Pumping Lemma: for Regular Languages:

- It satisfies Regular Language
- It proves regular language
- It can be used to prove non-regular languages - using contradiction.

Note: It should not be used to identify regular or non-regular.

Pumping Lemma for Regular(L):

1) choose pumping constant (P) for L

$P \geq$ no. of states in FA for L.

2) For every string w in L which has length $\geq P$

Such that $xyz = w$

   i) Divide w into 3 parts x, y, z

   ii) $|xy| \leq P$

   iii) $y \neq \varepsilon$

3) $\forall i \geq 0 \; xy^iz \in L$ iff L is Regular

Q1) If $L = aab(a+b)^*$ then which of the following can be pumping constant?

A) 2 ✗

B) 3 ✗

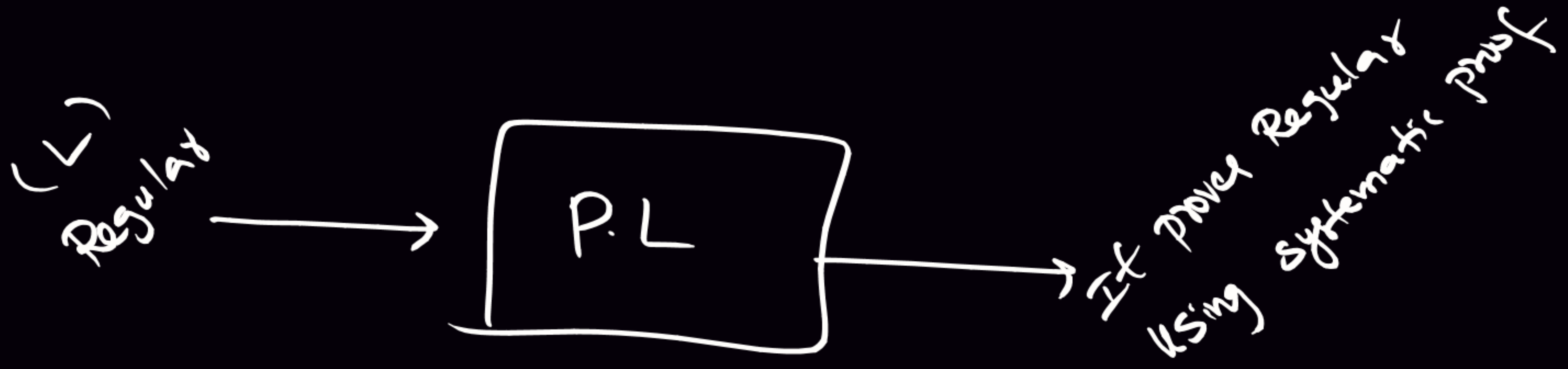
C) 10 ✓

D) 5 ✓

≥ 4

$L = aab(a+b)^*$

↳ 4 states in min NFA


(L)
Regular
P.L
It proves Regular
using systematic proof

$L = aab(a+b)^*$

≥ 4

1) choose $P = 5$

2) $\forall w$ in L, $|w| \geq P$ (5),

$w = aabab$

$x = aab$, $y = \boxed{a}$, $z = b$

| aabab: aab, ab, ε ✓

| aabab: aa, ba, b ✓ ...

$\forall i \geq 0 \; xy^iz \in L$

$\forall i \geq 0 \; aab(a)^ib \in L$ $\quad \therefore$ L is Regular

$L=\{a^n b^n\}$

Assume L is Regular

1) $P=2n$

2) $w=a^n b^n$

$a^{n-1}\ \boxed{ab}\ b^{n-1}$

$i=0 \Rightarrow a^{n-1}b^{n-1} \in L$

$i=1 \Rightarrow a^{n-1}ab\,b^{n-1}=a^nb^n \in L$

$i=2 \Rightarrow a^{n-1}abab\,b^{n-1} \notin L$

$\exists i,\ xy^iz \notin L$, ∴ contradiction.

$w=a^nb^n$

$a^n\ \ b^n\ \ \varepsilon$

$i=0 \Rightarrow a^n(b^n)^0=a^n \notin L$

So, L is not regular

. . .

