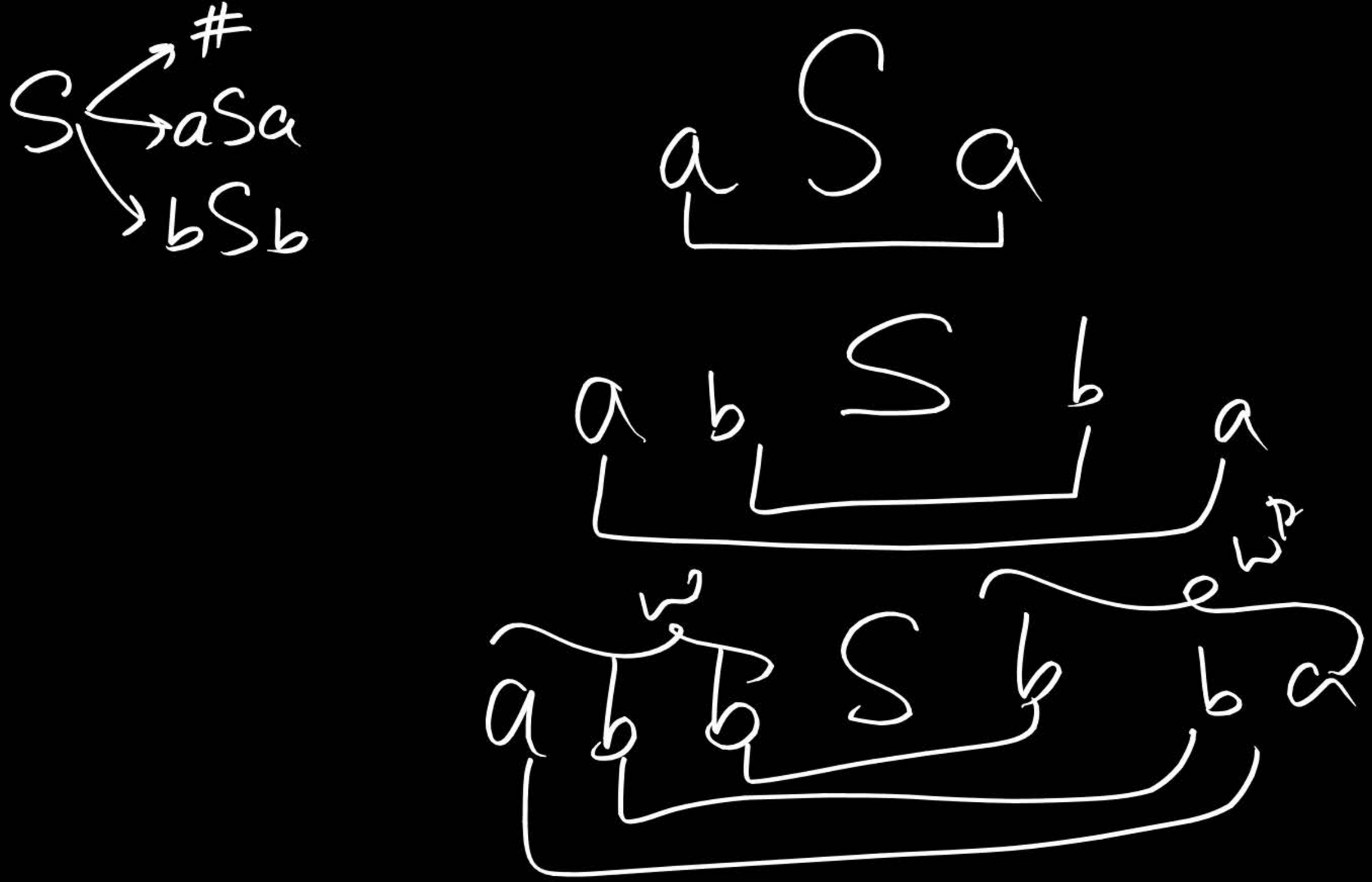
$= \{ w w^R \mid w \in \{a,b\}^* \}$

$= \{ w \mid w \in \{a,b\}^*, w = w^R, |w| = \text{even} \}$

(32) $S \rightarrow aSa \mid bSb \mid \#$

$$L = \{ w \# w^R \mid w \in \{a,b\}^* \}$$

$$= \{ \#, a\#a, b\#b, aa\#aa, bb\#bb, ab\#ba, ba\#ab, \ldots \}$$


#
S
aSa
bSb
a S a
a b S b a
a b b S b b a

(33) $S \rightarrow aSa \mid bSb \mid a \mid b$

$L = \{a, b, aaa, aba, bab, bbb, \ldots\}$

= Set of all odd length palindromes

$= \{w(a+b)w^R \mid w \in \{a,b\}^*\}$

$= \{wxw^R \mid w \in \{a,b\}^*, x \in \{a,b\}\}$

(34) $S \rightarrow aSa \mid bSb \mid \varepsilon \mid a \mid b$

L = Set of all palindromes over $\Sigma = \{a, b\}$

$= \{ w \mid w \in \{a, b\}^*, \; w = w^R \}$

(35) $S \rightarrow aSa \mid \#$

$L = \{a^n \# a^n \mid n \geq 0\}$

(36) $S \rightarrow aS \mid Sa \mid \#$

$L = \{a^* \# a^*\}$

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

$L = a^n a^n = a^{2n}$

$= (aa)^*$

(38) $S \rightarrow aSbS \mid bSaS \mid \varepsilon$

$L = \{ w / w \in \{a,b\}^*, n_a(w) = n_b(w) \}$

(39) $S \rightarrow SaSbS \mid SbSaS \mid \varepsilon$

$L = \{ w / w \in \{a,b\}^*, n_a(w) = n_b(w) \}$

(40) $S \rightarrow aSb \mid bSa \mid SS \mid \varepsilon$

$L = \{ w / w \in \{a,b\}^*, n_a(w) = n_b(w) \}$

$$L = \{ w \mid w \in \{a,b\}^*, \ \#a's(w) = \#b's(w) \}$$

$$= \{ \varepsilon, ab, ba, aabb, abab, abba, baab, \ldots \}$$

$$\boxed{\#a's = \#b's}$$

