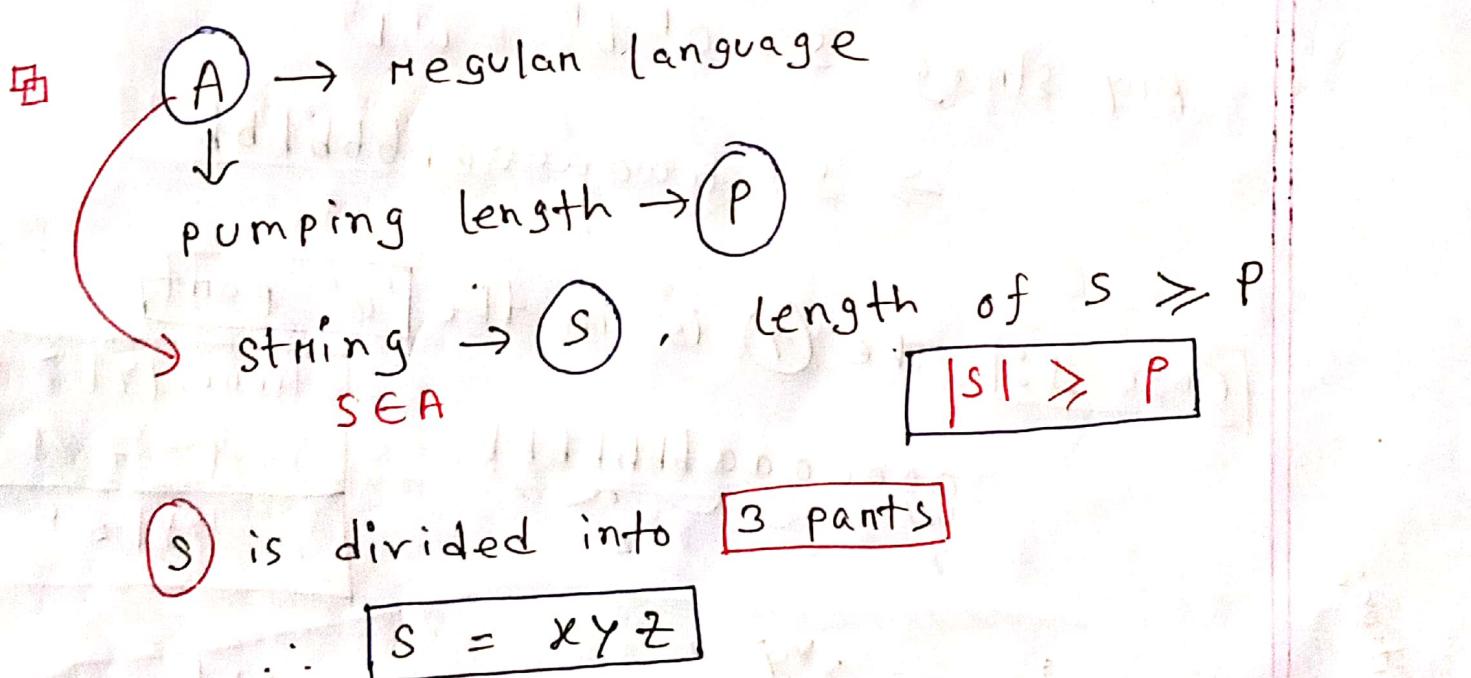
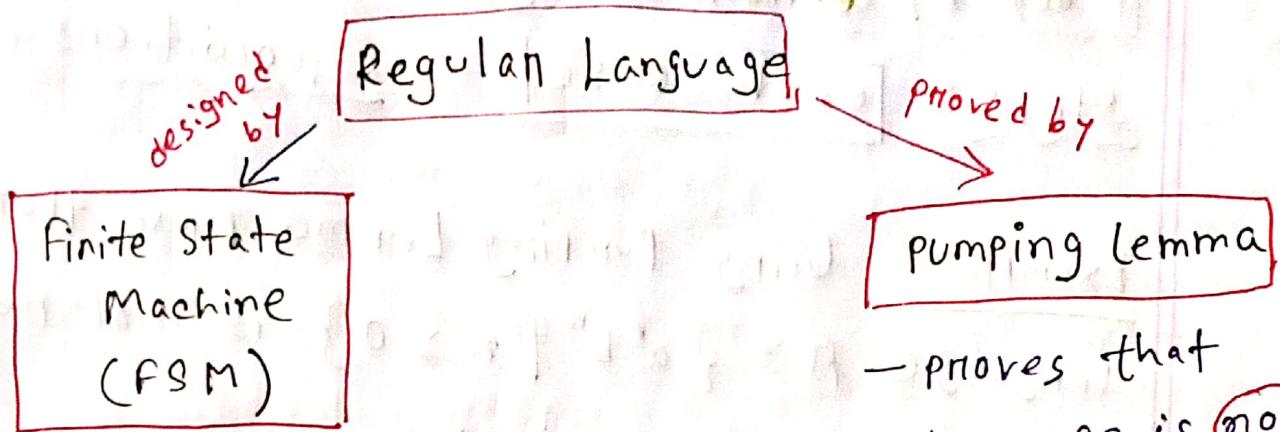


