Theory of Computation (Saiful sir)

Books: Introduction to theory of Computation
— Anil Maheshwari

Finite Automata:

→ CSL — TM — CSG → All solvable problem

→ CFL — PDA — CFG

→ RL — FA — RG

language → Generative Device

→ Accepting Devide

① Fundamental

Symbol: 0, 1, 2 ... a, b ... Buiding block

Alphabet: Finite set of Symbol. Σ

String: Sequance of symbol over an alphabet.

language: Set of string over an alphabet. finitly or infinitely.

$L = \{$ set of all string of lenth '2' over $\Sigma = \{0, 1\}\}$

$= \{00, 01, 10, 11\}$

$L_1 = \{$ " " " starting with '0' over $\Sigma = \{0, 1\}\}$

$= 0, 01, 011, 0101 \ldots$ infinite.

finite representation

FR → FA

finite acceptor

FSM

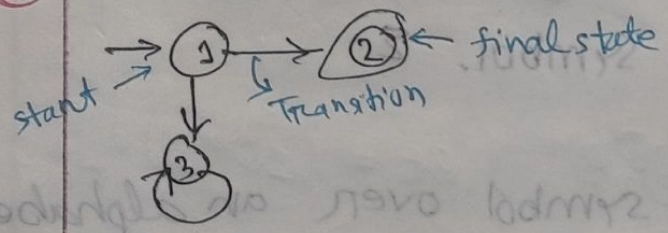
FSA

① Finite Automat & 3 type Representation

Finite automata → specific representation of infin string of a language.

Pictorial

① Transition Diagram: → visuallize of finite automata.



Accepted condition: ②

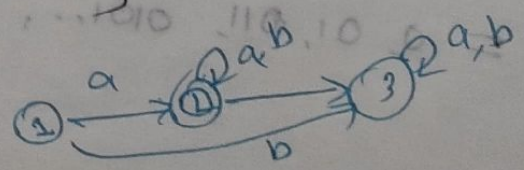
↳ Fully consumed

↳

tabular

② Transition Table:

States	a	b
starting → 1	2	3
2	2	2
Final → 3	3	3



③ ① Mathematical: Math model representati..

↳ FA is a 5 tupple.

① Five tuples.

$M = (Q, \Sigma, \delta, q_0, F)$

→ Q = Set of ~~finite~~ finite states/~~symbol~~

$Q = \{1, 2, 3\}$

→ Σ = Finite set of symbol — Alphabet

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

→ $q_0 = 1$

→ $F = 2$

→ $\delta : \underbrace{Q \times \Sigma}_{\text{Domain}} \rightarrow \underbrace{Q}_{\text{Range}}$

$Q \rightarrow$ states

$\Sigma \rightarrow$ Alphabet

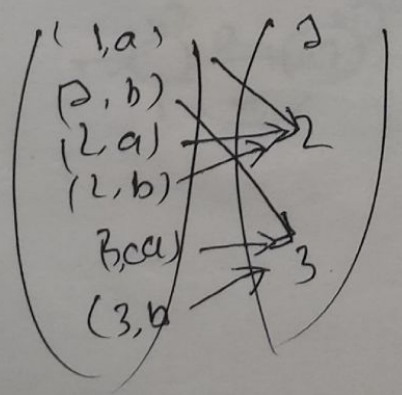
$\delta \rightarrow$ transition f

$q_0 \rightarrow$

$F \rightarrow$

$\{1,2,3\} \times \{a,b\} = \{(1,a), (1,b), (2,a), (2,b), (3,a), (3,b)\}$

Domain



$\delta(1,a) = 2$

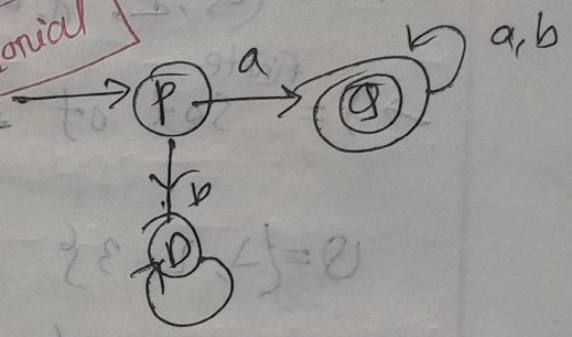
$\delta(1,b) = 3$

Construct a FA/FSA/FSM that accept language of all string starting with 'a' over $\Sigma = \{a, b\}$

$L = \{a, aa, ab \ldots\}$

$M(L) = (Q, \Sigma, \delta, q_0, F)$

pictorial



① Tabular

state	a	b
starting → P	Q	D
Final Q	Q	Q
	D	D

② $Q = \{P, Q, D\}$

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

$q_0 = P$

$F = \{Q\}$

$\delta : Q \times \Sigma \longrightarrow Q$

cartesian product

all are indicate Automata, machin or acceptor.

③

$(P, a) = Q$ $\quad$ $(Q, b) = Q$

$(P, d) = D$ $\quad$ $(D, a) = D$

$(Q, a) = Q$ $\quad$ $(D, b) = D$

clasification:

Finate Automata → without output → DFA, NFA, ε-NFA
↳ with output.
↳ Moore m/c → machin
↳ Mealy M/c

ε → empty, epsilon

transition without input

DFA → Diterminimistic Finate Automata

NFA → Non diternmistic FA.

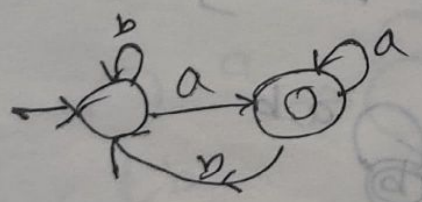
ε-NFA → " " " with ε- transition

③ DFA

DFA construction: → For each input have single diterministic path

construct a DFA that accepts language containing strings. ending with a over $\Sigma = \{a, b\}$

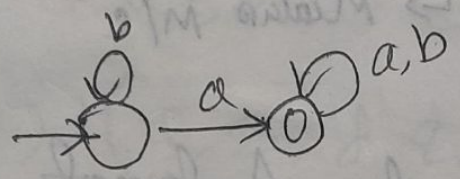
$L = \{a, aa, ba, aaa, bba \ldots\}$



For Σ they have deterin state same amount path.

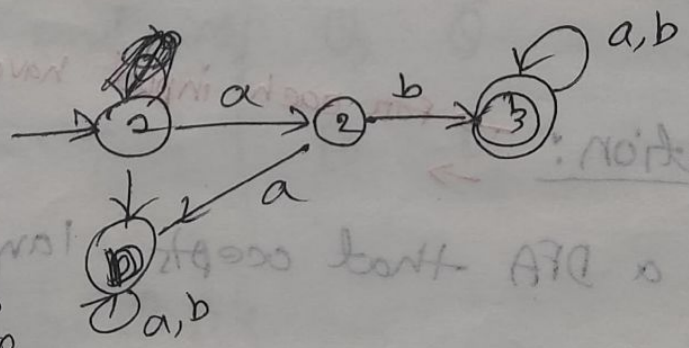
* containing a.

$L = \{a, ab, ba, bab, \ldots\}$



* starting with a,b.

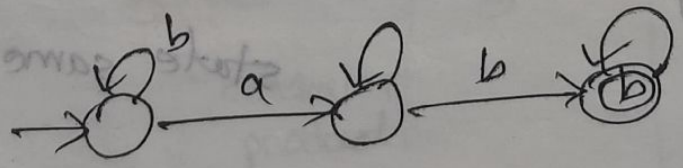
$L = \{a, b, aba, abb, \ldots\}$



Dade state
kill state

* ending with a,b

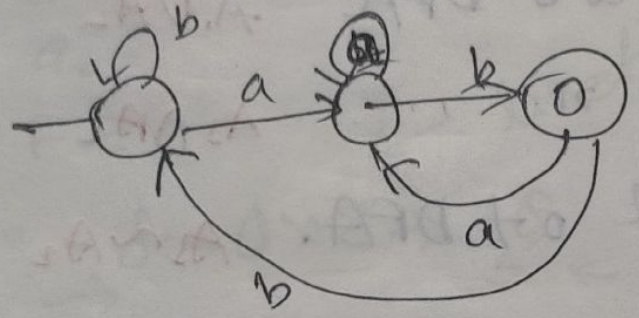
$L = \{a, b, aab, babb, \ldots\}$



Make ditemnistic

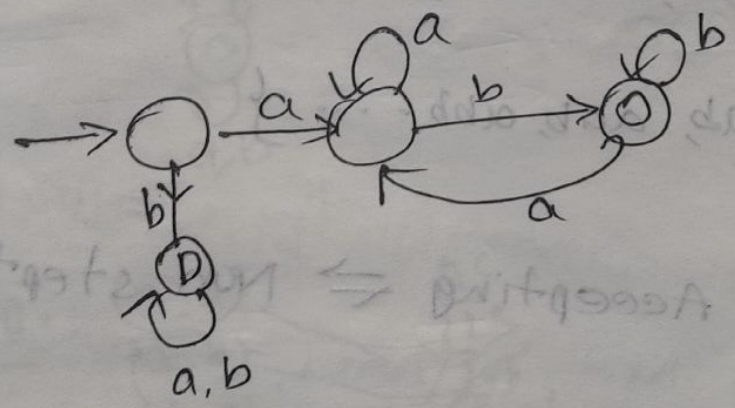
* Ending with ab

L = { a, b, aab, bab, ... }

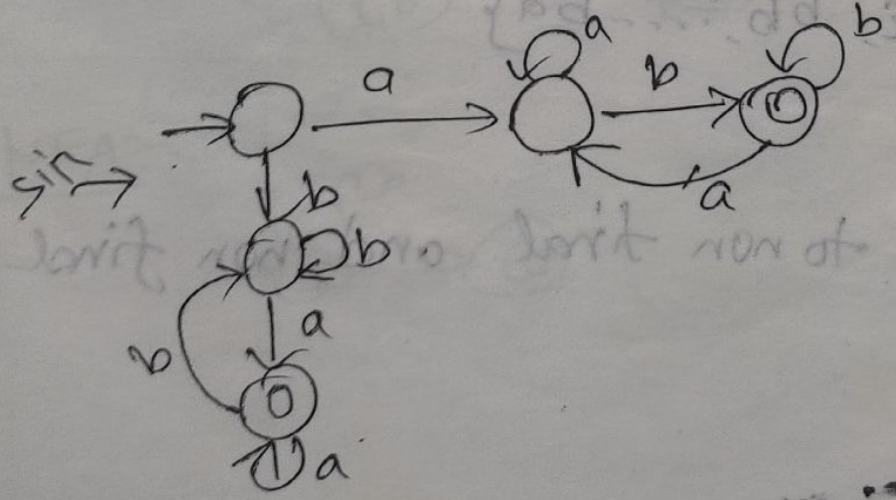


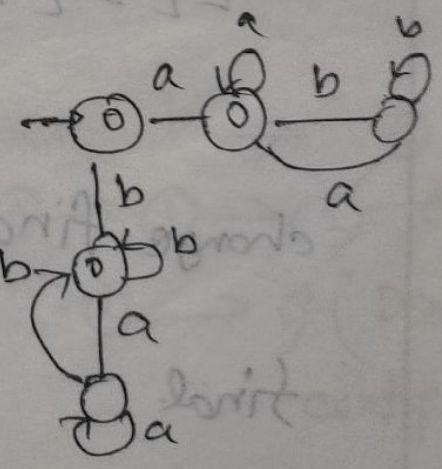
* Starting with a ending with b

L = {a, b, aab, abb, ... }



* Different starting and ending





④ Some operation of DFA

DFA OPERATION: → and For NFA

- ↳ concat of ~~DFA~~ Tw. DFA, $A_1 \cdot A_2$
- ↳ union " TWO DFA. $A_1 \cup A_2$
- ↳ intersection " " $A_1 \cap A_2$
- ↳ Complement of DFA. $A_1 \sim A_2$

Concat.

