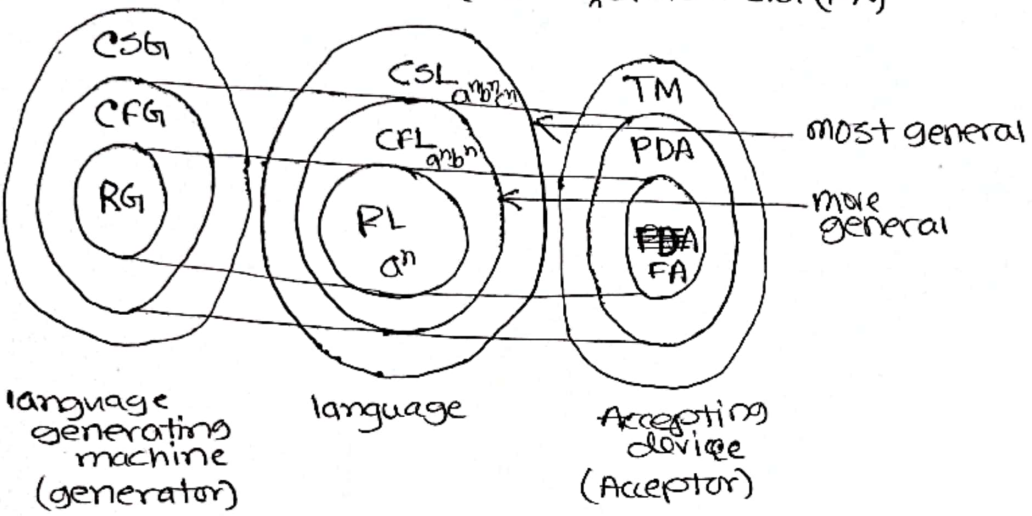
Date: 29.10.2024

Finite automata:

language → machine/device → grammer.

* All solvable problems are identified as context sensitive language → Turing machine দ্বারা use.

- context sensitive language (CSL) → Turing machine (TM) যে machine দ্বারা
- context free language (CFL) → pushdown automata (PDA)
- Regular language (RL) → finite automata (FA)



- a^n → RL
- $(abc)^n$ → RL
- $a^n (bc)^n$ → CFL
- $(ab)^n c^n$ → CFL
- $a^n b^n c^n$ → CSL.

- **RL:** $L = \{\varepsilon, 0, 00, 000, \ldots\}$ (ε ← empty)

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

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

n → length of zero.

$0^1 = 0$, $0^2 = 00$, $0^0 = \text{empty}(\varepsilon)$

regular language → memory থাকে না। (compare হচ্ছে না এইজন্য) ← সব language accepted হচ্ছে না।

- **Finite automata:** (building block of finite automata is symbol)
  - **Symbols:** building block : 0, 1 ..., a ... z, A ... Z.
  - **Alphabet:** finite set of symbols.

    Binary alphabet. $\Sigma = \{0, 1\}$

    English alphabet. $\Sigma = \{a \ldots z, A \ldots Z\}$

    Decimal alphabet. $\Sigma = \{0, \ldots 9\}$.
  - **String:** Sequence of symbols over an alphabet:

    0, 00, 01, 10, 010, 1000001 ...
  - **language:** set of strings / collection of strings over an alphabet finitely or infinitely.

    $L_1 = \{$set of all strings of length '2' over $\Sigma = \{0,1\}\}$

    $= \{00, 01, 10, 11\}$ → Finite language.

L_2 = {set of all strings starting with '0' over $\Sigma = \{0,1\}$}

= {0, 00, 01, 011, ...} ← Infinite language ⇒ finite representation:

- FSM: Finite state machine
- FSA: Finite state automata.

$\Sigma = \{a \cdots z, A \cdots Z\}$ a/A → symbol and word

a^n

$\Sigma = \{a \cdots z, A \cdots Z, \_\}$

$\Sigma = \{a \cdots z, A \cdots Z, \_, \square\}$ ← space

$a^n b^n$

$\Sigma = \{a \cdots z, A \cdots Z, \_, \square, \backslash n\}$ $a^n b^n c^n$

- $L = \{a, b\}$

$L = L^0 \cup L^1 \cup L^2 \cdots = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cdots$

$= \{\varepsilon\} \cup \{a, b\} \cup \{aa, bb, ab, ba\} \cup \cdots$

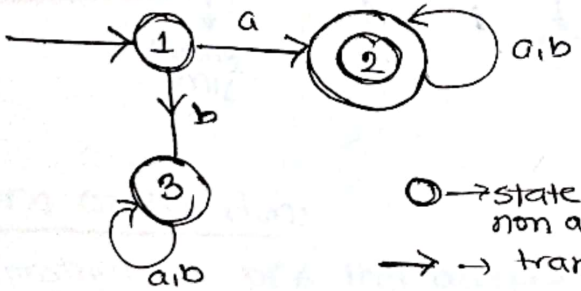
$L = \{\varepsilon, a, b, aa, ab, ba, bb, \cdots\}$

* More specific: (Starting with a)

* $L_1 = \{a, aa, ab, aaa, aba, abb \cdots\}$

⇓ finite representation (finite automata) → 3 ভাবে বলা যায়:

① Pictorial form (Transition diagram)



○ → state / non accepting state

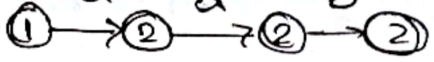
→ → transition

→○ → starting state

◎ → final/accepting state.

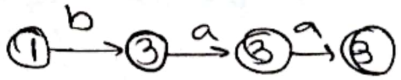
for aab

Transition diagram →



- fully consumed.
- whole length পড়া হবার পর final stage/state এ থাকে → ঐ language এ → accepted.

for baa



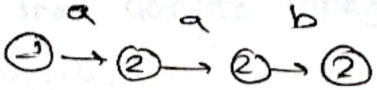
- finite state এ যায়নি fully consumed হবার পর → rejected → ঐ language এর কেউ না।

② Tabular form:

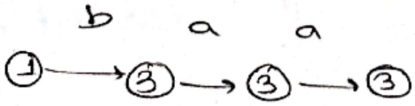
State	a	b
start → 1	2	3
final * 2	2	2
3	3	3

← Transition table:

Transition diagram:



(accepted in that language)



(rejected in that language)

(iii) Mathematical / Math model form:

Finite automata (FA) is a 5-tupple.

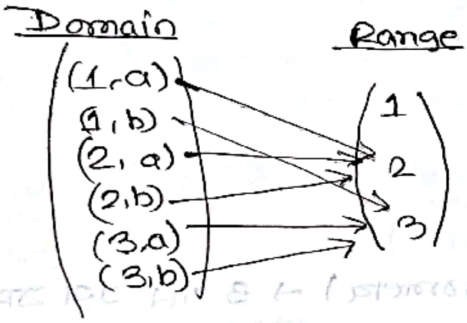
machine: $M = (Q, \Sigma, \delta, q_0, F)$

$Q \rightarrow$ finite set of states
$\Sigma \rightarrow$ finite set of symbols. (alphabet)
$\delta \rightarrow$ is a mapping function, (delta)
$q_0 \rightarrow$ starting state
$F \rightarrow$ final state.

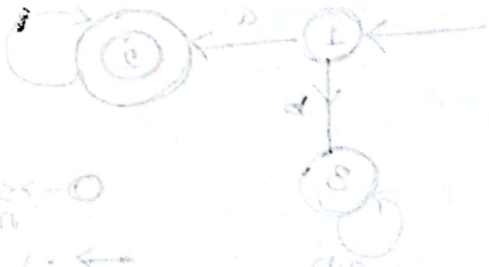
$Q = \{1, 2, 3\}$
$\Sigma = \{a, b\}$
$q_0 = 1.$
$F = 2$

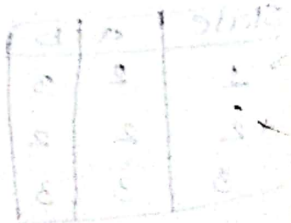
$\delta : Q \times \Sigma \longrightarrow Q.$
Domain range.

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



$\therefore \delta(1,a) = 2$
$\delta(1,b) = 2$
$\delta(2,a) = 2$
$\delta(2,b) = 2$
$\delta(3,a) = 3$
$\delta(3,b) = 3.$





Date: 11.11.2024

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

⇒ $L = \{a, aa, ab, \ldots\}$

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

$Q = \{P, Q, D\}$

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

$q_0 = P$

$F = \{Q\}$

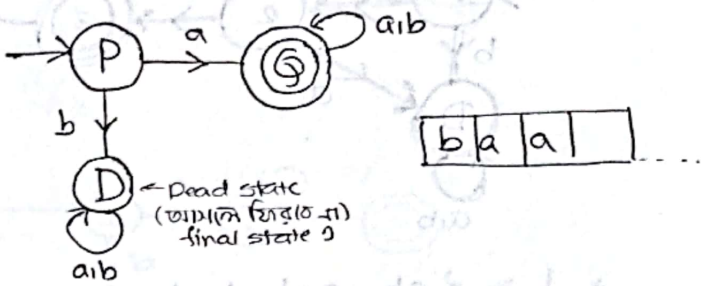
$\delta: Q \times \Sigma \to Q$.

$(P, a) = Q$ $(P, b) = D$

$(Q, a) = Q$ $(Q, b) = Q$

$(D, a) = D$ $(D, b) = D$

1.



2.

Starter	a	b
→ P	Q	D
*Q	Q	Q
D	D	D

language accept or reject করে → Machine ← language ছাড়া অস্তিত্ব নাই

M/C → Machine ← M(L)

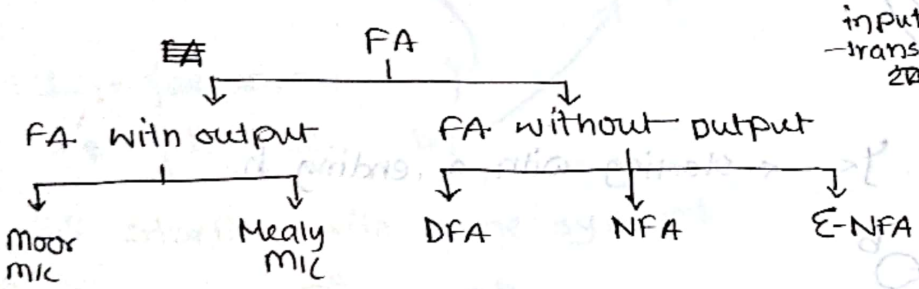
input ছাড়া transition হয় না,

DFA – Deterministic FA

NFA – Non নির্ধারিত deterministic FA

ε-NFA – Non-deterministic FA with ε-transition.

input ছাড়া transition হবে এক step থেকে অন্য step এ যাবে empty হলেও

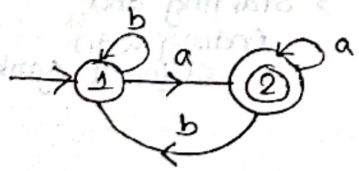


* DFA construction:

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

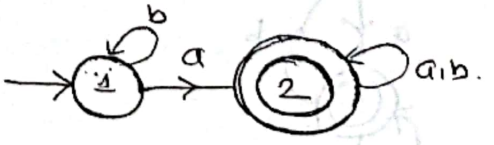
⇒ $L = \{a, aa, ba, aaa, baa \ldots\}$

* প্রতিটি input এর জন্য সুনির্দিষ্ট path থাকতে
* কোনো একটি input এর জন্য একটি path-ই থাকবে

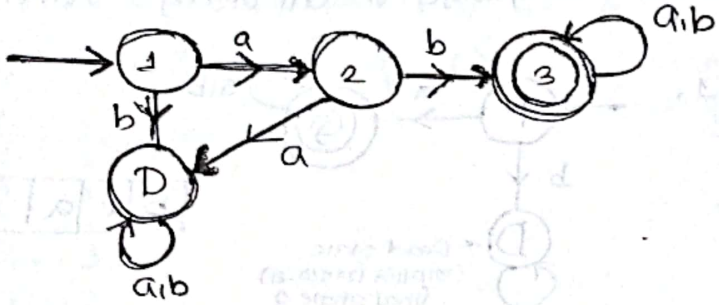


→ Construct a DFA that accepts language containing strings containing all 'a' over $\Sigma = \{a, b\}$.