F 7

Pumping Lemma



Conditions to be regular language:

- 1) $xy^iz \in A$ ($i \geq 0$)
 - 2) $|y| > 0$
 - 3) $|xy| \leq P$
- pumping conditions

Proof by contradiction

steps: each case → 3 conditions check

Example: Using Pumping Lemma prove that the language $A = \{a^n b^n \mid n \geq 0\}$ is not regular

Proof: Let $\rightarrow A$ be regular language.
pumping length $\rightarrow P = 7$

Any string, $s = a^P b^P = a^7 b^7$

$$\rightarrow s = aaaaabbbbb$$

Case 1: The y is in the 'a' part

aaaaaabbbbb
 x y z

$$\text{Hence, } |y| > 0$$

$$|ny| = 7 \quad \varphi = 7$$

$$\therefore |ny| \leq P$$

$$xy^1 z \rightarrow x y^1 z$$

$\rightarrow a \underbrace{aaaaaa}_{x} \underbrace{a}_{y^1} \underbrace{bbbbb}_{z}$

Hence, length of 'a' part = 11

length of 'b' part = 7

$$\therefore 11 \neq 7$$

① no condition

② no

③ no

Case 2: The y is in the ' b ' part.

$\underline{aaaaaaaaabbbbbb}$
 $x \quad y \quad z$

Hence, $|y| > 0$
 $|xy| = 9, p = 7$
 $\therefore |xy| \notin P$

$xy^iz \rightarrow xy^z$

$\rightarrow \underline{aaaaaaaaabbbbbb} b$
 $x \quad y^z \quad z$

Hence, length of ' a ' part = 7

length of ' b ' part = 11

$\therefore 7 \neq 11$

Case 3: The y is in the ' a ' and ' b ' part

$\underline{aaaaaaaabb}bbb$
 $x \quad y \quad z$

$|y| > 0$
 $|xy| = 9, p = 7$
 $\therefore |xy| \notin P$

$xy^iz \rightarrow xy^z$

$\rightarrow aaaaa \underline{aabb aabb} bbbb$
 y^z

It doesn't follow the pattern ' $a^n b^n$ '

So, none of these can satisfy all the 3 pumping conditions at the same time. So, S is not regular.

Example: Using pumping lemma prove that language $A = \{yy \mid y \in \{0,1\}^*\}$ is not regular.

Proof: Let A be a regular language with pumping length p

$$\text{let } s = \underbrace{0^p 1}_{y} \underbrace{0^p 1}_{y}$$

for $p=7$, $s = \underbrace{0000000}_{x} \underbrace{1}_{y} \underbrace{00000001}_{z}$

① $xy^iz \rightarrow xy^mz$
 $\rightarrow \underbrace{0000000}_{x} \underbrace{0}_{y^m} \underbrace{00000001}_{z} \notin A$ \times

② $|y| = 4 > 0 \quad \checkmark$

③ $|xy| = 6 \leq p = 7 \quad \checkmark$

Hence, ① no. condition isn't fulfilled, A is not a regular language.