(41) $S \rightarrow bSa \mid \varepsilon \Rightarrow L = \{b^n a^n\}$

(42) $S \rightarrow AS \mid \varepsilon \Rightarrow L = a^*$

$A \rightarrow a$

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

$A \to aAb \mid \varepsilon$

A) $L = \{a^n b^n \mid n \geq 0\}^*$

B) $L = \{(a^n b^n)^* \mid n \geq 0\}$

C) $L = \{a^{n_1} b^{n_1} a^{n_2} b^{n_2} \ldots a^{n_k} b^{n_k} \mid n_1, n_2, \ldots, n_k \geq 0, k \geq 1\}$

D) None of these

$\{a^n b^n / n \geq 0\}^*$

$= \{a^{n_1} b^{n_1}, a^{n_1} b^{n_1} a^{n_2} b^{n_2}, a^{n_1} b^{n_1} a^{n_2} b^{n_2} a^{n_3} b^{n_3}, \ldots\}$

ε ✓

ab ✓ , abab, . . .

ab aabb ✓

$\{\quad\}^2 = \{a^{n_1} b^{n_1}\}_{n_1=1} \{a^{n_2} b^{n_2}\}_{n_2=2}$

$= ab\ aabb$ ✓

$\{(a^n b^n)^* / n \geq 0\}$

$= \{a^n b^n, a^n b^n a^n b^n, a^n b^n a^n b^n a^n b^n, \ldots\}$

$(a^n b^n)^2$

$n=1$

abab ✓

ab aabb ✗ not possible

$n=2$

aa bb aa bb

Normal Forms :

→ Chomsky Normal Form (CNF CFG)

$S \to SA \mid a$
$A \to b$
} CNF

Every production must satisfy :

$V \to VV \mid T$

→ Greibach Normal Form (GNF CFG)

$S \to aSS \mid bS \mid c$
GNF

$V \to TV^*$

$S \to abcd$

not in CNF
not in GNF

$V \to VV \mid T$

CNF

$V \to TV^*$

GNF

CNF CFG

$S \to C_a C_b C_c C_d$

$S \to C_a X$
$X \to C_b Y$
$Y \to C_c C_d$
$C_a \to a$
$C_b \to b$
$C_c \to c$
$C_d \to d$

GNF CFG

$S \to a C_b C_c C_d$

$C_b \to b$
$C_c \to c$
$C_d \to d$

Note:

I) If CFG is not having null rules then we can always Convert into CNF CFG and GNF CFG.