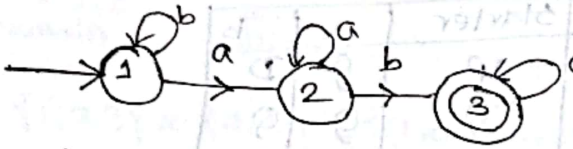
⇒ $L = \{a, ab, ba, bab \ldots\}$



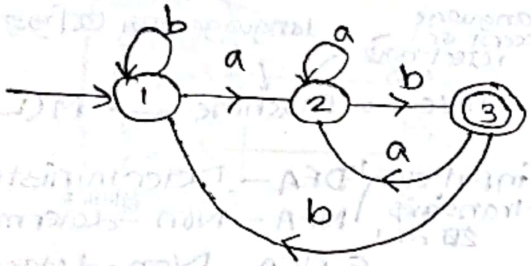
* $L_1 = \{ab, aba, abb, \ldots\}$ → starting with ab.



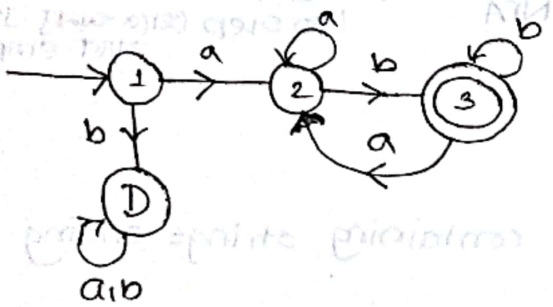
* $L_2 = \{ab, aab, bab, babb, \ldots\}$ ← containing ab.



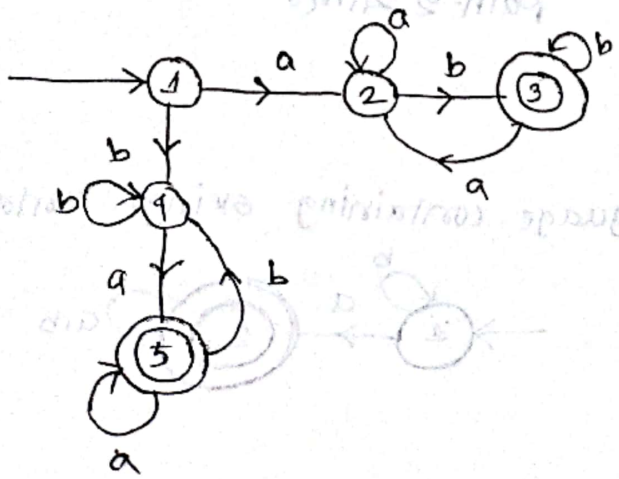
* $L_3 = \{ab, aab, bab, \ldots\}$ ← ending with ab.



* $L_4 = \{ab, aab, abb, \ldots\}$ ← starting with a, ending b.



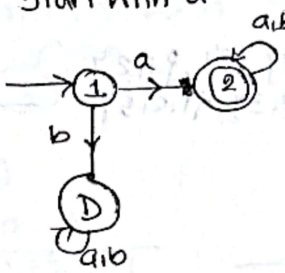
* $L_5 = \{ab, aab, abb, \ldots, ba, baa, baba \ldots\}$ →
  - start a, end b
  - start b, end a.
  - ↳ starting and ending with different symbo



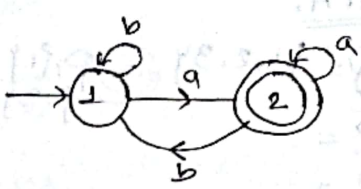
DFA operation:

(a) Concatenation of two DFA ⇒
(b) Union of two DFA.
(c) Intersection of two DFA.
(d) Inverse of DFA.

start with a



end with b

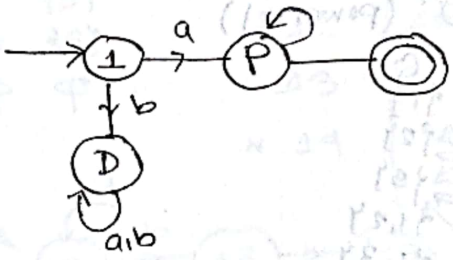


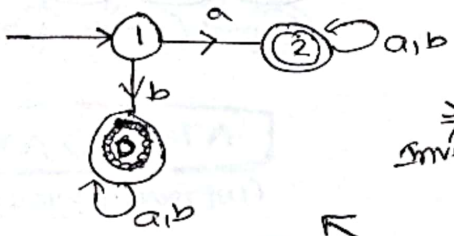
* NFA-ला-पण same operations.

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

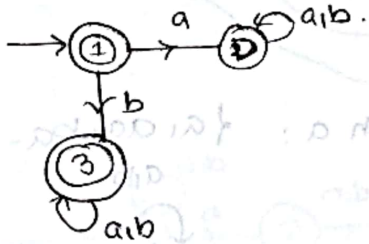
$L_2 = \{b, ab, bb, \ldots\}$

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





Inverse

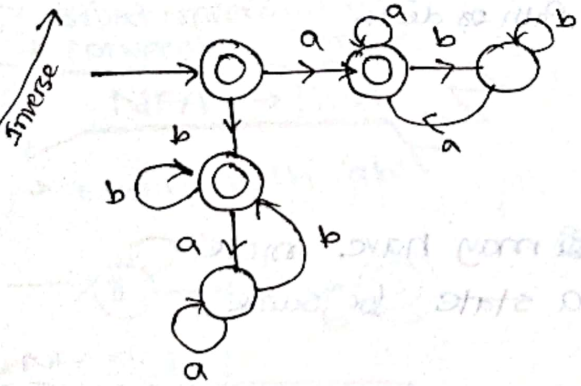


$L_1 = \{aa, aa, ab, \ldots\}$

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

Starting with same symbol

Inverse



NFA:

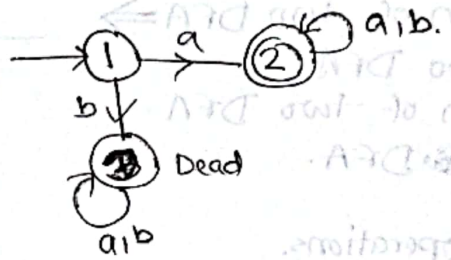
$Q = \{1,2,3\} = \{\phi, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$

$\Sigma =$

$q_0 =$

$F =$

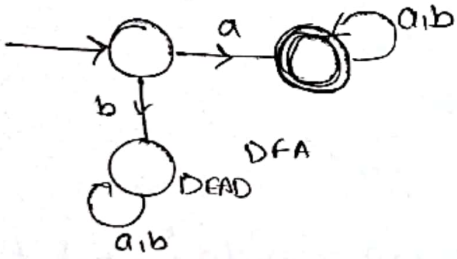
$\delta: Q \times \Sigma \rightarrow 2^Q$ (powerset)

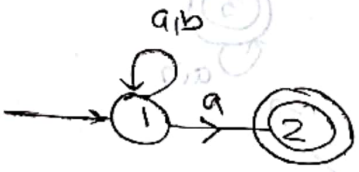


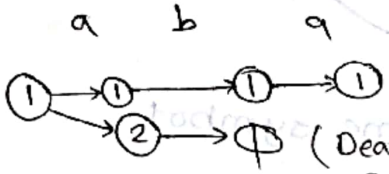
(1,a)	ϕ
(1,b)	$\{1\}$
(2,a)	$\{2\}$
(2,b)	$\{3\}$
(3,a)	$\{1,2\}$
(3,b)	$\{1,3\}$
	$\{2,3\}$
	$\{1,2,3\}$

→ DFA, NFA both language ক accept and reject করে, (equivalent)
But construct করতে [illegible],
NFA construct করা easy.

→ Language ending with a: $\{a, aa, ba \ldots\}$







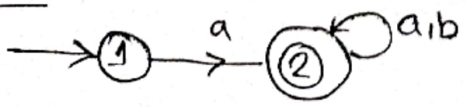
Date: 12.11.2024.

NFA

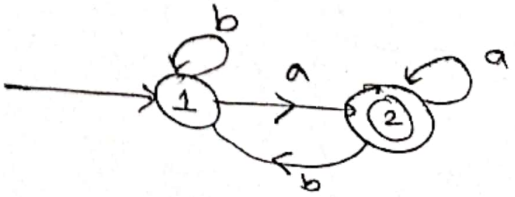
Same as DFA, except that in NFA there may have more than one state attainable from a state for same input alphabet.

$L_1 = \{$ starting with 'a' over $\Sigma = \{a,b\}\}$

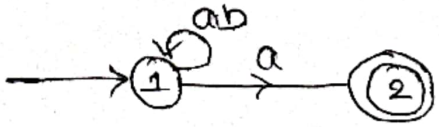
NFA



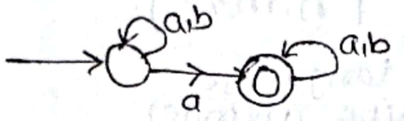
DFA : ending with 'a'



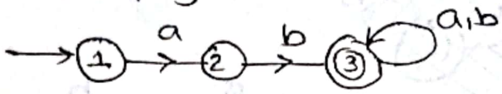
$L_2 = \{$ ending with 'a' over $\Sigma = \{a,b\}\}$



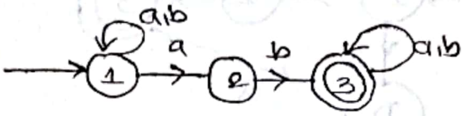
* L_3 = {containing with 'a'}



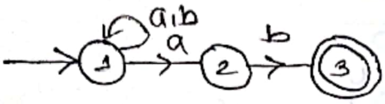
* starting with 'ab'



* containing 'ab'



* ending with 'ab'



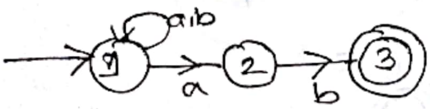
* DFA ⟺ NFA
(Equally powerful)

* 2^Q এর subset হলো DFA এর state.
* All DFA is NFA
not all NFA is DFA.

> (subset conversion method)

* conversion from:
NFA ⟶ DFA

* Ending with 'ab'



NFA এর T.T:

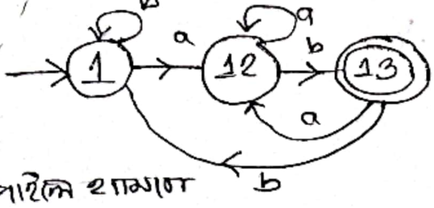
States	input a	b
→ 1	{1,2}	{1}
2	φ	{3}
* 3	φ	φ

union করে

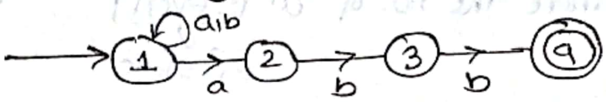
DFA এর T.T:

States	a	b
→ 1	12	1
12	12	13
* 13	12	1

↳ NFA এর Final state থাকলে হবে



* ending with abb:

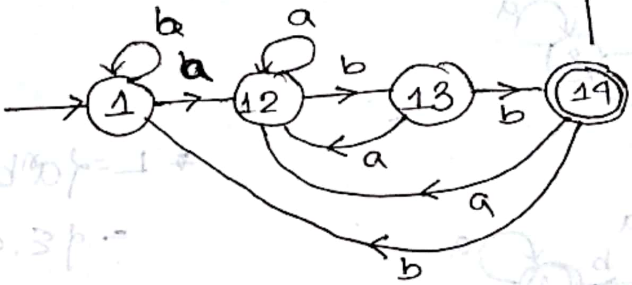


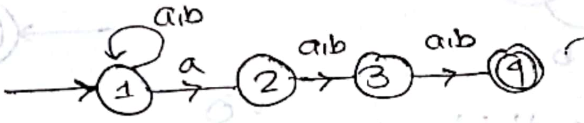
NFA

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

DFA

States	a	b
→ 1	12	1
12	12	13
13	12	14
* 14	12	1.





NFA. state define করা নাই।

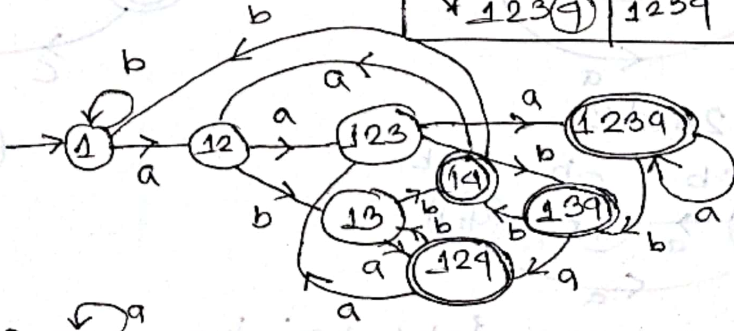
R.H.S is 'a' {aa, ab, abb, abb, aaa, aba, ababa, ...}

NFA.