Regulan Gramman

Context free grammar accepts

context free language

accepted by Automata used:

Pushdown automata

Regulan grammar → regulan language

automata : finite state automata

Gramman 'G' → described formally using
4 tuples

$$G = (V, T, S, P)$$

V → variables / non-terminal symbols' set

T → terminal symbols

S → start symbol

P → production rules of terminal and non-terminals

form: $\alpha \rightarrow \beta$ [α, β are strings on VUT]

at least a symbol of $\alpha \in V$

non-terminal ← variables

Ex: $G_1 = (\{S, A, B\}, \{a, b\}, S, \{S \rightarrow AB, A \rightarrow a, B \rightarrow b\})$

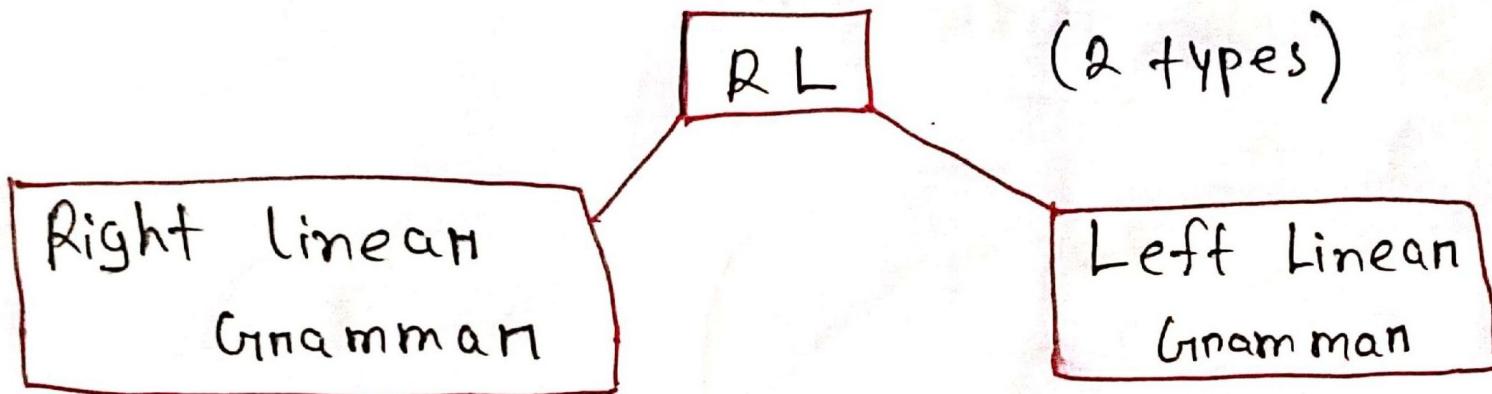
Hence, $V = \{S, A, B\}$

$T = \{a, b\}$

$S = S$

$P = S \rightarrow AB, A \rightarrow a, B \rightarrow b$

Eg: $S \rightarrow AB \rightarrow \underline{\underline{a}} \underline{\underline{B}} \rightarrow \underline{\underline{a}} \underline{\underline{b}}$ ——————



Eg : $A \rightarrow x B x$ *variable in the right side*
 $A \rightarrow x$

Eg : $A \rightarrow B x$
 $A \rightarrow x$

N.B. :
 at least
 a variable
A in the
LHS

Example :

$S \rightarrow ab \underline{S} b \rightarrow$ Right linear

$S \rightarrow \underline{S} bb b \rightarrow$ Left linear

Context Free Grammar

↳ CFG forms

described using 4 tuples.

$$G_1 = \{ V, \Sigma, S, P \}$$

(V) → set of variables / non-terminals

(Σ) → set of terminals

(S) → start symbol

(P) → production rule

$$A \rightarrow a$$

In the left, variable / NT

here, $A \in V \rightarrow$ variable / NT
 $a = \{ V \cup \Sigma \}^*$ → both non-term and terms