II) If we construct parse tree using CNF CFG then it always looks like binary tree.

$V \rightarrow VV$

$V \rightarrow T$

Note:

I) For deriving n length string using CNF grammar,

No. of steps = $2n-1$
(Length of derivation)

II) For deriving n length string using GNF grammar,

No. of steps = n

Context Free Grammar

Using CNF CFG:

$w = a$

$S \Rightarrow a$ ($S \to a$)

#steps = 1

$w = ab$

$S \Rightarrow XY$ ($S \to XY$)

$\Rightarrow aY$ ($X \to a$)

$\Rightarrow ab$ ($Y \to b$)

#step = 3

$w = abc$

$S \Rightarrow XY$ ($S \to XY$)

$\Rightarrow aY$ ($X \to a$)

$\Rightarrow aAB$ ($Y \to AB$)

$\Rightarrow abB$ ($A \to b$)

$\Rightarrow abc$ ($B \to c$)

#steps = 5

$w = abcd$

$\#steps = 2*n - 1$

$= 2*4 - 1$

$= 7$

Context Free Grammar

Using GNF CFG:

$w = a$

$S \Rightarrow a$ ($S \to a$)

#steps = 1

$w = ab$

$S \Rightarrow aA$ ($S \to aA$)

$\Rightarrow ab$ ($A \to b$)

#steps = 2

$w = abc$

$S \Rightarrow aBC$

$\Rightarrow abC$

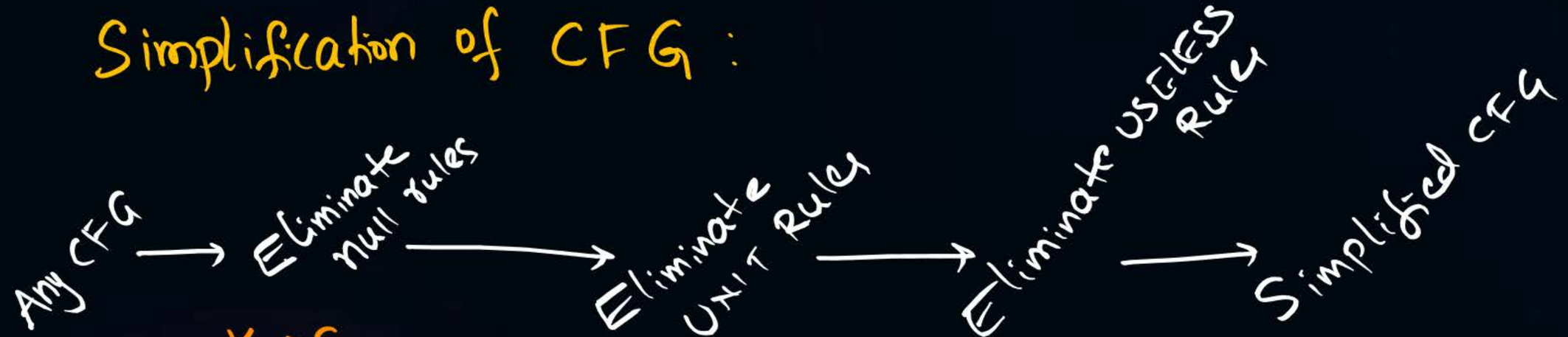
$\Rightarrow abc$

#steps = 3

$w = abcd$

#steps = n = 4

Simplification of CFG :



Elimination of NULL Rules:

$S \rightarrow aA \mid \varepsilon$

$A \rightarrow bSa \mid Bd$

$B \rightarrow aAe \mid \varepsilon$

Step1: Delete $S \rightarrow \varepsilon$

put ε in all places of S

$S \rightarrow aA$

$A \rightarrow bSa \mid Bd \mid ba$

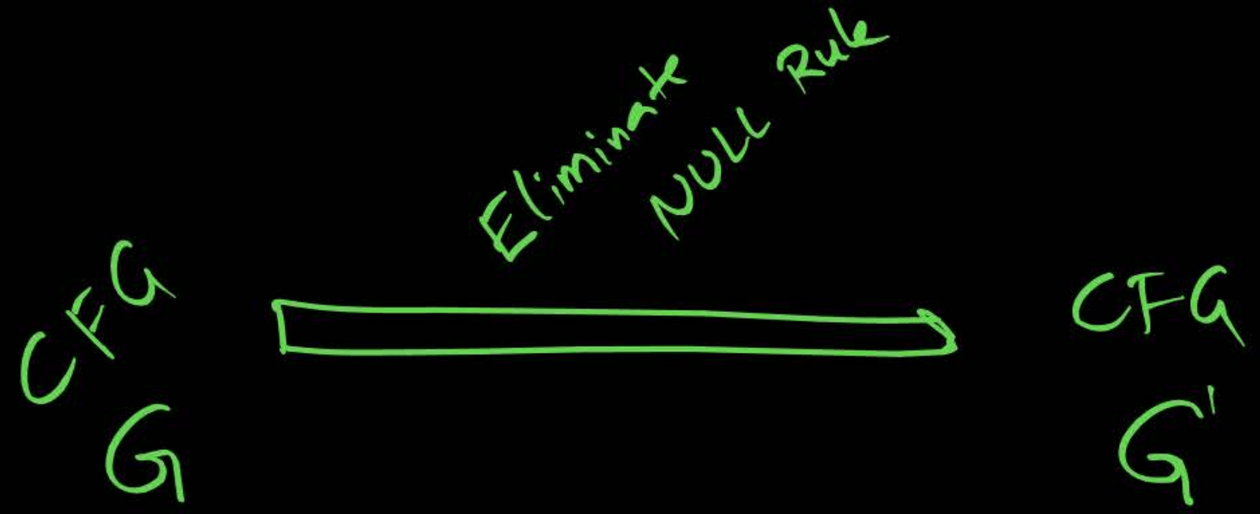
$B \rightarrow aAe \mid \varepsilon$

Step2: Delete $B \rightarrow \varepsilon$

$S \rightarrow aA$

$A \rightarrow bSa \mid Bd \mid ba \mid d$

$B \rightarrow aAe$



$$L(G) - \{\varepsilon\} = L(G')$$

Elimination of UNIT Rules:

$S \to aB \mid A$

$A \to bS \mid B$

$B \to aA \mid S$

Step1: Delete $S \to A$

Put all A productions in S

$S \to aB \mid bS \mid B$

$A \to bS \mid B$

$B \to aA \mid S$

Step2: Delete $S \to B$

$S \to aB \mid bS \mid aA \mid \cancel{S}$

$A \to bS \mid B$

$B \to aA \mid S$

Step3: Delete $A \to B$

already here, already here

$S \to aB \mid bS \mid aA$

$A \to bS \mid aA \mid S$

$B \to aA \mid S$

Step4: Delete $A \to S$

$S \to aB \mid bS \mid aA$

$A \to bS \mid aA \mid aB$

$B \to aA \mid S$

Step5: Delete $B \to S$

$S \to aB \mid bS \mid aA$

$A \to bS \mid aA \mid aB$

$B \to aA \mid aB \mid bS$

Elimination of USELESS Rules and Symbols :

Step1 : Eliminate Non-terminal if it can't derive any string

$S \to aA$

$A \to bB \mid dD$

$D \to e$

$F \to gF$

$E \to f$

S	A	B	D	E	F
⇓	⇓	⇓	⇓	⇓	⇓
Some string	✓	X	✓	✓	X

B and F can't derive any string

$S \to aA$
$A \to dD$
$D \to e$
$E \to f$

Step2 : Delete all unreachable symbols from Start S.