$L_1 = \{a, aa, ab \ldots\}$

$L_2 = \{b, ab, b, b \ldots\}$

$L_1 \cdot L_2 = L_3 = \{ab, aab, abb \ldots\}$

<u>complement:</u> Accepting ⇌ Non stepting

$L_1 = \{a, aa, ab \ldots\}$

$L_2 = \{b, \varepsilon, bb, \ldots ba\}$

change final to non final and non final to final.

⑤ NFA

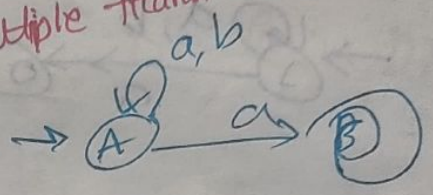
NFA: DFA, except that in NFA there have more than one state attainable from a state for same input alphabet.

→ Q =
→ Σ =
→ a_0 =
→ F =

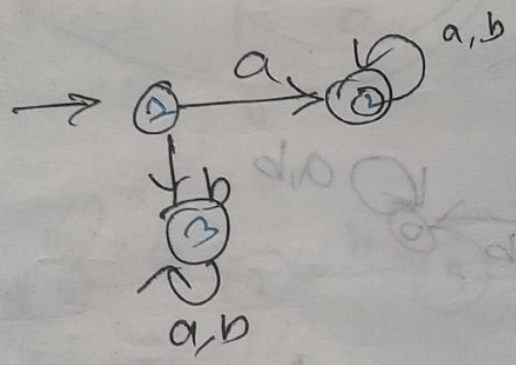
delta $\delta: Q \times \Sigma \to 2^Q$ ← power set / subset

← function

Multiple transition



* starting with a



baa

A → A → A → A
→ B → final
→ B → Q

aab

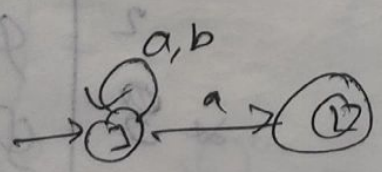
A → A → A → A
→ B → B → Q
→ Q

Maping:

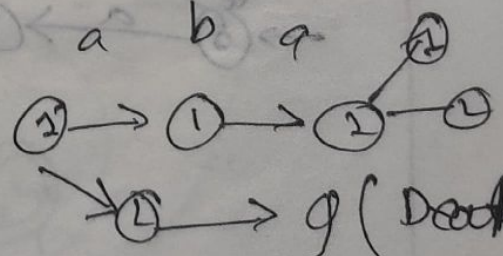
(1, a)
(1, b)
(2, a)
(2, b)
(3, a)
(3, b)

{1}
{2}
{3}
{1, 2}
{1, 3}
{2, 3}
{1, 2, 3}

NFA:

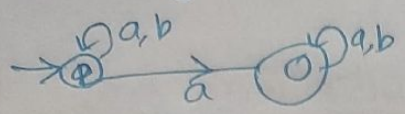


a b a

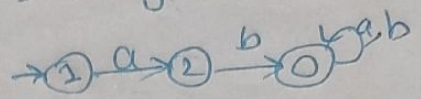


→ q (Dead configuration)

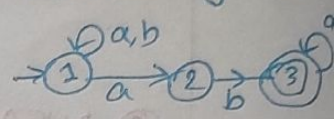
starting with a



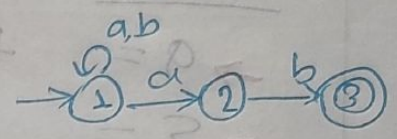
starting with ab



containing 'ab'

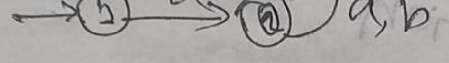


ending with ab

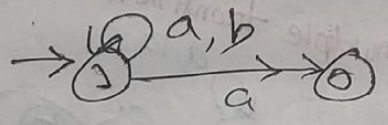


NFA:

→ starting with a of $\Sigma\{a,b\}$

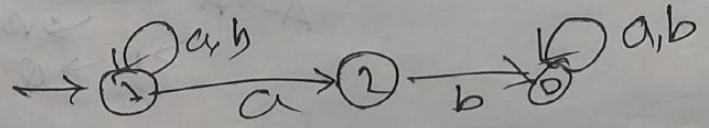


→ ~~starting~~ ending with a



[more than 1 state for a state]

→ containing ab



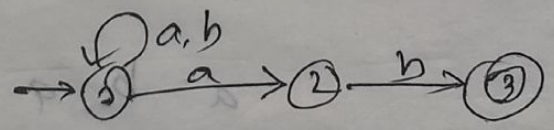
⑥ Conversion

[1] DFA, NFA conversion (DFA ⇌ NFA)

→ all DFA is NFA, no Need to convert DFA → NFA

Subset conversion

* conversion to NFA → DFA



Transition Table (NFA)

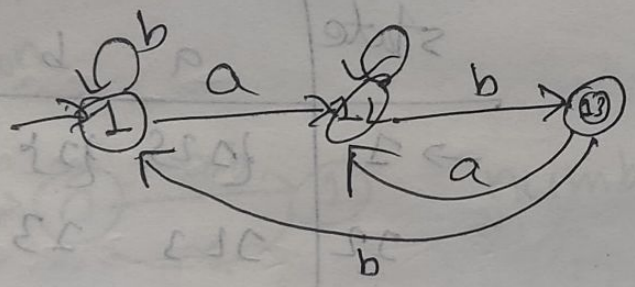
state	a	b
→ 1	{1,2}	{1}
2	ϕ	{3}
* 3	ϕ	ϕ

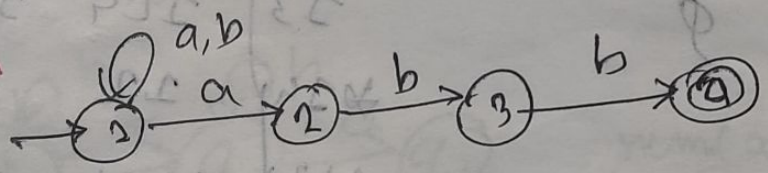
DFA

state	a	b
→ 1	12	1
12	12	13
* 13	12	1

12 → union {1,2} ∪ φ

*3 is final in NFA, state containing 3 is final in DFA





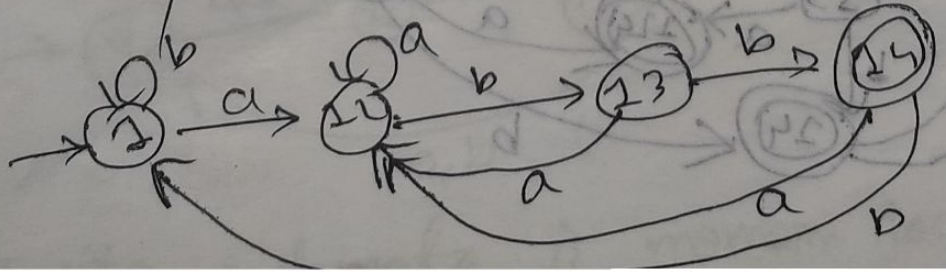
NFA - ending (abb)

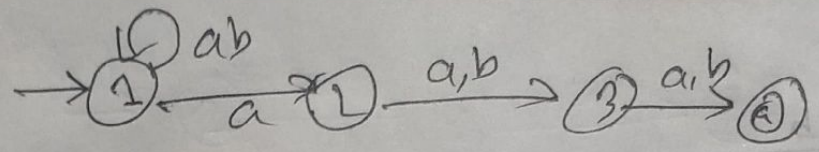
NFA

state	a	b
→ 1	{1,2}	{1}
2	φ	{3}
3	φ	{4}
* 4	φ	φ

DFA

state	a	b
→ 1	12	1
12	12	13
13	12	14
*14	12	1





RHS is 'a'

L = {aadob, abb, aaa, aba, ababa... }

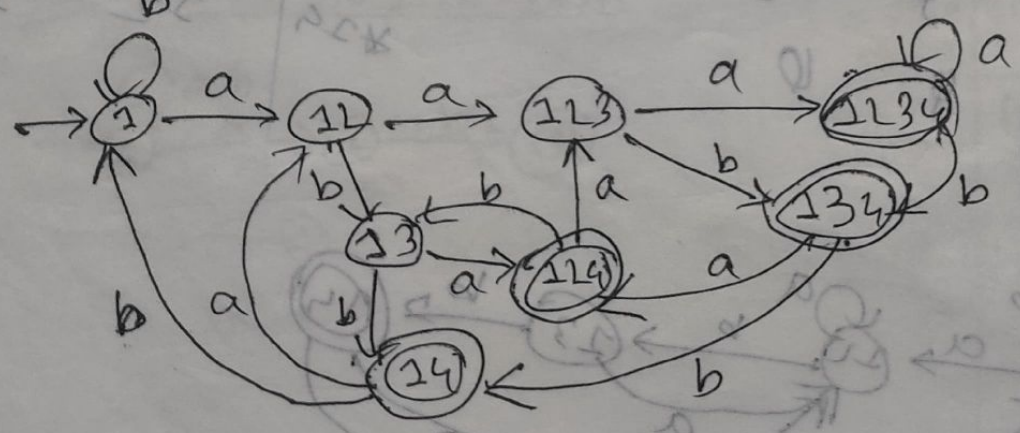
NFA

state	a	b
1	{1,2}	{1}
2	{3}	{3}
3	{4}	{4}
4	ϕ	ϕ

fi

DFA

state	a	b
→ 1	{12}	{1}
12	123	13
13	124	14
✓ 14	12	1
123	1234	134
✓ 124	123	13
✓ 134	124	14
✓ 1234	1234	134