In the right

Ex: For generating a language that generates equal number of a's and b's in the form $a^n b^n$, the CFG will be defined as $G_1 = \{ (S, A), (a, b), (S \rightarrow aAb, A \rightarrow aAb \mid \epsilon) \}$

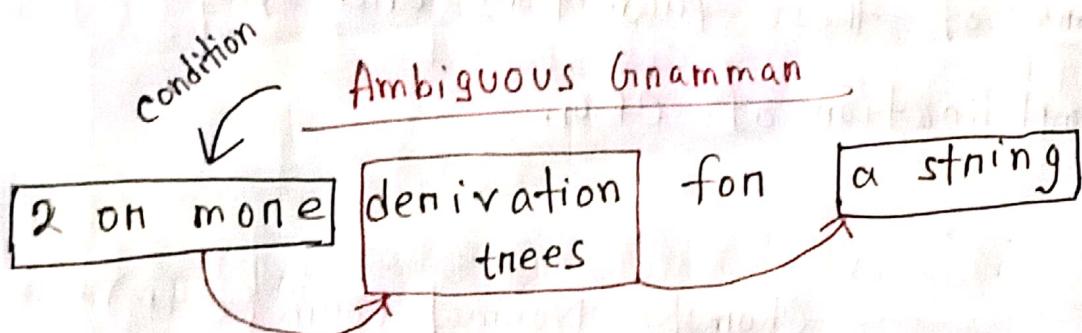
Proof:

$$\begin{aligned} \rightarrow S &\rightarrow a \underline{A} b \\ &\rightarrow a \underline{a} \underline{A} b \quad (A \rightarrow aAb) \end{aligned}$$

= $a a \underline{a A b} b b \dots$ ($A \rightarrow a \cap b$ again)

= $a a a b b b \quad (A \rightarrow \epsilon)$

= $a^3 b^3 \Rightarrow a^n b^n$



Example: $G_1 = (\{S\}, \{a+b, +, *\}, \{S \rightarrow S+S | S^*S | a+b\})$

The string $a+a^*b$ can be derived as:

$S \rightarrow S+S$
 $\rightarrow a+S$
 $\rightarrow a+S+S$
 $\rightarrow a+a^*S$
 $\rightarrow a+a^*b$

(2 on more ways)

$S \rightarrow S * S$
 $\rightarrow S + S * S$
 $\rightarrow a + S * S$
 $\rightarrow a + a^* S$
 $\rightarrow a + a^*b$

Thus this grammar G_1 is ambiguous

Simplification of CFG

* In CFG, sometimes all the production rules and symbols are not needed for the derivation of strings. Besides this, there may also be some null productions and unit productions. Eliminations of these productions and symbols is called simplification of CFG.

Chomsky Normal Form (CNF)

productions are in the following forms:

$$\boxed{A \rightarrow a}$$
$$\boxed{A \rightarrow BC}$$

Hence, A, B, C \rightarrow non terminals
a \rightarrow terminals

Conditions on RHS

- 1) either be two variables / NT
- 2) or a Terminal

$$NT \rightarrow NT \text{ (2 ft)}$$
$$\rightarrow T \text{ (1 ft)}$$

CFG \rightarrow CNF

Steps:

① Let start symbol be S

new start symbol be S'

new production $S' \rightarrow S$

②

remove null productions

③

remove unit productions

④

replace each production $A \rightarrow B_1 \dots B_n$ ($n > 2$)
with $A \rightarrow B_1 C \dots (C \rightarrow B_2 \dots B_n)$

more than 2
NT on RHS

To make
goal:
2 NT on
the
RHS

Repeat this step for all productions having
2 or more symbols on right side.