S, A, D

$S \to aA, A \to dD, D \to e$

CFG

⇓

Delete NULL Rules

⇓

Delete UNIT Rules

⇓

Delete USELESS Rules

⇓

Simplified CFG

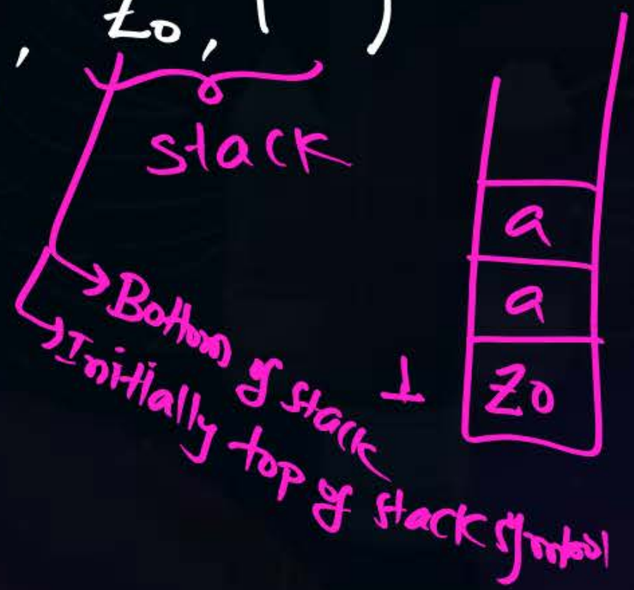
What is Push Down Automata (PDA) ?

- It represents (accepts) a Context Free Language
- It is by definition non-deterministic PDA