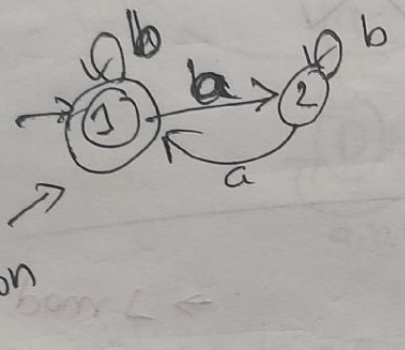


④ Epsilon NFA

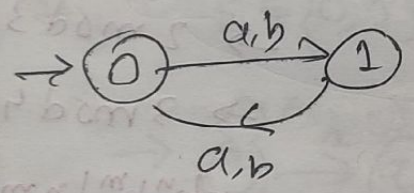
transition without input (ϵ)

① $L = \{$ string where numbe of a is <u>even</u> $\}$ $\Sigma(a,b)$

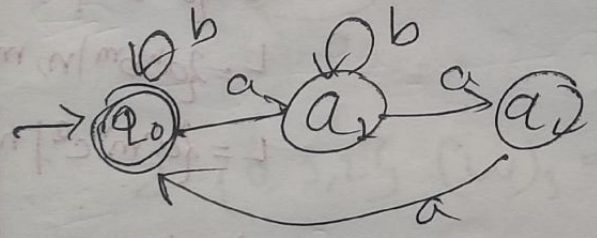
$= \{ \varepsilon, aa, aab, baa, baab, baababab, \ldots \}$



in E-NFA start state is always final state for ϵ input

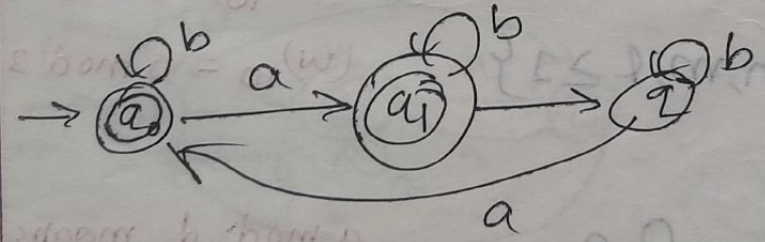
② 

number of string is <u>even</u>

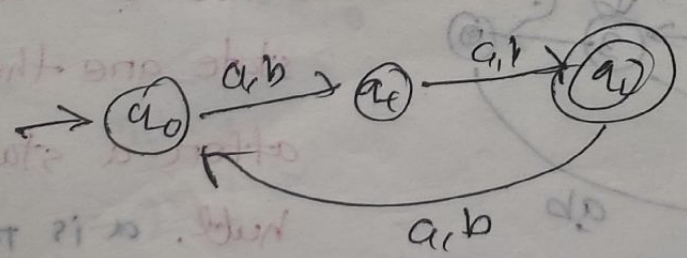
③ 

num(a) is divisible by 3

* epsilon দিলে ① state কে final করতে।

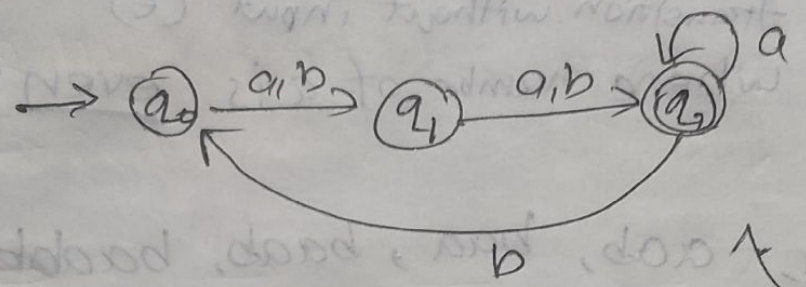


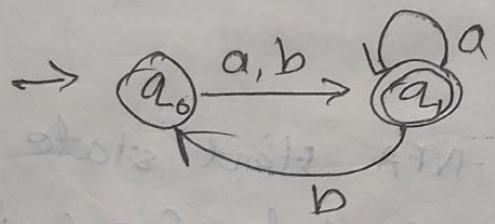
$(w) = 3 \bmod 4$



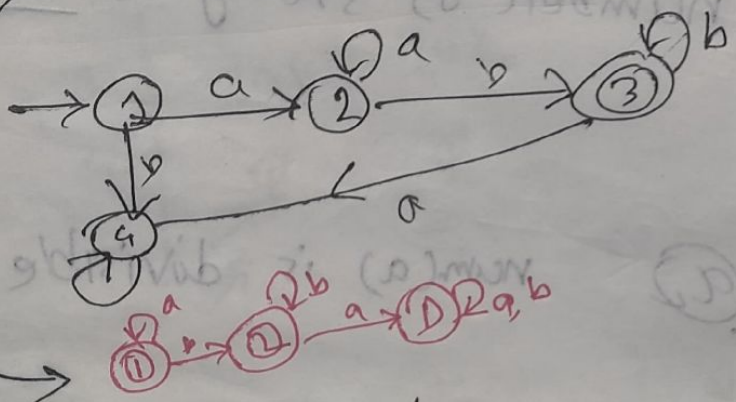
* Finite automata is memory less device

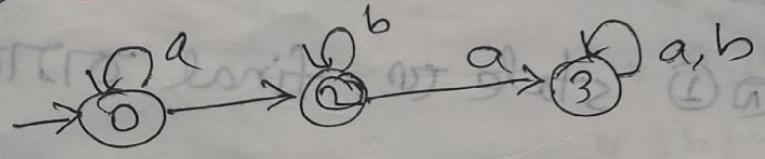
$n(a) > n(b)$ not possible





(a) $L = \{a^n b^m \mid n, m \geq 0\}$





→ 1 mod 3

→ 2 mod 3

→ 3 mod 4

$L = \{a^n b^m \mid n, m \geq 1\}$

$L = \{a^n b^m \mid n, m \geq 0\}$

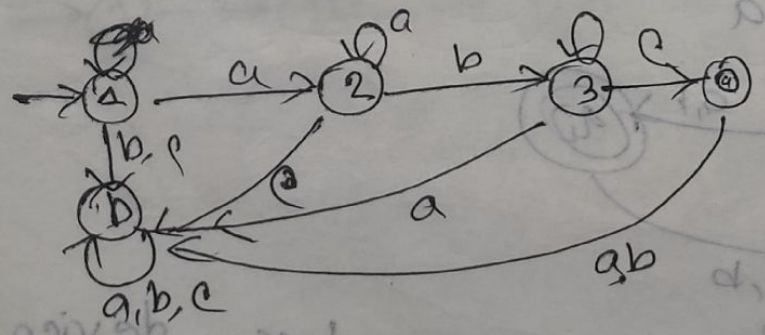
$L = \{a^n m^n c^l \mid n, m, l > 0\}$

$(w)_{10} = (w)_{\text{Docimal}} = 0 \bmod 2$

$(w)_{10} = 1 \bmod 3 \quad \Sigma\{0, 1, 2\}$

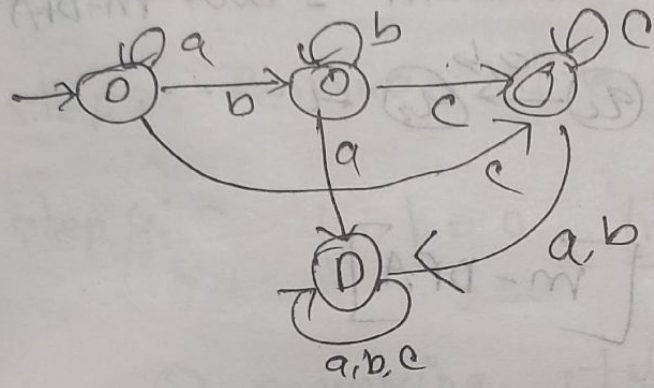
$(w)_{10} = 0 \bmod 3 \quad \Sigma\{0, 1, 2, 3\}$

(b) $L = \{a^n b^m c^l \mid n, m, l \geq 1\}$

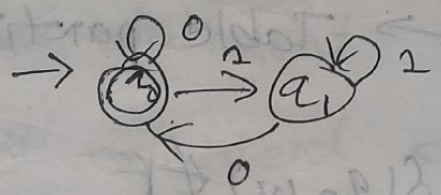


a mod d means d number state are there and after a state it is hult. a is remainder

⑥ $L = \{a^n b^m c^l \mid n, m, l \geq 0\}$

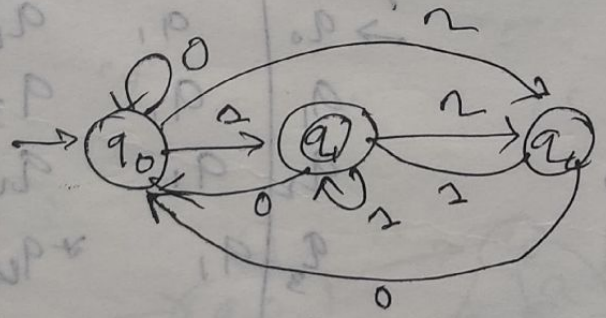


⑦ $\Sigma = \{0, 1\}$ $(w)_{10} = \text{mod } 2$



⑧ $\Sigma = \{0, 1, 2\}$ $(w)_3 = 1 \text{ mod } 3$

state	0	1	2
$\rightarrow q_0$	q_0	q_1	q_2
q_1	q_0	q_1	q_2
q_2	q_0	q_1	q_2

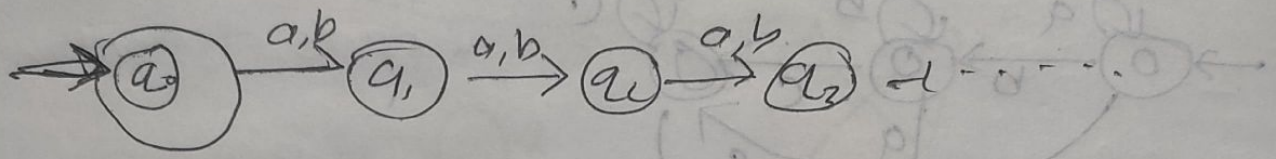


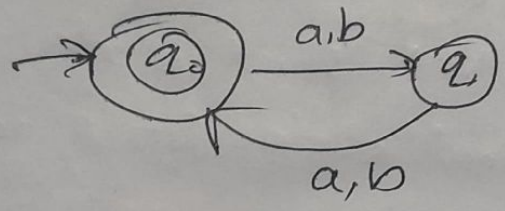
211

$= 22$

Most efficient machin (m-DFA) → minimul

→ DFA can be more than 1 but m-DFA is single.





