TOC — Theory of Computation

Time complexity is of 2 types:-

1 ;	2^n
$\log_2 n$;	
n ;	$n!$
$n \log_2 n$;	n^n
n^2	

Polynomial

P-class

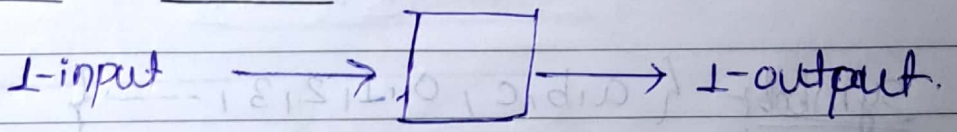
Polynomial class

Exponential

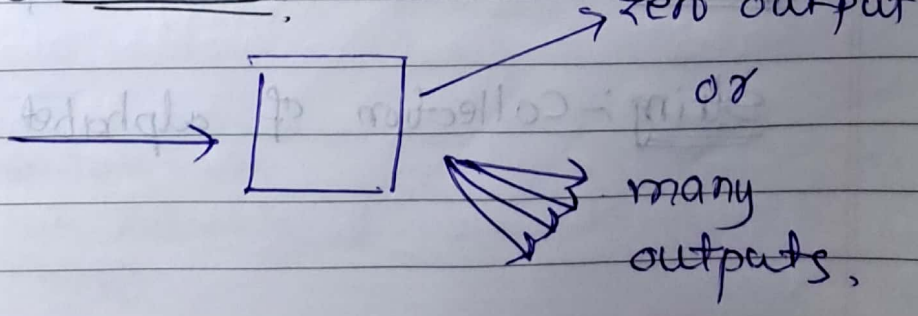
NP-class

Non deterministic polynomial

Deterministic Machine :-



Non-deterministic Machine :-



Theorem :- If $x \in NP$ can be solved then $y \in NP$ can also be solved.

Reason :- A NP problem can be mapped to another NP problem.

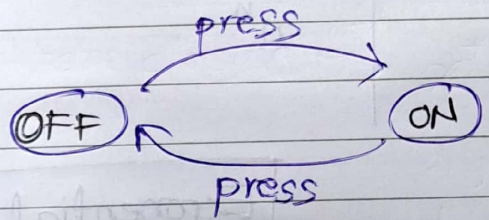
$$x \leq_p y$$

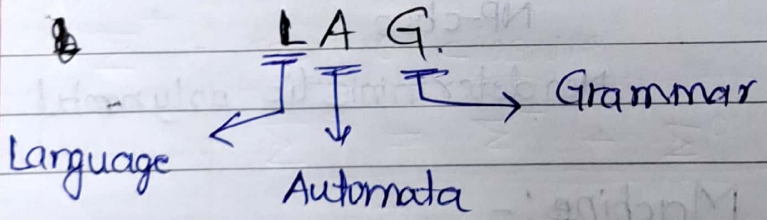
Q] Million Dollar Question. Is P=NP or not?

Books :- Peter Lynch

∴ States, Transitions :-

Take fan as simple machine with 2 states :-





symbol :- { a, b, c, 0, 1, 2, 3, ---- }

alphabet :- Σ (a, b) finite set of symbols.

string :- Collection of alphabet, sequence

Grammar :- Grammar G is defined as quadruple.

$G = \{ V, T, P, S \}$.

- V → variable
- T → Terminal
- P → Production Rule
- S → start

Language Definition :- collection of strings.

English is a language made up of alphabets (a, b, c ... z) their combinations, repetitions and also symbols like (.), (,), (" "), (/), (;), etc. and spaces.

similarly L is Language of given no. of alphabets & symbols

Note:- $ab \neq ba$

Σ - is set of alphabets.

ex:- if $\Sigma (a, b)$

then Language :- $\{a, b\}$ → Length 1

$\{aa, bb, ab, ba\}$ → Length 2.

$\{aaa, aab, aba, abb, baa, bab, bba, bbb\}$ → length 3

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

Language

- finite length
- infinite length

Automata:- Model / machine that tells whether a string belongs to a language.

Types of Automata

- Finite Automata (FA)
  - Deterministic Finite Automata (DFA)
  - Non-Deterministic finite Automata (NFA)
- Push Down Automata (PDA)
- Linear Bounded Automata (LBA)
- Turing Machine (TM)

Finite Automata

- Finite Automata without output
  - DFA
  - NFA
  - ϵ-NFA
- Finite Automata with output
  - Moore machine
  - Mealy Machine

Power of Σ :- $\Sigma = \{a, b\}$

Σ^0 - set of all strings with Length zero = $\{\lambda, \varepsilon, \wedge\}$ Null string.

Σ^1 = set of all strings with Length one = a, b.

Σ^2 = set of all strings with length two = aa, bb, ab, ba

Σ^3 = —"— Length three

$\{a,b\}\{a,b\}\{a,b\} = \{aaa, aba, aab, baa, bba, bab, abb, bbb\}$

Σ^* :- (kleene closure) x^* - means Language with repetition of x any no. of times. [zero no. of times also]

Σ^+ :- positive closure

$$\Sigma^* = \Sigma^+ + \Sigma^0$$

$$\therefore \Sigma^+ = \Sigma^* - \Sigma^0$$

ex:-

$a^* = \{\_\_, a, aa, aaa, \ldots\}$

$(ab)^* = \{\_\_, ab, abab, ababab, \ldots\}$

$(aba)^* = \{\_\_, aba, abaaba, abaabaaba, \ldots\}$

$(a+b)^* = \{\_\_, a, b, aa, bb, ab, ba, aab, bab, \ldots\}$

↳ choose a or b

* Finite Automata.

[a|b|a|b|b|a] ---

Finite control

No extra space
We cannot memorise anything that was previous alphabet.

* Deterministic Finite Automata (DFA) :-

DFA $(Q, \Sigma, \delta, q_0, F)$

ek hi state pe jayega
most rigid machine without zero flexibility.

Q :- Finite set of states (states should be countable)
Σ :- Finite set of alphabets.
δ :- Transition function.
q_0 :- Initial state
F :- Finite set of final states.

$q_0 \in Q$
$F \subseteq Q$

$\delta \Rightarrow$ δ is a function.
$\Rightarrow$ $\delta : Q \times \Sigma \longrightarrow Q$
$\Rightarrow$ on a state, read some symbol & land on a state.

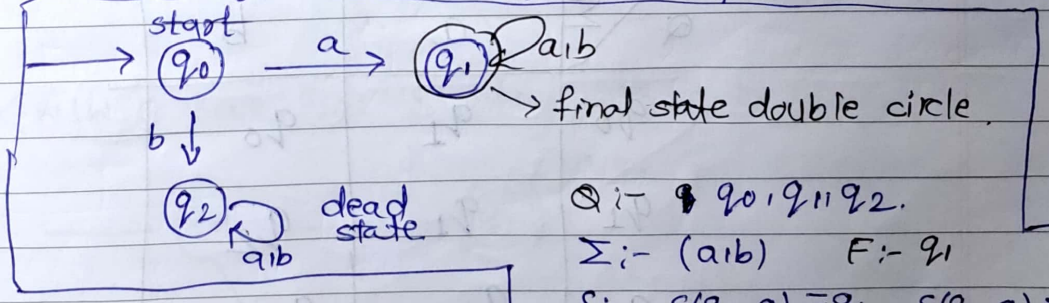
ex:- $\delta(q_0, a) = q_1$ $\quad (q_0) \xrightarrow{a} (q_1)$

** On every state, look for every symbol → DFA.

Q) strings starting with a should be accepted.

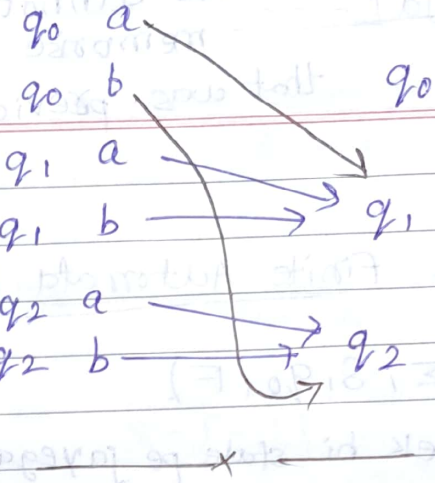
ex:- {a, aa, b, abb, baaa, bb, ba, abab, ---}
(b, bb, ba, baaa: x)

minimum string के लिए design करना पहले – (for easiness)



Q :- $\{q_0, q_1, q_2\}$.
Σ :- (a|b) F :- q_1
δ :- $\delta(q_0, a) = q_1$ $\quad \delta(q_2, a) = q_2$
$\delta(q_0, b) = q_2$ $\quad \delta(q_2, b) = q_2$
$\delta(q_1, a) = q_1$
$\delta(q_1, b) = q_1$

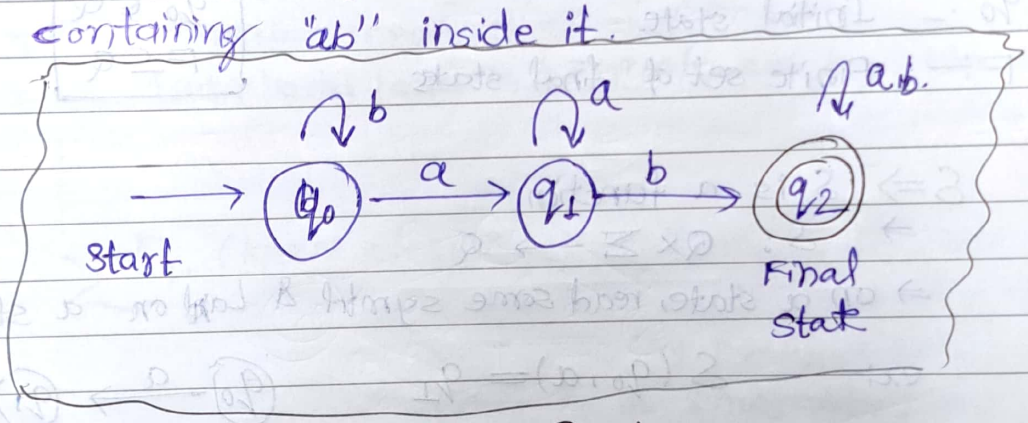
δ function domain codomain



Q.1) $L = \langle w \mid w \in (a,b)^*, ab \text{ anywhere} \rangle$

ans:- Language consists of w, such that w (belongs to) w is a string of any length made of a & b. containing "ab" inside it.

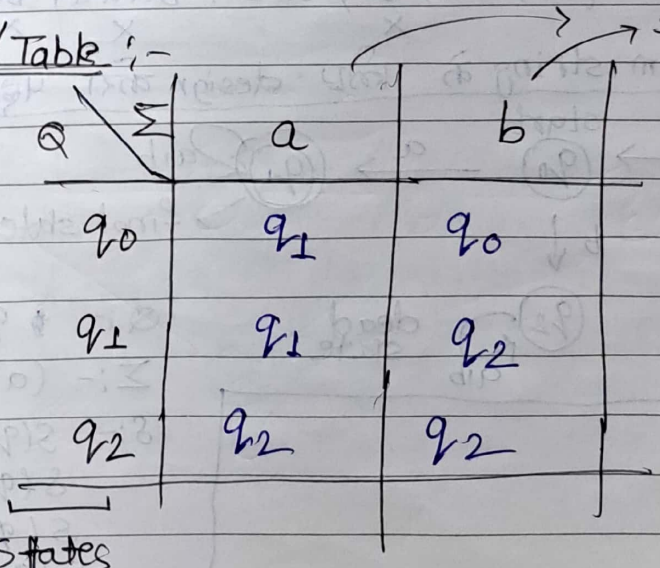
ex:- ab
ex:- bab
ex:- bbab
ex:- bbaaba



Transition Graph.

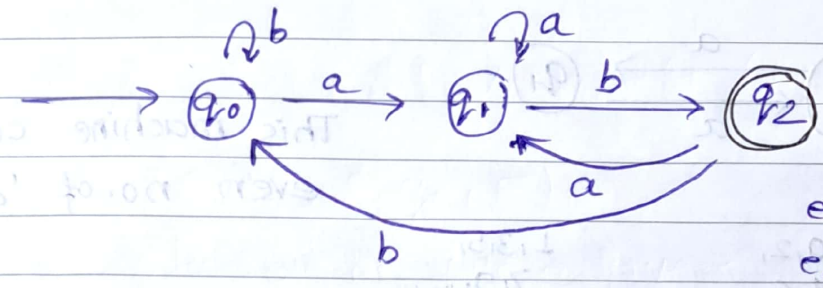
If after completing transitions, we are on final state, then string is accepted.

Transition / Table :-



Q.2) $L = \{w \mid w \in (a,b)^*$, ends with ab.$\}$

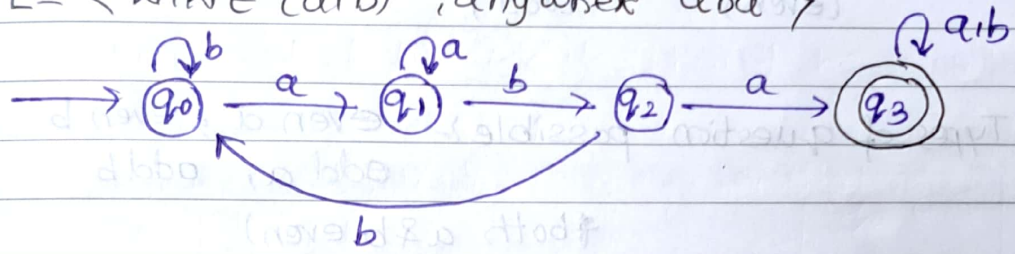
ans:-



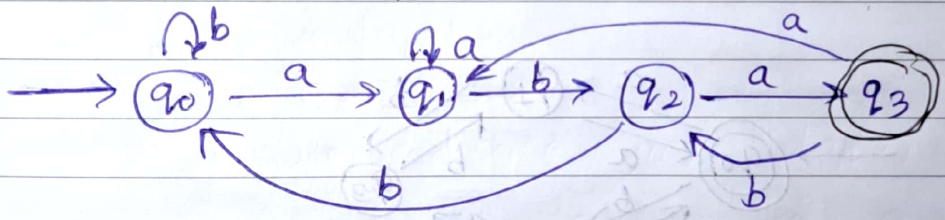
ex:- ab
ex:- bab
ex:- baab
ex:- abab
ex:- abaabab

Q.3) $L = \{w \mid w \in (a,b)^*$, anywhere aba$\}$

Sol:-

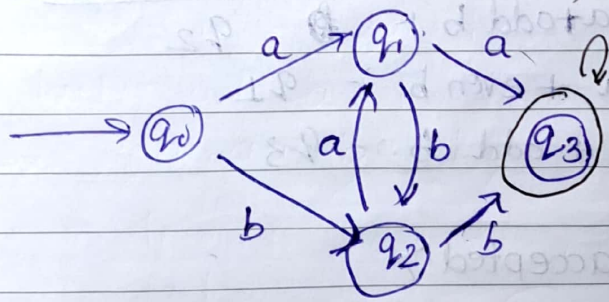


Q.4) $L = \{w \mid w \in (a,b)^*$, ends with abd$\}$



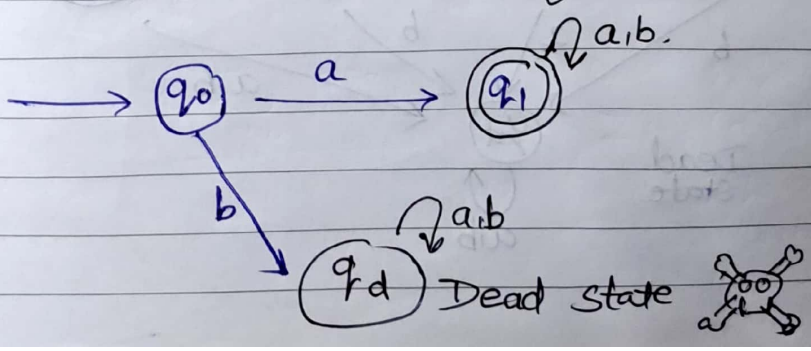
Q.5) $L = \{w \mid w \in (a,b)^*$, aa or bb$\}$

Sol:-



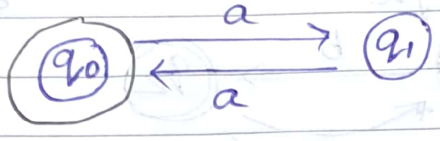
Q.6) $L = \{w \mid w \in (a,b)^*$: Starting with a.$\}$

ans:-



Q.7) $L = \{w \mid w \in (a,b)^*$, No. of a should be even$\}$

ans:-



This machine counts even no. of 'a'.

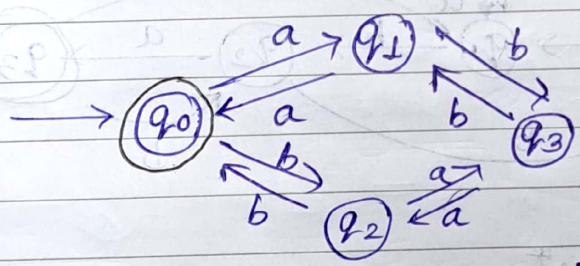
No. of a :- 0, 2, 4, 6, 8, 10, ... (even)

1, 3, 5, 7, 9, ... (odd).

4 Types of question possible :- even a, even b
odd a, odd b
* both a & b even)
* both a & b odd.

Q.8) $L = \{w \mid w \in (a,b)^*$ no. of a is even and no. of b even$\}$

ans:-



Final state

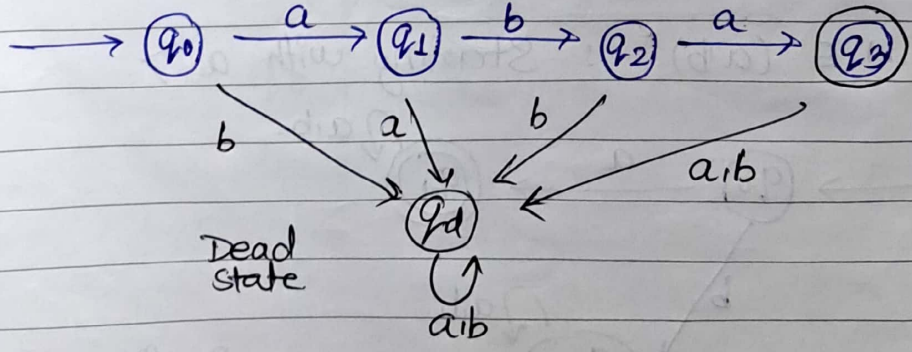
Slight changes :- Even a + odd b :- q_2

Odd a + even b :- q_1

odd a + odd b :- q_3

Q.9) $L = \{aba$ only accepted$\}$

ans:-

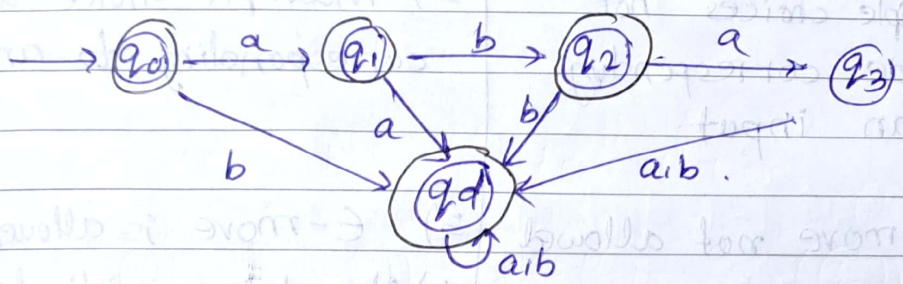


DFA works on complement & union :-

Q.10) $L = \langle w | w \in (a,b)^* - \{aba\} \rangle$

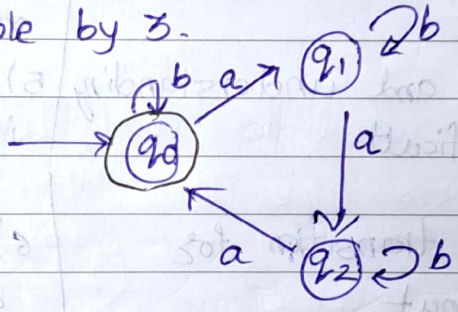
ans:- complement of Q.9.

Rule:- Final state becomes non-final.
Non-final state becomes final state.



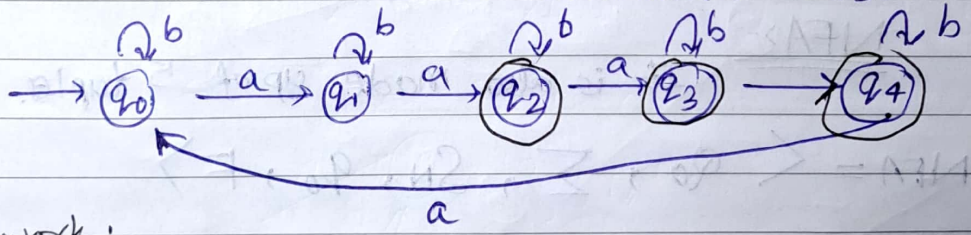
Q.11) $L = \langle w | w \in (a,b)^* \text{ and } n_a(w) \bmod 3 = 0 \rangle$

Sol:- No. of a is divisible by 3.



Q.12) $n_a(w) \bmod 5 > 1$ means remainder 2,3,4.

ans:-



Homework :-

Q.13) $L = \langle w | w \in (a,b)^*$, third symbol from end should always a $\rangle$.