FA with O/P : [Transducer]

$$FA = (Q, \Sigma, \delta, q_0, \Delta, \lambda)$$

- λ → O/P function
- Δ → O/P Alphabet

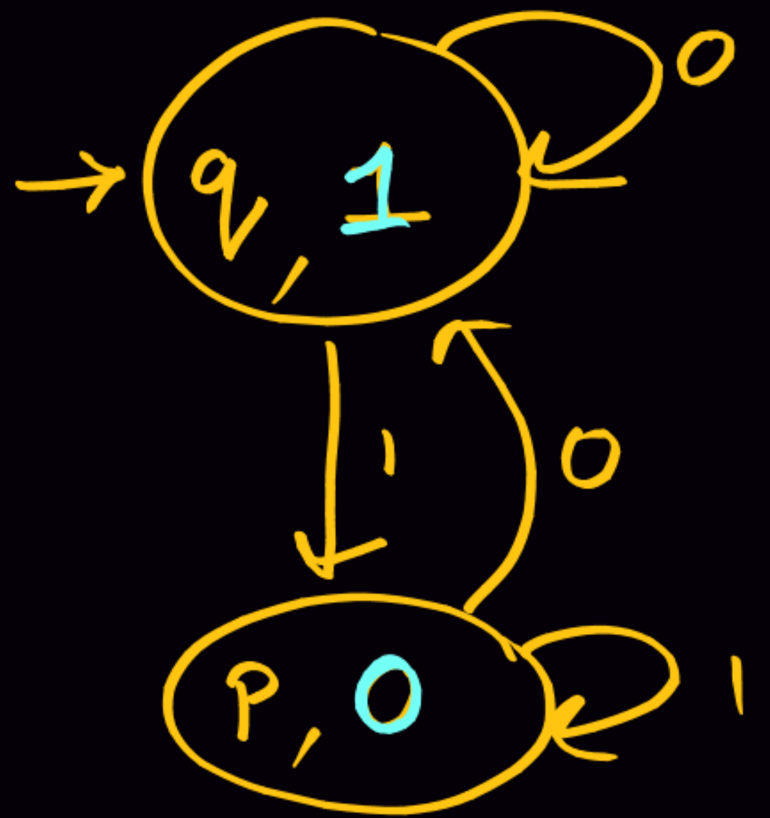
$$\delta_{DFA} \begin{cases} \delta_{Moore} : Q \times \Sigma \rightarrow Q \\ \delta_{Mealy} : Q \times \Sigma \rightarrow Q \end{cases}$$

$\lambda_{Moore} : Q \rightarrow \Delta$

$\lambda_{Mealy} : Q \times \Sigma \rightarrow \Delta$

$\Sigma = \{0, 1\}$ I/P Alphabet

$\Delta = \{0, 1\}$ O/P Alphabet



O/p associated with state in Moore m/c.

Input: 011

$$\underset{1}{q} \xrightarrow{0} \underset{1}{q} \xrightarrow{1} \underset{0}{p} \xrightarrow{1} \underset{0}{p}$$ Halt

O/p: $\boxed{1}$ 1 0 0

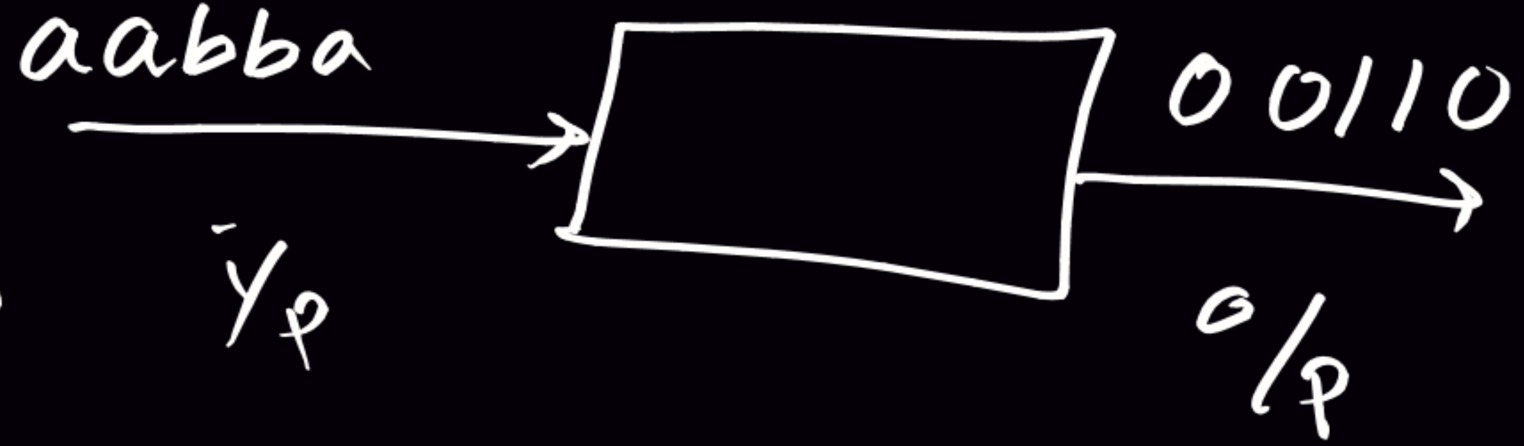
ignore

O/p associated with transition in Mealy m/c

$$A \xrightarrow[1]{0} A \xrightarrow[0]{1} A \xrightarrow[0]{1} A$$

O/p: 100

1) For every a, produce 0
For every b, produce 1.