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

DFA

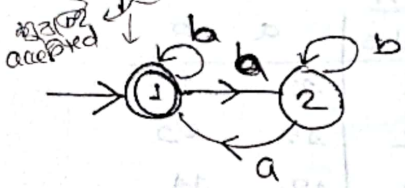
States	a	b
→ 1	12	1
12	123	13
13	124	14
* 14	12	1
* 123	1234	134
* 124	123	13
* 134	124	14
* 1234	1234	134



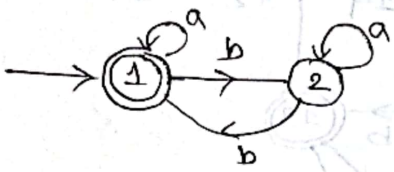
* L = {String where the no. of 'a' is even} (2 final state)

Σ = {a, b}

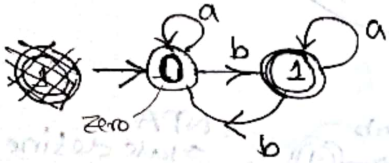
= {ε, aa, aab, aba, baa, ...}



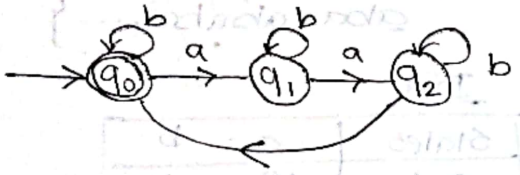
* 'b' is even



* 'b' is odd:



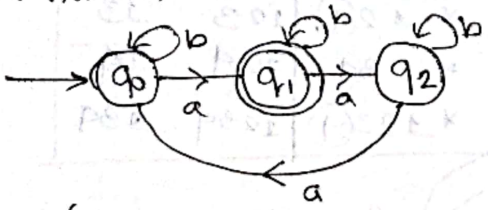
* num. of 'a' is divisible by 3: (0, 1, 2; 3rd state must)



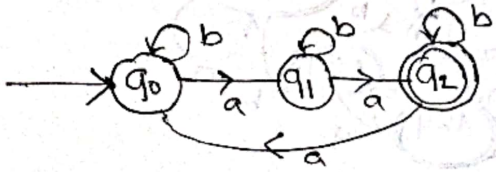
L = {ε, aaa, aaba, aaab, baaa, ...}

abab → reject.

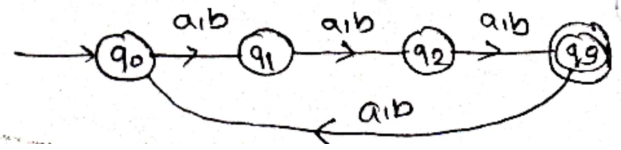
* num of 'a' → 1 mod 3.



* 'a' → 2 mod 3

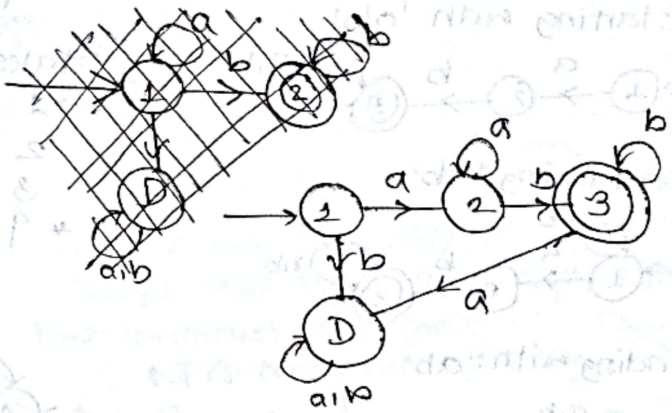


* |w| = 3 mod 4, {abaabaaba...}



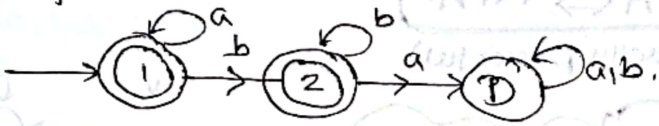
* $L = \{a^n b^m \mid n, m \geq 1\}$.

↳ regular language (Infinite)

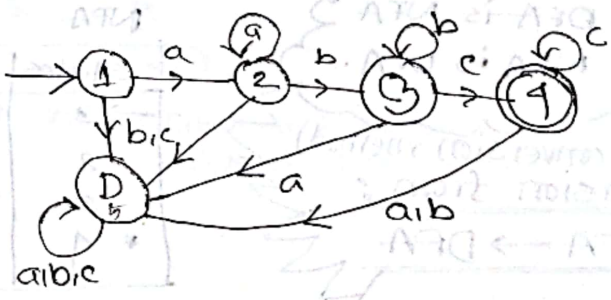


* $L = \{a^n b^m \mid n, m \geq 0\}$

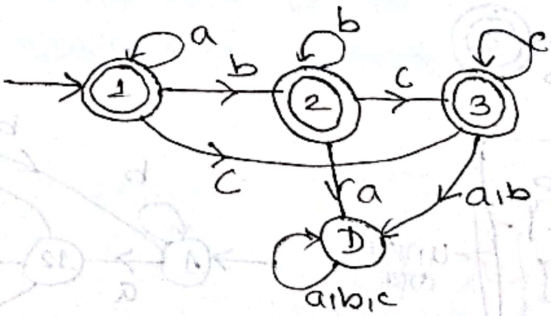
= {ε, aaa, ..., b, bb, ... aab, ...}



* $L = \{a^n m^m c^l \mid n, m, l \geq 1\}$.



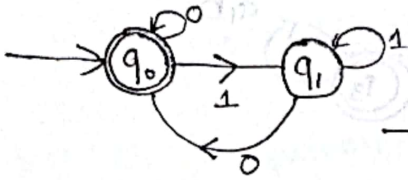
* $L = \{a^n m^m c^l \mid n, m, l \geq 0\}$.



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

$(w)_{Decimal} = (w)_{10} = 0 \bmod 2$. (even)

$L = \{\varepsilon, 0, 00, 000, \ldots, 10, 100, \ldots\}$



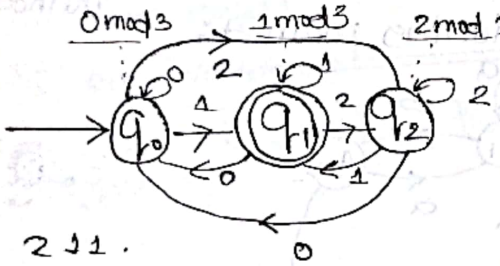
States	0	1
→*q_0	q_0	q_1
q_1	q_0	q_1

* $(w)_{10} = 1 \bmod 3$

↓ Ternary

* $\Sigma = \{0,1,2\}$ → num.

States	0	1	2
→q_0	q_0	q_1	q_2
q_1	q_0	q_1	q_2
q_2	q_0	q_1	q_2

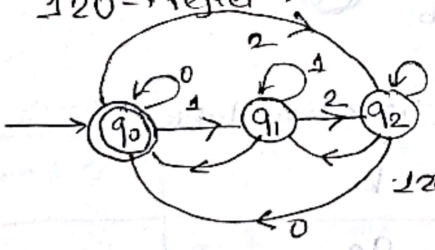


211.

$= 2\times3^2 + 1\times3^1 + 1\times3^0$

$= 18 + 3 + 1 = 22 \,\%\, 3 = 1$

120 → reject.



$\Sigma = \{0,1,2\}$.

$(w)_{10} = 0 \bmod 3$.

120 → accepted.

$\Sigma = \{0,1,2,3\}$

$(w)_{10} = 0 \bmod 3$.

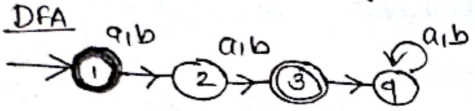
States	0	1	2	3
→q_0	q_0	q_1	q_2	q_0
q_1	q_1	q_2	q_0	q_1
q_2	q_2	q_0	q_1	q_2

+ Data structure
→ Schaum's outline
+ Discrete math
→ Rosen
+ DigSD
→ Tanci
+ Accounting
+
= Statistics

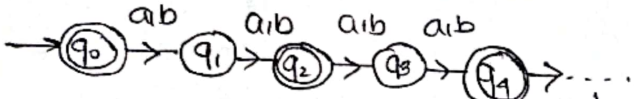
Date: 18.11.2024

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

DFA

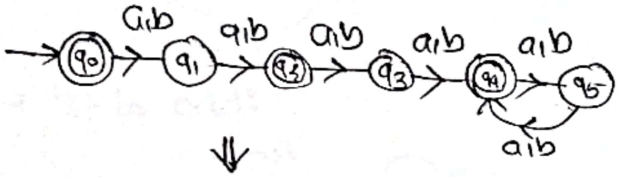


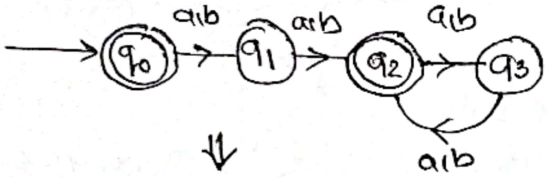
Divisable by 2:

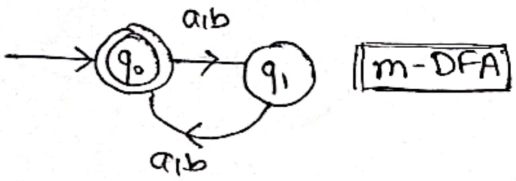


not FA. → infinite state.

finite করাও.



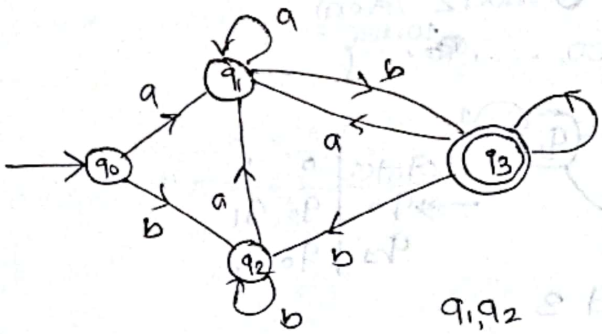




* language construct করার machine can be more than one.
  - machine বড় → efficiency ↓ valid but → space, time ↑
  - machine ছোট - most efficiency ↑
  + m-DFA - একটাই যত language → minimal DFA.

→ Subset conversion method:
NFA → DFA.
(আগের class এ)

→ Minimization of DFA:

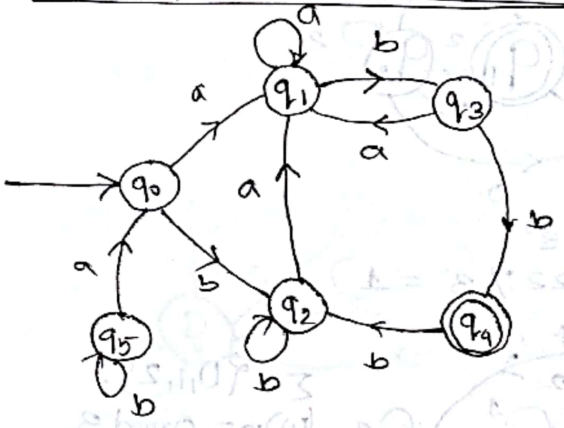


If → $\delta(q_1, w) \in F \Rightarrow \delta(q_2, w) \in F$ &
$\delta(q_1, w) \notin F \Rightarrow \delta(q_2, w) \notin F$.

if and only → $q_1 \Leftrightarrow q_2$. (equivalent হলে single state ধরে নিবো)

- Table filling method.
+ Table partitioning method } DFA minimize করার method.

* Table partitioning method:



Step-1: Draw the transition table.

States	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
X q_5	q_0	q_5

Step-2: final state গুলো mark করে আলাদা partition নিবো

Step-3: যেসব state গুলো থেকে initial state এ আসা যাবে না, সেগুলো বাদ

~~State 4~~:

'0' equivalent states

$|w|=0$ ← ↳ ↳

[non accepting state] [accepting state]

$\therefore |w|=0$. '0' equivalent states $[q_0,q_1,q_2,q_3][q_4]$.

'1' equivalent states.

$[q_0,q_1,q_2][q_3][q_4]$

'2' equivalent states.

$[q_0,q_2][q_1][q_3][q_4]$ ....

~~পরপর equivalent state same হলে stop করবো।~~

পরপর equivalent state same হলে stop করবো।

'3' equivalent states.