Q.14) $L = \langle w | w \bmod 5 = 0 \text{ , } w \in (0,1)^* \rangle$

—— x ——

Non-Deterministic Finite Automata (NDFA or NFA)

DFA	N-DFA
1) Dead configuration is not allowed.	1) Dead configuration is allowed.
2) Multiple choices not available corresponding to an input.	2) multiple choice available corresponding to an input.
3) ϵ-move not allowed	3) ϵ-move is allowed.
4) Digital computers are deterministic.	4) Non-deterministic feature is not associated with real computers.
5) designing and understanding is difficult.	5) design and understanding is easy.
6) Denote transition for every input	6) Not needeled to show all transitions

NFA:- It is also made up of 5 tuple.

$$NFA = \langle Q_0, \Sigma, \delta_N, q_0, F \rangle$$

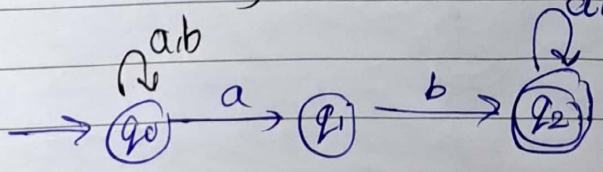
$$\delta_D \Rightarrow Q_0 \times \Sigma \longrightarrow Q_D$$

$$\delta_N \Rightarrow Q_N \times \Sigma \longrightarrow P(Q_N)$$

DFA $\subseteq$ NFA

Hence previously drawn DFA are also NFA.

Q) $L = \langle ab \text{ anywhere} \rangle$



δ / Σ	a	b
q_0	q_1	ϕ
q_1	ϕ	q_2
q_2	ϕ	ϕ

Homework :-
Answers

Q) 3rd symbol from end should be 'a'.

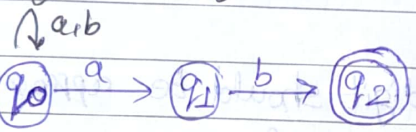
ans:- Acceptible strings :- aaa, aaab, ababb
Not acceptible :- ababba.

Name states as triplet, because last 3 letters can be :-
aaa, aab, aba, abb
bbb, bba, bab, baa
] 8 states possible

Note: You can start from any state.

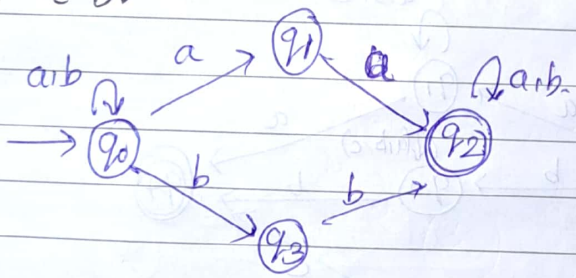
Q] String ending with ab.

ans:-



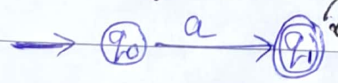
Q] aa or bb.

ans:-



Q] String starting with a.

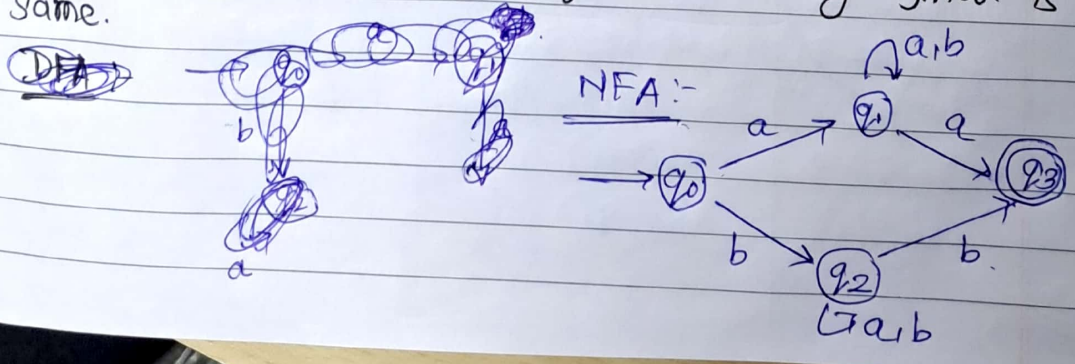
ans:-



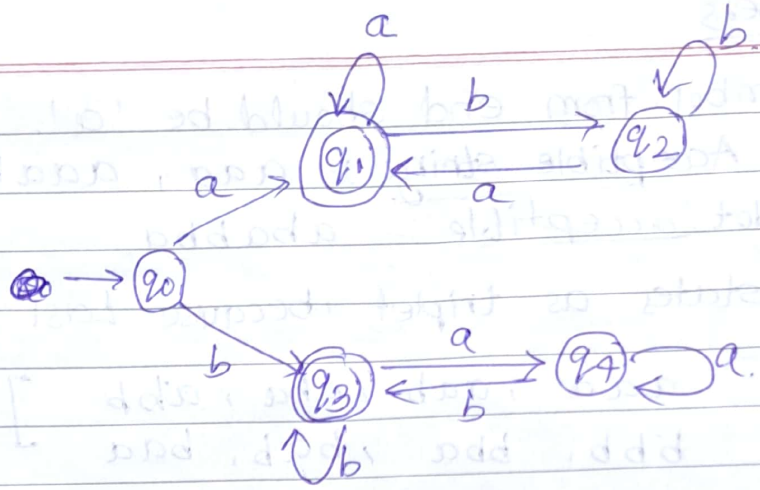
Q] No. of a even / odd → DFA, NFA same.

Q] DFA of (a,b)* where starting and ending symbol is same.

ans:-

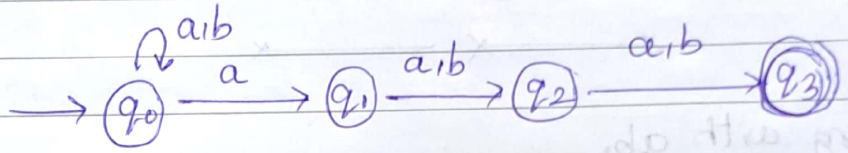


DFA:-



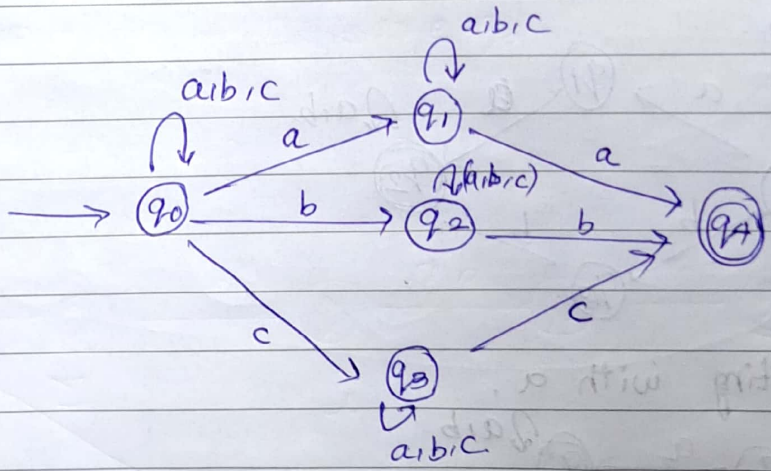
Q] NFA of 3rd Last digit should be a.

ans:-



Q] $(a|b|c)^*$ where Last digit should be appear atleast one more time. Draw NFA?

ans:-



Algorithm for NFA → DFA conversion

$NFA \Rightarrow \langle Q_N, \Sigma, \delta_N, q_0, F_N \rangle$

$DFA \Rightarrow \langle Q_D, \Sigma, \delta_D, q_0, F_D \rangle$

Step I) Initialise $Q_0 = \{q_0\}$

Step II) Read all symbols from Σ over Q_N and generate newer states (Also read Σ over new states).

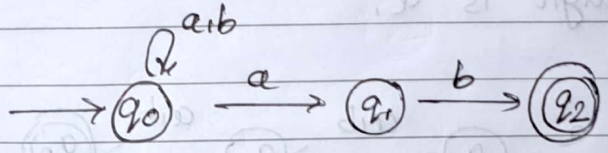
$$q \cup Q_D \longrightarrow Q_D$$

Step III) Repeat step-II until all transitions over Q_D are not covered.

Step IV :- F_D is all those states where F_N is present in it.

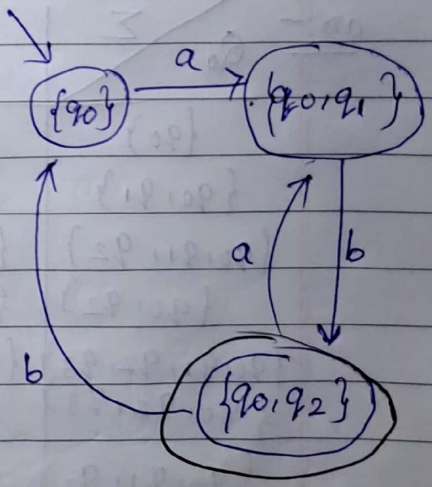
Examples:-

Q] NFA :-



ans:- Transition state

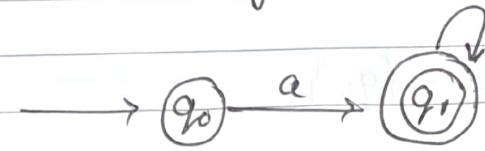
new state	Q_0 \ q	a	b
	$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
	$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$



q_2 was final state in NFA

∴ All states containing q_2 are final in DFA.

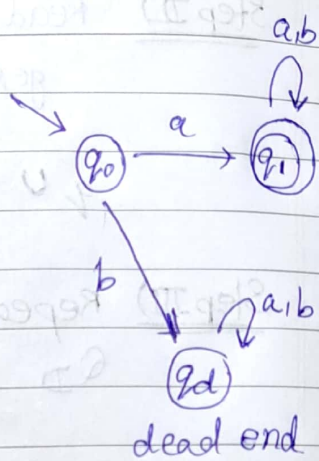
Q] string starts always with 'a'.



ans:-

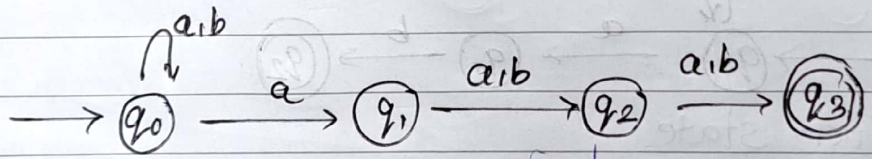
Q / Σ	a	b
q_0	$\{q_1\}$	$\phi \rightarrow \{q_d\}$
q_1	$\{q_1\}$	$\{q_1\}$
q_d	$\{q_d\}$	$\{q_d\}$

q_d — dead state



Replace all ϕ by same q_d.

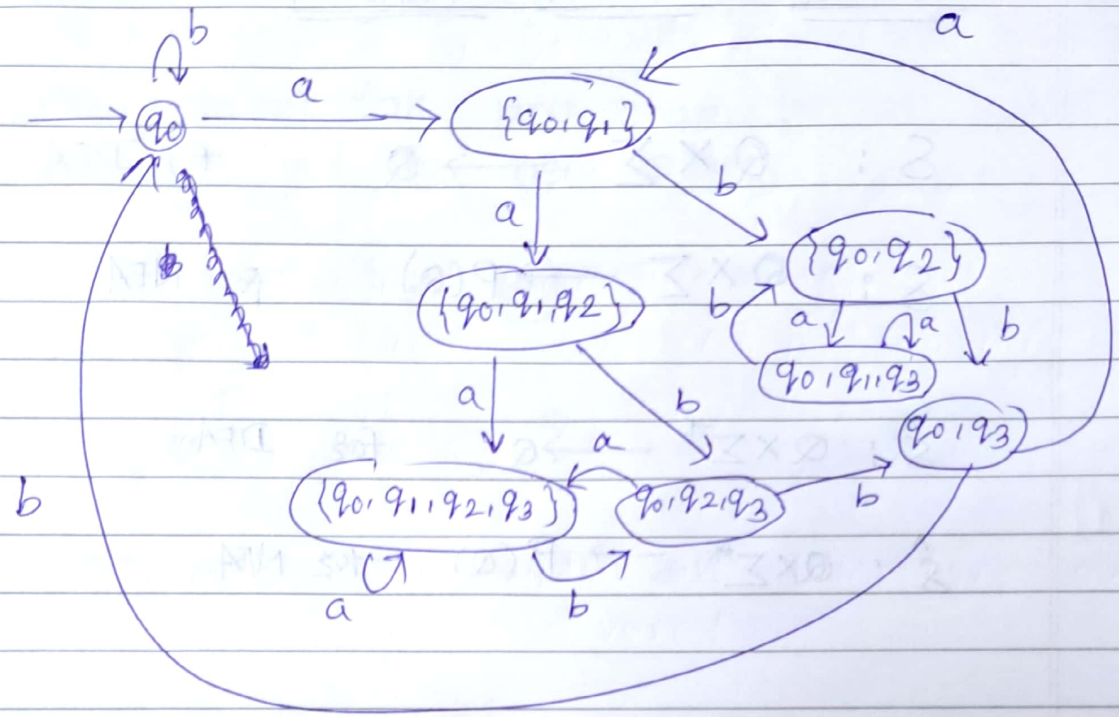
Q] Third Last digit is a.



ans:-

Q0 / Σ	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_2, q_3\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0, q_3\}$
$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_2, q_3\}$
$\{q_0, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_3\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1, q_3\}$	$\{q_0, q_2\}$
$\{q_0, q_3\}$	$\{q_0, q_1\}$	$\{q_0\}$

8 state


q0
{q0,q1}
{q0,q1,q2}
{q0,q2}
{q0,q1,q3}
{q0,q3}
{q0,q1,q2,q3}
{q0,q2,q3}
a
b

Start

Extended Transition Function :-

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

$\delta : Q \times \Sigma \longrightarrow P(Q)$ for NFA.

$\hat{\delta} : Q \times \Sigma^* \longrightarrow Q$ for DFA

$\hat{\delta} : Q \times \Sigma^* \longrightarrow P(Q)$ for NFA.

example

$\delta(q_0, a) = q_1$

$\hat{\delta}(q_0, abaab) = q_3$

$\hat{\delta}(q, a) = p$

$\hat{\delta}(q, aw) = \hat{\delta}(\delta(q,a), w)$

$\hat{\delta}(q, wa) = \delta(\hat{\delta}(q,w), a)$

$\hat{\delta}(q, \epsilon) = q$ (∵ not moved anywhere string of zero length)

$\hat{\delta}(q, w) = \bigcup_{p \text{ in } q} \hat{\delta}(p, w)$

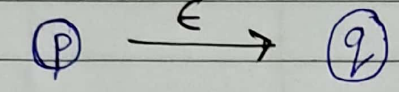
Here, If q represents set of states q_0, q_1, q_2.

w is string
Σ is alphabet
Σ^* is string

(ϵ-NFA)

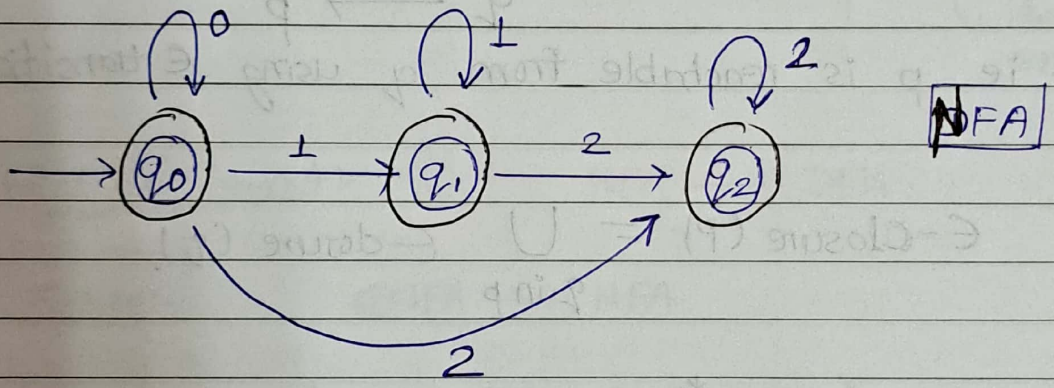
A more powerful machine :- (NFA with ϵ transitions)

- You do not take any character as input but still get to new state.



ex :- $0^i 1^j 2^k$; $i, j, k \geq 0$

String like 00 1111 222



Let find ϵ-NFA for above DFA :-

$$\epsilon\text{-NFA} = \langle Q, \Sigma, \delta, q_0, F \rangle$$

Transition function :- $\delta : Q \times \{\Sigma \cup \epsilon\} \longrightarrow P(Q)$

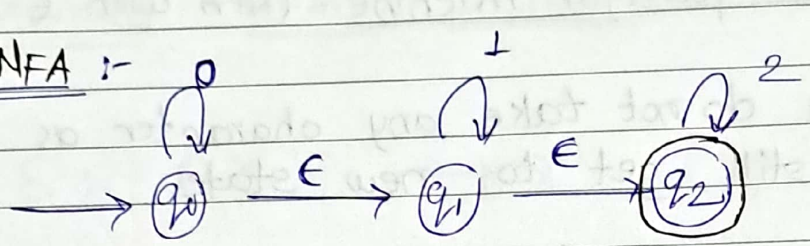
P(Q) : set of states

ϵϵ0ϵ1ϵϵϵ2
ex :- ϵ 0 1 2
ex :- 0 ϵ 1 ϵ 2
] Just working

⇓

012] Reality

∈-NFA :-



∈-closure Rules

Rule 1 ∈-closure (q) denotes all the states p such that there is a path from q to p, labelled by

$$q \xrightarrow{\in} p$$

ie p is reachable from q using ∈ transition.

Rule 2

$$\in\text{-Closure}(P) = \bigcup_{q \text{ in } p} \in\text{-closure}(q)$$

Rule 3 For $w \in \Sigma^*$ and $a \in \Sigma$
(w is string) (a is alphabet)

then

$$\hat{\delta}(q,wa) = \in\text{-closure}\left(\delta\left(\hat{\delta}(q,w),a\right)\right)$$

ex:- ∈-closure$(q_0) = \{q_0, q_1, q_2\}$
∈-closure$(q_1) = \{q_1, q_2\}$
∈-closure$(q_2) = \{q_2\}$.

Q \ (Σ ∪ ϵ)	0	1	2	ϵ
q_0	q_0	ϕ	ϕ	q_1
q_1	ϕ	q_1	ϕ	q_2
q_2	ϕ	ϕ	q_2	ϕ

↳ dead end.

What we learnt:- NFA ⟶ DFA

Future:- ϵNFA ⟶ NFA

$$NFA = \langle Q, \Sigma, \delta_N, q_0, F_N \rangle$$

$$\epsilon NFA = \langle Q, \Sigma, \delta_\epsilon, q_0, F_\epsilon \rangle$$

Steps:-

- Find ϵ-closure of a state
- Read an alphabet on ϵ-closure & take union.

ex:- $q_0 \xrightarrow{0} q_0$

$q_0 \xrightarrow{0} \xrightarrow{\epsilon} q_1$

$q_0 \xrightarrow{0} \xrightarrow{\epsilon} \xrightarrow{\epsilon} q_2$

$(q_0, q_1, q_2) \xrightarrow{0} [(q_0, q_1, q_2), \phi, \phi] \xrightarrow{\text{union}} [q_0, q_1, q_2]$

$(q_0, q_1, q_2) \xrightarrow{1} [(q_1, q_2), (q_1, q_2), (\phi)] \rightarrow [q_1, q_2]$

* conversion Transition table:-

Q / Σ	0	1	2
q_0	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
q_1	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
q_2	ϕ	ϕ	$\{q_2\}$

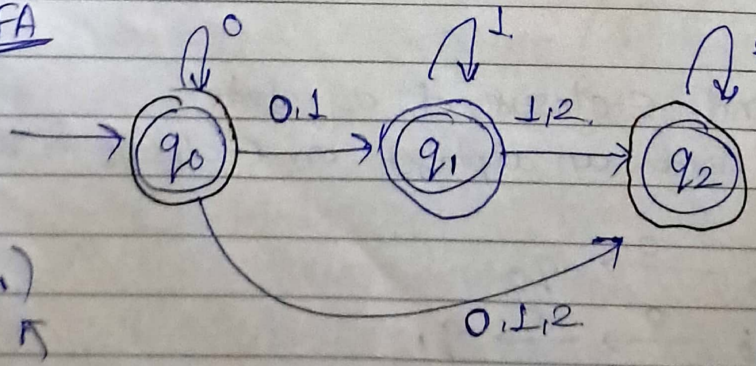
error

* Initial state will be added only under a condition:-

If ϵ-closure of q_0 (initial) state includes any ⊥ F-ϵ states.

ie You can directly jump from q_0 to q_2

NFA



(here q_0 & q_1)

(in ϵ-NFA, ie. q_2)

Any state which can reach final state via ϵ moves, will be final state, in NFA.

∴ q_0, q_1 can reach q_2, they will be final state.