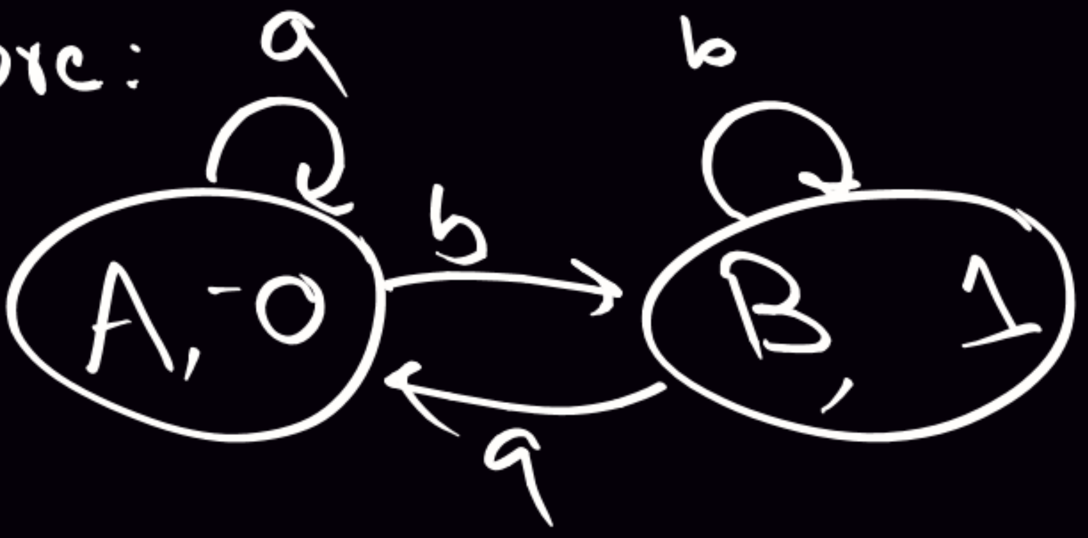


$\Sigma = \{a, b\}$

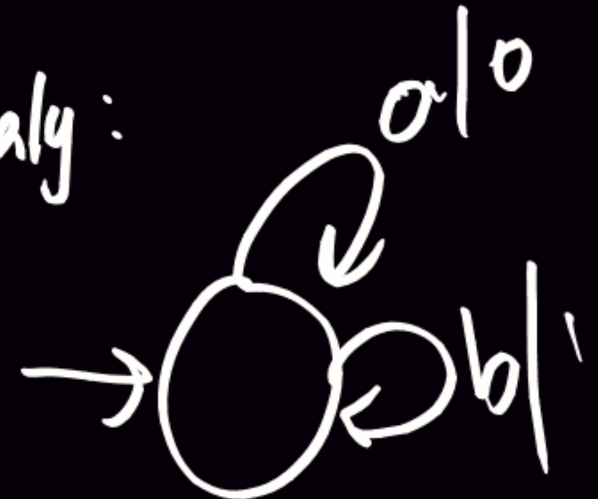
$\Delta = \{0, 1\}$

δ	a	b	o/p
$\rightarrow$A	A	B	0
B	A	B	1

Moore:



Mealy:



Moore M/C $\cong$ Mealy M/C

2) Increment of Binary:

$$f(x) = x + 1$$

reverse

0001

~~1111~~0100

reverse

carry stops

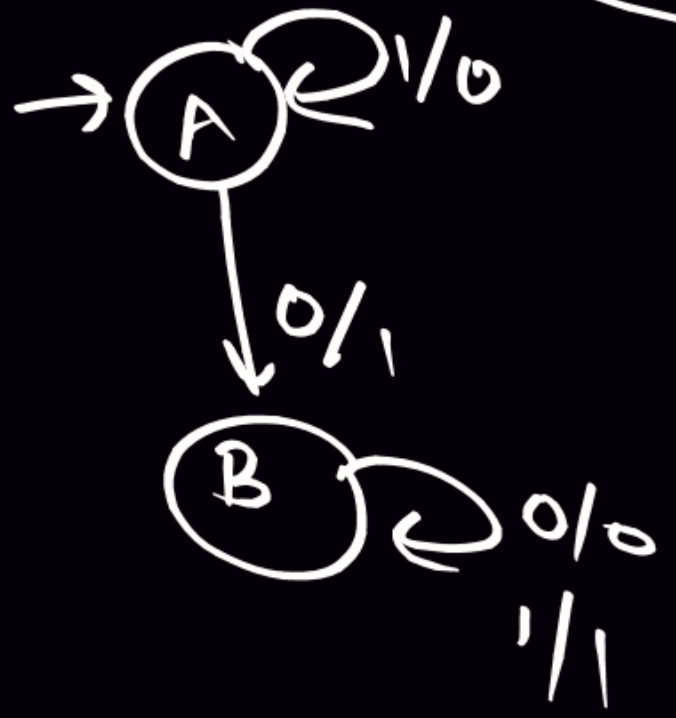
$x: \quad 0010\boxed{0}111$

$+1: \quad \quad \quad \quad \quad \; 1$

$\underbrace{0010}_{\text{same}}1000$

	i/p = 0		i/p = 1	
	NS	o/p	NS	o/p
→ A	B	1	A	0
B	B	0	B	1

Mealy

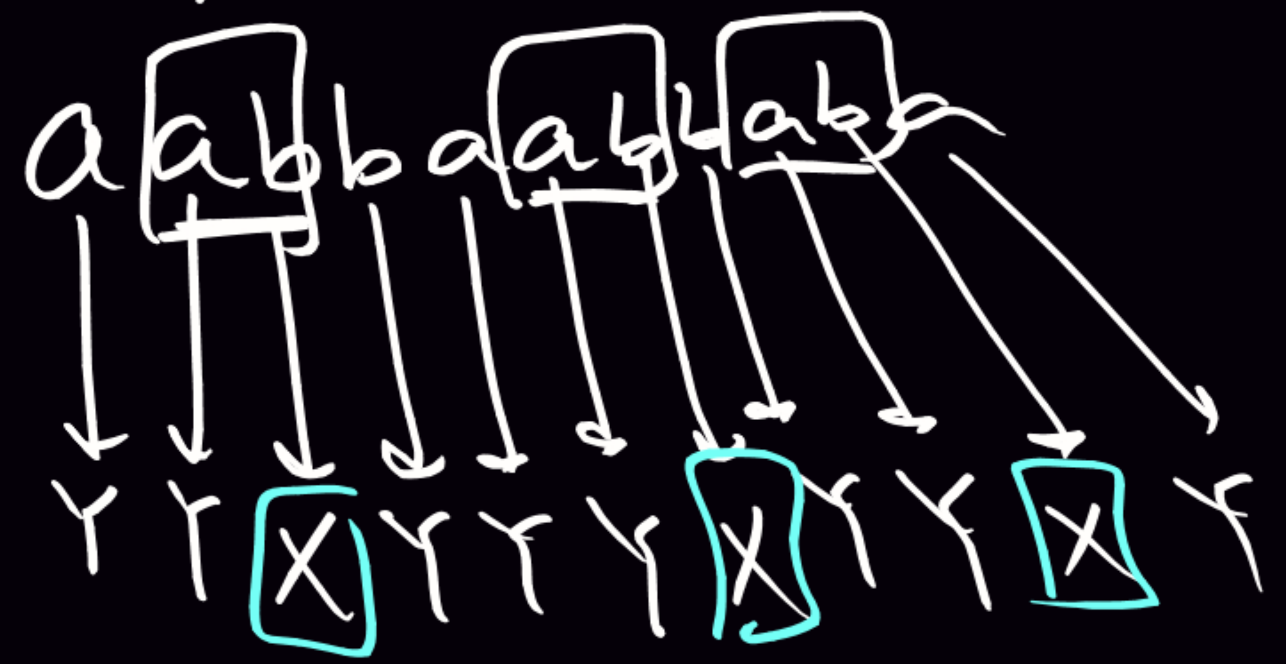


3) Decrement of Binary

4) Addition of 2 binary numbers

5) Subtraction of 2 " "

6) Count no. of occurrences of ab's.



CNF CFG

$$V \to VV \mid T$$

Example:

$$S \to SS \mid AS \mid b$$

$$A \to a \mid b \mid SA \mid c$$

To derive n length string, it require 2n-1 steps

GNF CFG

$$V \to T V^*$$

Example:

$$S \to aSSA \mid b$$

$$A \to cA \mid d$$

To derive n length string, it requires n steps.

← All the best →