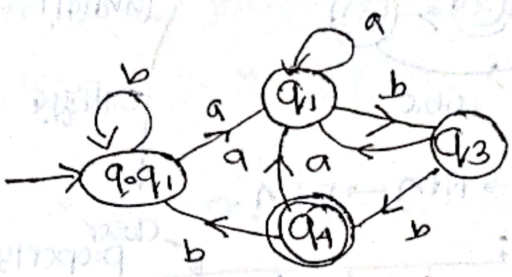
$[q_0,q_2][q_1][q_3][q_4]$

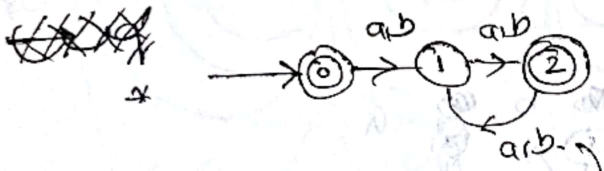
....

~~q_0,q_1,q_2~~

'2' $\begin{cases} \delta(q_0,a)=q_1 & \delta(q_0,b)=q_2 \\ \delta(q_1,a)=q_1 & \delta(q_1,b)=q_3 \end{cases}$

(q_0,q_1) $\begin{cases} \delta(q_0,a)=q_1 & \delta(q_0,b)=q_2 \\ \delta(q_2,a)=q_1 & \delta(q_2,b)=q_2 \end{cases}$





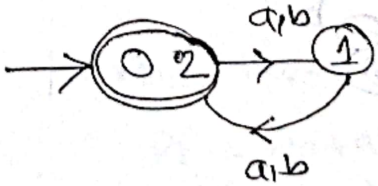
States	a	b
→ * 0	1	1
1	* 2	* 2
* 2	1	1

m-DFA

$(0,2)$:

$\delta(0,a)=1$ $\quad \delta(0,b)=1$

$\delta(2,a)=1$ $\quad \delta(2,b)=1$.

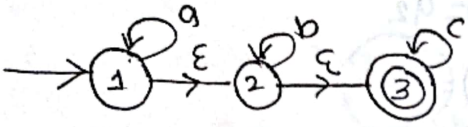


'0' equivalent states

$[1][0,2]$

'1' ↳ ↳ $[1][0,2]$

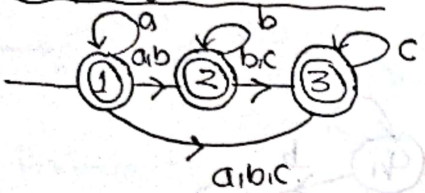
Date: 26.11.2024



$a^n b^m c^l \mid n,m,l \geq 0$

Thomson's construction method apply করার সময় আছে পরে দেখাবো!

* Hopcroft's method:



আক্ষিয়ার

* ε-NFA → NFA → DFA.

ε-NFA:

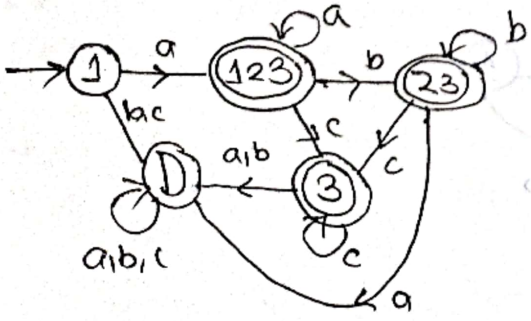
closer property

States	a	b	c	ε*
→ 1	{1}	ϕ	ϕ	{1,2,3}
2	ϕ	2	ϕ	{2,3}
* 3	ϕ	ϕ	{3}	{3}

DFA:

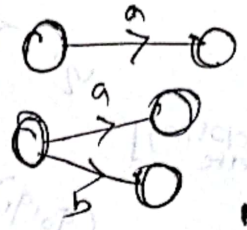
D → Dead state/flat state.

States	a	b	c
	ε^*a	ε^*b	ε^*c
→ 1	{123}	D	D
* 123	123	23	3
* 3	D	D	3
* 23	D	2,3	3
D	D	D	D



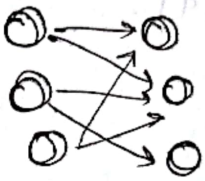
abbc → acc

abbcb → rej



DFA → current and next both states are নির্দিষ্ট

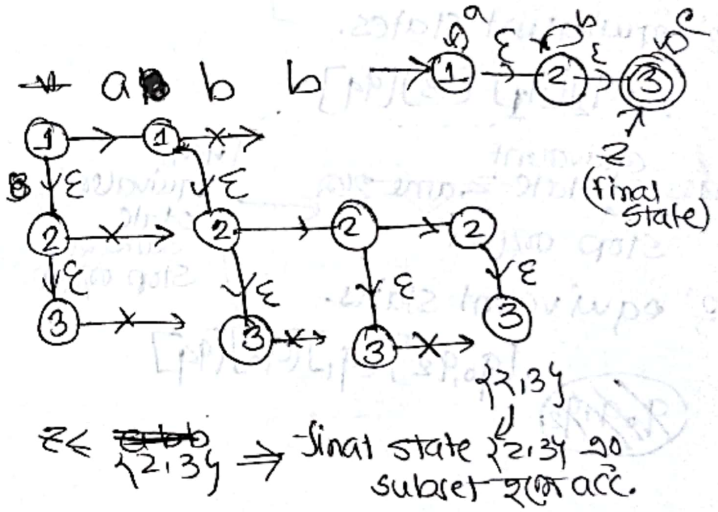
NFA → current state নির্দিষ্ট কিন্তু next state নির্দিষ্ট না।



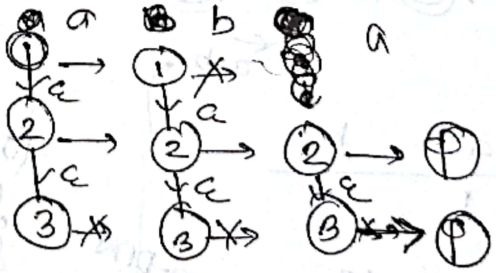
ε-NFA

current and next both states are অনির্দিষ্ট

* a b b



~~abb~~ {2,3} ⇒ final state {2,3} এর subset হলে acc.



Regular Expression:

Regulare expression is a pattern of string of a language.

* A language may have regular expression must be accepted by FA. Hence, it is regular language.
* ϕ is a regular expression. represent করে $\{\}$ → কোনো string নাই
* ε is also regular expression represents the language $\{\varepsilon\}$ ← empty string
* ρ " " " " " that " " of $\{\rho\}$

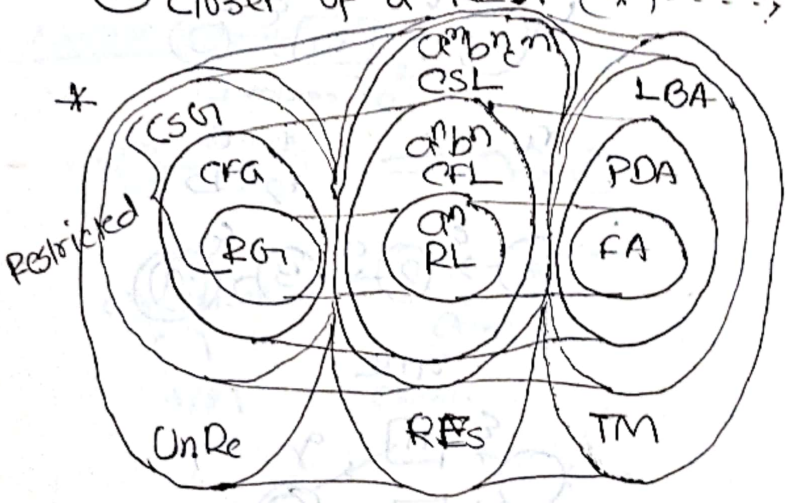
R.E. → Premitive/ Fundamental * $r_1 = a$; $r_2 = b$. এগুলো

↓ 3 operations:

(ক্রমিক সংযোগ) ① Concatenation of two REs. (.) ⋯→ $r_1.r_2 = a.b = ab$

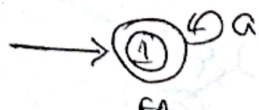
② Union of two REs (+/|) ⋯→ $r_1|r_2 = a|b$, $r_1+r_2 = a+b$

③ Closer of a RE. (*) ⋯→ $r_1^* = a^* = \{\varepsilon, a, aa, aaa \dots\}$



* REs → Recursively Enu...
Un Restricted grammer → UnRe.
* RL ক Represent করতে কোনো Regulare expression লাগে।

* $L = \{a^n \mid n \geq 0\} = \{\varepsilon, a, aa, aaa \dots\}$

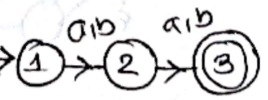


a^*
RE.

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

$L = \{aa, ab, ba, bb, \dots\}$
↳ at least two.

~~$L = (a+b)^{2n+1}$~~ RE $= (a+b)(a+b)(a+b)^*$



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

RE = aa+ab+ba+bb
= a(a+b) + b(a+b)
= (a+b)(a+b)

* RE can be more than 1.

* Most two → $L = \{aa, ab, ba, bb, a, b, \varepsilon\}$
RE = aa+ab+ba+bb+a+b+ε
= (a+b+ε)(a+b+ε)

* $(a+b)^{2n} \mid n \geq 0$ ⟶ even.
RE = $((a+b).(a+b))^*$

* $(a+b)^{2n+1} \mid n \geq 0$ ⟶ odd.
RE = $((a+b).(a+b))^*(a+b) = (a+b).((a+b).(a+b))^*$

RE = $a(a+b)^*$ → starting with a

= $(a+b)^*b$ → ending with b

= $+(a+b)^*b$ → both.

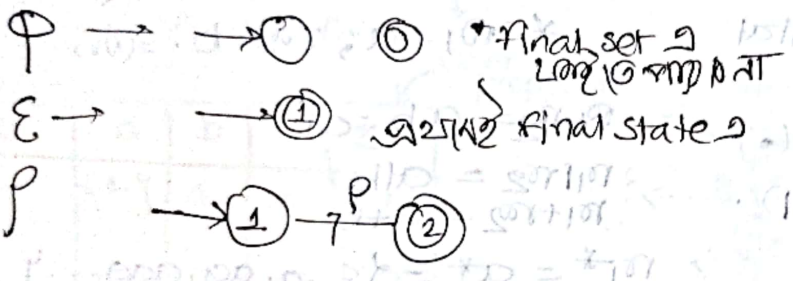
= $a(a+b)^*b$

= $a(a+b)^* + b(a+b)^*a$ → diff symbol

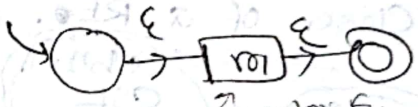
= $a(a+b)^*a + b(a+b)^*b$ ← starting same, ending diff. ?

= $a(a+b)^*a + b(a+b)^*b + (a+b+\epsilon)$ ← starting and ending with same symbol

$\{\epsilon, a, b \ldots y$

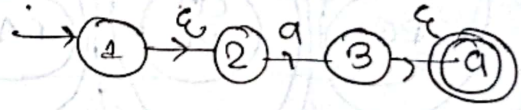


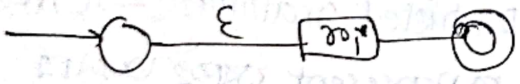
1. Concatetion of 2 FAs
2. Union of 2 FAs.
3. Closure of a FA.

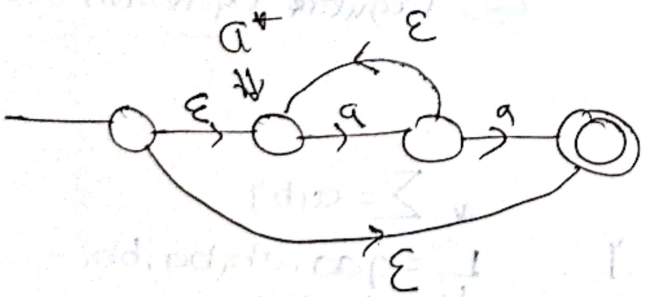


regular E.