Algorithm to convert ϵ-NFA $\longrightarrow$ DFA :-

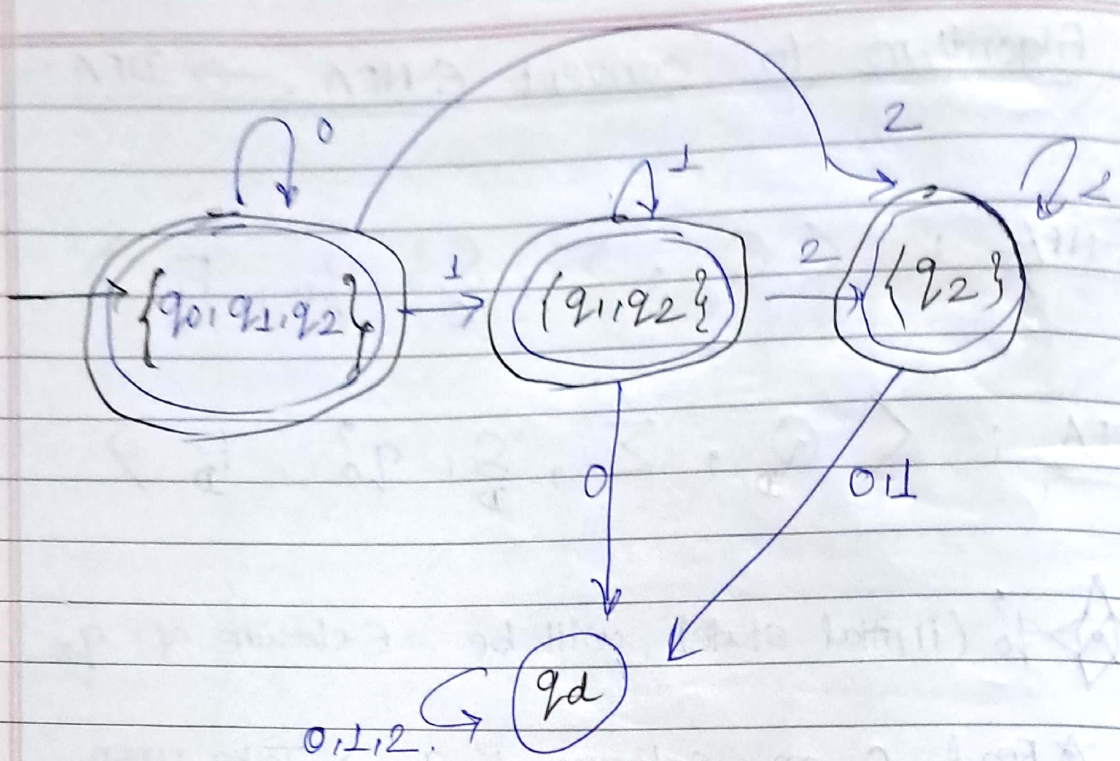
ϵ-NFA :- $\langle Q_\epsilon, \Sigma, \delta_\epsilon, q_0, F_\epsilon \rangle$

DFA :- $\langle Q_D, \Sigma, \delta_D, q_0', F_D \rangle$

q_0' (initial state) will be ϵ-closure of q_0

★ Read 0 on ϵclosure of q_0 $\Rightarrow$ Take union.
—11— 1 on —11— $\Rightarrow$ Take union.
—11— 2 on —11— $\Rightarrow$ Take union.

	0	1	2
ϵclosure q_0 $\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
ϵ-closure q_1 $\{q_1, q_2\}$	q_d	$\{q_1, q_2\}$	$\{q_2\}$
Eclosure q_2 $\{q_2\}$	q_d	q_d	$\{q_2\}$
Extra q_d	q_d	q_d	q_d



- ϵ is allowed to be an answer
  Hence $\{q_0, q_1, q_2\}$ can be final state

- only 1 allowed ⇒ $\{q_1, q_2\}$ also Final.
- only 2 allowed ⇒ $\{q_2\}$ also final.

- All final states of NFA, if present in DFA, will become final state in DFA.
  ex:- q_0, q_1, q_2 was final in NFA
  a) If any one of them present in DFA, it becomes final state.

OR

  b) q_2 was final in ϵ-NFA, any state containing q_2 will be final, ex:- $\{q_0, q_1, q_2\}, \{q_1, q_2\}$

- Miley & Murrey machine control output of display] Homework for Today

So far :- 3 automata [DFA, NFA, ϵ-NFA] studied

since they are inter convertible, ∴ DFA = NFA = ϵ-NFA.

Can we have algorithm to minimize DFA?

ans :- YES

Minimization of finite Automata :-

1) 2 states q_1 and q_2 are equivalent $q_1 \equiv q_2$ if

$$\hat{\delta}(q_1, x) \;\&\; \hat{\delta}(q_2, x)$$

(it means reading same string on different states) are final states or both of them are Non-final states $\forall\, x \in \Sigma^*$.

2) 2 states q_1 and q_2 are k-equivalent if both $\hat{\delta}(q_1, x)$ & $\hat{\delta}(q_2, x)$ are final states or non-final states for all strings x of length atmost k.

ie $|x| \leq k$

Equivalent relation (Reflexive, symmetric Transitive)

say we have n-states

$$Q = \langle q_1, q_2, q_3, \cdots\cdots q_N \rangle$$

we can make partition of states into final & non-final groups, etc.

$\pi_0 \Rightarrow$ 0 character
$\pi_1 \Rightarrow$ 1 character
$\pi_2 \Rightarrow$ 2 character
⋮
$\pi_k \Rightarrow$ k character
↓
$\pi_{k+1} \Rightarrow (k+1)$ character $= \pi_k$

ie keep continuing the group until

$$\pi_{k+1} = \pi_k$$

k is the reduced numbes of states

Let see DFA of 8 states as example,

Q \ Σ	a	b
q_0	q_1	q_5
q_1	q_6	q_2
q_2	q_0	q_2
q_3	q_2	q_6
q_4	q_7	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

$\pi_0 = \langle Q_1^0, Q_2^0 \rangle$

$Q_1^0 = \{q_2\}$, $Q_2^0 = \{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}$

$\pi_1 = \langle Q_1', Q_2', Q_3', Q_4' \rangle$

$Q_1' = \{q_2\}$

$Q_2' = \{q_0, q_4, q_6\}$

$Q_3' = \{q_1, q_7\}$

$Q_4' = \{q_3, q_5\}$

$\pi_2 = \langle Q_1^2, Q_2^2, Q_3^2, Q_4^2 \rangle$

$Q_1^2 = \{q_2\}$

$Q_2^2 = \{q_0, q_4\}$

$Q_3^2 = \{q_6\}$

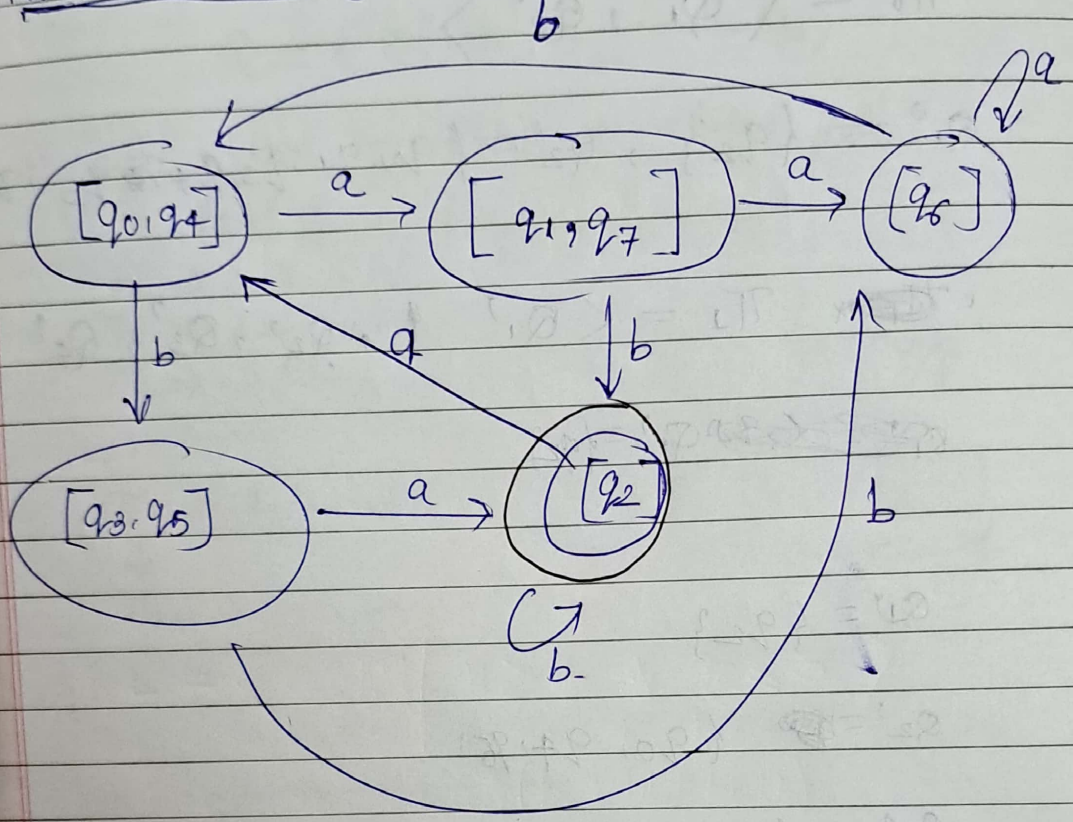
$Q_4^2 = \{q_1, q_7\}$

$Q_5^2 = \{q_3, q_5\}$

Now, π_3 will also have same part

Hence

Now, use this groups as minimized state



Regular Expressions

1) Σ
2) ϵ
3) (,)
4) *
5) +

$r_1 = a(a+b)^*$

$r_2 = (a+b)^* aba$

$r_3 = (a+b)^* aba (a+b)^*$

$r_4 = (a+b)^* (aa+bb) (a+b)^*$

$r_5 = a(a+b)^* b + b(a+b)^* a$

$r_6 = (a+b)^* a (a+b) (a+b)$

$r_7 =$ (even no. of a & even no. of b) — skip

Only aba selected.

$r_8 = aba$

$r_9 = \left((a+b)(a+b)(a+b)\right)^*$

Q] Language which should accept 1 'a' anywhere?

ans:- 1) $(a+b)^* a (a+b)^*$

ans 2) $b^* a b^* (a^* b^*)^*$

Q] Exactly 2 b's?

ans:- $a^* b a^* b a^*$

Q] $L = \{ a^n b^m \mid n \geq 4 \ \& \ m \leq 3 \}$

ans:- $aaaa\, a^* (b + bb + bbb + \epsilon)$

(or)

$aaaa\, a^* (b+\epsilon)(b+\epsilon)(b+\epsilon)$

Q] Strings composed of a and b; and at most 3 a.

ans:- $b^* (\epsilon + a) b^* (\epsilon + a) b^* (\epsilon + a) b^*$

Q] $L = \{ a^n b^m \mid n = \text{even},\ m \geq 0 \}$

ans:- $(aa)^* b^*$

Q] $L = \{ a^n b^m \mid n+m = \text{even} \}$

ans:- $a(aa)^* b (bb)^* + (aa)^* (bb)^*$ ~~#Regards~~

(or)

$(aa)^* (ab + \epsilon)(bb)^*$

Q] String composed of a, b ~~c~~. Atleast 1 occurence of each symbol should be there

ans:- $(a+b)^* a (a+b)^* b (a+b)^* + (a+b)^* b (a+b)^* a (a+b)^*$

or

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

Q] string composed of a, b & c, atleast 1 each occur.

$x^* a x^* c x^* b x^*$

$x^* a x^* b x^* c x^*$

$x^* b x^* a x^* c x^*$

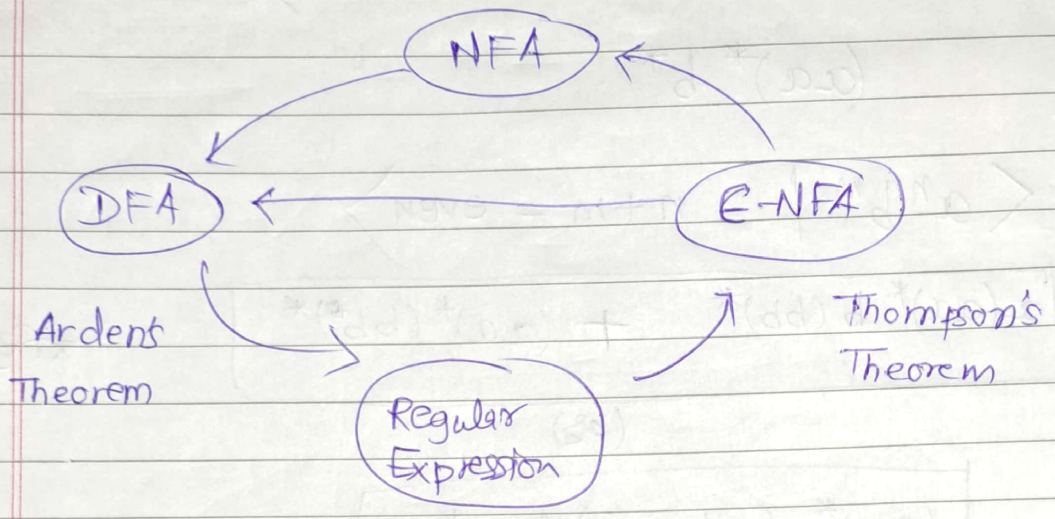
$x^* b x^* c x^* a x^*$

$x^* c x^* a x^* b x^*$

$x^* c x^* b x^* a x^*$

$\therefore$ ~~DFA < NFA < ϵ-NFA < Reg Expression~~ X wrong

$\therefore$ DFA = NFA = ϵ-NFA = Reg. Expression



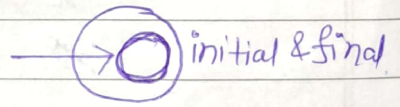
Thompson's Theorem

Let r be a regular expression.
Then there exist a ϵ-NFA which accepts $L(r)$

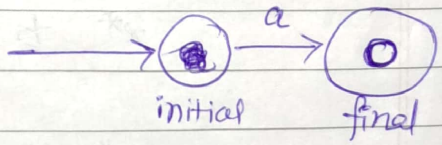
Proof :- Mathematical Induction on the no. of operators in Regular expressions are :-

1) There is ϵ-NFA having 1 final state and no transition goes out of this final state

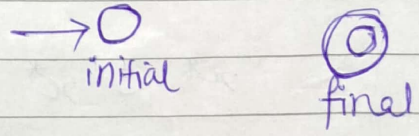
Basis :- $r_1 = \epsilon$



$r_2 = a$



$r_3 = \phi$



2) Now, we will increase operators 1 at a time.

Assume that the theorem is true for the regular expression with fewer than i operators ($i \geq 1$)

Let r have "i" operator, then there exist 3 cases.

case 1) $r = r_1 + r_2$

case 2) $r = r_1 r_2$

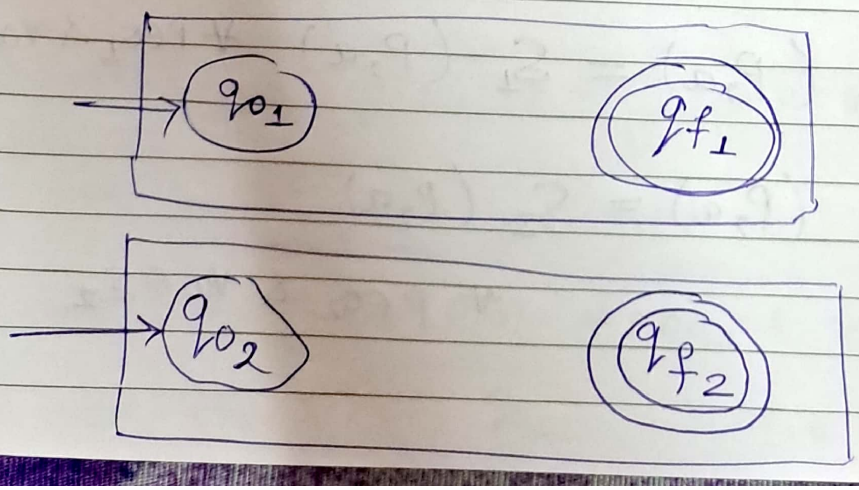
case 3) $r = r_1^*$

R_1 $L(R_1)$ M_1

R_2 $L(R_2)$ M_2

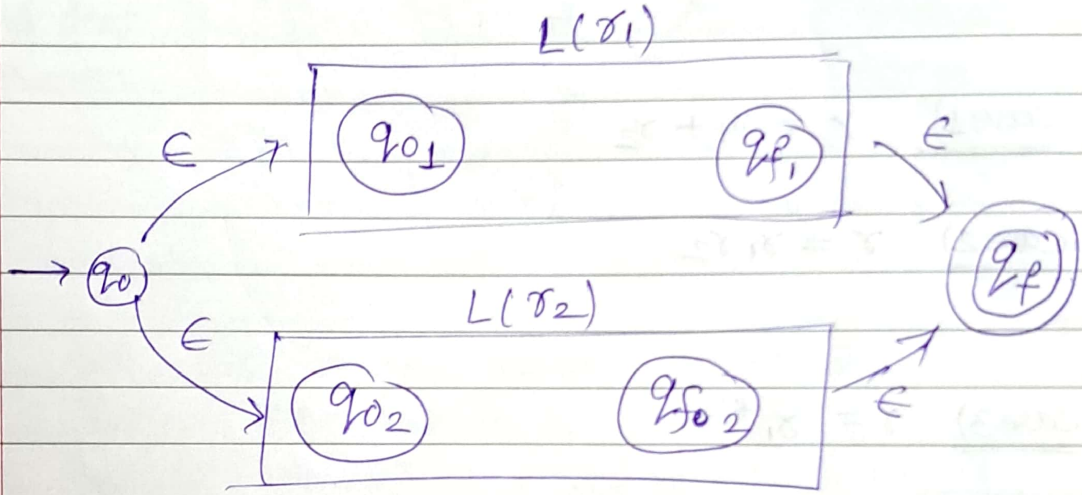
$M_1 = \langle Q_1, \Sigma_1, \delta_1, q_{0_1}, F_1 \rangle$

$M_2 = \langle Q_2, \Sigma_2, \delta_2, q_{0_2}, F_2 \rangle$



case 1) $r_1 + r_2$

$M_{union} = \langle Q_{union}, \Sigma_{union}, \delta_{union}, q_{0\,union}, F_{union} \rangle$



$Q_{union} = Q_1 \cup Q_2 \cup \{q_0, q_f\}$

$\Sigma_{union} = \Sigma_1 \cup \Sigma_2 \cup \{\epsilon\}$

$F_{union} = \{q_f\}$

1) $\delta_{union}(q_0, \epsilon) = \{q_{01}, q_{02}\}$

2) $\delta_{union}(P, a) = \delta_1(P, a) \quad \forall P \in Q_1 \& \forall a \in \Sigma_1$

3) $\delta_{union}(P, a) = \delta_2(P, a)$

$\forall P \in Q_2 \& \forall a \in \Sigma_2$

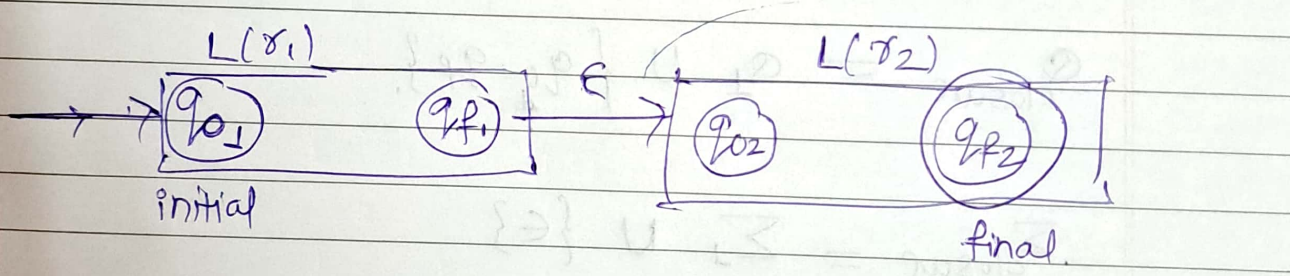
4) $\delta_{union}(q_{f_1}, \epsilon) = q_f \quad \forall\, q_{f_1} \in F_1$

5) $\delta_{union}(q_{f_2}, \epsilon) = q_f \quad \forall\, q_{f_2} \in F_2$

—— x ——

<u>case 2)</u> $\gamma_1\, \gamma_2$

$M_{concatenation} = \langle Q_{concat}, \Sigma_{concat}, \delta_{concat}, q0_{concat}, F_{concat} \rangle$



$Q_{concat} = Q_1 \cup Q_2$

$\Sigma_{concat} = \Sigma_1 \cup \Sigma_2 \cup \{\epsilon\}$

$F_{union} = F_2$

1) $\delta_{concat}(p, a) = \delta_1(p, a) \quad \forall\, p \in Q_1 \;\&\; \forall\, a \in \Sigma_1$

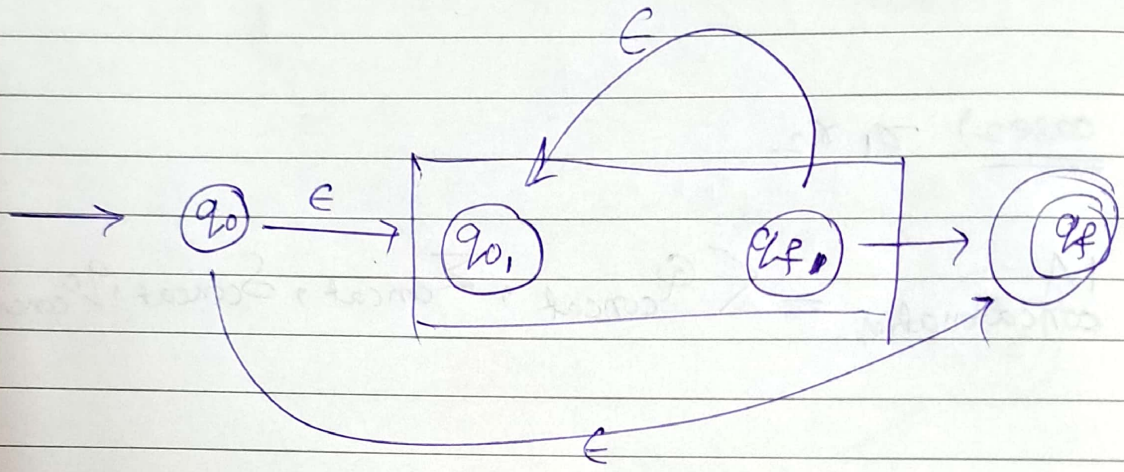
All transitions of both machines are retained.

2) $\delta_{concat}(p, a) = \delta_2(p, a) \quad \forall\, p \in Q_2 \;\&\; \forall\, a \in \Sigma_2$

3) $\delta_{concat}(q_{f_1}, \epsilon) = q0_2 \quad \forall\, q_{f_1} \in F_1$

Case 3. $\gamma = \gamma_1^*$

$$M_{closure} = (Q_{closure}, \Sigma_{closure}, \delta_{closure}, q_{0_{closure}}, F_{closure})$$



$Q_{closure} = Q_1 \cup \{q_0, q_f\}$.