[m-DFA]

⑧ Minimization

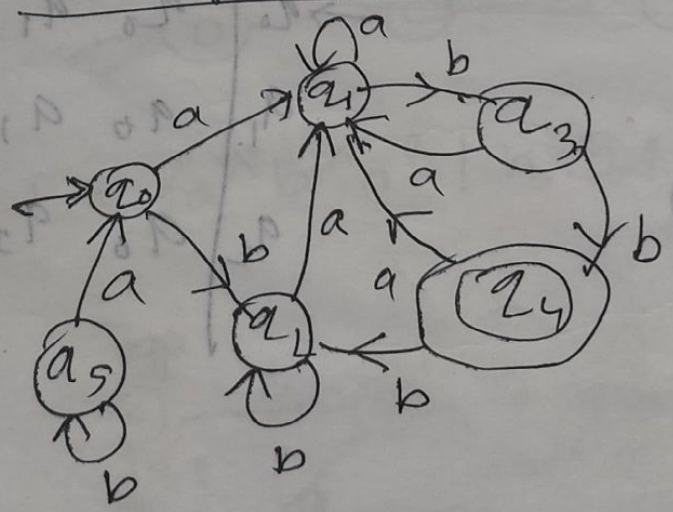
Minimization of DFA: → Table filling method

→ Table partitioning ,,

$\delta(q_1, w) \in F$

$\delta(q_2, w) \in F$ → $\delta(q_1, w) \notin F$, $\delta(q_2, w) \notin F$

Table partitioning:



state	a	b
→ q_0	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	* q_4
* q_4	q_1	q_2
q_5	q_0	q_5

Step: 1 T Table

step: 2 Mark final state.

step 3: Remove state that is not reachable from q_0

step 4: $|W| = 0$ [non occupying state]

⊙ occupying state

Two ~~set of~~ ~~|W| = 0~~ state .. $[q_0, q_1, q_2, q_3]$ $[q_4]$

zero equivalent {final}{nonfinal}

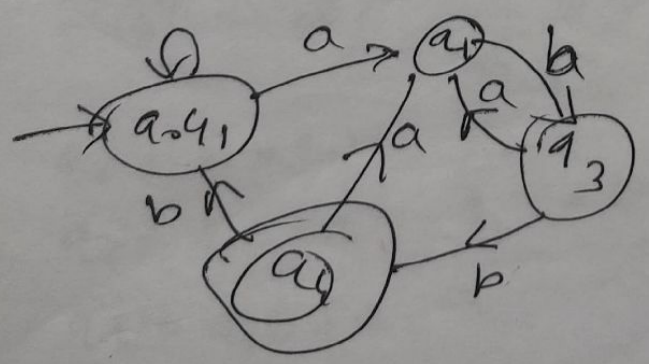
'1' equivalent state $[q_0, q_1, q_2]$ $[q_3]$ $[q_4]$

'2' " " $[q_0, q_1, q_2]$ $[q_3]$ $[q_4]$

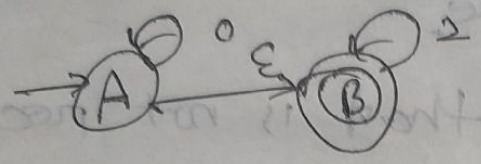
'2' " " $[q_0, q_2]$ $[q_1]$ q_3 $[q_4]$

'3' " " $[q_0, q_2]$ $[q_1]$ $[q_3]$ (q_4)

same



ε-NFA: NFA without ε transition



$L = \{0^n 1^m \mid n, m \geq 0\}$

$\varepsilon NFA(L) = (\{A, B\}, \{0, 1\}, \delta, A, \{B\})$

$\delta : Q \times \Sigma \cup \{\varepsilon\} \rightarrow 2^Q$

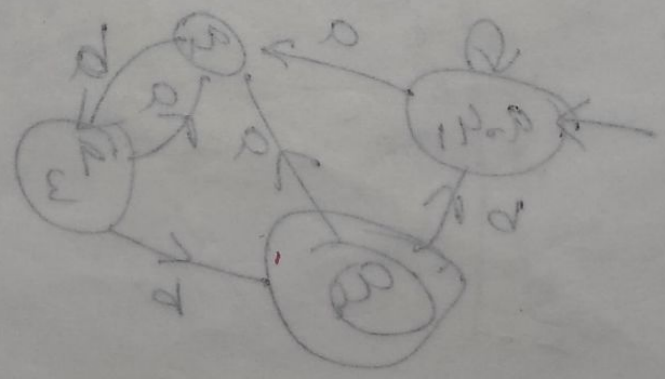
state	0	1	ε
A	{A}	∅	{B}
B	∅	{B}	∅

εNFA is a convenient construct of regular expression of a language

RE is a mathematical expression expression that present RL.

RE: 0^*1^*

E closer property:



Regular Expression :

Chapter 3. Regular Expression

→ union
↳ concatenation
↳ closure (kleen)

Alphabet (3 operator)

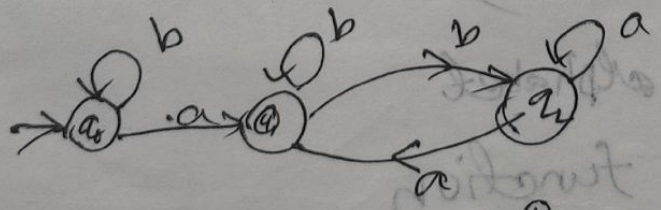
.	a.b
+/|	a+b
*	a* or (a+b)*

Conversion of FA to RE (Important)

i) state elimination method.

1. Initial state should not have incoming edge.
2. Final state should " " " outgoing edge.
3. should not more than one final state.

ii) Arden's Method : i) only NFA, DFA to RE
ii) Final state times constru



$q_0 = \varepsilon + q_0 b$ —— (i)

$q_1 = q_0 a + q_1 b + q_2 a$ —— (ii)

$q_2 = q_1 b + q_2 a$ —— (iii)

$$R = Q + RP$$
$$= QP^*$$

$$q_0 = \varepsilon b^* = b^*$$

$$q_1 = q_1 b a^*$$

$$q_1 = b^* a + q_1 b + q_1 b a^* a$$

$$= b^* a + q_1 (b + b a^* a)$$

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

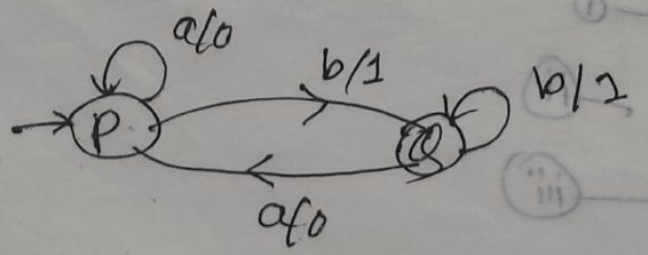
FA → with o/p → moor m/c
→ meaty m/c
→ without o/p → DFA
→ NFA
→ ε-NFA

Moor, Meaty Machin

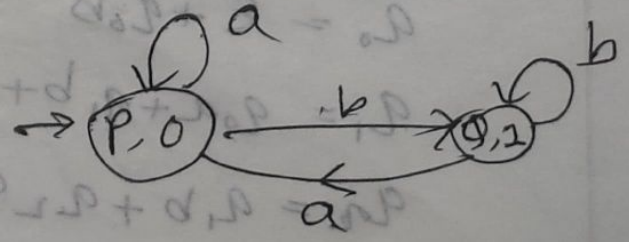
$$M = (Q, \Sigma, \delta, q_0, \lambda, \Delta)$$

Δ → output alphabet

λ → output function



Meaty Machin



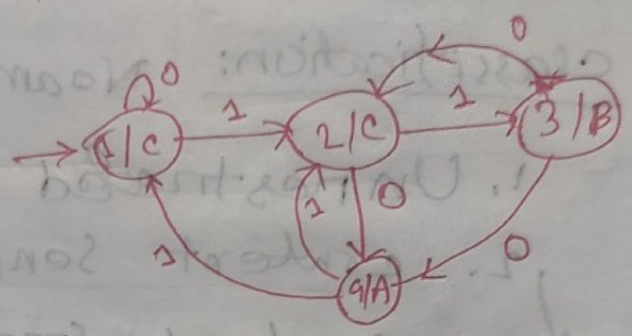
Moor Machin

D Construct a moore m/c that take input string over $\Sigma=\{0,1\}$ the count number of "ab" substring.

Ex: 10 → A, 11 → B otherwise C

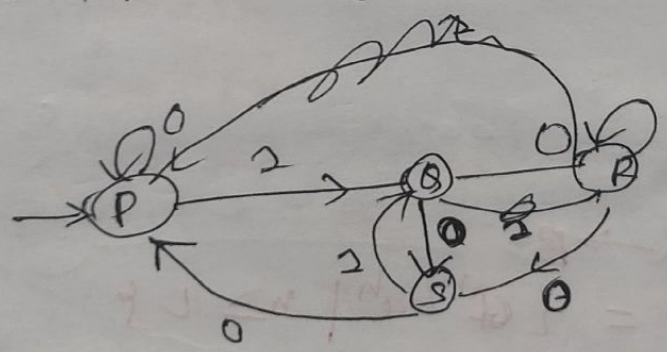
$\Sigma=\{0,1\}$

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



D $\Sigma\{0,1\}$ and ~~produce~~ ending with 10 or 11 otherwise C

A B



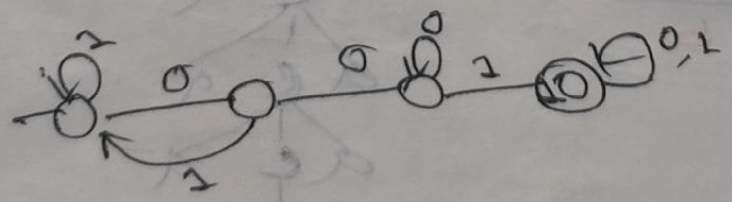
→ ~~start with ab~~

→ containing ab

→ base 3 input module-4 output

→ ending with aa or bb

→ divided by 3



Grammer: (Generative Device)

↳ Derivation → Generat String
↳ Pictorial Representation. / Derivation Tree

classification: Noam chomsk classificat.

1. Unrestricted G → Turing Machin
2. context Sensitive G
3. context-free G → Pushdown automata
4. Regular grammer G → Finite State Automata

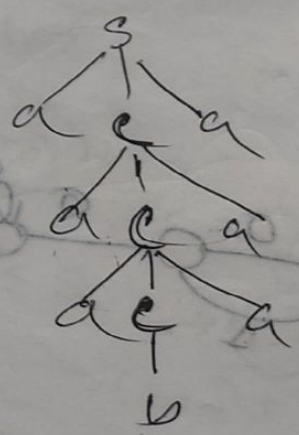
formal

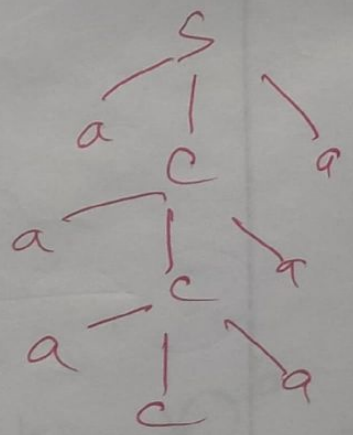
CSG:

CFG: $\alpha \to \beta$

$L(G) = \{a^n b a^n \mid n \geq 1\}$

1. $S \to aCa$
2. $C \to aCa$
3. $C \to b$





RG: $\alpha \to \beta$

$\alpha \in N$, β in the form of $a\beta$ or a, where $a \in T$, $B \in N$.