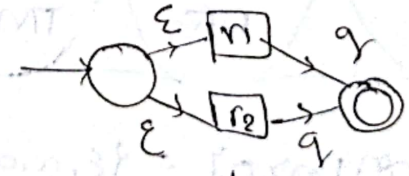
r_1 r_2 $r_1 = a$, $r_2 = b$

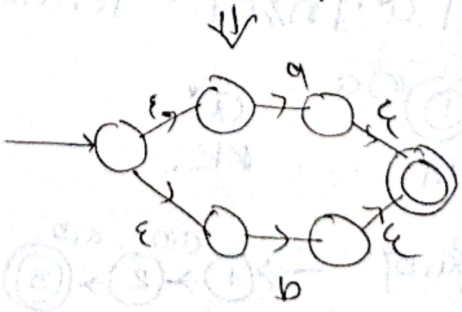




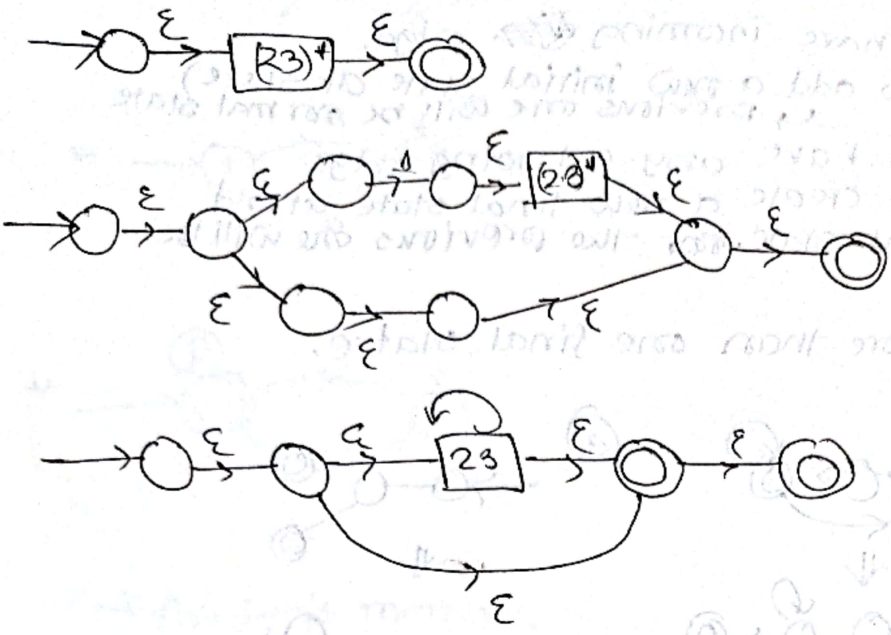


$r_1 + r_2$





1(23)* | 3 = r



2.12.24

Regular Expression:

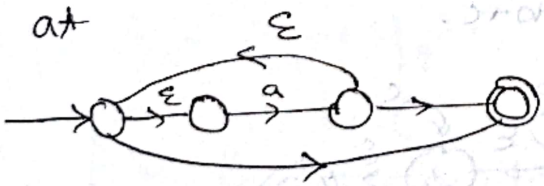
a^+ b^2

alphabet
( )
+ / | |
+

$a^* = \{a, \}$

closer.

a+

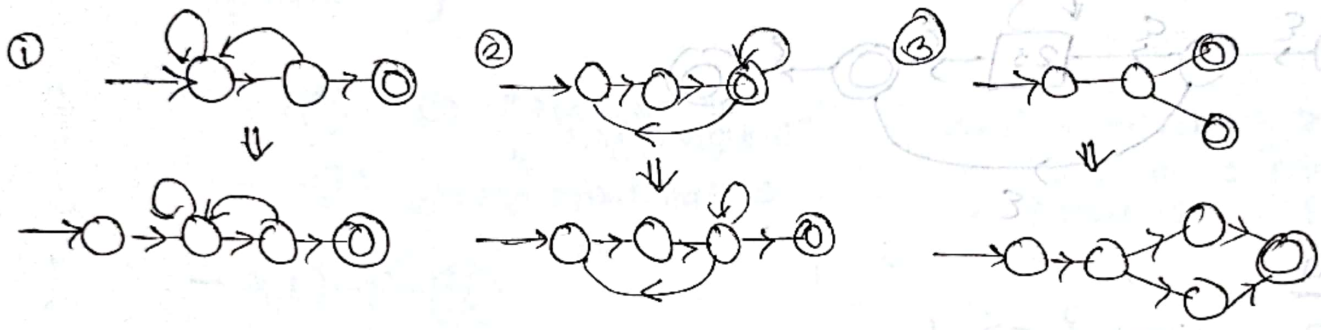


P.T.O.

Conversion of FA to RE:

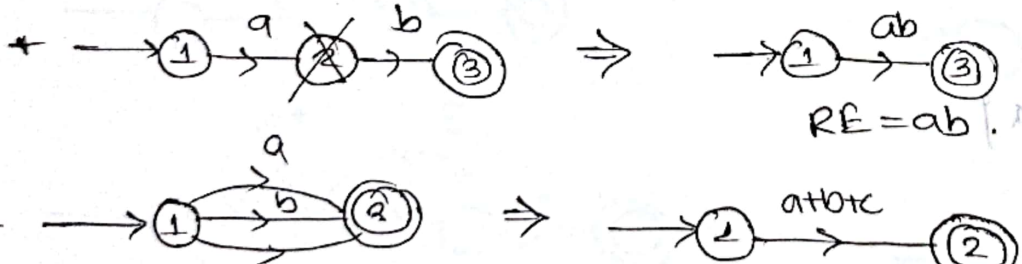
State elimination method:

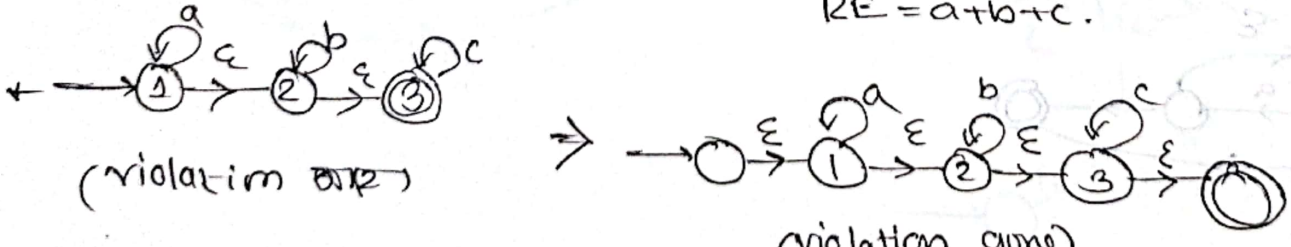
1. Initial state should not have incoming edge.
   (If there is one, we have to add a new initial state at state)
   ↳ previous one will be normal state
2. Final state(s) should not have any outgoing edge.
   (If there is one, we have create a new final state at end and make it the final state, the previous one will be normal state)
3. There should not be more than one final state.

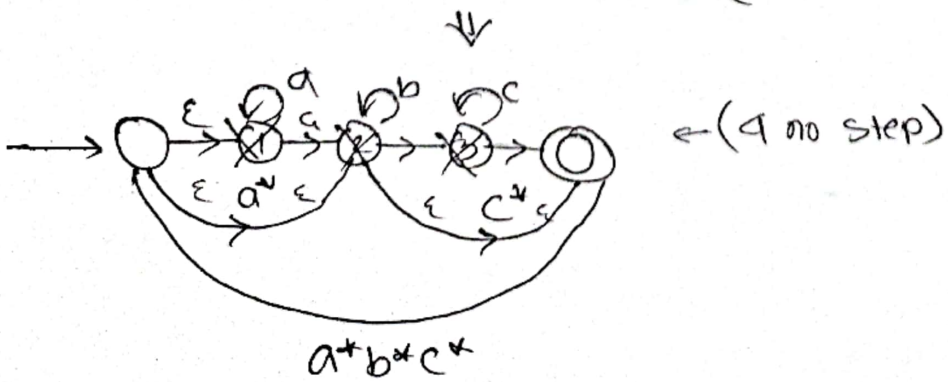


1, 2, 3. violation না থাকলে,

4. যাকের্ড state বাদিবো।

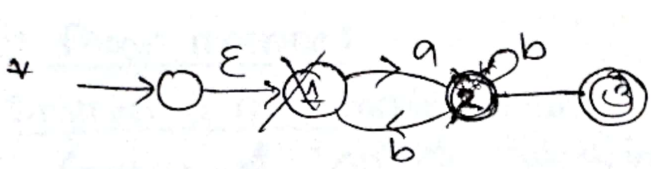


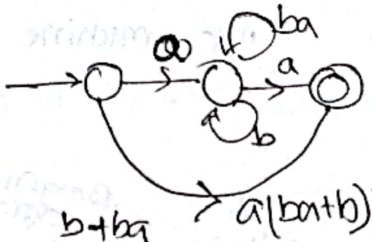


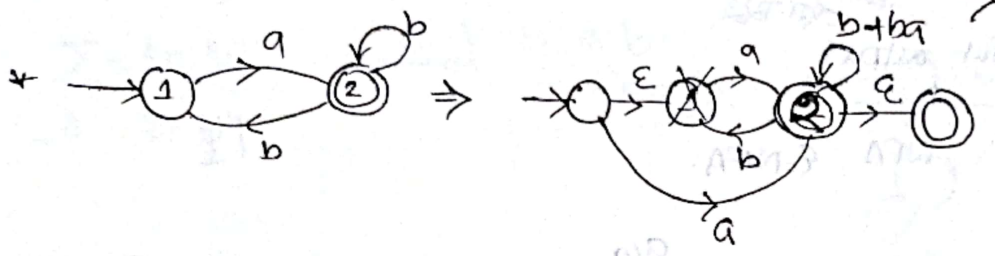


← (4 no step)

∴ RE = $a^*b^*c^*$

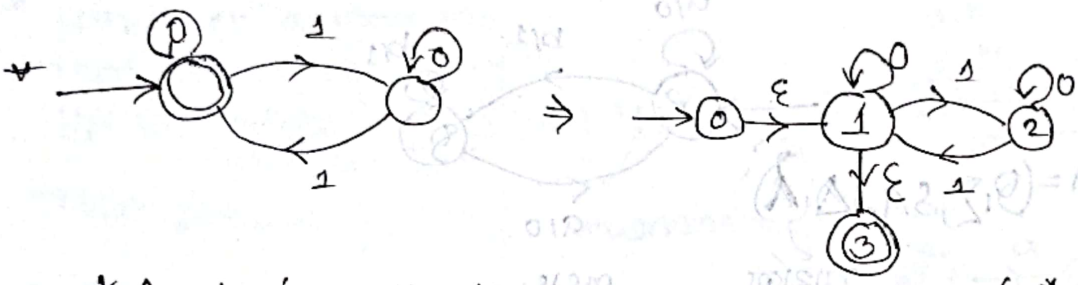


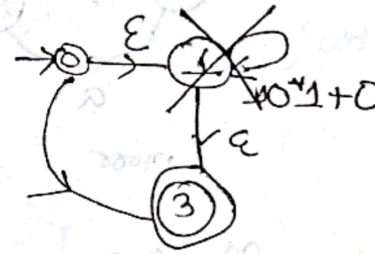




RE = a(b+ba)

?





(10*1+0)*

*Arden's method:

2* কোনো ε থাকবে না

1* only NFA and DFA Transmission হবে।

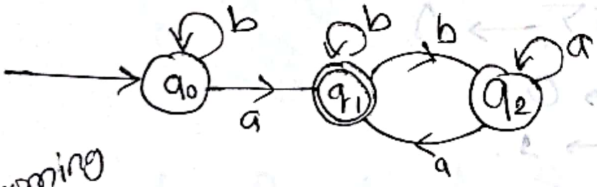
3* more than one final state থাকলে,

প্রতিটা final state এর জন্য RE generate করা।

RE = (10*1+0)*

= {ε, 0, 10, 11, 101, ...}

↑ 1 এর সংখ্যা always even.



incoming edges

$q_0 = \varepsilon + q_0 b$ ... ①

$q_1 = q_0 a + q_1 b + q_2 a$ ... ②

$q_2 = q_1 b + q_2 a$ ... ③

$R = Q + RP \rightarrow R = QP^*$ → Arden's rules.

using Arden's theorem,

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

$\Rightarrow q_1 = b^* a + q_1 b + q_1 ba^* a$

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

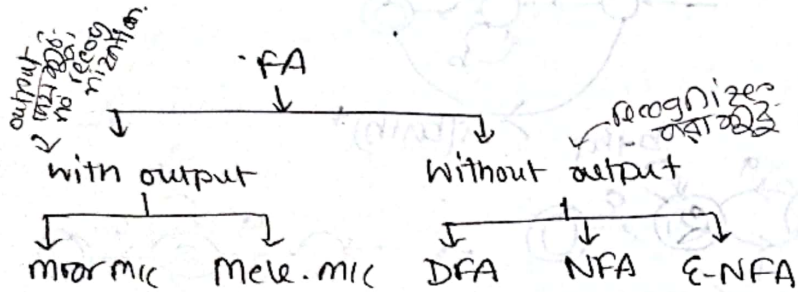
$= b^* a (b + ba^* a)^*$

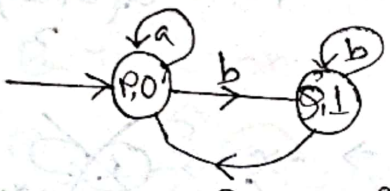
$\Rightarrow q_2 = q_1 ba^*$

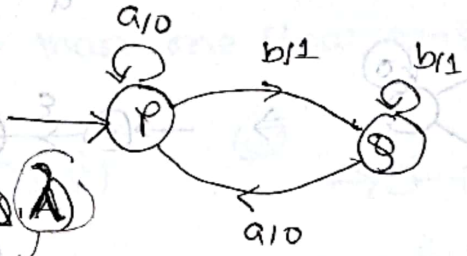
RE = final state এর RE

$\therefore RE = b^* a (b + ba^* a)^*$

Date: 09.12.2024 m/c → machine.