$\Sigma_{closure} = \Sigma_1 \cup \{\epsilon\}$

$F_{closure} = \{q_f\}$

1) $\delta_{closure}(q_0, \epsilon) = \{q_{0_1}, q_f\}$

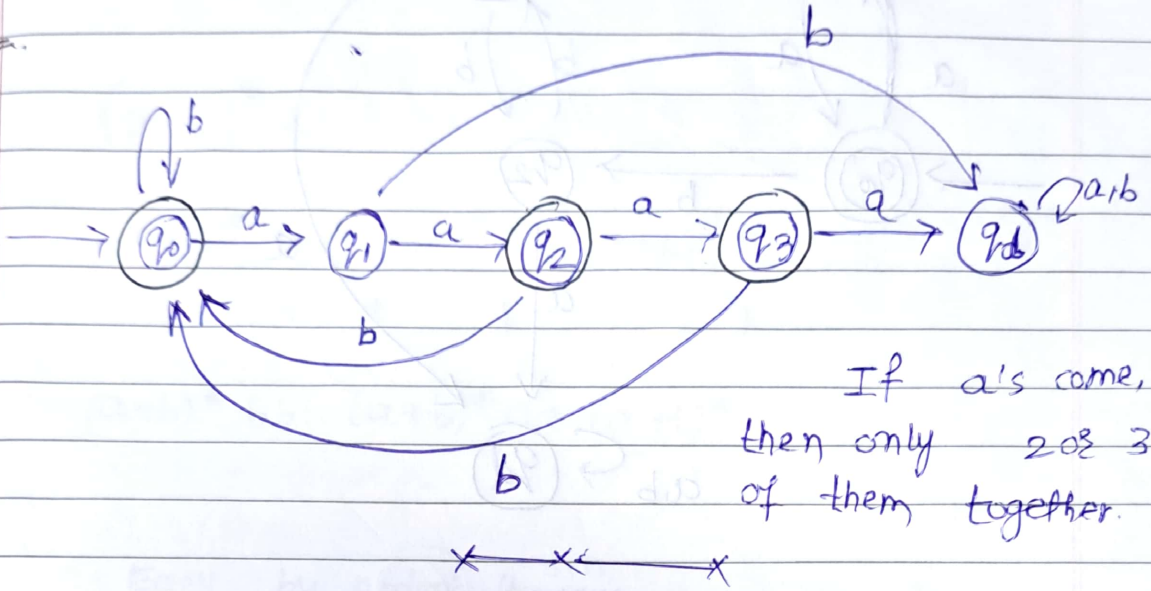
2) $\delta_{closure}(P, a) = \delta_1(P, a)$. $\forall P \in Q_1$ & $\forall a \in \Sigma_1$

3) $\delta_{closure}(q_{f_1}, \epsilon) = \{q_f, q_{0_1}\}$

* Test solutions *

Book- Peter Lynch

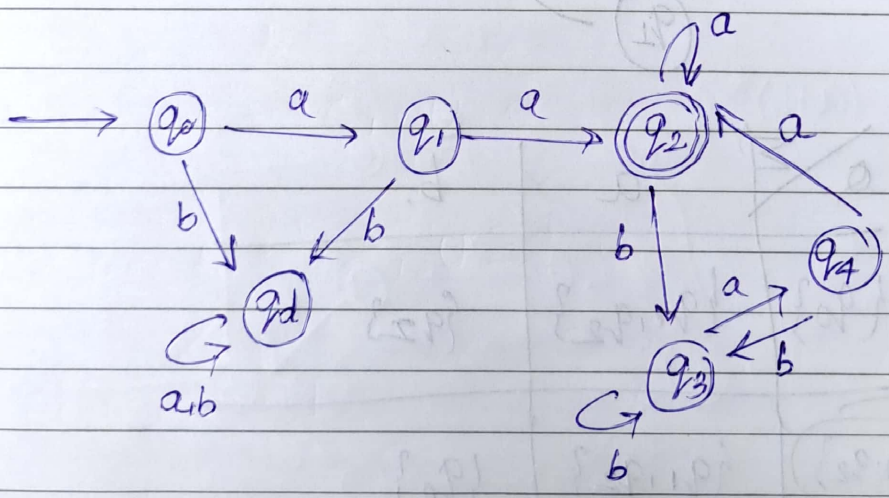
Q.1)



Q2) string should start and end with 'aa'

ans: acceptible strings:- aa, aaa.

starts & ends with 'aa'

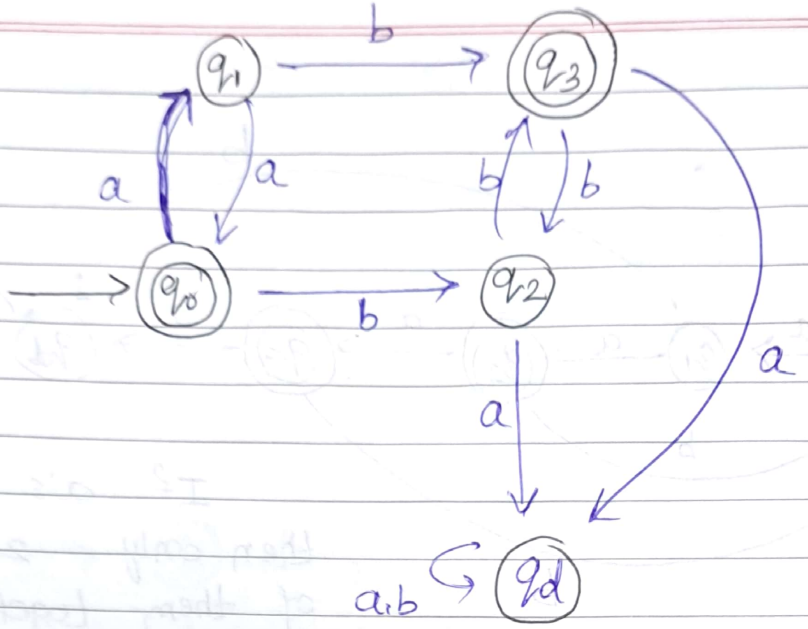


$r = aa(a+b)^*aa + aa + aaa.$

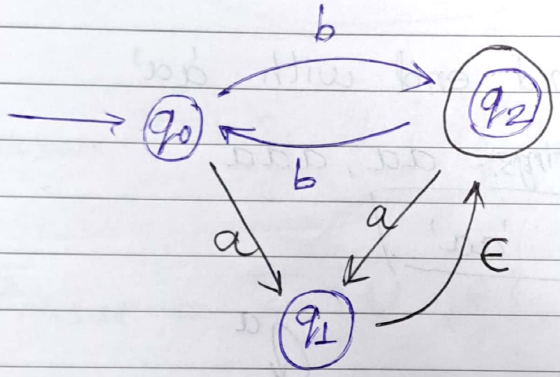
C, C++, Java ↑ most imp: DSA

Python ~~Python~~ STL problem

Q.3)



Q.4)



q / Σ	a	b
{q0}	{q1, q2}	{q2}
{q1, q2}	{q1, q2}	{q0}
{q2}	{q1, q2}	{q0}

TOC → directed graph

classmate
Date
Page 3

Q.5)

a) $(0+1)^* 1$

b) $(a+b)^* aa\ (a+b)^* bb\ (a+b)^*$

$+$

$(a+b)^* bb\ (a+b)^* aa\ (a+b)^*$

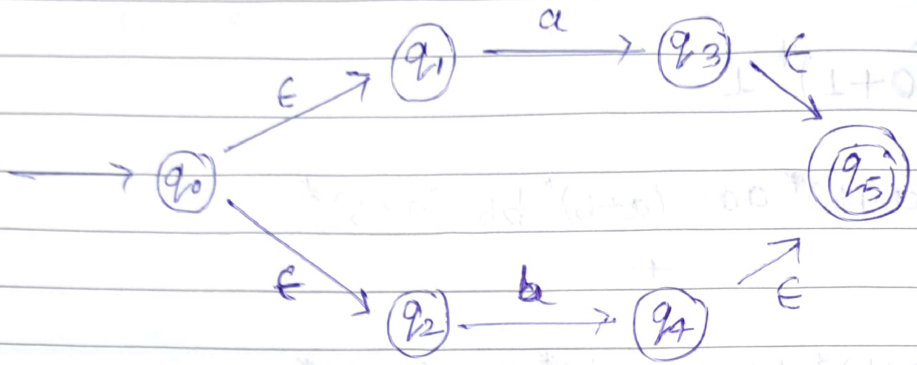
c) Easy by Ardon's theorem

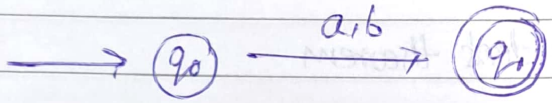
—x—

TOC
4

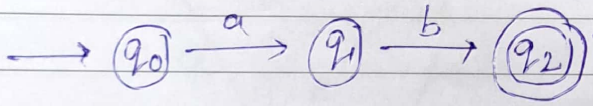
Conversions :- Regular Expression to ϵ-NFA.

$R_1 = a+b$

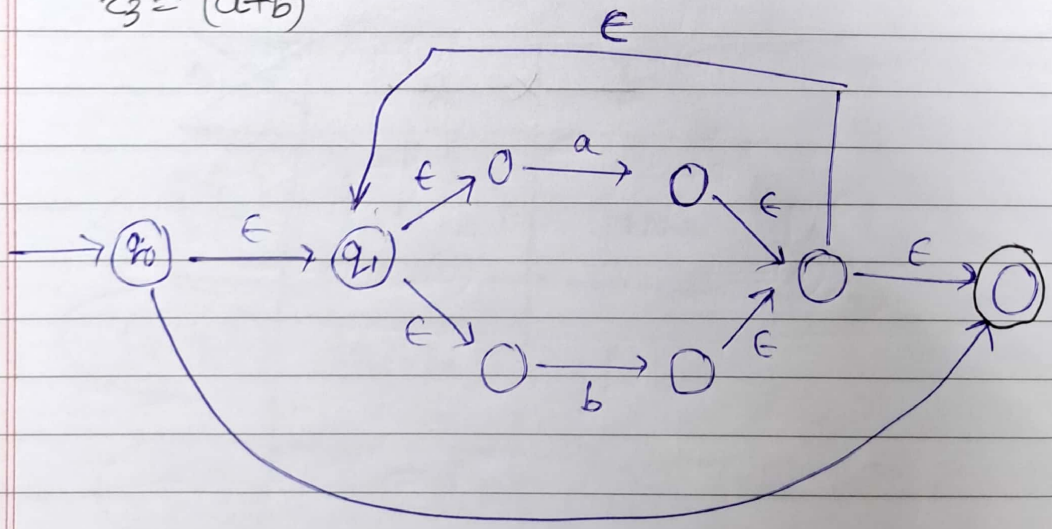


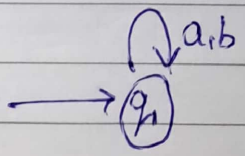


$R_2 = ab$

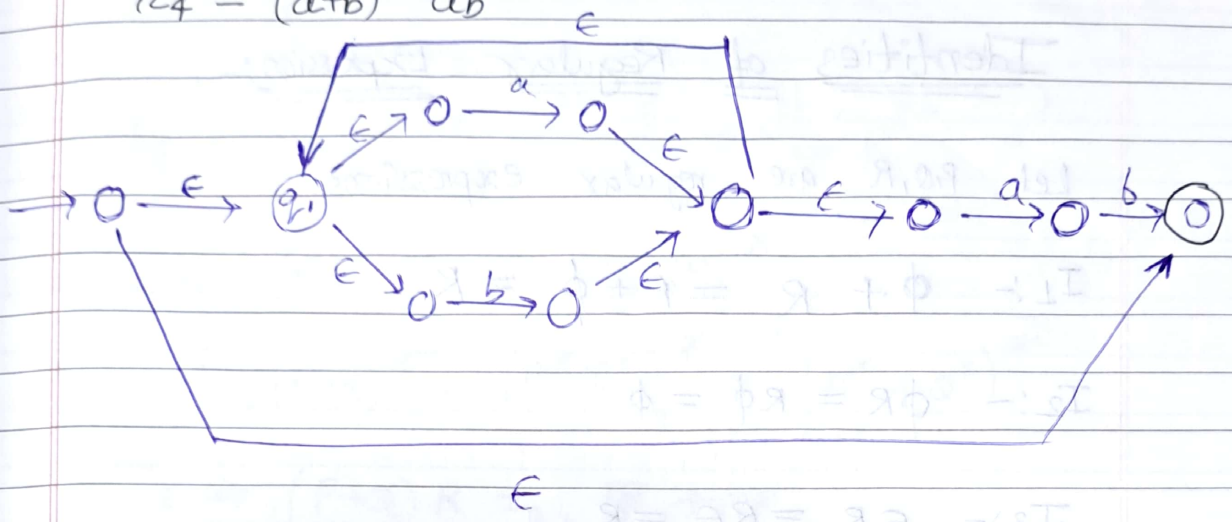


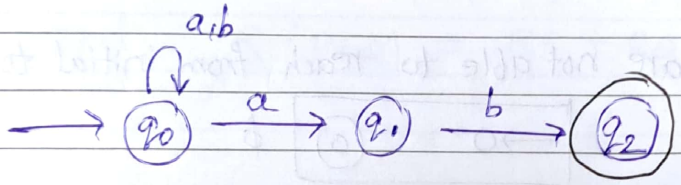
$R_3 = (a+b)^*$



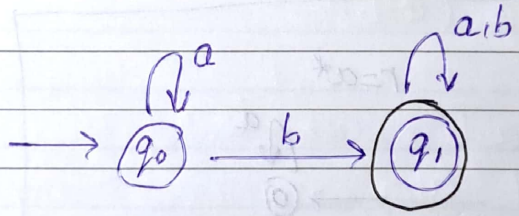


$R_4 = (a+b)^* ab$

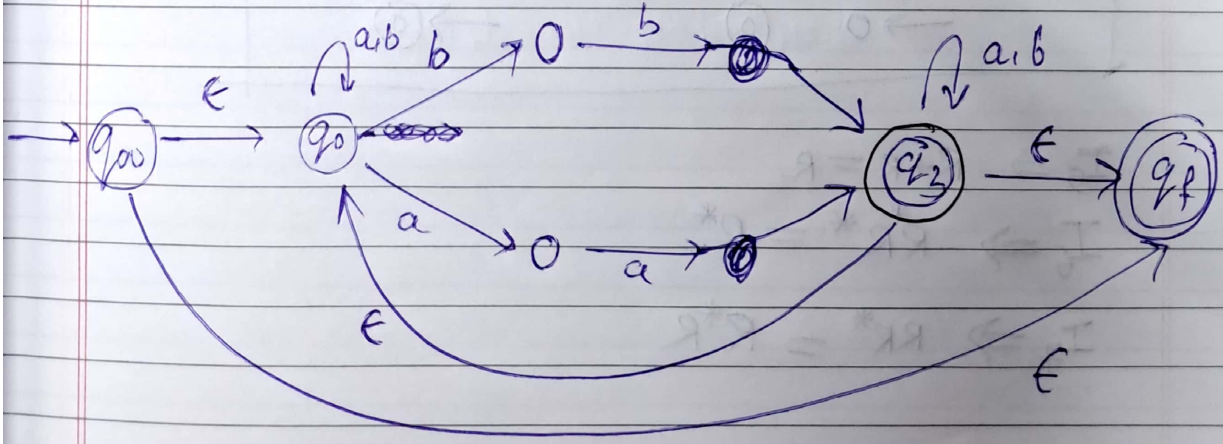




$R_5 = a^* b\ (a+b)^*$



$R_6 = (a+b)^*\ (aa+bb)\ (a+b)^*$



Identities of Regular Expression:-

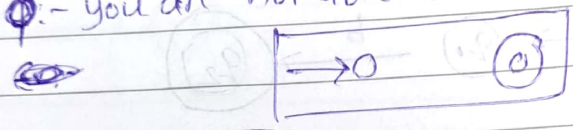
Let P, Q, R are regular expressions.

I_1 :- $\phi + R = R + \phi = R$

I_2 :- $\phi R = R\phi = \phi$

I_3 :- $\epsilon R = R\epsilon = R$

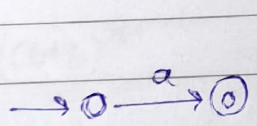
ϕ - you are not able to reach from initial to final.

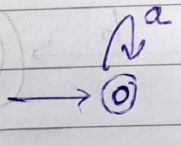


$I_4 \Rightarrow \epsilon^* = \epsilon$

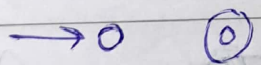
$\Rightarrow \phi^* = \epsilon$

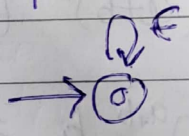
ex:- $r = a$ $\quad r = a^*$





similarly $r = \phi$ $\quad r = \phi^*$





$I_5 \Rightarrow R + R = R$

$I_6 \Rightarrow R^* R^* = R^*$

$I_7 \Rightarrow RR^* = R^* R$

$I_8 \Rightarrow (R^*)^* = R^*$

$I_9 \Rightarrow \epsilon + RR^* = RR^* + \epsilon = R^*$

$I_{10} \Rightarrow (PQ)^* P = P(QP)^*$

$I_{11} \Rightarrow (P+Q)^* = (P^* Q^*)^* = (P^* + Q^*)^*$

$I_{12} \Rightarrow (P+Q)R = PR + QR$

and $R(P+Q) = RP + RQ$.

—x—

Q] Prove

$$\epsilon + 1^*(011)^*\left(1^*(011)^*\right)^* \simeq (1+011)^*$$

ans:- Ig, I11

$$= \left(1^*(011)^*\right)^*$$

$$= \left(1^* + 011\right)^*$$

$$= \left(\left(\epsilon + 01\right)1\right)^*$$

Q] Now, you will be given DFA, NFA, Regular expression, ...
Find the Language it corresponds to :-

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

ans:-

Language over a, b, such that atleast 1 occurence of aba is present anywhere

$$L = \{ w \mid w \in (a,b)^* \text{ and } w \text{ contains aba atleast once anywhere in string} \}$$

self loop ⇒ both incoming & outgoing

Ardon's Theorem :-

Let P, Q be 2 regular Expressions on a finite alphabet Σ, then if P does not contain ϵ.

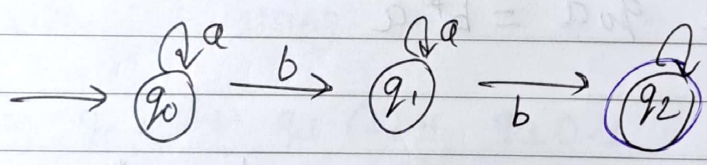
The equation of this form

$$R = Q + RP$$

has a unique solution given by

$$R = QP^*$$

Q] Write down regular expression corresponding to this?



ans:- Lets write equation for q_0 :- check incoming edge.
:- write state & symbol

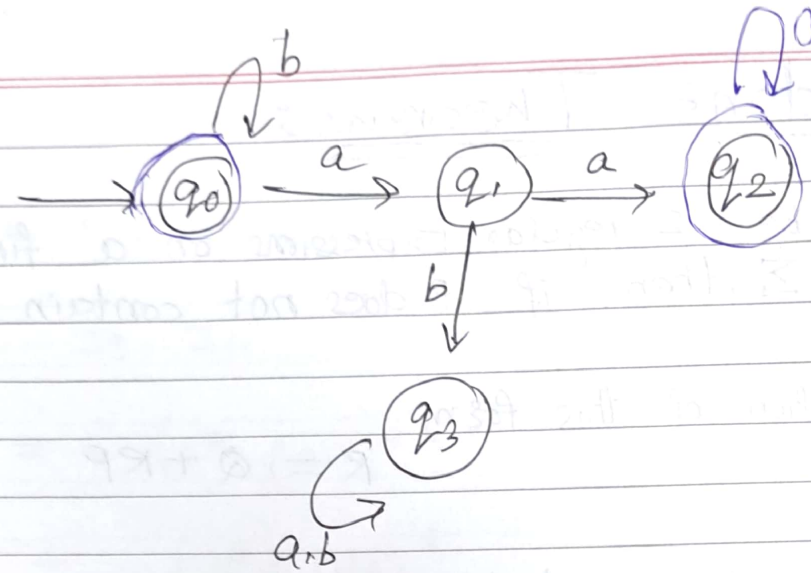
$$q_0 = \epsilon + q_0 a = \epsilon a^* = a^*$$

$$q_1 = q_0 b + q_1 a = a^* b + q_1 a = a^* b a^*$$

$$q_2 = q_1 b + q_2 a + q_2 b = a^* b a^* b + q_2 (a+b)$$
$$= a^* b a^* b (a+b)^*$$

Final answer simply depends on final state q_2.
here all are interdependent, we need to solve all using Ardons theorem.

Q]



ans-

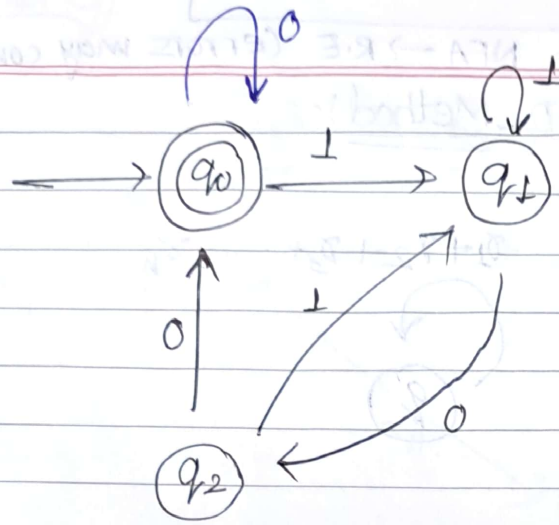
$$q_0 = \epsilon + q_0 b = \epsilon b^* = b^*$$

$$q_1 = q_0 a = b^* a$$

$$q_2 = q_1 a + q_2 a + q_2 b = b^* a\, a + q_2 (a+b)$$
$$= b^* a\, a\, (a+b)^*$$

$$q_3 = q_1 b + q_3 (a+b)$$
$$= b^* a b + q_3 (a+b)$$
$$= b^* a b\, (a+b)^*$$

Q



$R = Q + RP$

$R = QP^*$

ans: $q_0 = \epsilon + q_0 0 + q_2 0$ ——— (i)

$q_1 = q_1 1 + q_0 1 + q_2 1$ ——— (ii)

$\boxed{q_2 = q_1 0}$ ——— (iii)

Rewrite (ii) using (iii).