$$PDA = (\underbrace{Q, \Sigma, \delta, q_0, F}_{FA}, \underbrace{Z_0}_{(\perp)}, \Gamma)$$

Γ → stack alphabet

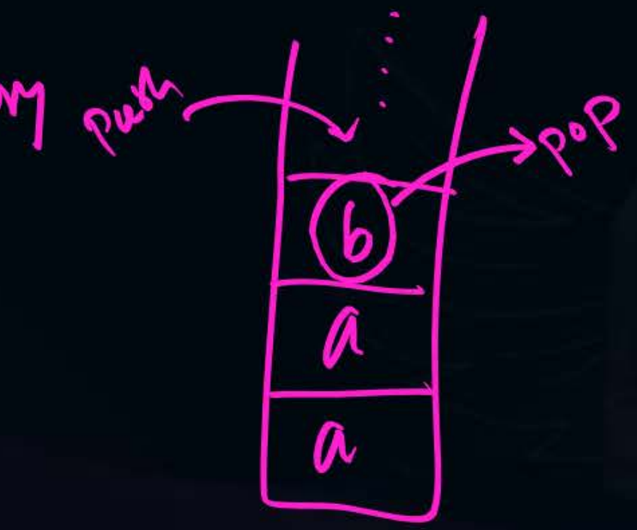
FA + 1 stack $\cong$ PDA



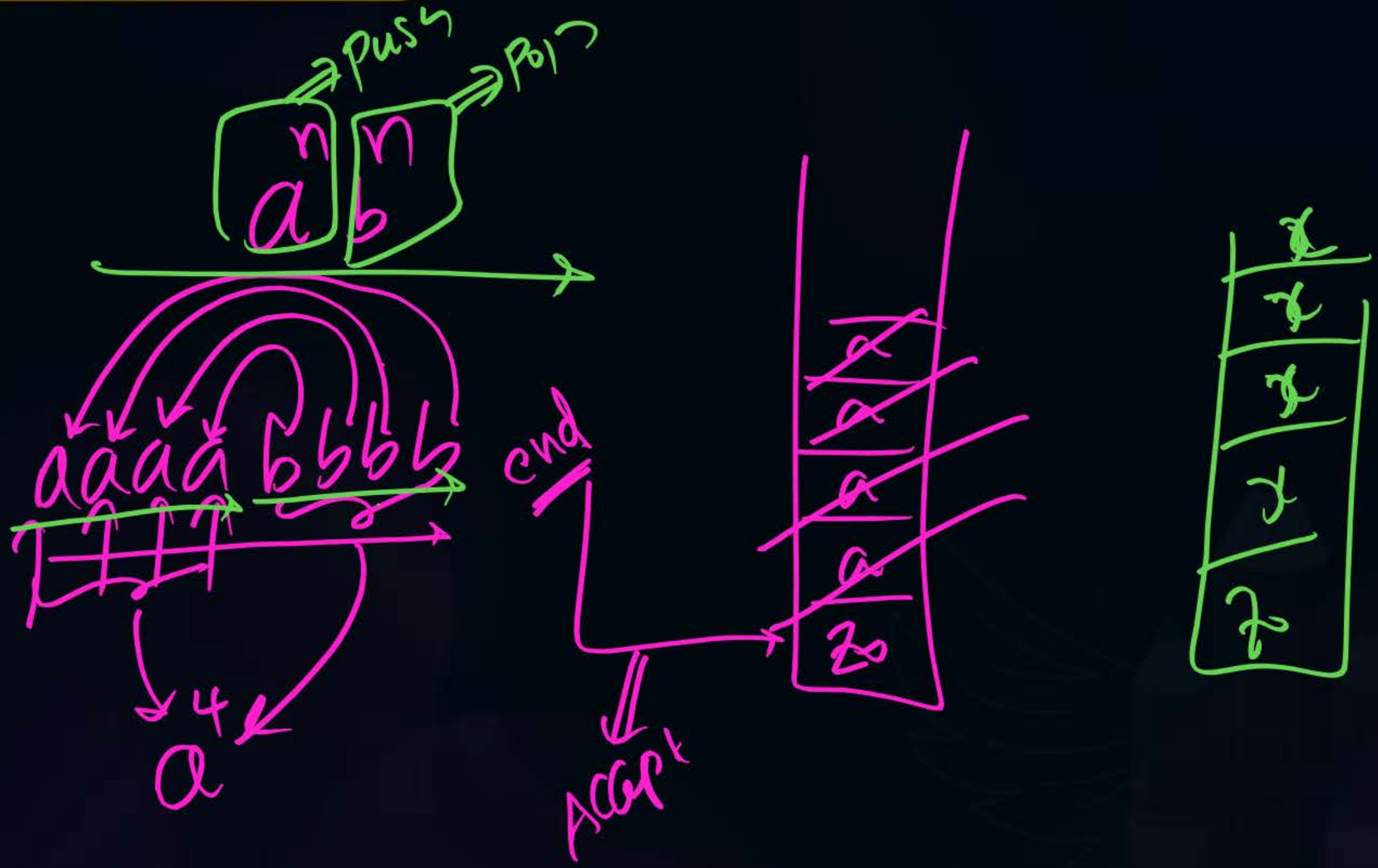
$\Gamma = \{a, Z_0\}$ = set of stack symbols