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

moor, mele ই

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

$\delta: Q \times \Sigma \to Q.$

Δ: output Alphabet. $\Delta = \{0, 1\}$

output is related to transition

$\lambda: Q \times \Sigma \to \Delta$

$\lambda: Q \to \Delta.$

output is related to state.

$p \to 0$

$q \to 1$

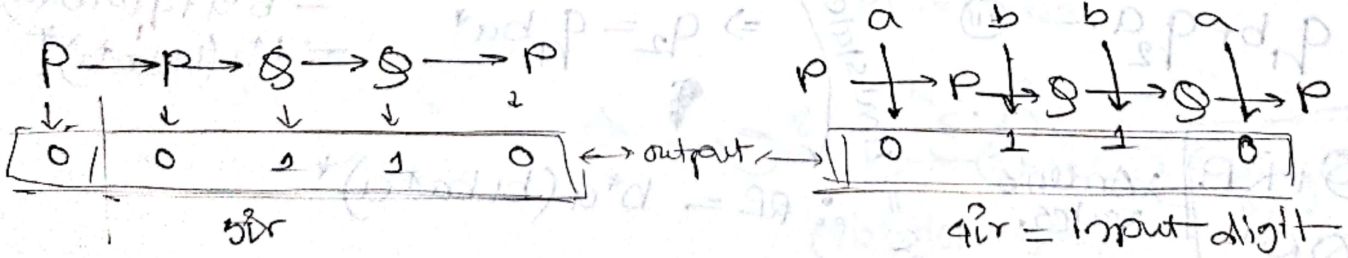
$(p, a) \to 0$

$(p, b) \to 1.$

$(q, a) \to 0$

$(q, b) \to 1.$

a b b a



* Moor machine:

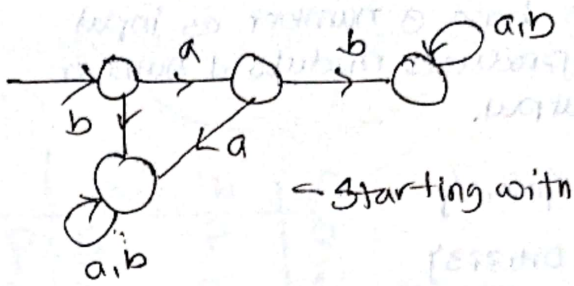
* Construct a moor machine that takes input string over $\Sigma = \{a, b\}$.

Count no. of 'ab' as substring.

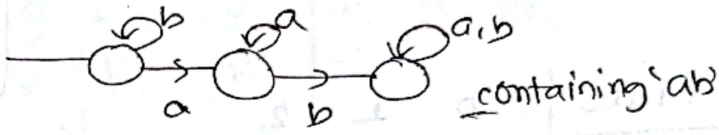
$\Sigma = \{a, b\}$

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

a b b a b ⟹



← Starting with 'ab'

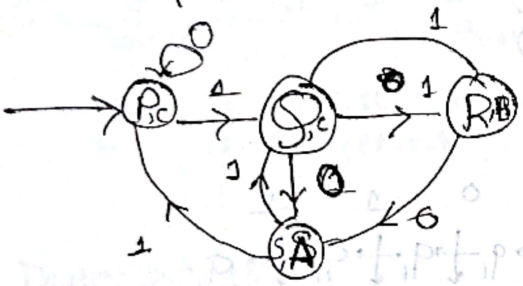


containing 'ab'

* construct a moor mlc that takes ~~any~~ {0,1} as i/p and ~~produces~~ '10' ending with and '11' ending with then produces A, B; otherwise C.

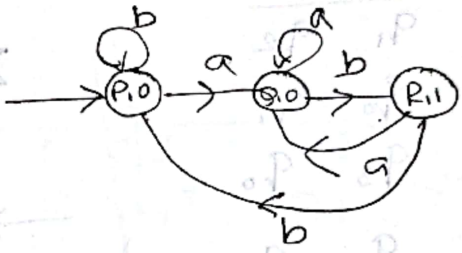
$\Sigma = \{0, 1\}$

$\Delta = \{A, B, C\}$



1 0 1 1 0

P → Q → S → Q → R → S



a b b a b

P → Q → R → P → Q → R

0 0 1 0 0 1 = 1+1 = 2

~~PRACTICE~~.

Regular Expression ~~and~~ CT

17.12.2024.

Date: 10.12.2024

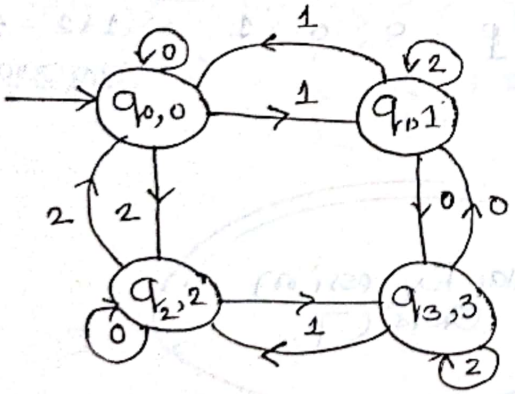
* moor machine:

construct a moor machine that takes base-3 number as input and produces modulo-4 number as output.

$\Sigma = \{0,1,2\}$

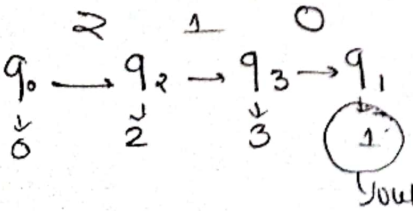
$\Delta = \{0,1,2,3\}$.

states	0	1	2
$\rightarrow q_0$	q_0	q_1	q_2
q_1	q_3	q_0	q_1
q_2	q_2	q_3	q_0
q_3	q_1	q_2	q_3



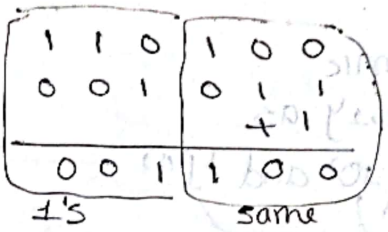
$(210)_3 = 2\times3^2 + 1\times3^1 + 0\times3^0$

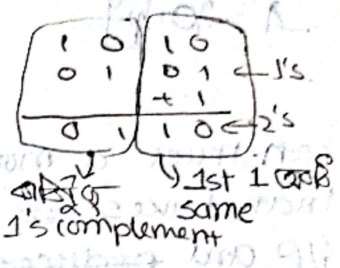
$= 21 \% 4 = 1.$



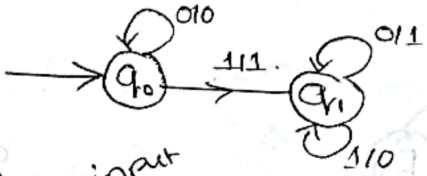
* mele machine:

Construct a mele machine that takes binary number as input and produces it's equivalent 2's complement number as output. provided that the input should be taken from LSB first.





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



LSB (zero input

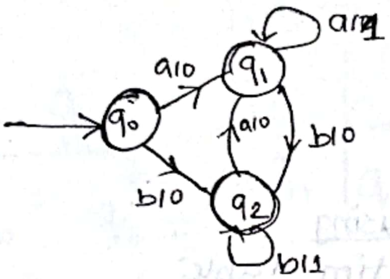
0 0 1 0 1 1

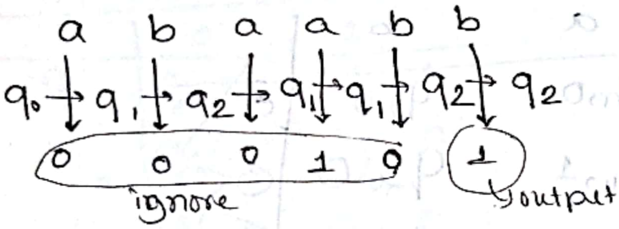
$q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_1 \rightarrow q_1 \rightarrow q_1 \rightarrow q_1$

0 0 1 1 0 0

* construct a mele m/c. that accepts the string of a language over $\Sigma=\{a,b\}$ where the string should be ending with 'aa' or 'bb'.

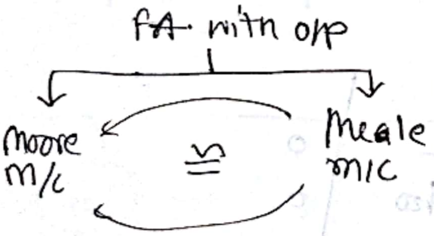
$\Sigma=\{a,b\}$; $\Delta=\{0,1\}$.





output = 1 → accepted
" = 0 → rejected.

Date: 29.12.29

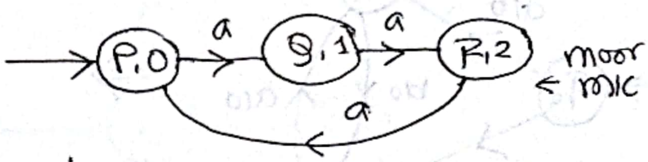


* কোনটা strong বেশি?
both, conversion সম্ভব তাই

Moore ⟶ mele convert
2 ভাবে → Transition diagram
→ " table.

Moore থেকে mele m/c এ converse:

L = { numbers devided by 3 }

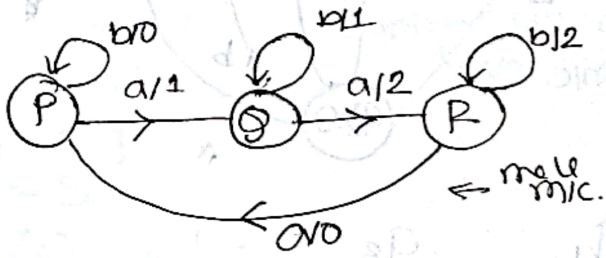


	a	b	Δ
P	Q	P	0
Q	R	Q	1
R	P	R	2

← Moor m/c.

	a	b
P	Q,1	P,0
Q	R,2	Q,1
R	P,0	R,2

← mele m/c.



moor → mele.

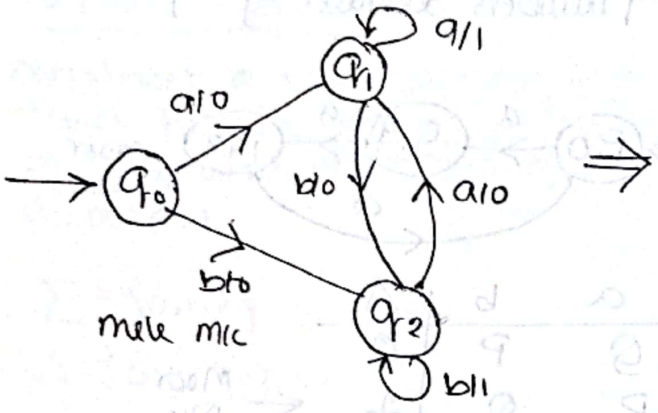
* number of states of moor m/c
= " " " " mele m/c.

mele ⟶ more এ বেশি হবে number of states.

* mele m/c
n input n output
moor m/c
n input n+1 output (starting state এর জন্য বেশি (state))

* first state । output ignore হবে same and equally powerful.