$q_1 = q_0 1 + q_1 (1) + q_1 0 1$

$= q_0 1 + q_1 (1 + 01)$

$\therefore \boxed{q_1 = q_0 1 + q_1 (1+01) \\ = q_0 1 (1+01)^*}$

$q_2 = q_1 0 = q_0 1 (1+01)^* 0$

$\therefore q_0 = \epsilon + q_0 0 + q_0 1 (1+01)^* 00$

$= \epsilon + q_0 (0 + 1 (1+01)^* 00)$

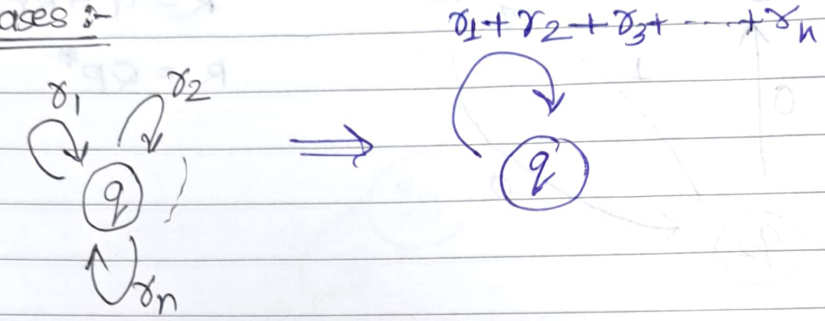
$= (0 + 1 (1+01)^* 00)^*$

Note:- DFA → R.E 100% work
Ardon Theorem
but NFA → R.E (errors may come)

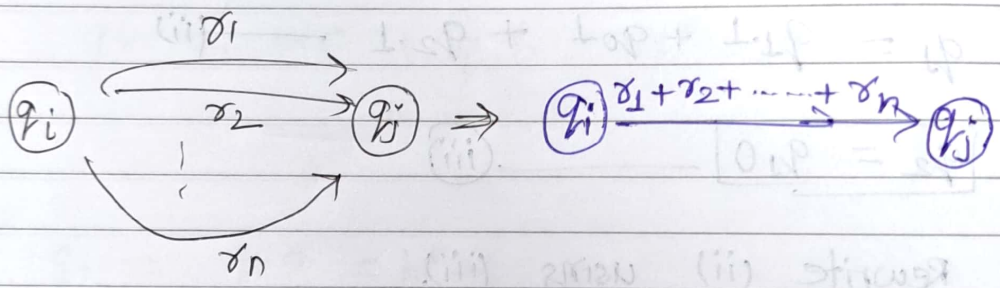
★ State Elimination Method:-

Cases :-

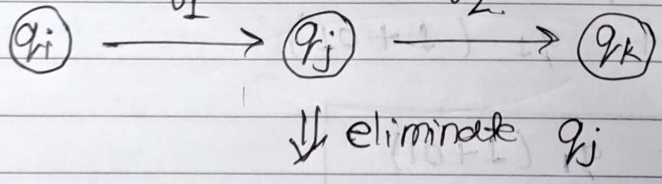
1)



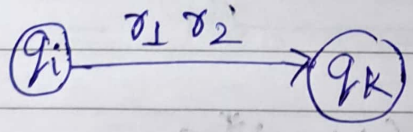
2)



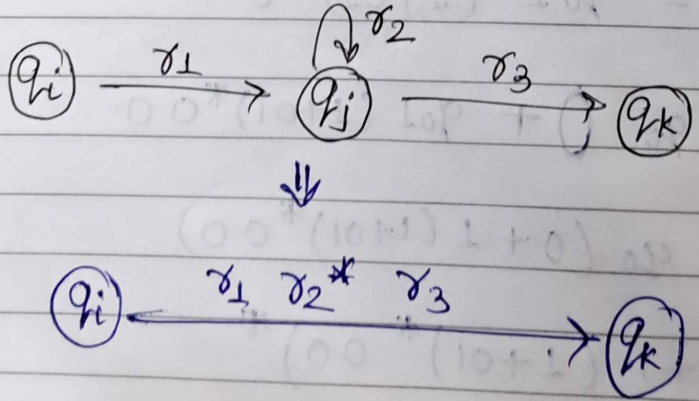
3)



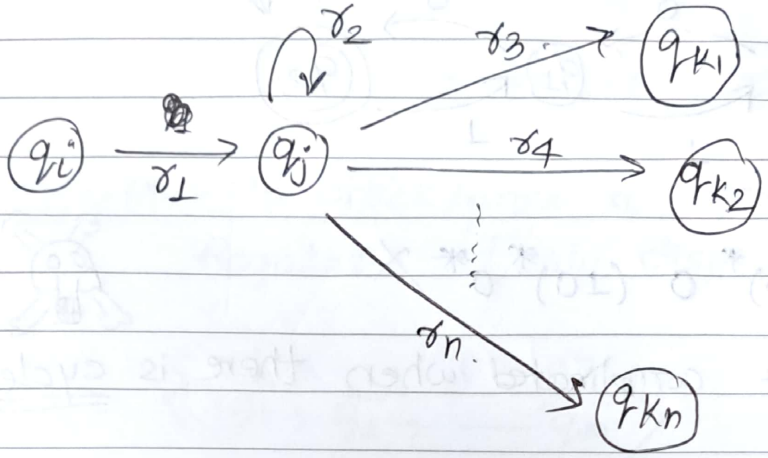
⇓ eliminate q_j



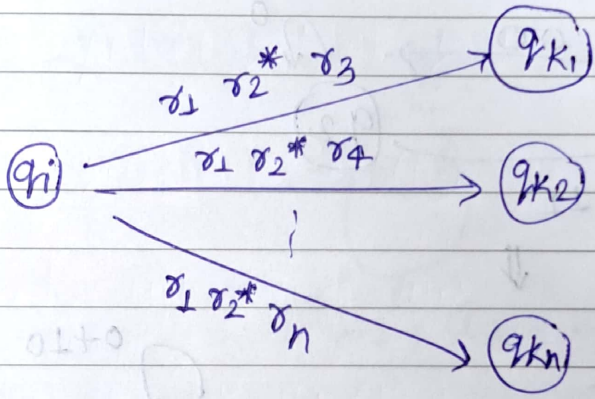
4)



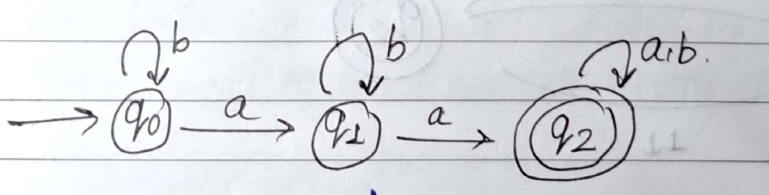
5)



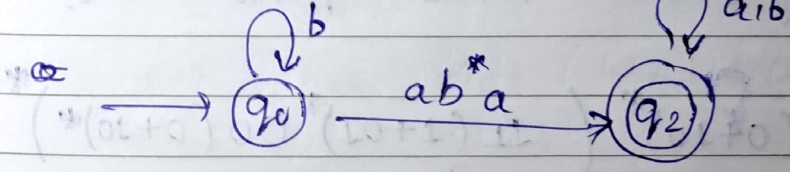
⇓ eliminate



6)

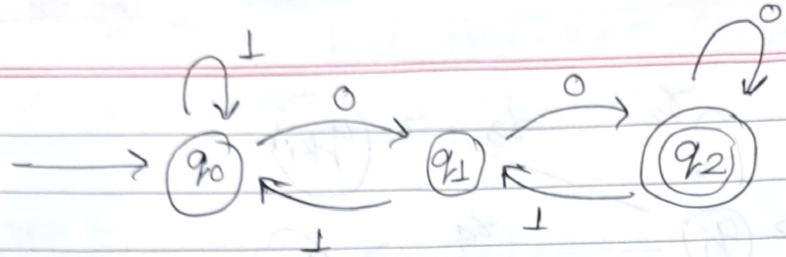


⇓ eliminate q_1.



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

(7)

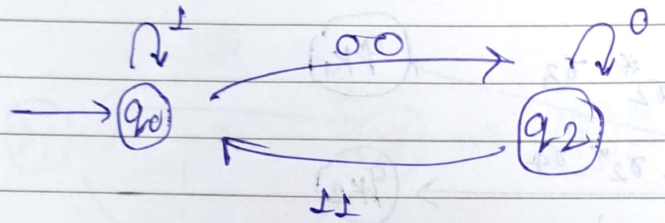


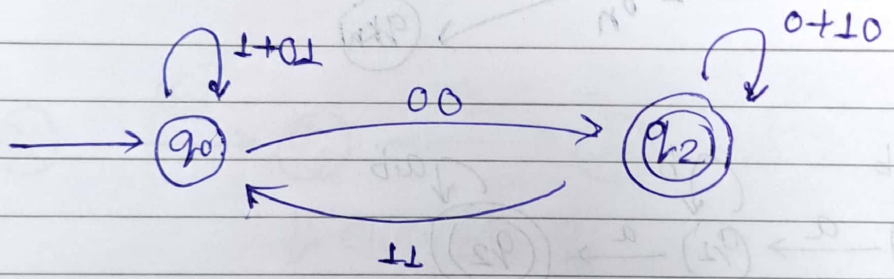
class:-

$1^* 0 (10)^* 0 (10)^* 0^*$ ✗

Things get complicated when there is cycle

cycle will convert into self loop.





ans:- Taking ϵ common, means happened 1 time.

$$(1+01)^* 00 (0+10)^* \left(11 (1+01)^* 00 (0+10)^* \right)^*$$

or.

$$(1+01)^* 00 \left(0+10 + 11 (1+01)^* 00 \right)^*$$

Pumping Lemma :-

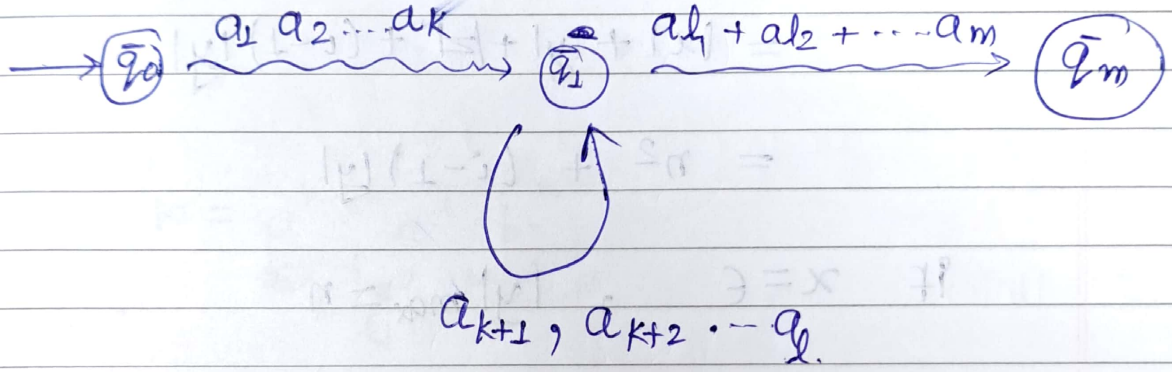
It is not a regular Language

Objective:- To check/prove a Language is not Regular. [only disprove available]

$Q = \langle \bar{q_0}, \bar{q_1}, \cdots \bar{q_m} \rangle$

$|w| > m.$

$w = a_1 a_2 \cdots a_n$ string



Let L be a regular Language. and const n, such that w is string in L, and

$|w| \geq n$

Then we may write; $w = xyz$.

In such a way that mod $|xy| \leq n$

where $|y| \geq 1$.

and for all $i \geq 0$, the new string xy^iz is in L.

L	$\|w\| \geq n$
$w = xyz$	
$\|xy\| \leq n$	$\|y\| \geq 1$
$i \geq 0$	xy^iz

Q] $L = \{a^{p^2} \mid p \geq 1\}$

ans:

1) $w = a^{n^2}$

$|w| = n^2 \geq n$

$w = xyz \qquad |xy| \leq n \qquad |y| \geq 1$

$|xy^iz| = |x| + i|y| + |z|$

$= |x| + |y| + |z| + (i-1)|y|$

$= n^2 + (i-1)|y|$

If $x = \epsilon$, $|y|_{max} = n$

and $i = 2$

$|xy^iz| = n^2 + n$, which is never perfect square

$xy^iz \notin L$

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

ans:- You cannot remember, no. of a & b.
∴ Not Regular.

Proof:- $w = a^n b^n$

$|w| = 2n \geq n$

$w = xyz$

$|xy| \leq n$, $|y| \geq 1$

$w = a^{n-l} a^l b^n$

x y z

$xy^i z = a^{n-l} b^{il} b^n$

If $i = 2$

$xy^2 z = a^{n+l} \cdot b^n \notin L$

$xy^2 z = a^{n+l} b^n \notin L$

∴ it is not regular

Q.3) $L = \langle w w^R \mid w \in (a,b)^* \rangle$

i.e palindrome

ans: Not Regular Expression/Language.

ex: fibonacci, etc.

★ Closure properties of Regular Language:

1) Union.
2) Concatenation.
3) Closure
4) Complementation

$$L^c = \Sigma^* - L$$

5) Intersection.

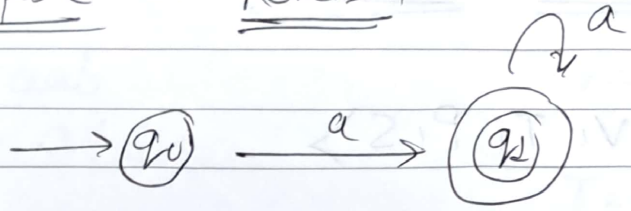
$$L_1 \cap L_2.$$

note: L_1, L_2, L_1^c, L_2^c.

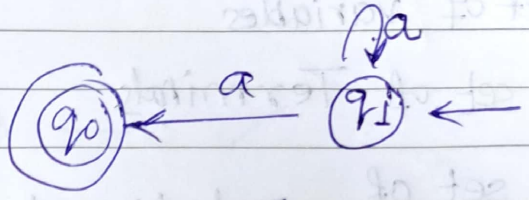
ans: $\left(L_1^c \cup L_2^c\right)^c$

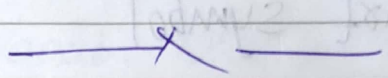
6) Transpose or Reversal

ex:-



ans:-





Regular Grammar

$G = \langle V, T, P, S \rangle$

V ← set of variables

T ← set of Terminals

P ← set of production rules

S ← Start symbol

P is in form

$\alpha \longrightarrow w_1 \beta \mid w_2$ (Right Linear Grammar.)

↓ or

$\alpha \longrightarrow w_1 \beta$

$\alpha \longrightarrow w_2$

$\alpha, \beta \in V$

$w_1, w_2 \in T^*$

any length of terminals

A → variable
a → terminals
→ by Default.

Q] $S \rightarrow aA \mid bB$

$A \rightarrow aab$

$B \rightarrow a \mid \epsilon$

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

$T = \{a, b\}$

string b

$S \rightarrow bB \rightarrow b$ ✓

string aaab

$S \rightarrow aA \rightarrow aaab.$ ✓

string aaaaab

$S \rightarrow aA \rightarrow aaaab$ ✗ rejected

Q] ex:- aaaab

$S \rightarrow aA \mid bB \mid aS$

$A \rightarrow aab$

$B \rightarrow a \mid \epsilon$

ans:- $S \rightarrow aS \rightarrow aaA \rightarrow aaaab$

— x —

DFA & Grammar

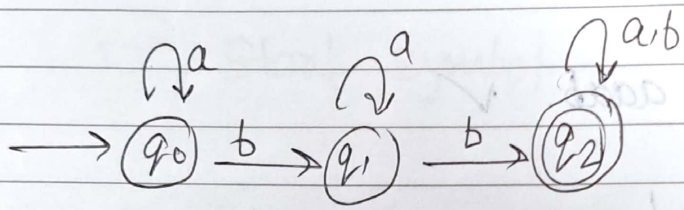
$V \iff Q$

$T \iff \Sigma$

$S \iff q_0$

$P \iff \delta$

ex:-

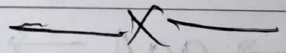


$q_0 \longrightarrow aq_0 \mid bq_1$

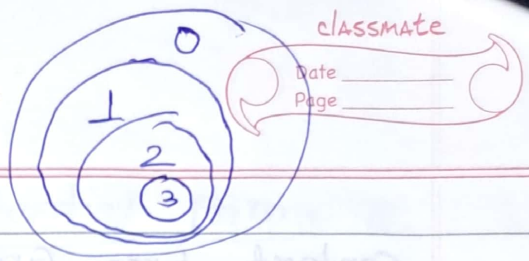
$q_1 \longrightarrow aq_1 \mid bq_2$

$q_2 \longrightarrow aq_2 \mid bq_2 \mid \epsilon$

final states pe ϵ production rule lagaye



∴ Chomsky Hierarchy :-



Type	Language	Form of Production Rule	Acceptor
0	Recursive Enumerable Language.	$\alpha \longrightarrow \beta$ $\alpha \in (V+T)^* V (V+T)^*$ $\beta \in (V+T)^*$	Turing Machine
1	Context Sensitive Language.	$\alpha \longrightarrow \beta$ $\alpha \in (V+T)^* V (V+T)^*$ $\beta \in (V+T)^*$ $\lvert\alpha\rvert \leq \lvert\beta\rvert$	Linear Bounded Automata.
2	Context Free Language.	$\alpha \longrightarrow \beta$ $\alpha \in$ Variables $\beta \in$ (variables + Terminals)*	Pushdown Automata.
3	Regular Language	$\alpha \longrightarrow w_1 \beta \mid w_2$ $\alpha, \beta \in$ variables, $w_1 \& w_2 \in T^*$	Finite Automata.

Terminals ie. constants.

Context Free Grammar :- (CFG)

$G = \langle V, T, P, S \rangle$

$\alpha \longrightarrow \beta$

$\alpha \in V$

$\beta \in (V+T)^*$

$S \longrightarrow \langle sign \rangle \langle integer \rangle$

$\langle integer \rangle \longrightarrow \langle digit \rangle \langle integer \rangle \mid \langle digit \rangle$ (or)

$\langle sign \rangle \longrightarrow +/-$

$\langle digit \rangle \longrightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

e.g. -716

$V = \{ \langle sign \rangle, \langle integer \rangle, \langle digit \rangle, S \}$

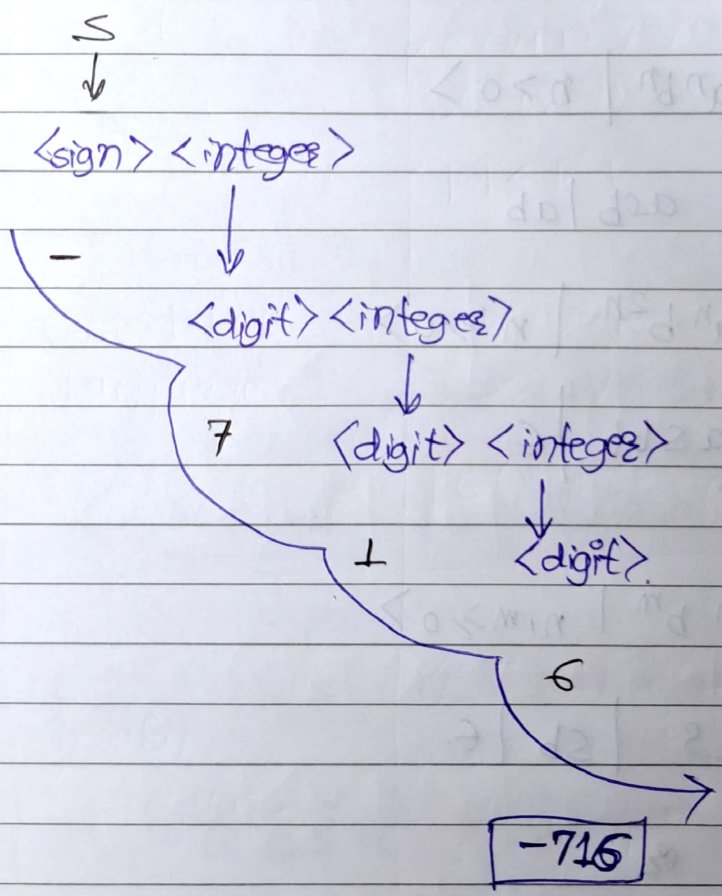
$T = \{ +, -, 0, 1, 2, \ldots\ldots, 9 \}$

Derivation on next page.

Derivation Sentential Form

S → <sign> <integer>
→ - <digit> <integer>
→ - 7 <digit> <integer>
→ - 7 1 <digit>
→ -716

Parse Tree :-



In pumping lemma, we did not possess memory so we could not write following using Lemma.