Replace production $A \rightarrow aB$ (a - Terminal,
 $B, A, x - NT$)

with $A \rightarrow xB$ and $x \rightarrow a$

Repeat this step for all productions having
terminals and non-terminals on right side.

when
both NT
and T
are in RHS

Goal:

- 1) simplifying grammar
- 2) reducing structural ambiguity

* Example: Convert from CFG to CNF:

$$P: S \rightarrow ASA | aB, A \rightarrow B | s, B \rightarrow b | \epsilon$$

→ ① Let new start symbol be S'
and new production $S' \rightarrow S$

$$P: S' \rightarrow S, S \rightarrow ASA | aB, A \rightarrow B | s, B \rightarrow b | \epsilon$$

② Removing null productions: $B \rightarrow \epsilon, A \rightarrow \epsilon$

a) Removing $B \rightarrow \epsilon$:

$$P: S' \rightarrow S, S \rightarrow ASA | aB | a, A \rightarrow B | S | \epsilon, B \rightarrow b$$

b) Removing $A \rightarrow \epsilon$:

$$P: S' \rightarrow S, S \rightarrow ASA | . aB | a | AS | SA | S, A \rightarrow B | S, B \rightarrow b$$

③ Removing unit productions:

$$S \rightarrow S$$

$$S' \rightarrow S$$

$$A \rightarrow B$$

$$A \rightarrow S$$

(a) Removing $S \rightarrow S$:

p: $S' \rightarrow S, S \rightarrow ASA | aB | a | AS | SA, A \rightarrow B | S, B \rightarrow b$

(b) Removing $S' \rightarrow S$:

p: $S \rightarrow ASA | aB | a | AS | SA, S' \rightarrow ASA | aB | a | AS | SA,$

$A \rightarrow B | S, B \rightarrow b$

(c) Removing $A \rightarrow B$:

p: $S \rightarrow ASA | aB | a | AS | SA, S' \rightarrow ASA | aB | a | AS | SA,$

$A \rightarrow b | S, B \rightarrow b$

(d) Removing $A \rightarrow S$:

p: $S \rightarrow ASA | aB | a | AS | SA, S' \rightarrow ASA | aB | a | AS | SA,$

$A \rightarrow b | ASA | aB | a | AS | SA,$

$B \rightarrow b$

④ finding out the productions that has more than 2 variables in RHS

$$\left\{ \begin{array}{l} S \rightarrow ASA \\ S' \rightarrow ASA \\ A \rightarrow ASA \end{array} \right.$$

After removing these, we get:

$$P: S \rightarrow AX | aB | a | AS | SA$$

$$S' \rightarrow AX | aB | a | AS | SA$$

$$A \rightarrow b | AX | aB | a | AS | SA$$

$$B \rightarrow b$$

$$X \rightarrow SA$$

⑤ changing the productions $\frac{S \rightarrow aB}{A \rightarrow aB}, \frac{S' \rightarrow aB}{}$

finally we get:

$$P: S \rightarrow AX | YB | a | AS | SA$$

$$S' \rightarrow AX | YB | a | AS | SA$$

$$A \rightarrow b | AX | YB | a | AS | SA$$

$$B \rightarrow b$$

$$X \rightarrow SA, Y \rightarrow a$$

which is the required CNF for the given CFG

Pushdown Automata

[PDA] is a way to implement a context-free grammar.

So, {Context-free grammar is a language which is recognized by PDA}.

* Diff. between FSM and PDA:

FSM

- ① less powerful
- ② limited memory
- ③ don't have stack
- ④ implements regular language.