Stack

- It is a data structure
- LIFO
- Push and Pop happens in LIFO manner
- Unbounded memory



Context Free Grammar



$$PDA = (Q, \Sigma, \delta, q_0, F, z_0, \Gamma)$$

$$\delta_{PDA} : Q \times \Sigma_\varepsilon \times \Gamma^* \longrightarrow 2^{Q \times \Gamma^*}$$

$$\delta_{DPDA} : Q \times \Sigma \times \Gamma \longrightarrow Q \times \Gamma^*$$

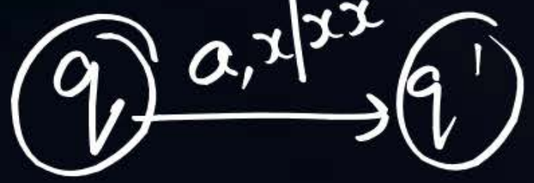
PDA Stack :

↳ I) PUSH

$\delta(q, a, x) = (q', xx)$

(present state, input symbol, top of stack; new push)

$Q \times \Sigma \times \Gamma \longrightarrow Q \times \Gamma^*$



$a, x \rightarrow xx$

a, x, xx

Context Free Grammar

PDA

DPDA:

$\delta(q, a, \varepsilon) = (q, x)$

$\delta(q, a, x) = (q, xx)$

$Q \times \Sigma \times \Gamma \quad Q \times \Gamma^*$

$\delta(q, a, xx) = (q, xxx)$

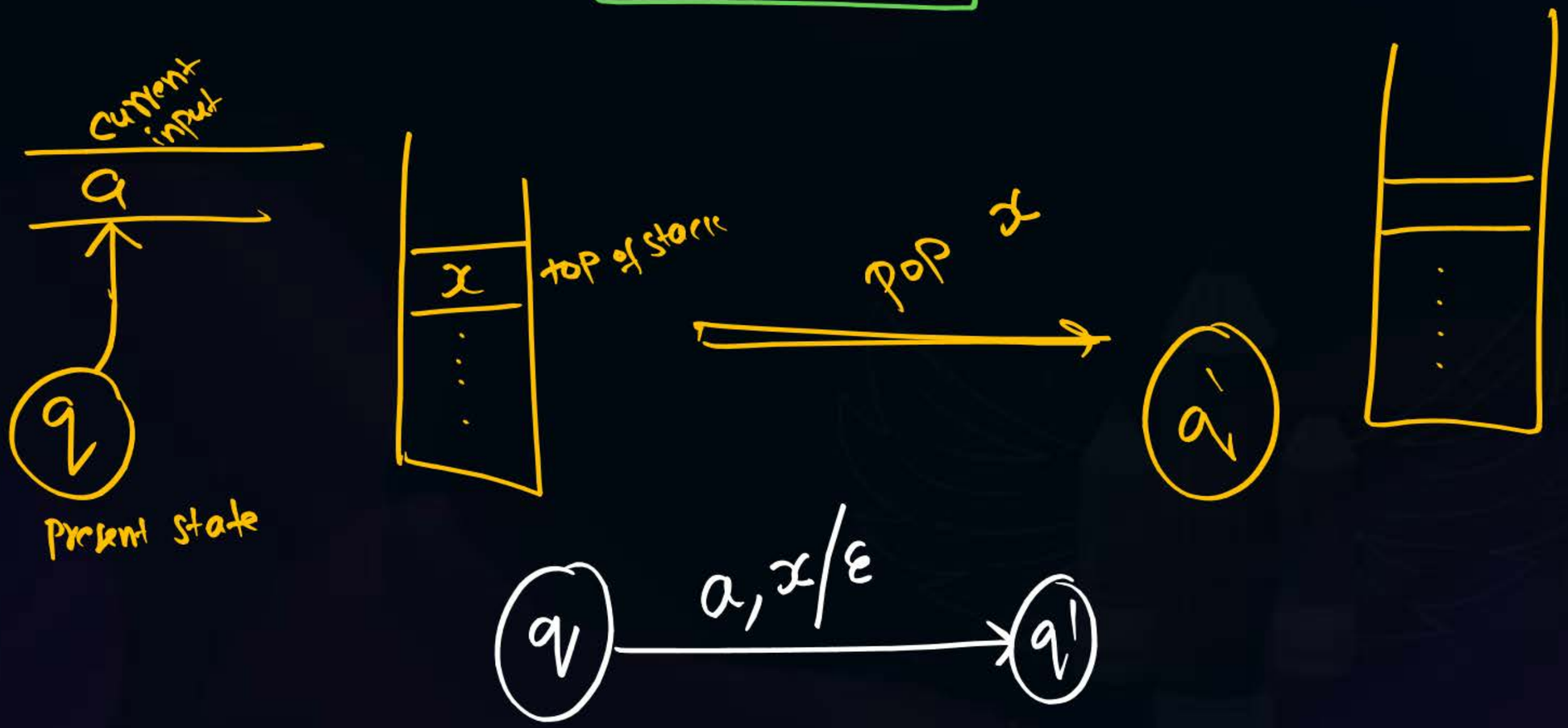
$\delta(q, a, X) = (q, xX)$

Push x

DPDA:

II) Pop :

$$\delta(q, a, x) = (q', \varepsilon)$$



Context Free Grammar

PDA

DPDA

$\delta(q, a, x) = (q', \varepsilon)$ — deleted x

$\delta(q, a, xx) = (q', x)$ — deleted x

$\delta(q, a, xX) = (q', X)$ — deleted x

Pop x

III) No operation:

$$\delta(q, a, x) = (q', x)$$

no change

Context Free Grammar

PDA

IDPDA: $\delta(q, a, x) = (q, x)$

$\delta(q, a, X) = (q, X)$

$\delta(q, a, \varepsilon) = (q, \varepsilon)$

nothing happened on stack

Context Free Grammar

1) $x/x \Rightarrow$ no operation

2) $x/\varepsilon \Rightarrow$ Pop x

3) $\varepsilon/\varepsilon \Rightarrow$ no operation

4) $\varepsilon/x \Rightarrow$ push x

5) $xx/x \Rightarrow$ Pop 2 x's

Note:

I) Every DPDA is PDA

II) PDA & DPDA are not equivalent

III) DPDA $<$ PDA

less powerful

IV) L(DPDA) is DCFL

V) L(PDA) is CFL

PDA Acceptance :

- Using Final state
  ≅
- Using Empty stack

DPDA Acceptance

- Using Final state

2 mins Summary

Topic

CFG Vs CFL

CFG Normal forms

Basics of PDA
- what is PDA?
- Transitions?

Next: PDA construction

THANK - YOU