$L_1 = \langle a^n b^n \mid n \geq 0 \rangle$

ans:- $S \longrightarrow aSb \mid \epsilon$

$S \longrightarrow aSb \longrightarrow aaSbb$

$\longrightarrow aaaSbbb$

$\longrightarrow aaabbb.$

Q] $L_2 = \langle a^n b^n \mid n > 0 \rangle$

ans:- $S \longrightarrow aSb \mid ab$

Q) $L3 = \langle a^n b^{2n} \mid n \geq 0 \rangle$

ans:- $S \longrightarrow aSbb \mid \epsilon$

Q] $L_4 = \langle a^n b^m \mid n, m \geq 0 \rangle$

ans:- $S \longrightarrow aS \mid Sb \mid \epsilon.$

or

$S \longrightarrow AB$

$A \longrightarrow aA \mid \epsilon$

$B \longrightarrow bB \mid \epsilon$

Q] $L_5 = \langle a^n b^m c^{n+m} \mid n, m \geq 0 \rangle$

ans:-

$a^n \; b^m \; c^m \; c^n$

$S \rightarrow A \rightarrow bAc \mid bc$

$S \rightarrow aSc \mid A$

$A \rightarrow bAc \mid \epsilon$

Q] $L_6 = \langle a^n b^m c^n \mid n \geq 1, m \geq 0 \rangle$

$S \rightarrow aS\epsilon \mid aAc$

$A \rightarrow bA \mid \epsilon$

Q] $L_7 = \langle W c W^R \mid W \in (a,b)^* \rangle$

W^R → Reverse of W

WcW^R → palindrome of odd length.

$S \rightarrow asa \mid bsb \mid c$

ex:- a b s a

a b s b a

a b a s a b a

a b a a s a a b a

a b a a c a a b a

aaasbs
aasbs
aaasbb

aaaBb

Q] $L_8 = \langle$ palindrome strings over a & b $\rangle$

ans:- $S \to aSa \mid bSb \mid \epsilon \mid a \mid b.$

× ——— × ——— × TOC

Q] $L_9 = \langle a^n b^n \mid n \text{ is even} \rangle$

ans:- $S \to aaSbb \mid \epsilon$

Q] $L_{10} = \langle a^n b^n \mid n \text{ is odd} \rangle$

ans:- $S \to aaSbb \mid ab$

Q] $L = \langle a^n b^n \mid n \text{ is multiple of } 3 \rangle$

ans:- $S \to aaaSbbb \mid \epsilon$

Q] $L^B = \langle a^n b^n \mid n \text{ is not multiple of } 3 \rangle$

ans:- $S \to aaaSbbb \mid ab \mid aabb$

Q] $L = \langle w \mid w \in (a,b)^* \;\&\; n_a(v) \geq n_b(v)$

where v is a prefix of $w \rangle$

ans:- ✓ a, ab, aba, aaa, abaab.

× b, abb,

$S \to aS \mid asbs \mid \epsilon$

or

$S \rightarrow as \mid asb \mid ss \mid \epsilon$

Note:- $S \rightarrow abs \mid asb \mid sab \mid \epsilon$

means $S \rightarrow asbs$

Q] $L_{14} = \{ w \mid w \in (()^*$ where only valid parenthesis are acceptable. $\}$

ans:- ~~$S \Rightarrow (S)S \mid \epsilon$~~ $S \rightarrow S(S)S \mid \epsilon$

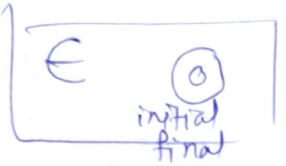
or

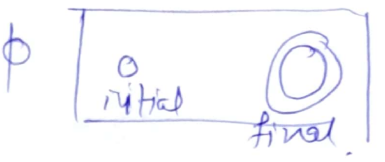
~~$S \Rightarrow (S)S \mid \epsilon$~~

or

$S \rightarrow (S) \mid SS \mid \epsilon$

ex:- $\underset{d}{(}\underset{c}{(}\underset{b}{(}\underset{b}{)}\underset{c}{)}\underset{d}{(}\underset{a}{)}\underset{a}{)}\underset{d}{)}$





Q $L_{15} = \langle w | w \in (a,b)^*$ are valid regular expressions $\rangle$

ans:-

$S \rightarrow S+S \mid SS \mid S^* \mid (S) \mid a \mid b \mid \in \mid \phi$

Q $L_{16} = \langle w \mid w \in (a,b)^* \rangle$

ans:- $S \rightarrow aS \mid bS \mid \in$

[a] $L = \langle w | w \in (a,b)^* \ \& \ n_a(w) = n_b(w) \rangle$

ans:- $S \rightarrow asbs \mid bsas \mid \in$

$S \rightarrow asb \mid bsa \mid ss \mid \in$

a] $L = \langle a^n b^m \mid n \leq m+3 \rangle$

ans:- $S \rightarrow aSb \mid Sb \mid aaa \mid aa \mid a \mid \in$

a] $L = \langle a^n b^m \mid n \neq 2m \rangle$

~~$S \rightarrow asbb \mid a \mid b \mid ab \mid ba \mid \in$~~

$S \rightarrow aaSb \mid$ 😕 ??

New way

$n \neq 2m$	$n < 2m$ and $n > 2m$

aaab

$n = 3$ $m = 1$

S ⟶ A | B

A ⟶ aaSbbb

B ⟶ .

Homework.

Derivations:-

(1) Leftmost Derivation (LMD)
(2) Rightmost Derivation (RMD)

} This for Compilers

$S \rightarrow S+S \mid S*S \mid a \mid b \mid c$

$a+b*c$

ans:-

LMD

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

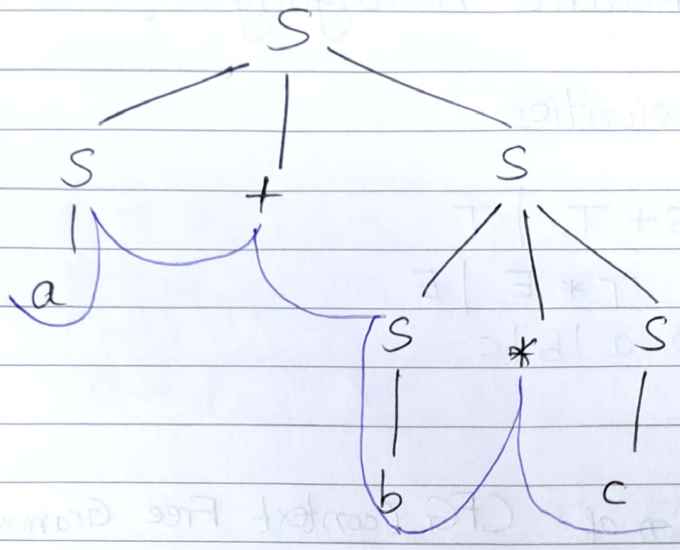
RMD

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

Note:- we could also begin with S*S

$S \rightarrow S*S$
$S \rightarrow S+S*S$ etc

TREE



Ambiguity :-

* English is never used for writing compilers.
* In python, java, C, we use english because high level Language
* Grammar should be Unambiguous
* Sanskrit is unambiguous.

ex :- Previous Language is ambiguous because different derivations (commands) are giving same output.

If 2 LMD or 2 RMD give same result ⇒ Ambiguous

ex :-

LMD1	LMD2
S → S * S	S → S+S
S → S+S * S	S → S+S*c
⇒⇒ a+b * c	⇒⇒ a+b*c

How to Reduce Ambiguity :- ?

ans:- Set priorities.

$S \rightarrow S+T \mid T$

$T \rightarrow T*E \mid E$

$E \rightarrow a \mid b \mid c$

Simplification of CFG (context Free Grammar) :-

ex.1

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow b \mid C$

$E \rightarrow c \mid \epsilon$

1st Problem Useless symbols

* C can be reached, but cannot defined.
* E is defined, but E cannot be reached.

ex-2

$S \rightarrow ACbB$

$A \rightarrow BC$

$B \rightarrow b \mid \epsilon$

$C \rightarrow D \mid \epsilon$

$D \rightarrow d$

Rectify Nullables

Identify nullable symbols & rectify it.

correction:-

$S \rightarrow ACbB$

$A \rightarrow BC$

$B \rightarrow b$

$C \rightarrow D$

$D \rightarrow d$

more rectifications :-

$S \rightarrow ACbB \mid ACb \mid AbB \mid Ab \mid b \mid CbB \mid cb \mid bB.$

$A \rightarrow BC \mid B \mid C$

$B \rightarrow b$

$C \rightarrow D$

$D \rightarrow d.$

ex-3

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow c \mid b$

$C \rightarrow D$

$D \rightarrow E$

$E \rightarrow a$

Unit productions.
(Too many derivations interconnected)

ans:-

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow b$

$E \rightarrow a$

$B \rightarrow a$

$C \rightarrow a$

$D \rightarrow a$

Chomsky Normal Form (CNF)

$G = \langle V, T, P, S \rangle$

Rule:
- $A \to BC$
- $A \to a$

Q) $S \to aXbX$
$X \to aY \mid bY \mid \epsilon$
$Y \to X \mid c$

ans:-

$S \to aXbX \mid aXb \mid abX \mid ab$
$X \to aY \mid a \mid bY \mid b$
$Y \to X \mid c$

Now remove unit productions

$S \to aXbX \mid aXb \mid abX \mid ab$
$X \to aY \mid a \mid bY \mid b$
$Y \to aY \mid a \mid bY \mid b \mid c$

We need to every RHS should be one of the chomsky normal form.

Take new variables for deriving 1 terminal to remove "aA" type derivations & convert to "ZA" where $Z \to a$.

Let $X_a \rightarrow a$

$X_b \rightarrow b$.

$S \rightarrow X_a X X_b X$

still not CNF

$\therefore S \rightarrow X_a Y_1$

$Y_1 \rightarrow X(X_b)X$

still not CNF

$\therefore$ All Rules of X.

$S \rightarrow X_a Y_1$	$S \rightarrow X_a Y_3$	$S \rightarrow X_a Y_2$
$Y_1 \rightarrow X\ Y_2$	$Y_3 \rightarrow X X_b$.	$S \rightarrow X_a X_b$
$Y_2 \rightarrow X_b X$		

Now, $X \rightarrow aY \mid a \mid bY \mid b$.

$X \rightarrow X_a Y$

$X \rightarrow a$

$X \rightarrow X_b Y$

$X \rightarrow b$

Now, $Y \rightarrow aY \mid a \mid bY \mid b \mid c$

$Y \rightarrow X_a Y$

$Y \rightarrow a$

$Y \rightarrow X_b Y$

$Y \rightarrow b$

$Y \rightarrow c$

Q) $S \rightarrow AbA$

$A \rightarrow Aa \mid \epsilon$

ans:

$X_b \rightarrow b$

$X_a \rightarrow a$

$S \rightarrow AA_1$

$A_1 \rightarrow X_b A$

$A \rightarrow AX_a$

$A \rightarrow \epsilon$

$S \rightarrow AbA \mid Ab \mid bA \mid b.$

$A \rightarrow Aa \mid a$

Removed Nullables

$S \rightarrow AA_1$

$A_1 \rightarrow X_b A$

$S \rightarrow AX_b$

$S \rightarrow X_b A$

$S \rightarrow b$

$X_b \rightarrow b$

$X_a \rightarrow a$

$A \rightarrow AX_a$

$A \rightarrow a$

9 rules.

***Substitution Rule:-

If

$A \rightarrow \alpha_1 B \alpha_2$

$B \rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_n$

$A \rightarrow \alpha_1 \beta_1 \alpha_2 \mid \alpha_1 \beta_2 \alpha_2 \mid \dots \mid \alpha_1 \beta_n \alpha_2$

Left Recursion:-

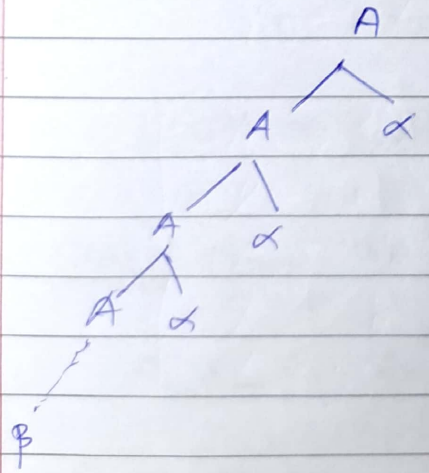
$A \rightarrow A\alpha \mid \beta$

If symbol on LHS is matching with 1st symbol of RHS., called Left Recursion.

$A \rightarrow \beta B \rightarrow \beta$

$B \rightarrow \alpha B \mid \alpha$ ☹ but it is Right recursion

ans:- Yes, will discuss Later



Identification :-

$$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \cdots \mid A\alpha_m \mid \beta_1 \mid \beta_2 \mid \beta_3 \cdots \mid \beta_n$$

$$\Downarrow$$

ex:- $A \rightarrow A\,BcD \mid DF \mid a \mid aDE \mid aA$

$A\alpha_1$ $\quad \beta_1$ $\quad \beta_2$ $\quad \beta_3$ $\quad \beta_4$

Process

Write Rules of β as it is

$$A \rightarrow \beta_1 \mid \beta_2 \mid \cdots \beta_n \mid \beta_1 B \mid \beta_2 B \mid \cdots \mid \beta_n B$$

$$B \rightarrow \alpha_1 \mid \alpha_2 \cdots \alpha_m \mid \alpha_1 B \mid \alpha_2 B \cdots \mid \alpha_m B$$

Greibach Normal Form (GNF):-

* Every first Letter will be terminal, so that it will easy for compilers.

$$A \longrightarrow a\alpha$$
$$\alpha \in V^*$$

Basic :-

CFG context free gram $\longrightarrow$ Chomsky Normal form $\longrightarrow$ Greibach Normal form

Step 1) Eliminate null production, unit production & useless production & construct G' grammar

$G = \langle V, T, P, S \rangle$
$G' = \langle V', T', P', S' \rangle$
in chomsky normal form & generate:

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

Step 2) Rename the variables like:-

$A_1, A_2, \ldots, A_n$.

Step 3) Modify the Rules in P', so that if you are having a rule,

$$A_i \longrightarrow A_j \gamma \in P'$$

then $(j > i)$.

$A_1 \longrightarrow A_4 A_1$ ✓

$A_1 \longrightarrow A_1 A_4$ ✗

$A_4 \longrightarrow A_1 A_2$ ✗

Step 2) Starting with A_1 and proceeding to A_n.

4a) Assume that productions have been modified so that $1 \leq i \leq k$

$$A_i \longrightarrow A_j \gamma \in P' \quad j \geq i$$

4b) If $A_k \longrightarrow A_j \gamma$ is a production with $j < k$, generate a new set of productions substituting for A_j. The body of each A_j production.

4c) Repeating 4b) at most $k-1$ times, we obtain rules of the form.

$$A_k \longrightarrow A_p \gamma \quad , \quad p \geq k$$

4d) Replace rules $A_k \longrightarrow A_k \gamma$, by removing Left recursion.

Step 5) Modify rule $A_i \rightarrow A_j\gamma$

for $i = n-1, n-2, \ldots 1$ in the derived form at the same time change the original production rules after next recursion.

—x—

example

Q] $S \rightarrow XA \mid BB$
$B \rightarrow b \mid SB$
$X \rightarrow b$
$A \rightarrow a$

ans:- $A_1 \rightarrow S$
$A_2 \rightarrow X$
$A_3 \rightarrow A$
$A_4 \rightarrow B$

Now it becomes $A_1 \rightarrow A_2A_3 \mid A_4A_4$ (nope, nope)

$A_4 \rightarrow b \mid A_1A_4$ ✓ (problem)

$A_2 \rightarrow b$ ✓

$A_3 \rightarrow a$ ✓

change:- $A_4 \rightarrow A_1A_4$

at this step check $i \geq j$

$A_4 \rightarrow A_2A_3A_4 \mid A_4A_4A_4 \mid b$ (problem)

$A_2 \rightarrow b$

$A_3 \rightarrow a$

Now solve Left recursion, but this time u need new variable, say A_5.

$A_1 \longrightarrow A_2A_3 \mid A_4A_4$

$A_4 \longrightarrow bA_3A_4 \mid b \mid bA_3A_4A_5 \mid bA_5$

$A_5 \longrightarrow A_4A_4 \mid A_4A_4A_5$

$A_2 \longrightarrow b$

$A_3 \longrightarrow a$

$j > i$ not followed 😐 !

ans: It has already been check. It will not be checked for new variables like A_5 which are introduced to solve Left Recursion. instead of A_5, we could use Z_1.

$A_1 \longrightarrow A_2A_3 \mid A_4A_4$

$A_4 \longrightarrow bA_3A_4 \mid b \mid bA_3A_4Z_1 \mid bZ_1$

$Z_1 \longrightarrow A_4A_4 \mid A_4A_4Z_1$

$A_2 \longrightarrow b$

$A_3 \longrightarrow a$

Next step ↓

$A_1 \longrightarrow bA_3 \mid bA_3A_4A_4 \mid bA_4 \mid bA_3A_4Z_1A_4 \mid bZ_1A_4$

$A_4 \longrightarrow bA_3A_4 \mid b \mid bA_3A_4Z_1 \mid bZ_1$

$Z_1 \longrightarrow bA_3A_4A_4 \mid bA_4 \mid bA_3A_4Z_1A_4 \mid bZ_1A_4$
$bA_3A_4A_4Z_1 \mid bA_4Z_1 \mid bA_3A_4Z_1A_4Z_1 \mid$
$bZ_1A_4Z_1$

$A_2 \rightarrow b$

$A_3 \rightarrow a$

H.W $S \rightarrow AB$

$A \rightarrow BS \mid a$

$B \rightarrow SA \mid b$.

Closure Properties :-

$G_1 = \langle V_1, T_1, P_1, S_1 \rangle$

$G_2 = \langle V_2, T_2, P_2, S_2 \rangle$

$G_U = \langle V_U, T_U, P_U, S_U \rangle$

union

$V_U = V_1 \cup V_2 \cup \{S_U\}$

$T_U = T_1 \cup T_2$

$P_U = P_1 \cup P_2 \cup \{S_U \rightarrow S_1 \mid S_2\}$

Hence, closure is followed.

$G_c \Rightarrow \langle V_c, T_c, P_c, S_c \rangle$

$V_c = V_1 \cup V_2 \cup \{S_c\}$

$T_c = T_1 \cup T_2$.

$P_c = P_1 \cup P_2 \cup \{S_c \rightarrow S_1 S_2\}$

ex:- $a^n b^n a^m b^m$ string $L_1 = \langle a^n b^n \rangle$ $L_2 = \langle a^m b^m \rangle$

ans:- $S \rightarrow AB$
$A \rightarrow aAb \mid ab$
$B \rightarrow aaBbb \mid \epsilon$

closure:-

$G_{cl} = \langle V_{cl}, T_{cl}, P_{cl}, S_{cl} \rangle$

$V_{cl} = V_1 \cup \{S_{cl}\}$

$T_{cl} = T_1$

$P_{cl} = P_1 \cup \{S_{cl} \rightarrow S_{cl} S_1 \mid S_1 S_{cl} \mid \epsilon\}$

Lec-10 Pumping Lemma Version 2

Pumping Lemma :- (To disprove only)

Let 'L' be a context free language, then there is constant 'n' depending only on L such that if 'z' is in L & $|z| \geq n$, then we may write

$z = uvwxy$ such that.

$|vx| \geq 1$

and $|vwx| \leq n$

and $\forall i \geq 0$

$uv^iwx^iy \in L$

(If it fails for any i → then fails).

Example :-

① a^nb^n → Is this context free language ??

ans:- Yes, CFG exist $S \rightarrow asb \mid \epsilon$

② $a^nb^nc^n$ → Is this CFL ??

ans:- NOT CFL.

Lets try to prove ki CFL nahi hai by pumping Lemma

Let $z = a^nb^nc^n$ be a string.

$|z| = n+n+n = 3n \geq n$

Now, $z = \underbrace{a \ldots a}_{uvwx} \underbrace{b^n c^n}_{y}$

Let $|x| = l_1$, $|w| = l_2$, $|v| = l_3$,

$|u| = n - l_1 + l_2 + l_3$

obviously → $|uvx| \leq n$

and assume :- $l_3, l_1 > 0$

So $|vx| \geq 1$

Let $i=0$,

then, $uv^i wx^i y = a^{n-l_1-l_3} b^n c^n \notin L$

(∵ not same as $a^n b^n c^n$).

Note:- similarly using pumping Lemma someone could prove that $a^n b^n$ is not CFL, but it is in CFL.

(3) $WW^R \longrightarrow$ CFG ✓

(4) $WW \rightarrow$ eg abaaba → where is middle, need to identify

NOT CFL

For $L=w$

$S \rightarrow asIbs \mid \epsilon$

for $L=ww$?

will disprove by taking $w = a^n b^n$, $z = ww$

Now $|z| = n+n+n+n = 4n \geq n$.

$z = \underbrace{a \ldots\ldots a}_{uvwx} \underbrace{b^n a^n b^n}_{y}$

$uvwx$: $n-l_1-l_2-l_3$, l_3, l_2, l_1

Let $i=0$

$uv^i wx^i y = a^{n-l_1-l_3} b^n a^n b^n \notin L$

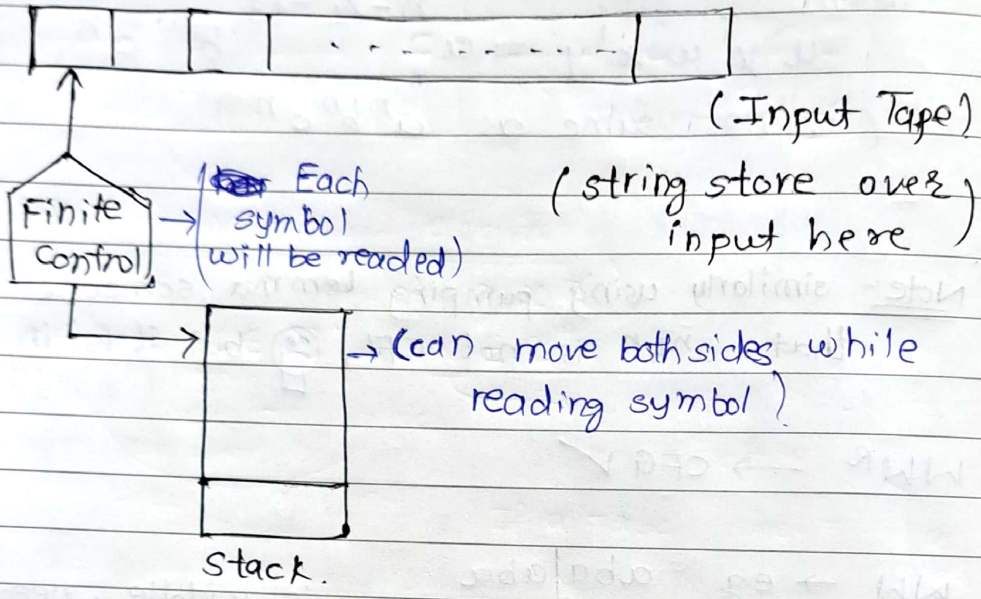
as $l_1, l_3 > 0$

as $|vx| \geq 1$

$l_1 + l_3 \geq 1$

PDA : Pushdown Automata

- We get memory in terms of stack! (LIFO)



PDA = ϵ-NFA + Stack

PDA = $\langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$

Γ , Z_0 → related to stack.