PDA

- ① more powerful
- ② more memory
- ③ has a stack with infinite memory
- ④ implements context-free grammar.

push — new element added

pop — top element read and removed

PDA components : (3)

- 1) input tape — takes input
- 2) finite control unit — accept/reject
- 3) stack — push/pop

* PDA is formally defined by 7 tuples :

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

Q — finite set of states

Σ — finite set of input symbols

Γ — finite stack of alphabets

δ — transition function

q_0 — start state

z_0 — start stack symbol (\$)

F — set of final / accepting states

δ (transition function)

takes triple $[\delta(q, a, X)] \rightarrow \text{input}$

Hence, (1) - state $\in Q$

(2) - input $\in \Sigma$

on empty symbol

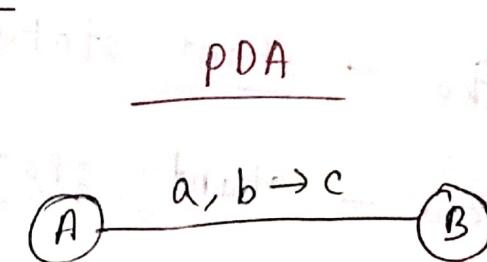
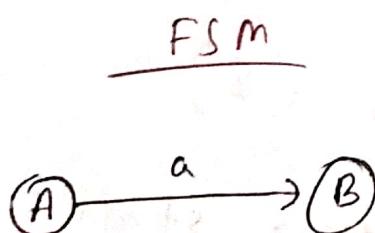
(3) - stack symbol $\in \Gamma$

\rightarrow output pairs (P, γ)

(1) - new state

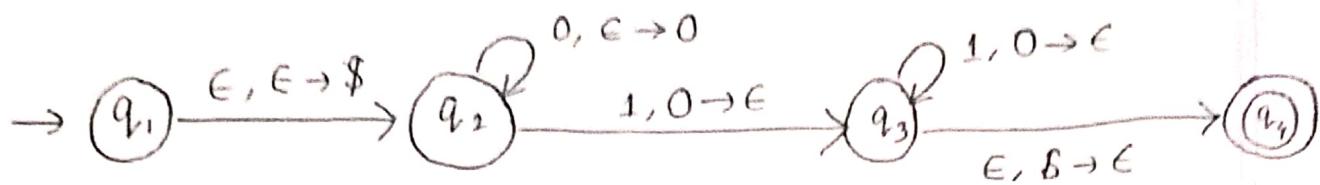
(2) - string of stack symbols

Graphical notation

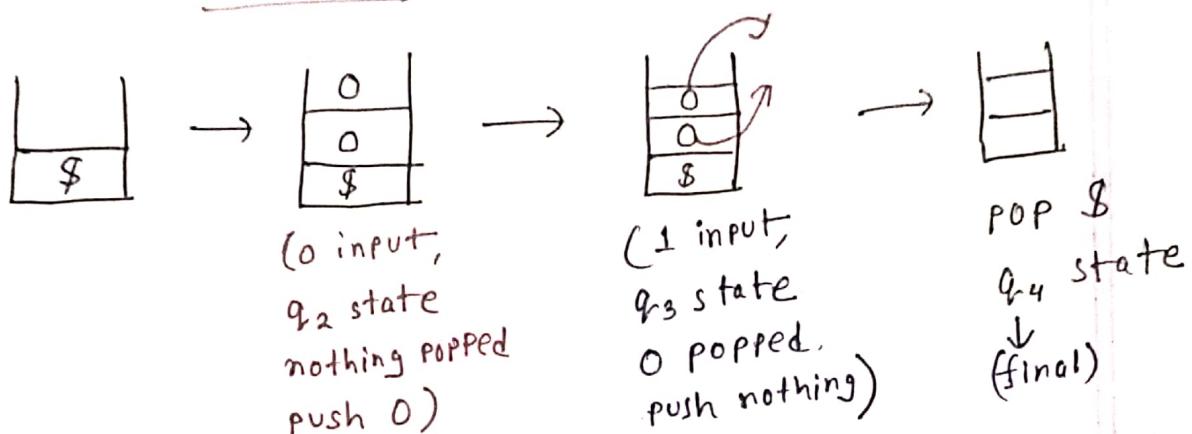


Hence,
a - input
b - if top element, then
pop
c - push

Ex: construct a PDA that accepts $L = \{ 0^n 1^n \mid n \geq 0 \}$



Suppose, [0 0 1 1]



Ex: PDA that accepts even palindromes of

the form

$$L = \{ w w^R \mid w = (a+b)^+ \}$$