RG
↳ LLG
↳ RLG

G ↳ UG
↳ RG
↳ CSG
↳ CFG
↳ RG
↳ LLG
↳ RLG

RLG vs LLG

Computation: 21.01.2025

Regular Expression: Way of Regular Language mathematic

premitive expression:

$\emptyset$: empty set $\{\}$

ϵ : empty string $\{\epsilon\}$

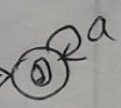
Σ : set of state

this like power → $a^* \rightarrow \{\epsilon, a, aa, aaa, aaaa \ldots\}$

$a^+ \rightarrow \{a, aa, aaa, aaaa \ldots\}$ $\quad a^+ = a.a^*$

$(a+b)^* \rightarrow \{\epsilon, a, b, aa, ab, ba, bb, aaa \ldots\}$

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

a^* → 

RE FA

Ex: $\{0, 1, 2\} \rightarrow R = 0+1+2$

Ex: $\{a, b\}$ for length 2

$L = \{aa, ab, ba, bb\}$

$RE = aa+ab+ba+bb$

$= a(a+b)+b(a+b)$

$= \underline{(a+b)}\underline{(a+b)}$ → 2 length by concat

Ex: $\{a, b\}$ length at least 2

$RE = \underline{(a+b)} \cdot \underline{(a+b)} \cdot (a+b)^*$

Ex: $\{a, b\}$ at most length 2

$L = \{\epsilon, a, b, aa, ab, ba, bb\}$

$RE = \epsilon + a + b + aa + ab + ba + bb$

$= \underline{(a+b+\epsilon)} \cdot \underline{(a+b+\epsilon)}$

Ex: even length $((a+b)(a+b))^*$

Ex: $\{a,b\}$ ~~even~~ odd length

$$RE = ((a+b)\cdot(a+b))^*\cdot(a+b)$$

Ex: 2 mod 3

$$RE = ((a+b)\cdot(a+b)(a+b)^*\cdot(a+b)(a+b)$$

Ex No of "a" exactly 2

$$RE = b^*a\ b^*a\ b^*$$

Ex: No of a is even

$$RE = (b^*ab^*ab^*)^* + b^*$$ for a=0 number

Ex: start with a

$$RE = a(a+b)^*$$

$\cdot \rightarrow$ and
$+ \rightarrow$ or

Ex: start and end with different symbol

$$RE = a(a+b)^*b + b(a+b)^*a$$

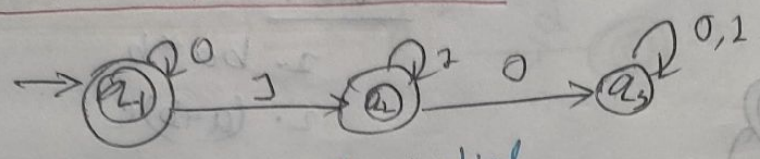
Ex: Should not contain "aa"

$$RE = (b+ab)^* + (b+ab)^*a$$
$$= (b+ab)^*(\epsilon+a)$$

$RE = a(a+b)^*$ → start with a
$= (a+b)^*b$ → end with b
$= a(a+b)^*b$ → start with a end with b
$= a(a+b)^* + b(a+b)^*$

→ কোনো ∈ থাকবে না।
→ only NFA, DFA transimission হবে।
→ একাধিক final state থাকলে প্রতিটির জন্য RE হবে।

Arden's Theorem:



Sol: $q_1 = \epsilon + q_1 0$ → for initial

$q_2 = q_1 \cdot 1 + q_2 \cdot 1$

$q_3 = q_2 \cdot 0 + q_3 \cdot 0 + q_3 \cdot 1$

must 1 final

$q_1 = \epsilon + q_1 \cdot 0$
$= \epsilon 0^*$
$= 0^*$

$q_2 = q_1 \cdot 1 + q_2 \cdot 1$
$= 0^* 1 + q_2 \cdot 1$
$= 0^* 1 1^*$

$RE = 0^* + 0^* 1 1^*$
$= 0^* (\epsilon + 1 1^*)$
$= 0^* 1^*$

multiple final state থাকলে এভাবে ∪ বসাবে

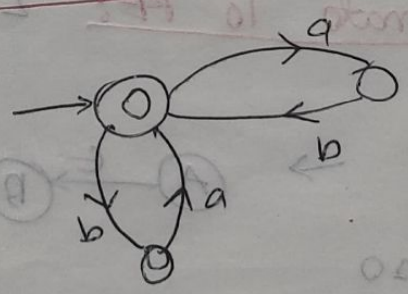
Regular Expression to FA:

(NFA is easy)

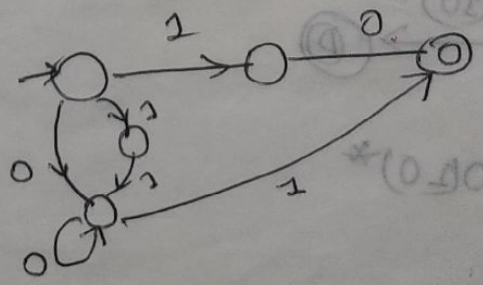
$a+b = a|b$

always initial is final for ∈

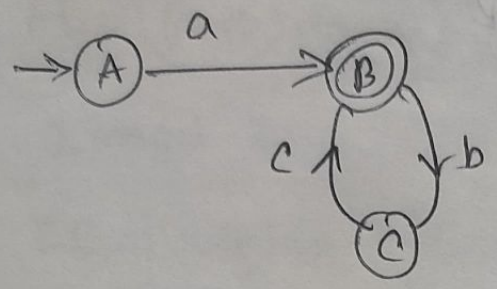
Ex: $(ab+ba)^*$



Ex $10+(0+11)0^*1$



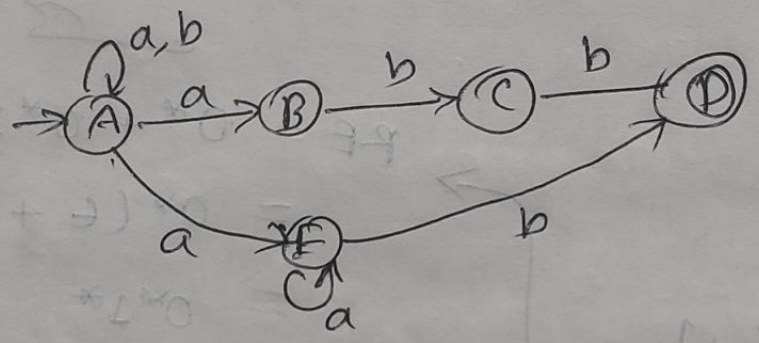
Ex: $a(bc)^*$



More:

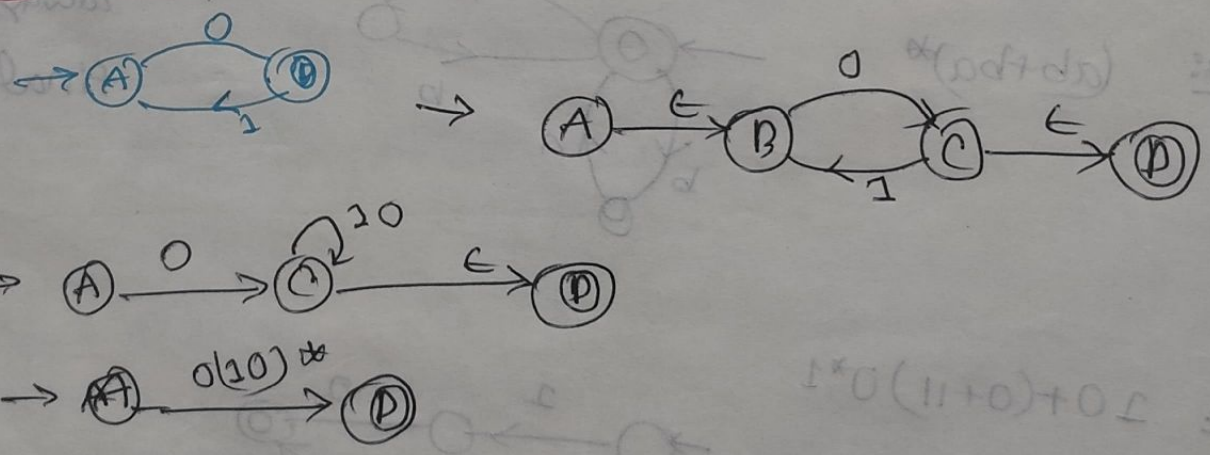
1. ba^*b
2. $(a+b)c$
3. $a(bc)^*$
4. $(a+b)^*$
5. a or b never comes together

Ex: $(a|b)^*(abb|a+b)$



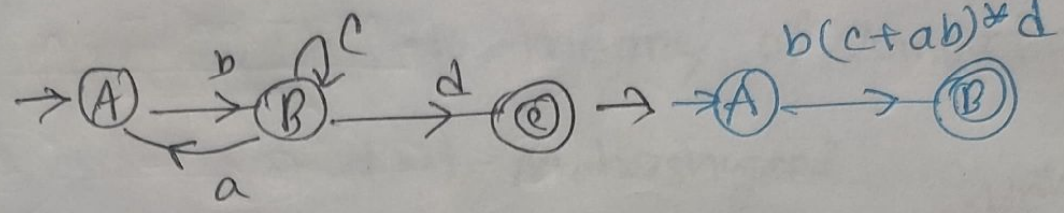
Finite Automata To RE:

2 process → 3 rules
→ Arden's theory

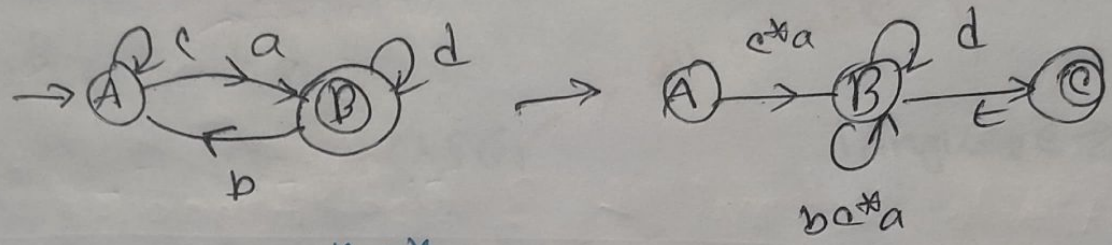


RE = $0(10)^*$

Ex:

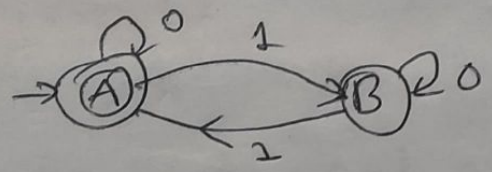


Ex:



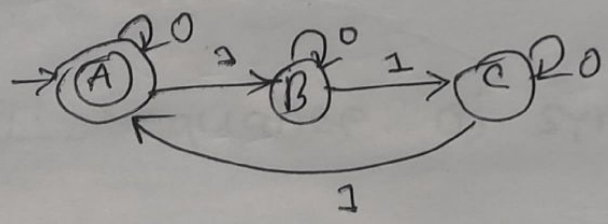
$\rightarrow$ RE = $c^*a(d+bc^*a)^*$

Ex:



$R = (10^*1+0)^*$

Ex:

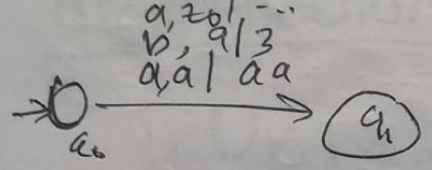


$(RE = (10^*10^*1+0)^*$

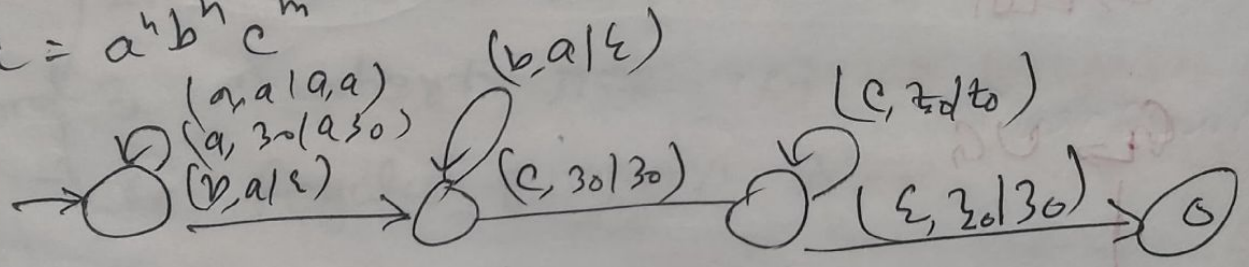
Pushdown Automata.

PDA can successfully halt with/without final sta

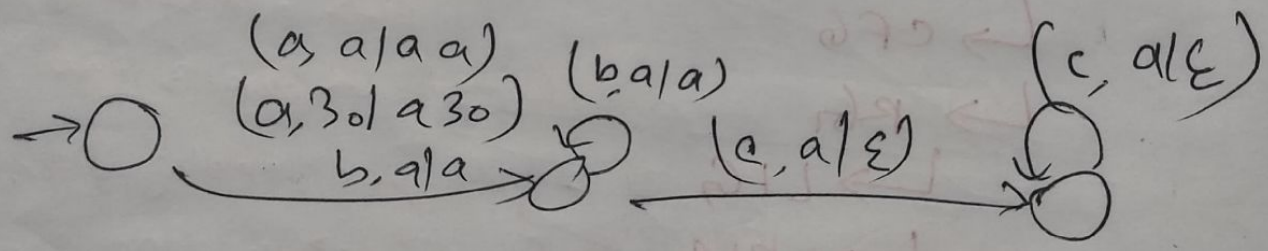
$L = w \in \{a,b\}^* \mid$ such that $n_a(w) = n_b(w)$



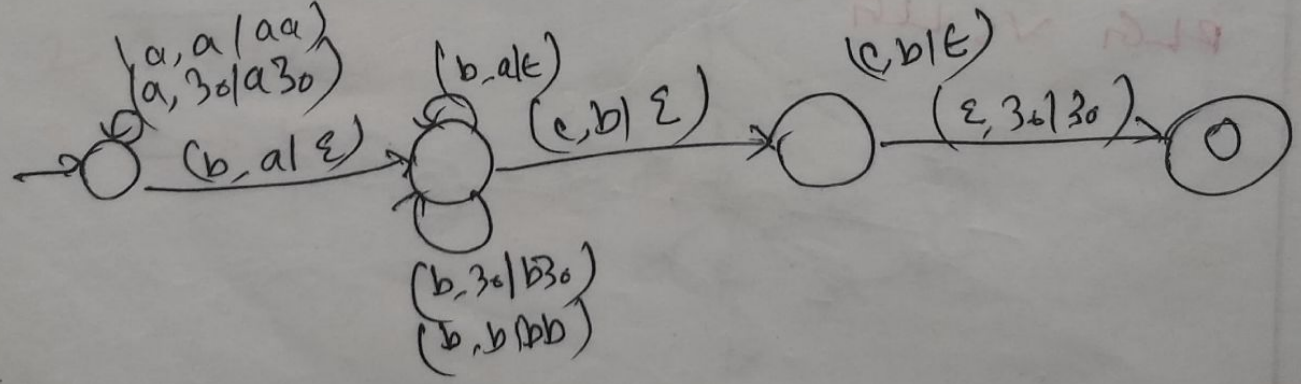
$L = a^n b^n c^m$



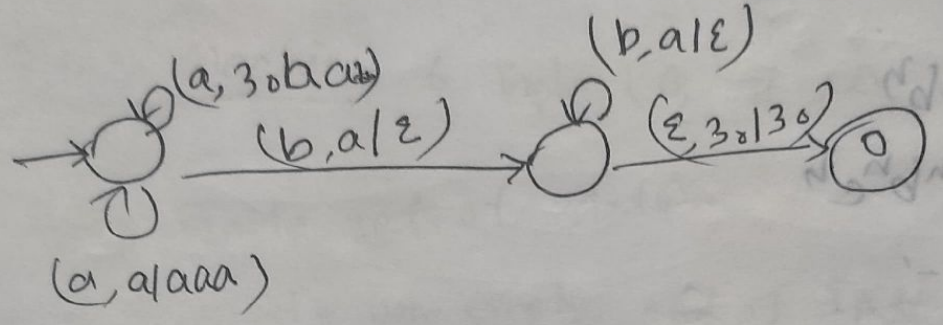
$L = a^n b^m c^n$



$L = a^m b^{m+n} c^n$



$L = \{ a^n b^{2n} \mid n \geq 1 \}$

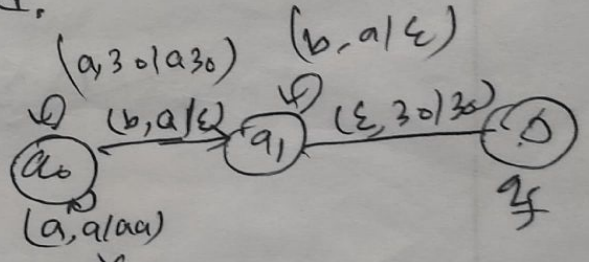


containing ab

Instantaniace Description (ID)

ID is a formal notation of PDA that copute string and make a decision whether the string is accepted or rejected.

→ Triple (q, w, a)



$\delta : Q \times (\Sigma \cup \varepsilon) \times \Gamma \rightarrow Q \times \Gamma^*$

Pictorial & mathmatical representation

$(q_0, a, z_0) = (q, a z_0)$

$(q_0, a, a) = (q, aa)$

$(q_0, b, a) = (q_1, a z_0)$

$(q_1, b, a) = (q_1, z_0)$

$(q_1, \varepsilon, z_0) = (q_f, z_0)$

(accepted)

Turing Machine

$L_1 = a^n b^n$

$L_2 = a^n b^n c^n$

$L_3 = a^{-1}$

$n_a(w) = n_b(b)$

20 (monday) CT → Moore, Milli → Pushdown

Finite Automata

with output
↳ moore
↳ mealy

without output
↳ DFA, NFA, E-NFA

DFA

Finite Automata With Output: Mealy, Moore

Mealy Machin: 6 Tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

$Q \rightarrow$ Finite set of state

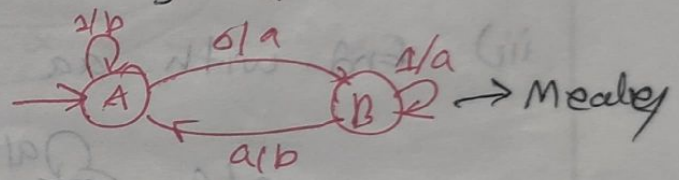
Sum — Σ = Finite non-empty set of Input Alphabet

Del — Δ = set of output alphabet

Delta — δ = Transition function $Q \times \Sigma = Q$

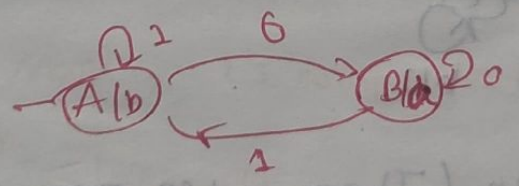
Lamda — λ = output function: $\Sigma \times Q \rightarrow \Delta$

q_0 = Initial start

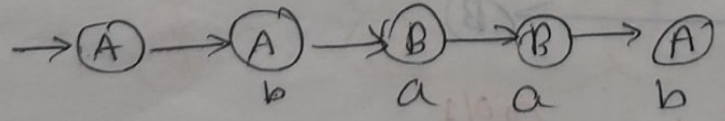


Moore Machin:

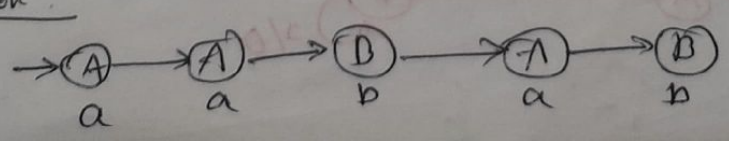
λ = output function $Q \rightarrow \Delta$



Mealy Example: 10p



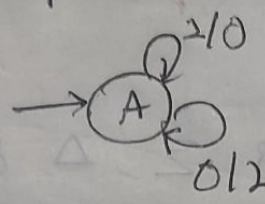
Moore Machin:



(n+1) length

Mealy

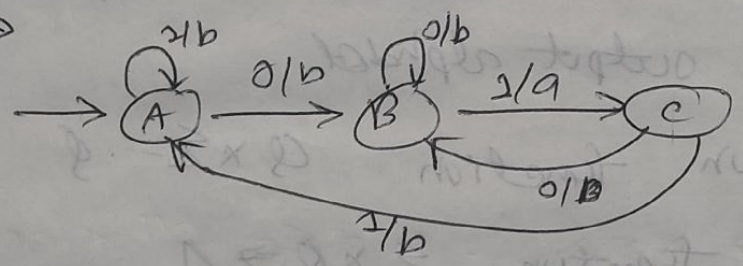
i) 1's complement



ii) print a when get 01

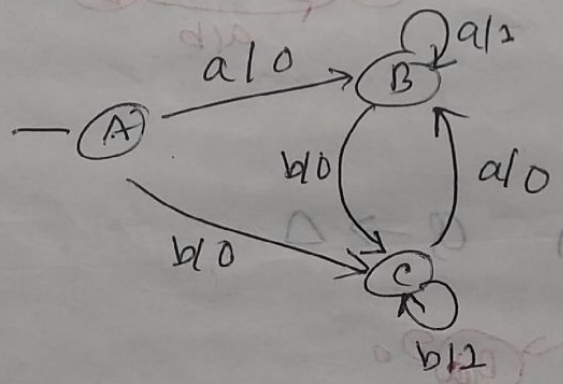
$\Sigma = \{0, 1\}$ $\Delta = \{a, b\}$

first DFA >>

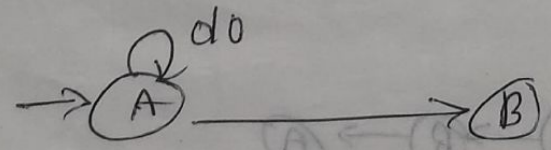


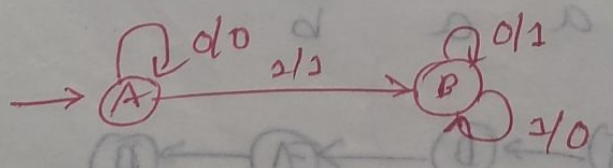
0110
baab

iii) End with aa or bb



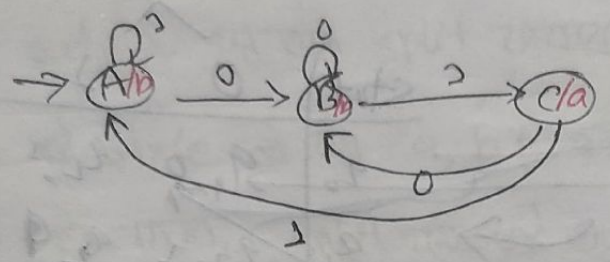
iv) 2s complemen (Theory → LSB থেকে শেষ পর্যন্ত (১ম) হলে থেকে তার বিপরীত)





Moore Machine: print a for 01 $\Sigma = \{0,1\}$ $\Delta = \{a, b\}$

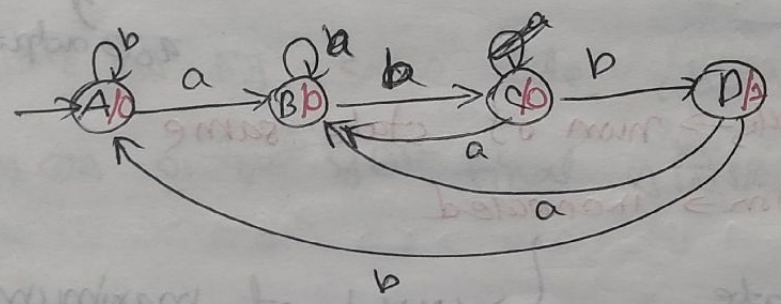
(i)



0110

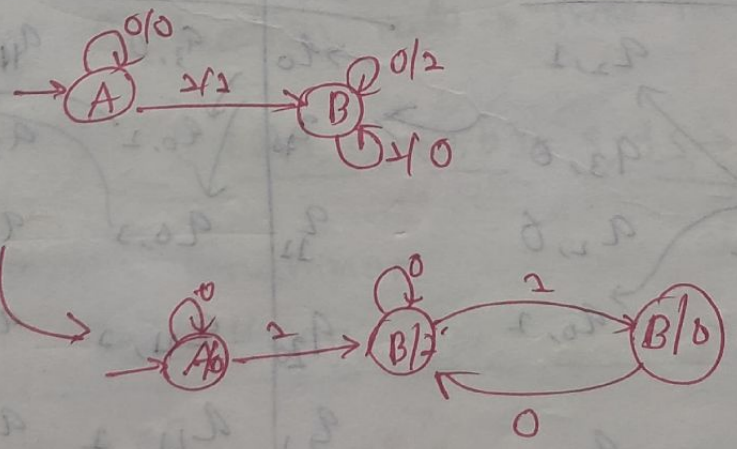
A B C A B
b b a b b

(ii) print a when abb



যদি Moore থেকে Mealy or Mealy থেকে moore করতে চাই তবে শুধু output A/0 → 1/0 বা 1/0 → A/0

Mealy to Moore → 2's complement



→ Depend on transition not state

Moore to Mealy → Table:

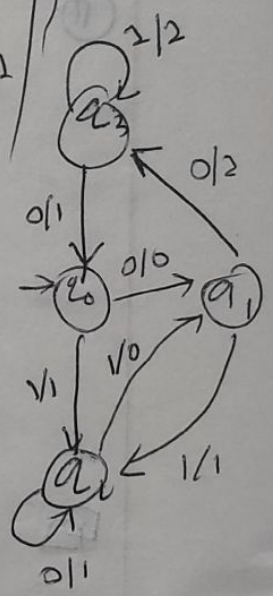
State	0	1	output
→ q_0	q_1	q_2	1
q_1	q_3	q_2	0
q_2	q_2	q_1	1
q_3	q_0	q_3	1

input

State	0	1
q_0	$q_1, 0$	$q_2, 1$
q_1	$q_3, 1$	q_2
q_2	$q_2, 1$	$q_1, 0$
q_3	$q_0, 1$	$q_3, 1$

input

q_0 → output



Moore → Mealy → num of state same

Mealy → Moore → increased

x → state

y → output

→ $(x \times y)$ at maximum.

Mealy to moore → Transition Table

State	a	b
→ q_0	$q_3, 0$	$q_2, 1$
q_1	$q_0, 1$	$q_3, 0$
q_2	$q_2, 1$	$q_2, 0$
q_3	$q_1, 0$	$q_0, 1$

State	a	b	Final
→ q_0	$q_3, 0$	$q_{10}, 1$	1
q_{10}	$q_0, 1$	$q_3, 0$	0
q_{11}	$q_0, 1$	$q_3, 0$	1
q_{20}	$q_{11}, 1$	$q_{10}, 0$	0
q_{21}	$q_{11}, 1$	$q_{10}, 0$	1
q_3	$q_{10}, 0$	$q_0, 1$	0

q_1: $q_1, 0$ (0) ; $q_1, 1$ (1)

q_2: $q_2, 0$ (0) ; $q_2, 1$ (1)

i) state এ থাকলে, ঐ state এর জন্য a, b তে দেখব যদি একই output হয় তবে split করবো।

ii) ঐ state এর জন্য a, b তে গেলে, যদি single হয় তবে output এর জন্য, split হলে ঐ output এর জন্য যে split branch আছে তার নিজস্ব output হবে।

iii) split গুলোর শেষে যতটা split moore এ আছে same হবে।

iv) state এর জন্য same state ab এর কোনো একে যেসব নিজের output গুলো final এ নিবো।

Grammer:

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

V → set of variable, non-terminal symbol → not a component of string

T → set of terminal symbol

S → start symbol

P → Production rule → Those rule replace V to T.

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

→ Regular Grammer

Right linear		Left linear	
$A \to xB$ $A \to x$	$A \to xB \mid x$	$A \to Bx$ $A \to x$	$A \to Bx \mid x$
where $A, B \in V$, and $x \in T$		$A, B \in V$ and $x \in T$	

Ex:

$S \to abS \mid b$

Ex:

$S \to Sbb \mid b$

Derivation from a Grammar: (Derive language)

$G1 = (\{S, A\}, \{a, b\}, S, \{S \to aAb, aA \to aaAb, A \to \epsilon\})$

$S \to \underline{aA}b$

$\to a\underline{aA}bb$

$\to aaa\underline{A}bbb$

$\to aaabbb$

$G2 = (\{S, A, B\}, \{a, b\}, S, \{S \to AB, A \to a, B \to b\})$

$S \to \underline{A}B$

$\to a\underline{B}$

$\to ab$

$B \to bB \mid b$

also

$L(G2) = \{ab\}$

$G3 = (\{S, A, B\}, \{a, b\}, S, \{S \to AB, A \to aA \mid a, B \to bB \mid b\})$

$S \to \underline{A}\underline{B}$

$\to a\underline{A}b\underline{B}$

$\to aabb$

$S \to AB$

$\to ab$

$S \to AB$

$\to a\underline{A}b$

$\to aab$

$S \to AB$

$\to abB$

$\to abb$

$L(G3) = \{a^2b^2, ab, a^2b, ab^2, \ldots\}$

$= \{a^m b^n \mid m \geq 0 \text{ and } n \geq 0\}$ ← General lang from grammer

→ For all

CFG → left hand side of all rules non terminal

CSG → one or more rules are terminal $aA \to ab$

$S \to aSb$
$A \to aBb$
$B \to b$

Context Free Language: Generated from CFG

Defined by 4 tuples $G = \{V, \Sigma, S, P\}$

$V \to$ set of variable or non terminal symbol.

$\Sigma \to$ Terminal symbol

$S \to$ Start symbol

$P \to$ Production rule $\to$ $A \to a$

$a = \{V \cup \Sigma\}^*$ and $A \in V$

Ex: Generate a language of equal number a, b that $a^n b^n$

$G = \{(S, A), (a, b), (S \to aAb, A \to aAb \mid \epsilon)\}$

↳ Epsilon
Empty symbol

$S \to aAb$
$\to aaAbb$
$\to aaaAbbb$
$\to aaabbb \Rightarrow a^3 b^3 \to a^n b^n$

Find if a string belongs to a Grammar or not?

$G = S \to 0B \mid 1A,\ A \to 0 \mid 0S \mid 1AA \mid \wedge \to \varepsilon,\ B \to 1 \mid 1S \mid 0BB$

00110101

$S \to 0B$
$\to 00BB$
$\to 001B$
$\to 0011S$
$\to 0011 0B$
$\to 00110 1S$
$\to 00110 10B$
$\to 00110101$

* Context → non-terminal এর আগে পরে terminal বা Symbols থাকে মাত্র replace করা যাবে। $aA \to c$ → context a
$A \to c$ → No context

$S \to aAb$ $A \to aAb \mid 1$ → string → aabbb

$S \to aAb$
$\to aaAbb$
$\to aaaAbbb$
$\to aaabbb$

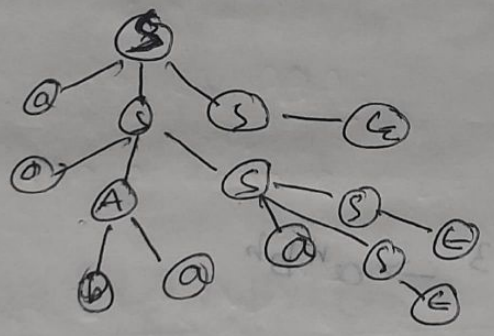
(Not Belong)

Derivation Tree/ Parse tree

$G = \{V, T, P, S\}$ $S \to 0B$, $A \to 1AA \mid \epsilon$, $B \to 0AA$

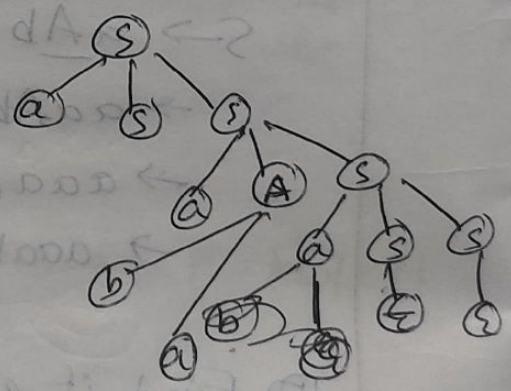
LEFT Derivation $S \to aAS \mid ass \mid \epsilon$, $A \to SbA \mid ba$ Right Derivation

→ Always select left variable



aabaa

→ Always select right variable



aabaa

FA + Infinite Stack

replace
push
pop
No change
Automatic halt

PushDown Automata: Finite state machin + stack (infinite memory)

Have 3 component-
↳ Input string, Finite control Unite, infinite size stack.