Γ = Finite set of stack symbols.

Z_0 = the initial stack symbol
(useful to tell if we reach end of stack - Emptystack)
(like top variable in DSA).

If we are on Z_0 means nothing done
stack empty

$\delta(q_i, a, Z) = ?$ $\quad \delta : Q \times \{\Sigma \cup \epsilon\} \times \Gamma \longrightarrow Q \times \Gamma^*$

Input set of symbols union null symbol

(Read Top symbol)

(Pop only ⊥ symbol)

PDA will read input symbol and stack symbol, (Top of stack) and move to a new state and change symbol of stack.

Instantaneous description (ID) :- informal notation of how PDA computes a input string and make decision whether it is accepted or rejected.

input (q_1, a, z)

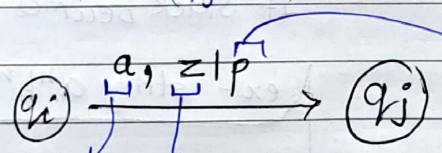
$\delta(q, a, z) = \{(p_1, \gamma_1), (p_2, \gamma_2), (p_3, \gamma_3) \cdots\}$

- q, p_i's $\in Q$
  $a \in \Sigma$
  $z \in \tau$
  γ_i's $\in \tau^*$

$\delta(q_i, a, z) = (q_j, p)$ (general Equation)
for understanding pop-push operation.

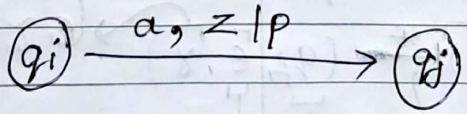
ex:- If $p = \epsilon$

$\delta(q_i, a, z) = (q_j, \epsilon)$

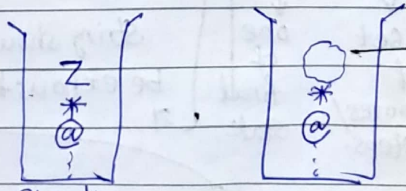


z is changed to p and state changed from q_i to q_j

(a is read) (top most element was z)



If $p = \epsilon$



nothing here, means ϵ

This is pop operation.

Stack

upon reading 'a'

$\delta(q_i, a, z) = (q_j, z)$

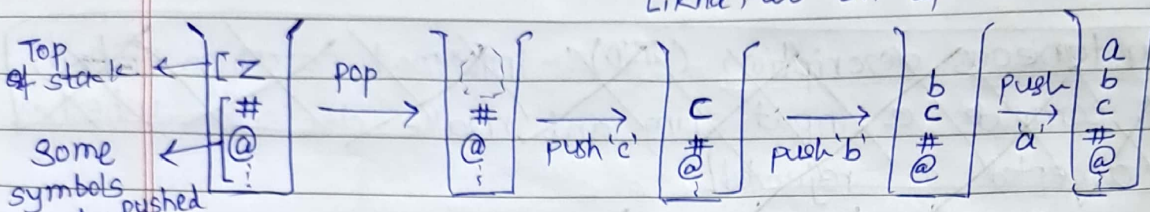
pop z

push z

2 operations happened:

Meaning of $\delta(q_i, a/z) = (q_j, abc)$? (replacement mein sabse pehle pop)

Jo starting mein likha, wo on top



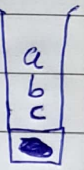
ID (Instantaneous Description)

Written in form of triplet (q, x, α)

where $q \in Q$, $x \in \Sigma^*$, $\alpha \in \tau^*$

- q: current state.
- x: string which you want to check is accepted or not.
- α: content of stack from top to bottom.

If $\alpha = abc$ stack =



How to know, if string accepted or rejected?

Acceptance by PDA

Case (1) If finally it reaches final state So, ACCEPTED

ex:- string abc only

$PDA_f = \langle Q, \Sigma, \delta, q_0, Z_0, F \rangle$

PDA_f → PDA for final state acceptance

$L(PDA_f) = \langle w \mid (q_0, w, Z_0) \overset{*}{\vdash} (q_f, \epsilon, \gamma) \rangle$

- (q_0, w, Z_0): Beginning ID
- w: string want to check
- q_0: Initial state
- $\overset{*}{\vdash}$: set of moves/steps
- q_f: one of final state
- ϵ: string should be exhausted
- γ: final ID.
- (ϵ, τ^*)

Case (2) Empty stack

If stack becomes empty, then accepted

ex:- string $a^n b^n$

$PDA_e = \langle Q, \Sigma, \tau, \delta, q_0, Z_0, \phi \rangle$

- PDA_e: acceptance by Empty stack
- ϕ: Final state not required to check acceptance

$L(PDA_e) = \langle w \mid (q_0, w, Z_0) \overset{*}{\vdash} (P, \epsilon, \epsilon) \rangle$

ex:- $a^n b^n$, $n > 0$

* Pushdown Automata *

$FA = \langle Q, \Sigma, \delta, q_0, F \rangle$

FA ∪ stack

$PDA = \langle Q, \Sigma, \Gamma, q_0, z_0, F \rangle$

$\delta(q_0, a, z) = (p, \alpha)$

We read 'a' and reached p state.

This is equivalent to 2 operations

1) popping a, z
2) pushing α instead

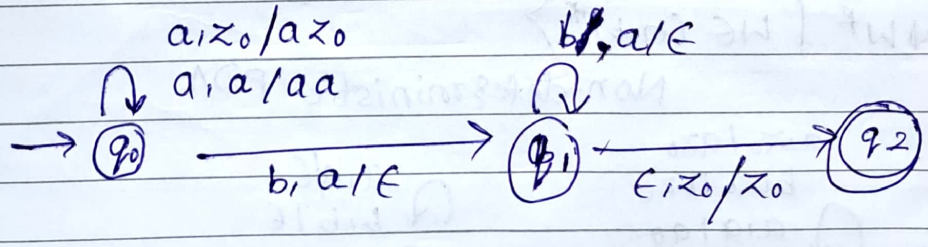
$$q \xrightarrow{a,\ z/\alpha} p$$

Q] $L = \langle a^n b^n , n > 0 \rangle$

$S \rightarrow asb \mid ab$

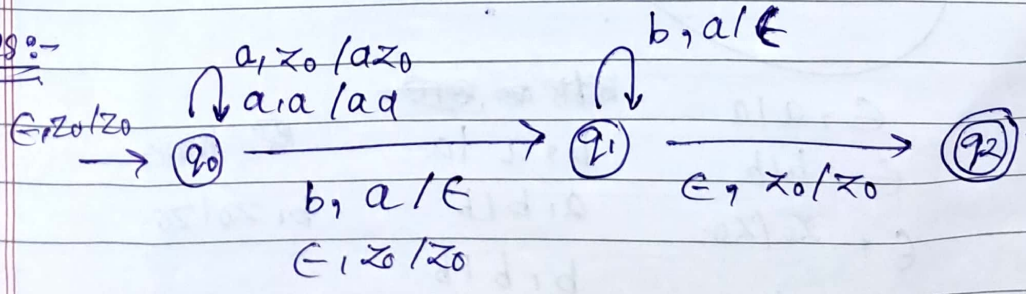
α
α
z_0

ans:



Q] $L = \langle a^n b^n , n \geq 0 \rangle$

ans:-

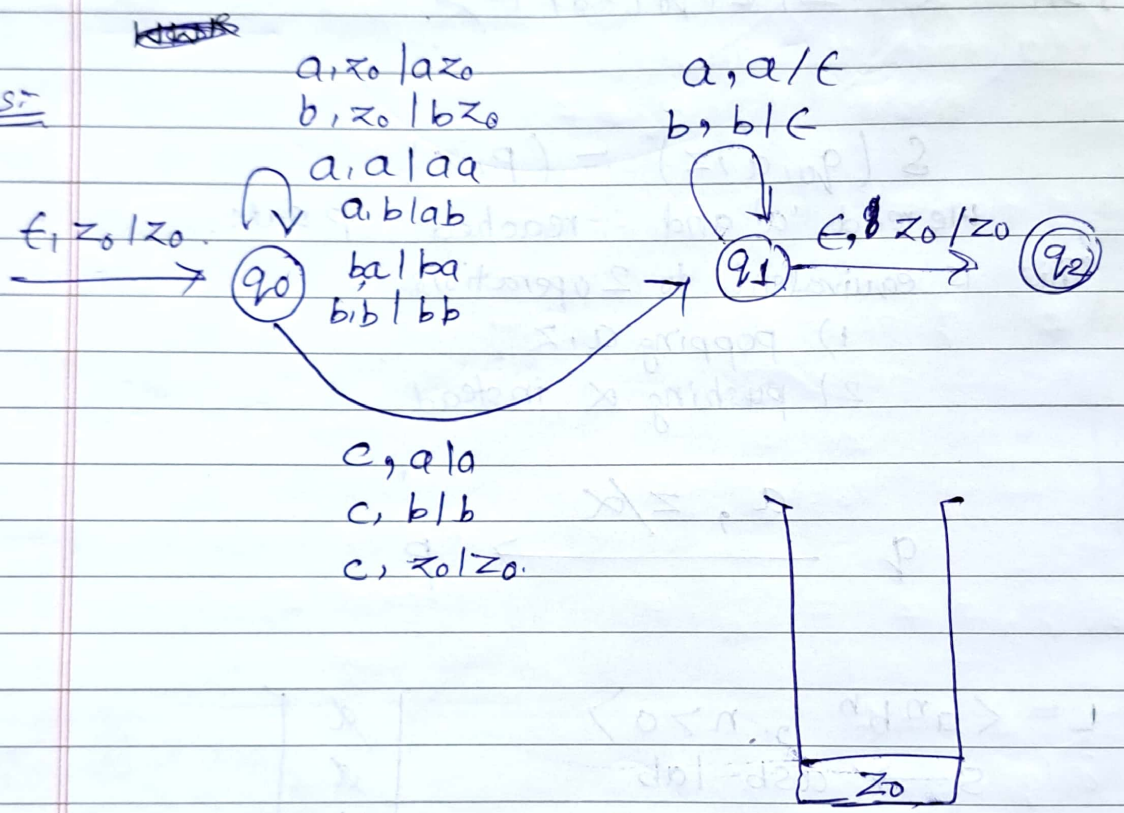


$WW^R \Longrightarrow$ non-deterministic PDA

Q $L = \{ WcW^R \mid W \in (a,b)^* \}$

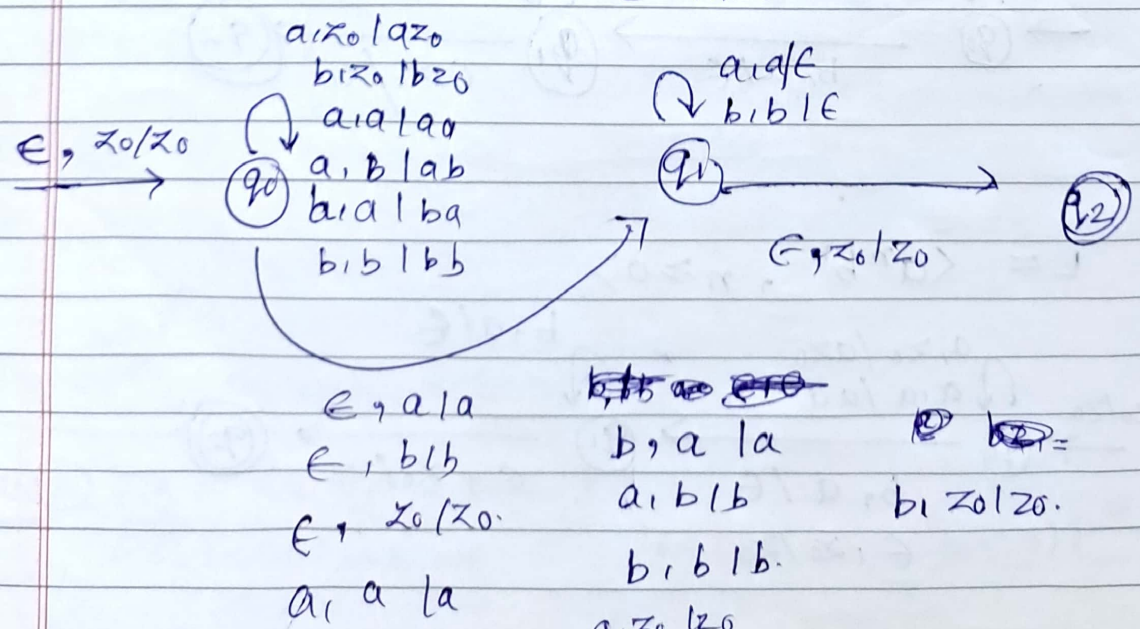
↳ Deterministic PDA

ans:



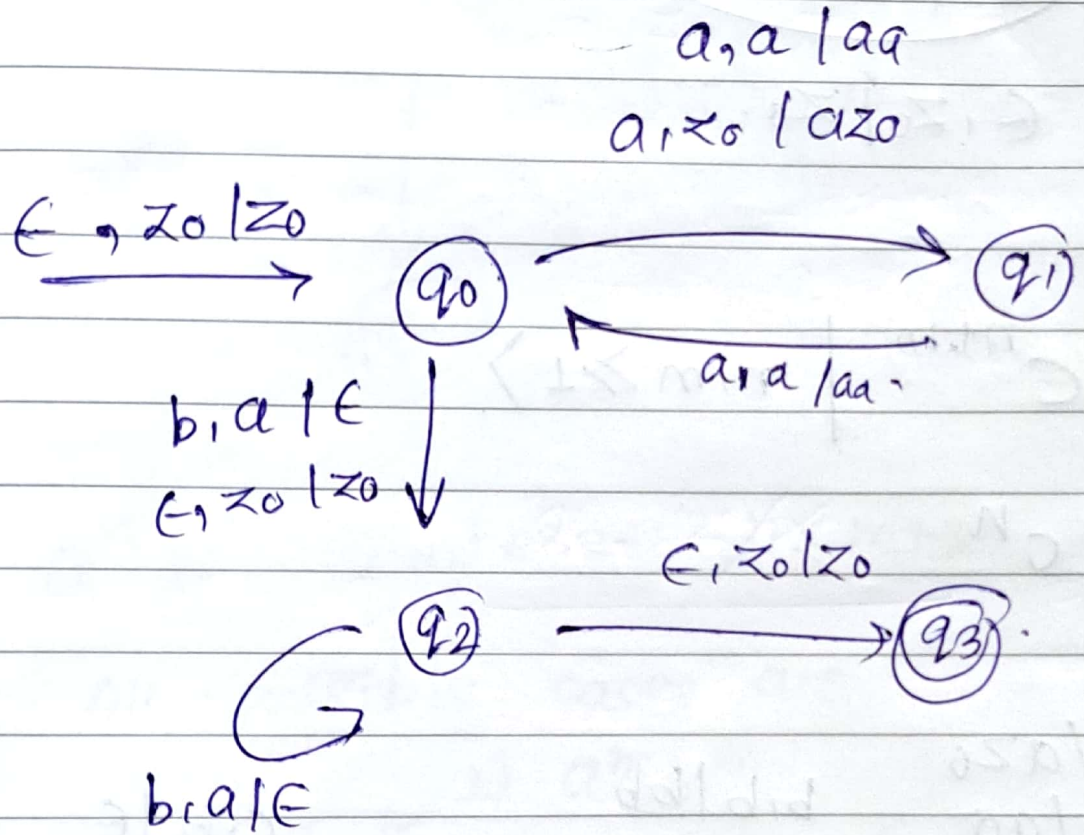
Q $\{ WW^R \mid W \in (a,b)^* \}$

Non-deterministic PDA

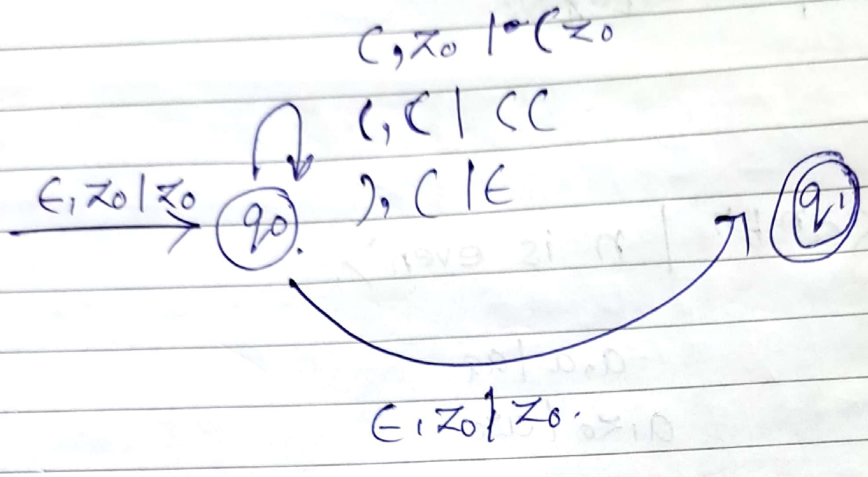


Q. $L = \{a^n b^n \mid n \text{ is even}\}$.

ans:

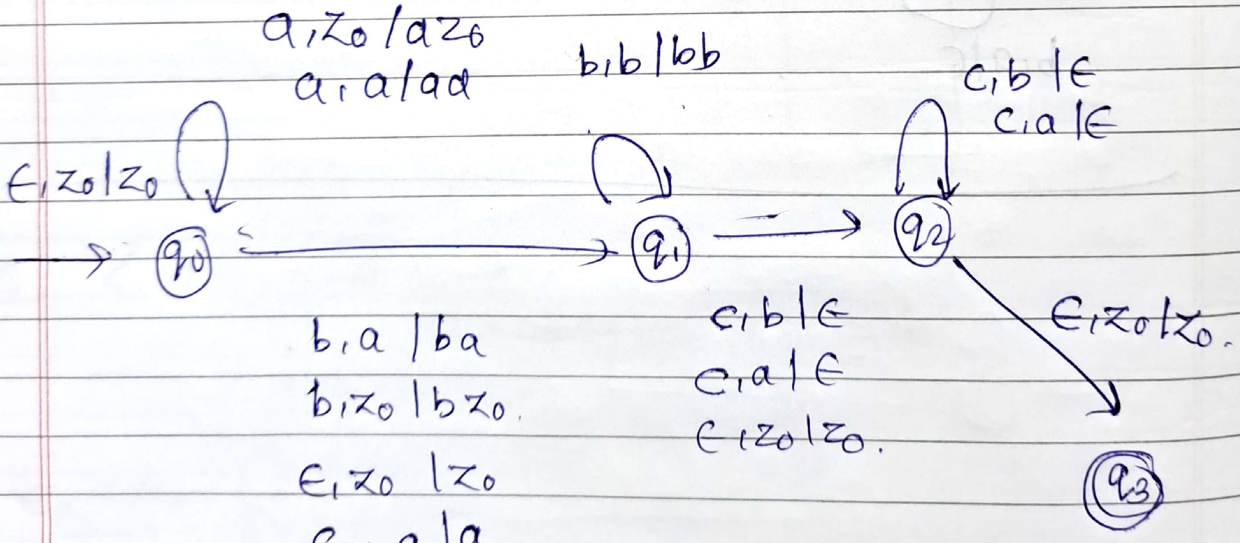


Q Balanced parenthesis :-



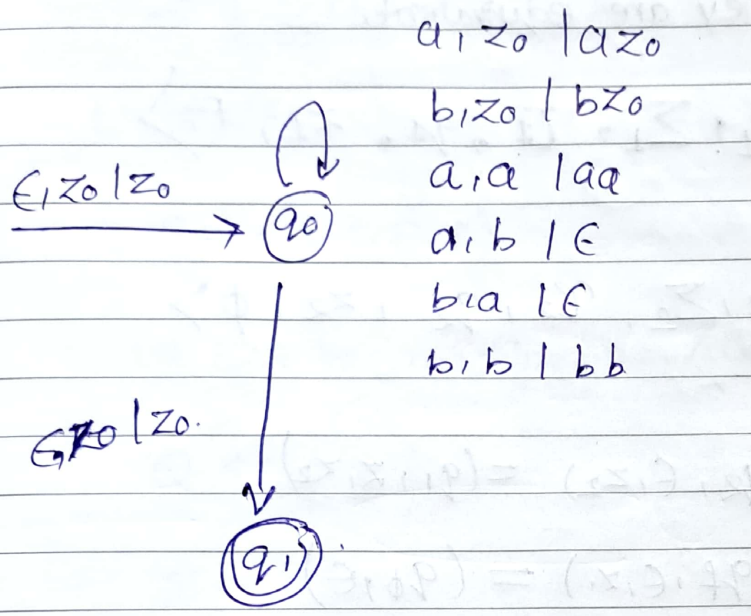
Q $L = \langle a^n b^m c^{n+m} \mid n,m \geq 1 \rangle$

ans :- $a^n b^m c^m c^n$



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

ans:-



Q] $a^n b^m$, where $n \leq m+3$

ans:- All possible cases are.