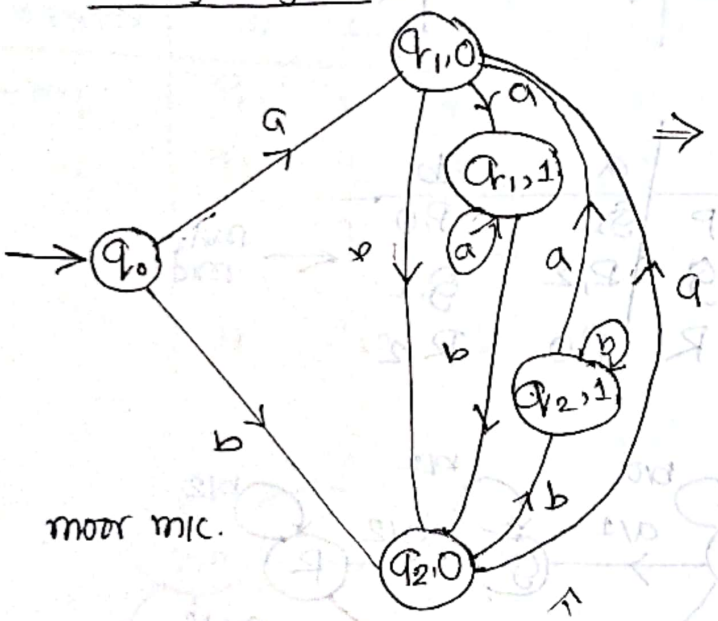
Mele m/c to moor m/c:



Transition table for mele m/c:

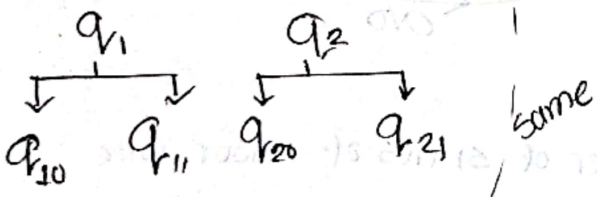
	a	b
q_0	$q_1, 0$	$q_2, 0$
q_1	$q_1, 1$	$q_2, 0$
q_2	$q_1, 0$	$q_2, 1$

using diagram:

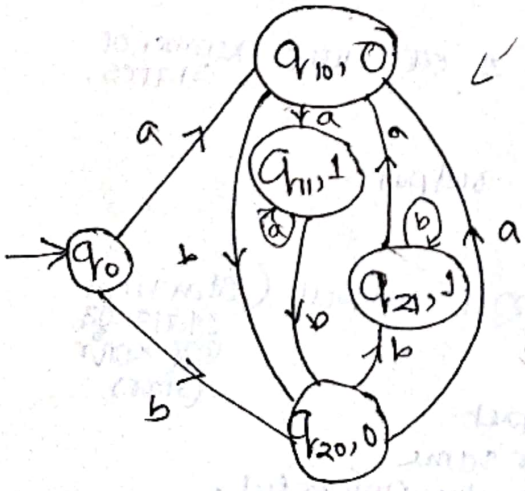


using moor m/c transitim table:

	a	b	
q_0	$q_1, 0$	$q_2, 0$	0
q_{10}	$q_{11}, 1$	$q_{20}, 0$	0
q_{11}	$q_{11}, 1$	$q_{20}, 0$	1
q_{20}	$q_{10}, 0$	$q_{21}, 1$	0
q_{21}	$q_{10}, 0$	$q_{21}, 1$	1



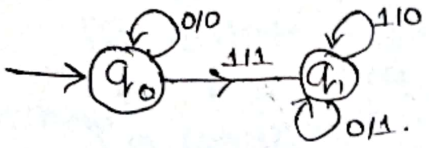
	a	b	
q_0	q_1	q_2	0
q_{10}	q_{11}	q_{20}	0
q_{11}	q_{11}	q_{20}	1
q_{20}	q_{10}	q_{21}	0
q_{21}	q_{10}	q_{21}	1



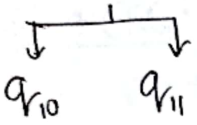
Date: 06.01.2025

* binary input নিয়ে 2's com. output দিবে।

Mealy m/c:



	0	1
q_0	q_0,0	q_1,1
q_1	q_1,1	q_1,0



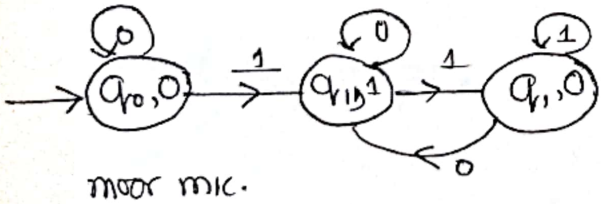
Moor m/c transition table:

	0	1
q_0	q_0,0	q_{11},1
q_{10}	q_{11},1	q_{10},0
q_{11}	q_{11},1	q_{10},0

⇓

	0	1	
q_0	q_0	q_{11}	0
q_{10}	q_{11}	q_{10}	0
q_{11}	q_{11}	q_{10}	1

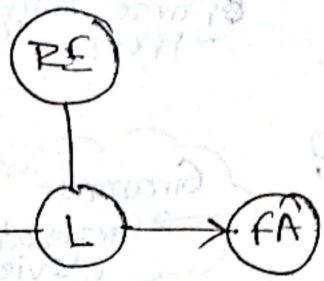
⇓



moor m/c.

* mele m/c এ input N output M হলে,

moor m/c এ সর্বোচ্চ output হবে N×M.



Grammer is represented by 4 tuple.

$G_1 = (N, T, S, S)$

N → set of non terminal
T → set of terminals.
S → starting symbol, $S \in N$.
S → set of productions/ productions rules/ rules.

(i) $S \rightarrow aSB \mid aB$ (ii)

(ii) $B \rightarrow b$

N = {S, B}
T = {a, b}
S = { }
S = S.

Capital letters ↳ variable.

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

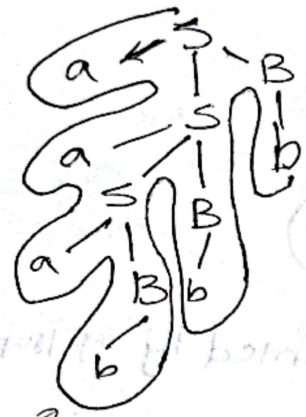
Getting strings from the grammer is called derivation.

$S \Rightarrow aSB$ [using (i)] $w = "aaabbb"$
$\Rightarrow aaSBB$ [using (i)]
$\Rightarrow aaaBBB$ [using (ii)]
$\Rightarrow aaabBB$ [using (iii)]
$\Rightarrow aaabbB$ "
$\Rightarrow aaabbb$ " → sentence.

sentential forms

← LMD
Left most Derivation
RMD ↓ opposite

Pictorial form → Derivation (TOC) tree / Sparse tree (compile design)



traverse.

Grammer → Generating device.

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

* How to create G from RL:

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

$S \to aa|ab|ba|bb.$

$RE = aa + ab + ba + bb.$

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

$= \underbrace{(a+b)}_{A}\underbrace{(a+b)}_{A}.$

$S \to AA$

$A = a/b.$

$S \to aA|bA.$

$A \to a|b.$

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

exactly (2)

at least 2).

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

~~$S \to aS +$~~ $a(a+b) + b(a+b)$

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

$S \to AAB$

$A \to a|b.$

$B \to aB|bB|\varepsilon$

$S \to aS|bS|\varepsilon.$

$a(a+b)^*b$

$S \to aBb.$

$B \to aB|bB|\varepsilon.$

ending with.

different symbol

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

$S \to aBb|bBa.$

$B \to aB|bB|\varepsilon.$

RL থেকে যে যে মেশিন দ্বারা recognize করা যায় FA দ্বারা।

memoryless device

$a^m b^m \to$ memory লাগে. FA হবে না,

$a^m b^m \to$ FA হবে।

Classification of Grammar:

Noam Chomsky classify grammer in four categories; based on different restrictions:

1. Unrestricted G_1.
2. Context sensitive G_1. (CSG)
3. " Free G_1. (CFG)
4. Regular G_1. (RG)

(2, 3, 4 — restricted)

* CSG:

$\alpha \rightarrow \beta$;

α is defined as

$\varphi_1 A \varphi_2$

β is defined as

$\varphi_1 \tau \varphi_2$

$\varphi_1, \varphi_2 \in (N \cup T)^*$ (context)

$A \in N$

$\tau \in (N \cup T)^+$

A can be rewrite as τ in the context of φ_1 and φ_2,

$\varphi_1 A \varphi_2 \rightarrow \varphi_1 \tau \varphi_2$

$\alpha \rightarrow \beta$; $|\alpha| \leq |\beta|$.

Date: 07.12.2025

* Chomsky classification/Chomsky Hierarchy of Formal Grammer:

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

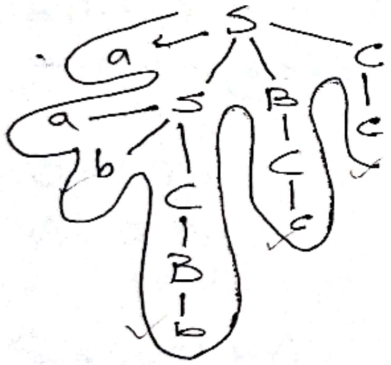
1. $S \rightarrow aSBC$.
2. $S \rightarrow abC$.
3. $bB \rightarrow bb$
4. $bc \rightarrow bc$
5. $CB \rightarrow BC$
6. $cC \rightarrow cc$

→ restriction যেমন সবচেয়ে কম: ডানে কম বামের terminal থাকে

restriction ↓, general (most) ↑

Context sensitive → most general

$w =$ "aabbcc". $n = 2$.



aabbcc

CFG: (Context free grammar)

Restriction: CSG এর (চেয়ে বেশি) production রুলে যেমন বিদ্যমান,

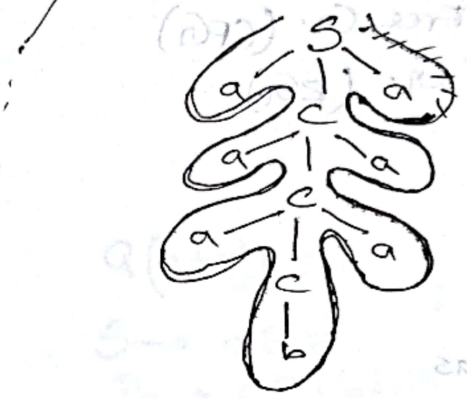
$\alpha \rightarrow \beta$

$\alpha \in N$. (Non-terminal)

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

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

$w =$ "aaabaaa"



* Regular grammer - G of G.

Regular Grammer (RG):

(সবচেয়ে বেশি restrict...)

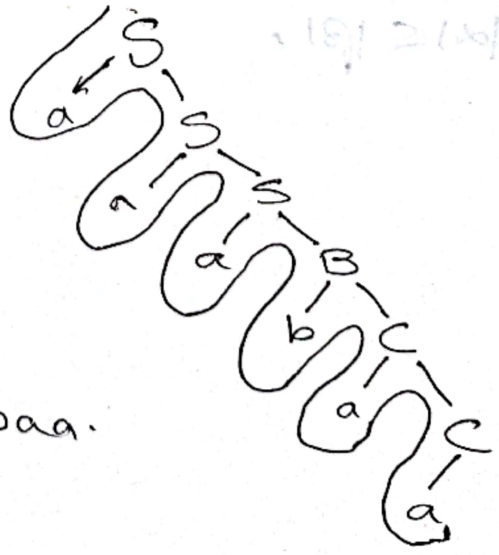
$\alpha \rightarrow \beta$

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

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

1. $S \rightarrow aS$
2. $S \rightarrow aB$
3. $B \rightarrow bC$
4. $C \rightarrow aC$.
5. $C \rightarrow a$.

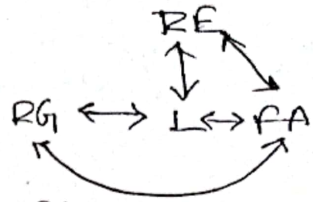
$w =$ "aaabaa"



aaabaa.

RG

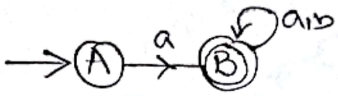
- LLG (Left linear Grammer)
  - *Left দিকে expand করে
  - *B will be form of a or Ba
- RLG (Right linear Grammer)
  - *Right এ expand করে
  - *B will be form of a or aB



RG ⟷ FA:

A → aB.
B → aB|bB|ε

*variable গুলো state হবে।



* ε থাকলে হলে end state এ আসতে rules violet করবে।

G

- UG (T_1)
- RG
  - CSG (T_2)
  - CFG (T_3)
  - ReG (T_4)
    - LLG
    - RLG.

$L(T_1) \supseteq L(T_2) \supseteq L(T_3) \supseteq L(T_4)$

most general . . . . most restricted

* RLG vs LLG.

RLG:

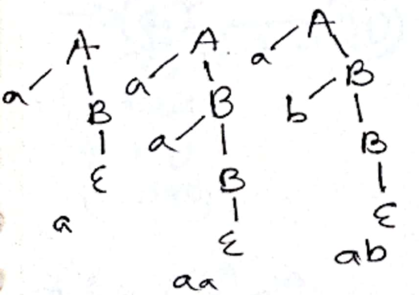
A → aB

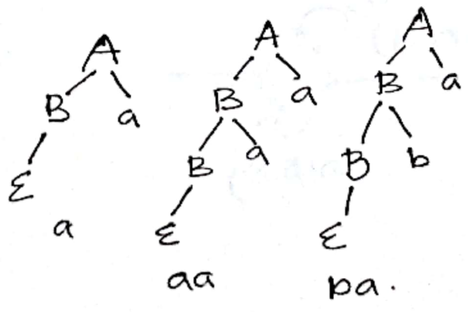
B → aB|bB|ε

LLG:

A → Ba

B → Ba|Bb|ε.