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

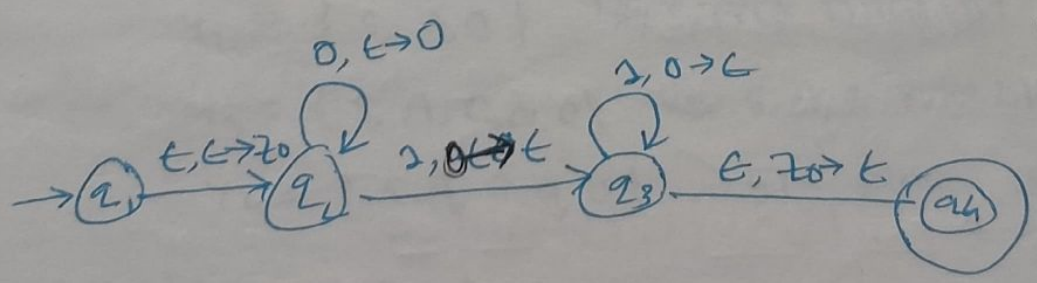
$Q \rightarrow$ Finite set of state
$\Sigma \rightarrow$ Finite set of input symbol
$\Gamma \rightarrow$ Finate Stack Alphabet
$\delta \rightarrow$ Transition function $\rightarrow \delta(q, a, X)$
$q_0 \rightarrow$ Start state
$z_0 \rightarrow$ start stack symbol.
$F \Rightarrow$ Set of final states

q → state of Q
a → Input symbol Σ or ϵ
X → Stack symbol

output → finite pair (p, γ) set
$p \rightarrow$ new state
$\gamma \rightarrow$ stack symbol that replaces X at the top of the stack

$\gamma = \epsilon$ stack is popped
$\gamma = X$ unchanged
$\gamma = YZ$ X is replaced by Z and Y is pushed onto stack

$A \xrightarrow{a, b \rightarrow c} B$ a → Input b → symbol top of stack that is popped → ϵ mean read, popped nothing happened



PDA accept "001"

CFL → CFG (language to grammer)

i) $a^n b^n, n \geq 1$ $\quad S \to aSb \mid ab$

ii) $a^n b^{n+2}, n \geq 0$ $\quad S \to aSb \mid bb$

iii) $a^{2n} b^n, n \geq 0$ $\quad S \to aaSb \mid \lambda$

$n \geq 1$ $\quad S \to aaSb \mid aab$

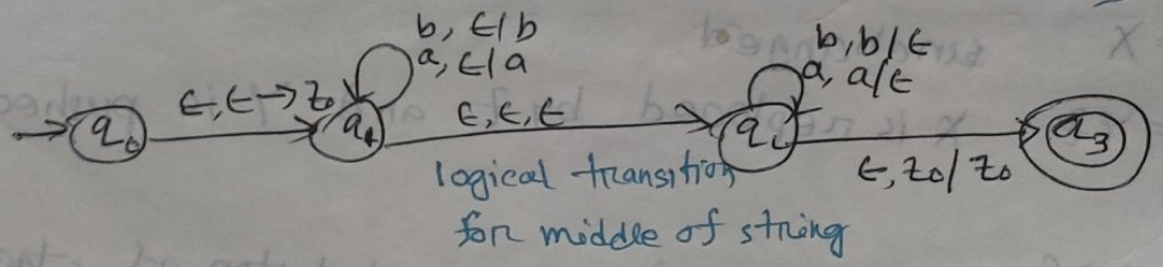
iv) $a^{2n+3} b^n, n \geq 0$ $\quad S \to aaSb \mid aaa$

v) $a^m b^n, m \geq 1, n \geq 0$ $\quad S \to aSb \mid \lambda$

$A \to aA \mid \lambda$

Example of PDA:

1. Even palindrom $\quad L = \{ww^R \mid w = (a+b)^+\}$



Simplication of CFG

↳ Reduction of CFG — (I)

↳ Removal of Unite production — (II)

↳ Removal of Null Production — (III)

(I)

Phase 1: Given $S \to AC/B$, $A \to a$, $C \to c|Bc$, $E \to aA|e$

St 1:

↳ $V = \{A, B, C, E, S\}$ Non Terminal $T = \{a, c, e\}$ Terminal

$W_1 = \{A, C, E\}$ → who are derive terminal

$W_2 = \{A, C, E, S\}$ → who are derive W_2

$W_3 = \{A, C, E, S\}$ If two are same then stop

New → $G' = \{\overline{(A, C, E, S)}, \overline{(a, c, e)}, P, S\}$

P:
$S \to AC$
$A \to a$
$C \to c$
$E \to aA|e$

যারা new V, T তে নেই তাদের সকল production rule G থেকে সরাই।

Phase 2:

$Y_1 = \{S\}$ শুরু থেকে কাদের derive করা যাবে তাদের নিয়ে

$Y_2 = \{S, A, C\}$ ↳ S থেকে যাদের পাই

$Y_3 = \{S, A, C, a, c\}$ ↳ S, A, C থেকে যাদের পাই

$Y_4 = \{S, A, C, a, c\}$ > যদি same তাই terminate

$G'' = \{(S, A, C), (a, c), P, S\}$

$P = S \to AC$, $A \to a$, $C \to c$

(ii) Any production rule of the form $A \to B$ where $A, B \in$ Non terminal, is called Unite production

$A \to B, \quad B \to x$

$\therefore A \to x$

(iii) A non terminal symbol 'A' is a null variable if there is a production $A \to \epsilon$ or there is a derivation that start at A and leads to ϵ.

$A \to B, \; B \to C, \; C \to \epsilon$

$\therefore A \to \epsilon$

Ex. $S \to aSa \mid bSb \mid \epsilon$

$S \to asb \mid bsb \mid aa \mid bb$ → ϵ দিয়ে replace করলে যা আসবে সেটা আবার add করব

III CFG to Chomsky Normal Form (CNF)

↳ No ϵ symbol

↳ In right side max 2 Non terminal or 1 terminal

Ex. $S \to aSa \mid bSb \mid a \mid b \mid aa \mid bb$

$A \to a \quad B \to b \to S \to ASA + BSB \mid a \mid b \mid AA \mid BB$

$R_1 \to SA, \; R_2 \Rightarrow SB \to S \to AR_1 + BR_2 \mid a \mid b \mid AA \mid BB$

Enumerable – গণনাযোগ্য
Halt → থামা

CFG to Greibach Normal Form:

↳ $A \to a\alpha$ $\quad$ $a \to$ ~~non~~ terminal
$\alpha \to$ non terminal

Turing Machin: That accept recursively enumerable

↳ math model - have infinite length tape divided into cell
↳ a head than scan/read and the replace on move ⇄

$$A \xrightarrow{a \to b, R} B$$

$a \to$ read
$b \to$ write
$R \to$ Right / Left

7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, b, F)$

$Q \to$ Non empty set of state
$\Sigma \to$ Non empty set of symbol
$\Gamma \to$ Tape symbol
$\delta \to$ Transition function. $Q \times \Sigma \to \Gamma \times (R/L) \times Q$

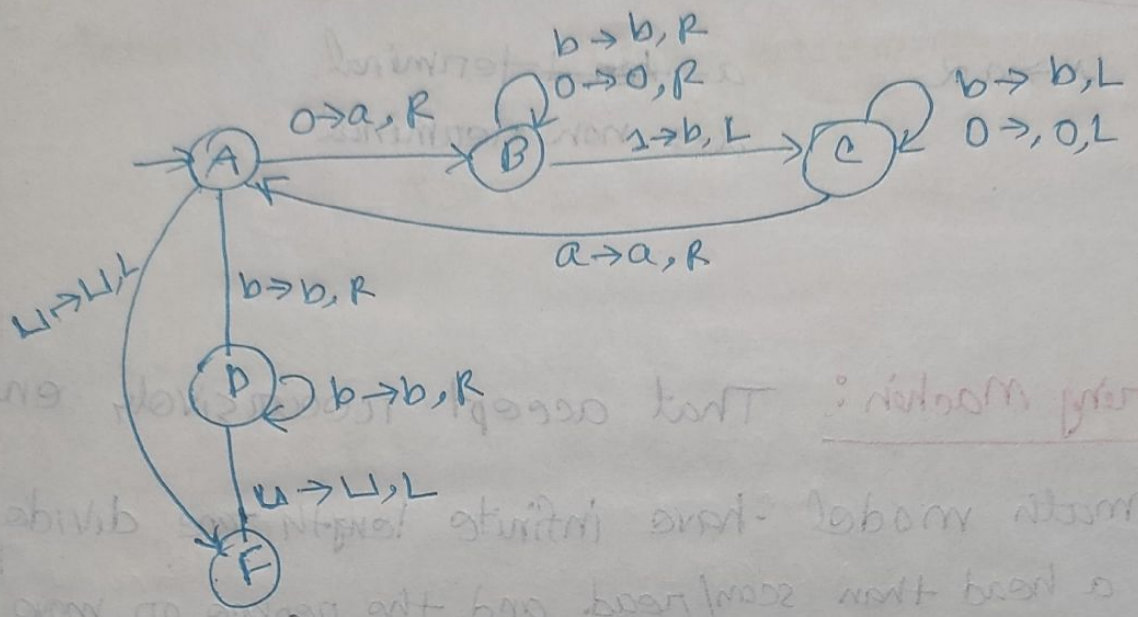
write, move R/L, Go new state

$\delta(q_2, a) \to (q_1, Y, R)$ → move

From, input, Go to, write

$q_0 \to$ Initial State
$b \to$ Blank symbol $\sqcup$
$F \to$ Final state

Ex: Turing machin for $L = 0^n 1^n$



Turing Thesis: any computation that can be carried out by mechanical means can be solve by turing machin.

Recursively Enumerable language: A language L and Σ is said to be recursively enumable if there exist a turing machin that accept it.