L = {a, aa, ab, ...}
starting with a.

L = {a, aa, ba ...}
ending with a.

* RLG থেকে direct conversion করলে language change হয়ে যাবে।

∴ RLG and LLG is not equivalent; conversion করতে হবে।

* RLG, LLG conversion:

1. RLG → FA
2. FA কে reverse করে new FA.
3. FA → RLG.

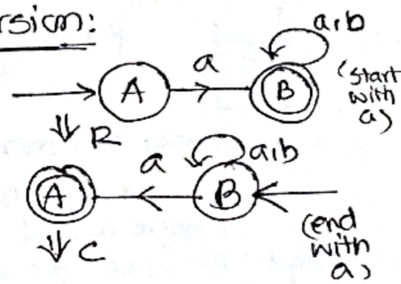
   B → aB|bB|aA

   A → ε

4. RLG কে reverse করে LLG.

   B → Ba|Bb|Aa

   A → ε.

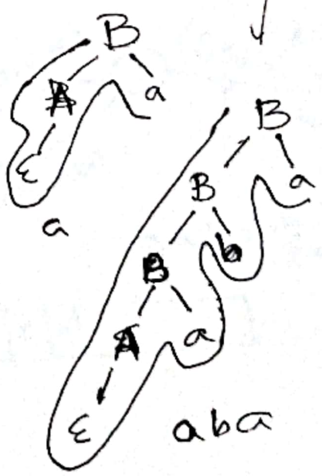


⇓C

RLG

⇓R

LLG.



L

↓R

L^R

↓R

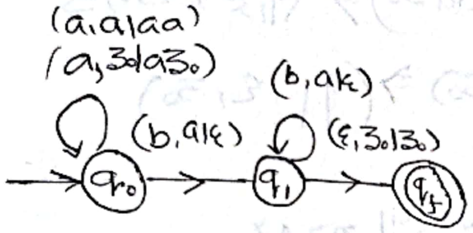
$(L^R)^R$

↓

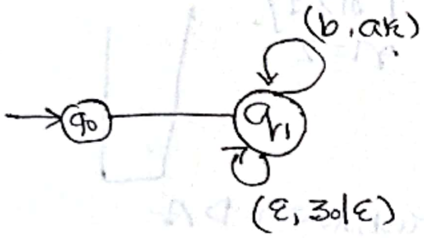
L

Date: 19. 1. 25

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



Halt using finishing state.

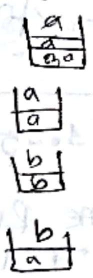


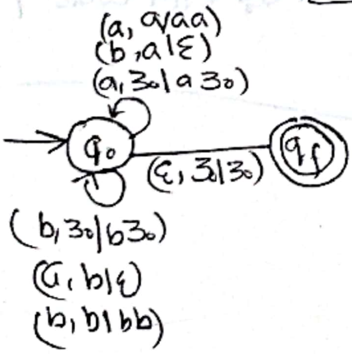
Halt using emptying stack

* PDA can successfully halt with/without final state.

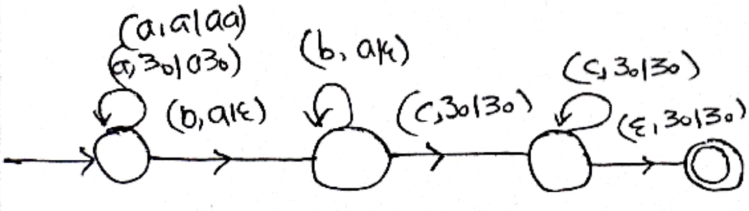
* $L = \{w \in \{a,b\}^* \mid$ such that $n_a(w) = n_b(w)\}$.

abab
baba
aa bb
bbaa
abba
baab.

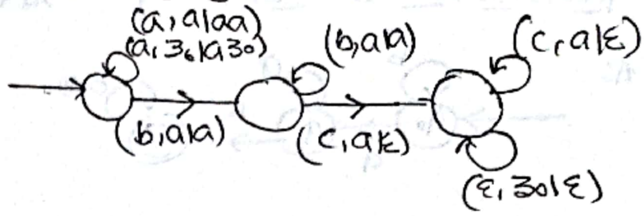




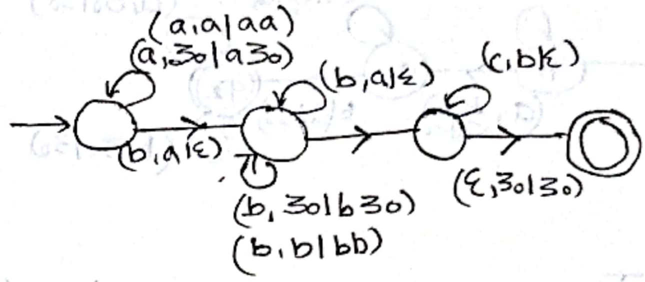
* $L = a^n b^n c^m$.



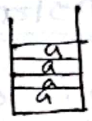
* $L = a^n b^m c^n$



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



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



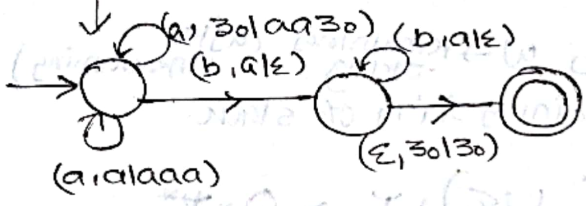
aabbbb

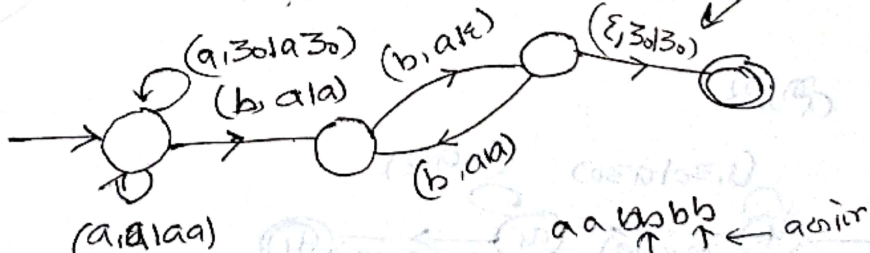
একটা a এর জন্য দুইবার a push হয়

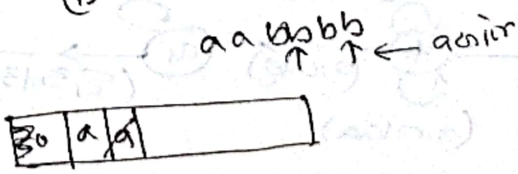
একটি করে a রাখলাম।

↳ first a কে ignore করলাম.

and b এর জন্য একটি করে a pop করলাম।

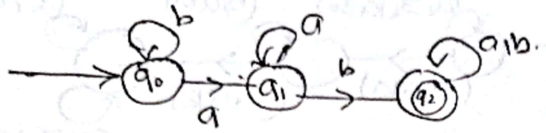




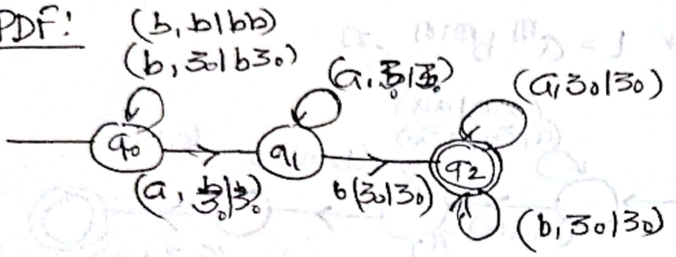


{ab, aab, aba, bab, ...} ← containing ab

DFA:



PDF:



Instantaneous Description: (ID):

ID is a formal notation of how a PDA computes strings and makes a decision whether the string is accepted or rejected.

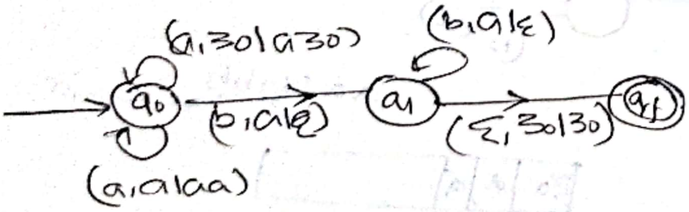
Triple (q, ω, α)

$q \rightarrow$ state, $\omega \rightarrow$ remaining string (after consumming)

$\alpha \rightarrow$ remaining string of stack.

$\delta : Q \times (\Sigma \cup \epsilon) \times \Gamma \rightarrow Q \times \Gamma^{*}$

$a^n b^n$



$(q_0, a, z_0) = (q_0, az_0)$

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

$(q_0, b, a) = (q_0, az_0)$

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

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

(accepted)

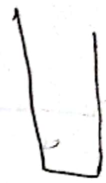
$(q_0, aabbz_0) \Rightarrow (q_0, abb, az_0) \Rightarrow$

$(q_0, bb, aaz_0) \Rightarrow (q_1, b, az_0) \Rightarrow$

$(q_1, \epsilon, z_0) \Rightarrow (q_f, \epsilon, z_0)$

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

$n = 2$



CFL

CSL → Pushdown DA

দিয়ে Handle করা যায় না

TM:

$L_1 = a^n b^n$

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

~~L~~

$n_a(\omega) = n_b(b)$

20.1.25 → 10:30 এর

CT

~~Moore me~~ RE এর পর থেকে

TM এর আগ পর্যন্ত।

12.01.25

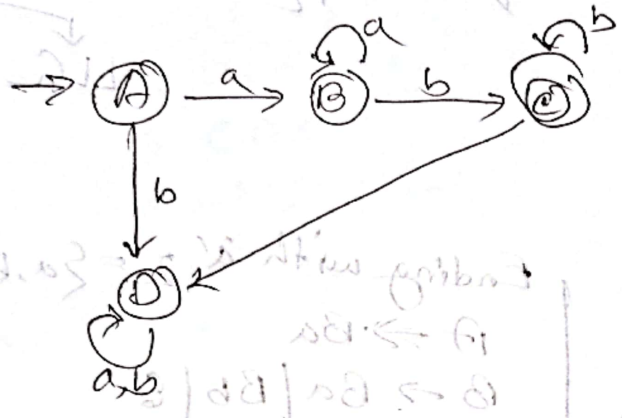
* Is FA able to accept every language?

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

$= \{ab, aabb, \ldots\}$

Push down Automata

FA + Infinite mem (stack) = ~~FA~~ PDA



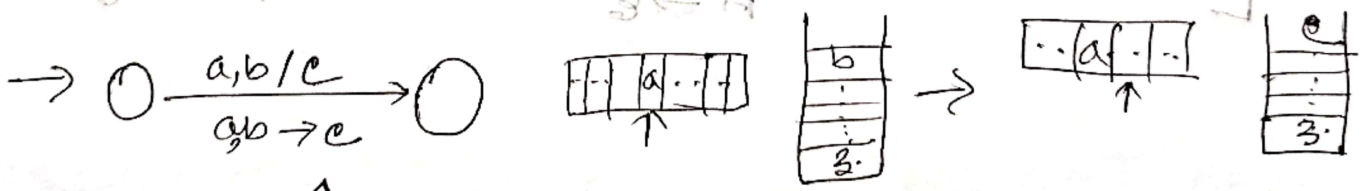
Q = Finite set of states
Σ = " " " symbols
q_0 = initial state $\in Q$
F = Final state $\subseteq Q$
Z_0 = initial symbol of stack
Γ = push down symbol

PDA can be defined with 7 tupple $M = (Q, \Sigma, \delta, q_0, F, Z_0, \Gamma)$

~~Q = fin~~ $\delta : Q \times \{\Sigma \cup \varepsilon\} \times \Gamma \to Q \times \Gamma^*$

$\delta : Q \times \{\Sigma \cup \varepsilon\} \times \Gamma \to 2^{Q \times \Gamma^*}$

PDA → Deterministic / Non Deterministic



Operations
- replace
- push
- pop
- no change
- Automatic halt

a, ε → c
or, a, f → fc
push

a, b → ε
pop

a, b → b
a, ε → ε
no chg

ε, ε → ε
Automatic halt

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