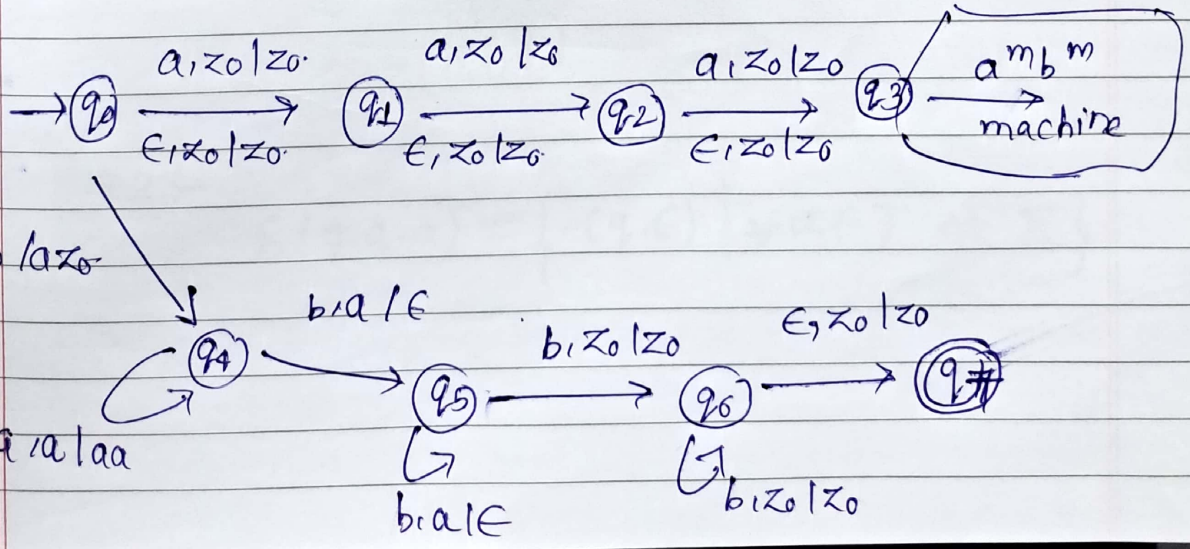
1) $a^m b^m$

2) $a^{m+1} b^m$

3) $a^{m+2} b^m$

4) $a^{m+3} b^m$

5) $a^n b^m$ $\quad n < m$



Q] There are 2 machines M_1 and M_2. Prove they are equivalent.

ans:-

$$M_1 = \langle Q_1, \Sigma_1, \tau_1, q_1, z_1, F_1 \rangle$$

$$M_2 = \langle Q_2, \Sigma_2, \tau_2, q_2, z_2, \phi \rangle$$

$$\delta(q_2, \epsilon, z_2) = (q_1, z_1 z_2)$$

$$\delta(q_f, \epsilon, z_1) = (q_\phi, \epsilon)$$

$$\delta(q_\phi, \epsilon, z_2) = (q_\phi, \epsilon)$$

15 marks out of 30 — 9th oct — ~~next~~ - ~~next~~ - monday

5 marks quiz

5 marks Question & 5 marks Execution

classmate

Date

Page

:- PDA & CFL :-

L is CFL.

$G = \langle V, T, P, S \rangle$

Let PDA be machine $M = \langle Q, \Sigma, \tau, \delta, q_0, z_0, \phi \rangle$

$Q \leftarrow \{ q \}$

$\Sigma \leftarrow T$

$\tau \leftarrow V \cup T$ (∪ → union)

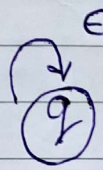
$q_0 \leftarrow q$

$z_0 \leftarrow S$

First Step :- CFL → PDA

Rule 1 :-

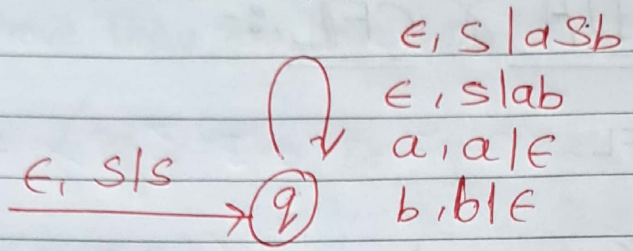
$$\delta(q, \epsilon, A) = \{ (q, \alpha) \mid A \rightarrow \alpha \text{ in } P \}$$



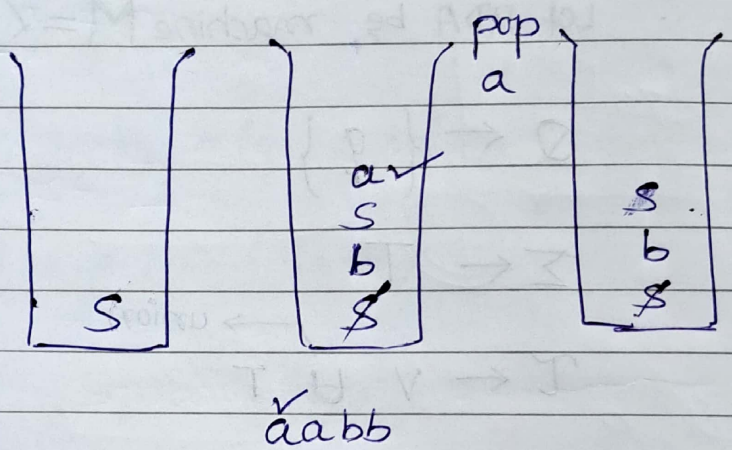
Rule 2 :-

$$\delta(q, a, a) = \{ (q, \epsilon) \mid \forall a \in T \text{ or } \Sigma \}$$

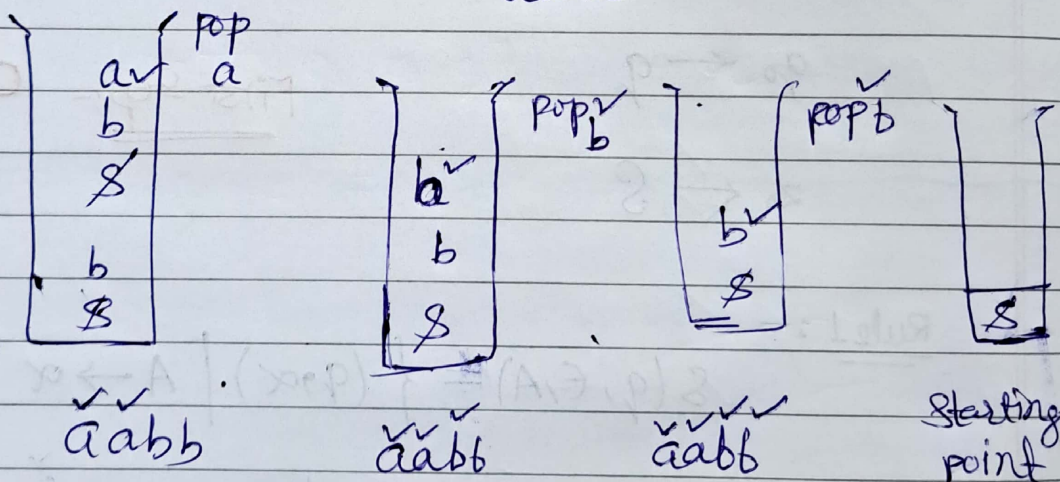
$S \rightarrow aSb \mid ab$



ex:- aabb



aabb

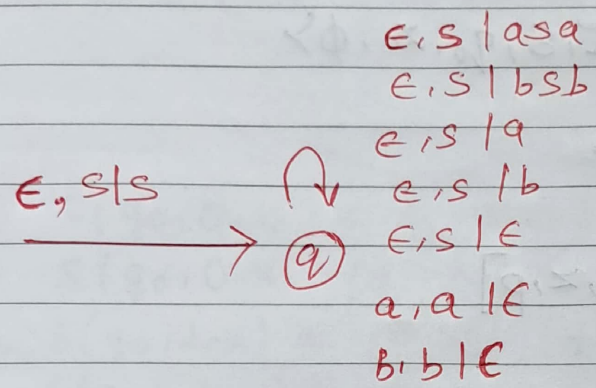


aabb aabb aabb

Note:- In exam be sure if empty stack or final state mentioned

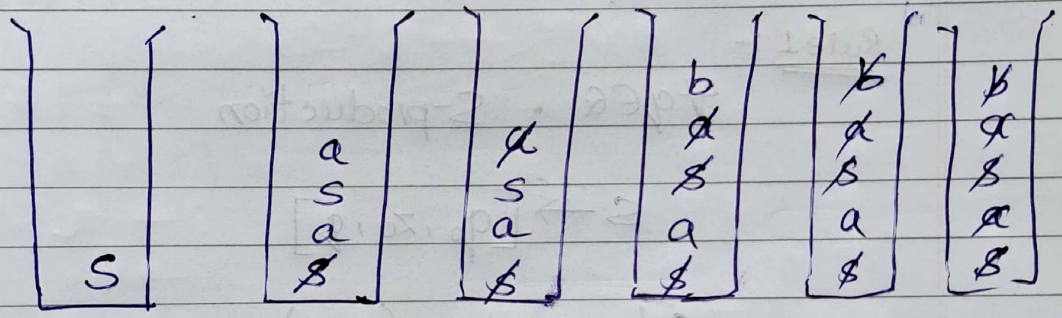
2] Palindrome with (a,b).

$S \rightarrow asa \mid bsb \mid a \mid b \mid \epsilon$

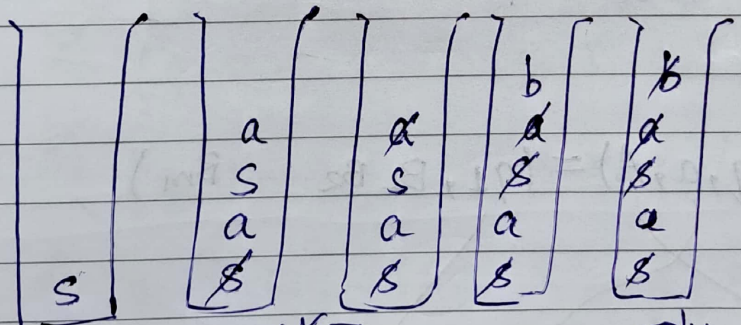


ex:- aba

ϵ, S | asa a, a | ϵ ϵ, S | b b, b | ϵ a, a | ϵ



ex:- abb



✓ ✓
a b b
x

only a can be popped now, Hence rejected.

Second step : PDA → CFL

You have PDA, you have to generate Grammar.

$$M = \langle Q, \Sigma, \Gamma, \delta, q_0, z_0, \phi \rangle$$

$$G = \langle V, T, P, S \rangle$$

$$V = \{S\} \cup [q, z, p]$$

$$q, p \in Q \quad , z \in Z$$

Rule 1 :-

$\forall q \in Q$, S-production

$$S \rightarrow [q_0, z_0, q]$$

Rule 2 :- $\delta(q, a, z) = (p, \epsilon)$

$$[q, z, p] \rightarrow a$$

Rule 3 :-

$$\delta(q, a, A) = (q_1, B_1 B_2 \ldots\ldots B_m)$$

$$[q, A, q_{m+1}] \rightarrow a\,[q_1, B_1, q_2]$$

$$[q_2, B_2 q_3]$$

$$\vdots$$

$$[q_m B_m q_{m+1}]$$

where $a \in \Sigma \cup \{\epsilon\}$

$A, B_1, B_2, B_3, \ldots B_m \in \tau$

Q
1) $\delta(q_0, 0, z_0) = (q_0, Xz_0)$
2) $\delta(q_0, 0, X) = (q_0, XX)$
3) $\delta(q_0, 1, X) = (q_1, \epsilon)$
4) $\delta(q_1, 1, X) = (q_1, \epsilon)$
5) $\delta(q_1, \epsilon, X) = (q_1, \epsilon)$
6) $\delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$

$Q = \{q_0, q_1\}$
$\Sigma = \{0, 1\}$

ans:-

1) $S \rightarrow [q_0, z_0, q_0]$
2) $S \rightarrow [q_0, z_0, q_1]$
3) $[q_0, z_0, q_0] \rightarrow 0[q_0, X, q_0][q_0, z_0, q_0]$
4) $[q_0, z_0, q_0] \rightarrow 0[q_0, X, q_1][q_1, z_0, q_0]$
5) $[q_0, z_0, q_1] \rightarrow 0[q_0, X, q_0][q_0, z_0, q_1]$
6) $[q_0, z_0, q_1] \rightarrow 0[q_0, X, q_1][q_1, z_0, q_1]$
7) $[q_0, X, q_0] \rightarrow 0[q_0, X, q_0][q_0, X, q_0]$
8) $[q_0, X, q_0] \rightarrow 0[q_0, X, q_1][q_1, X, q_0]$
9) $[q_0, X, q_1] \rightarrow 0[q_0, X, q_0][q_0, X, q_1]$
10) $[q_0, X, q_1] \rightarrow 0[q_0, X, q_1][q_1, X, q_1]$

11) $[q_0, X, q_1] \rightarrow 1$

12) $[q_1, X, q_1] \rightarrow 1$

13) $[q_1, X, q_1] \rightarrow \epsilon$

14) $[q_1, Z_0, q_1] \rightarrow \epsilon$

Not all 14 rules are useful, one must remove the useless productions.

Rules like $S \rightarrow aSb$ are non-terminating, hence useless.

also rules like $S \rightarrow aSB$

$B \rightarrow aBa$

both are useless then.

* Terminating rules may also be useless if not used.

---

* Till this is syllabus. for Lab Evaluation *

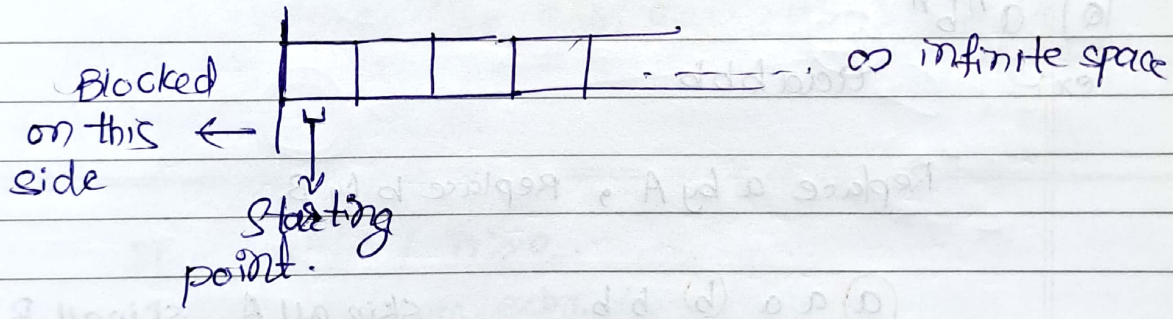
CFL is closed under Union, Concatenation, & kleen closure, but not intersection.

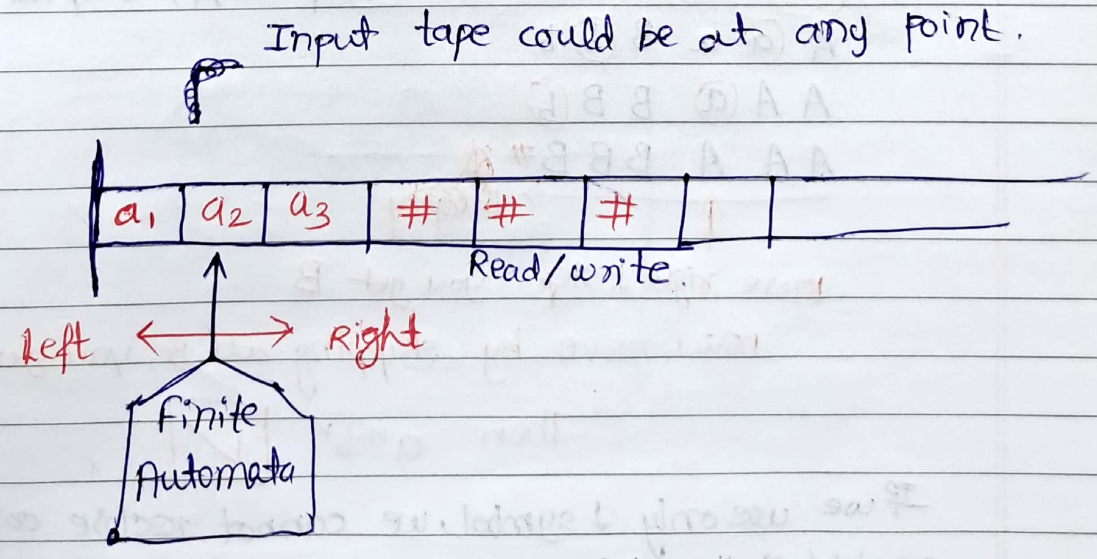
CFL could not :- WW, WWW, copying a string, arithmetic operations, etc.

Till now, we were only reading & verifying but not writing

Now, movement of header (/input tape) is allowed in any direction with help of Turing Machine

Turing Machine





$TM = \langle Q, \Sigma, \tau, \delta, q_0, \#, F \rangle$

Blank symbols

$(\#, B, \square, b)$

$\tau \leftarrow \Sigma \cup \{\#\} \cup \bigcirc$

↳ new symbol read on input tape.

$\delta : Q \times \tau \longrightarrow Q \times \tau \times \{L, R\}$

left Right.

Q $a^n b^n$

ex- aaabbb

Replace a by A, Replace b by B.

(a) a a (b) b b. skip all A, skip all B

A (a) a B (b) b

A A (a) B B (b)

A A A B B B #

move right 1 step, you get B

now, move by skipping all B, you get #

Then $a^n b^n$ ✓

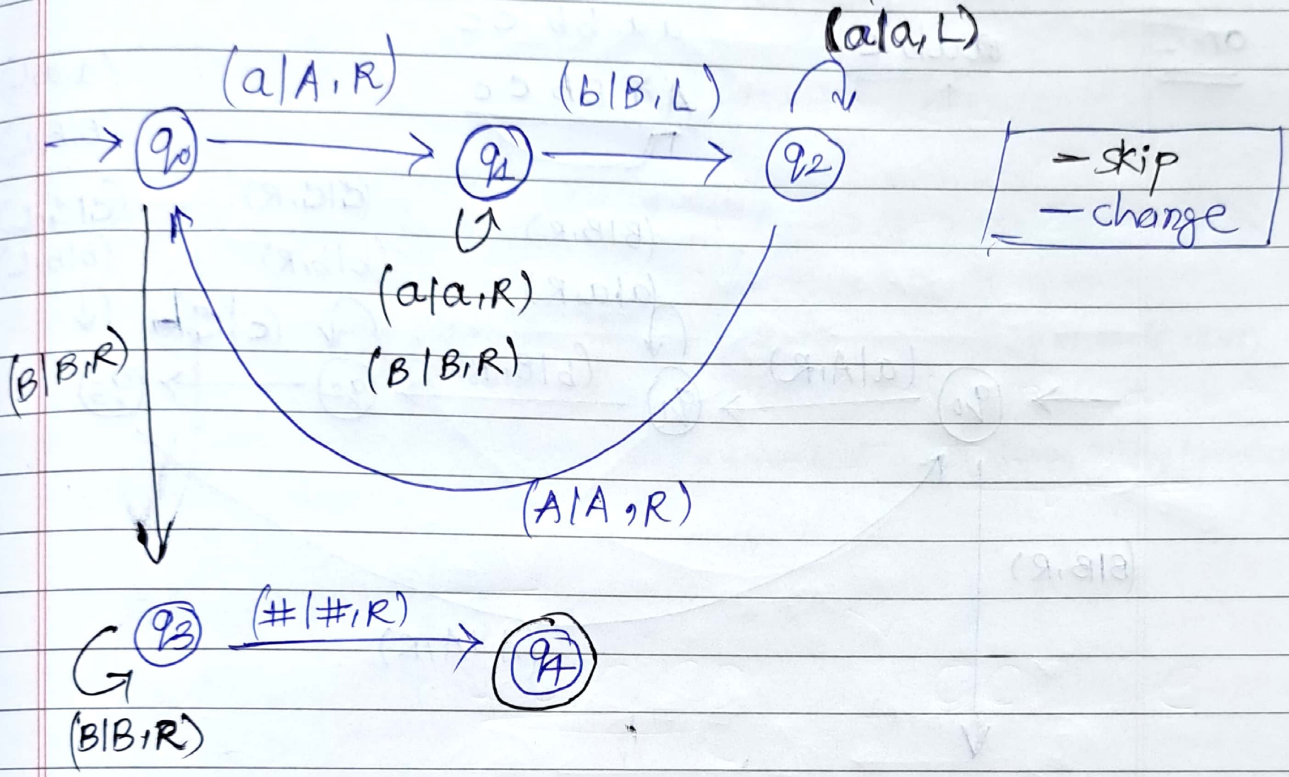
If we use only 1 symbol, we cannot restore original string.

aaabbb ⇒ #aa#bb ⇒ ### ### ⇒ × aaabbb.

Turing Machine

$TM = \langle Q, \Sigma, \tau, \delta, \#, F \rangle$

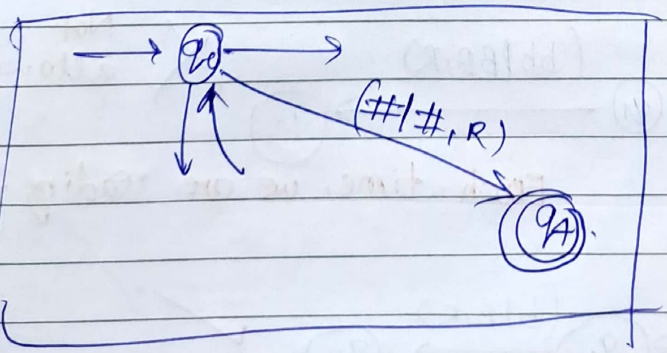
(τ → Tape symbols in total)



strings like abb, aab, abab will be rejected.

If $a^n b^n \mid n \geq 0$.

same diagram, extra step.

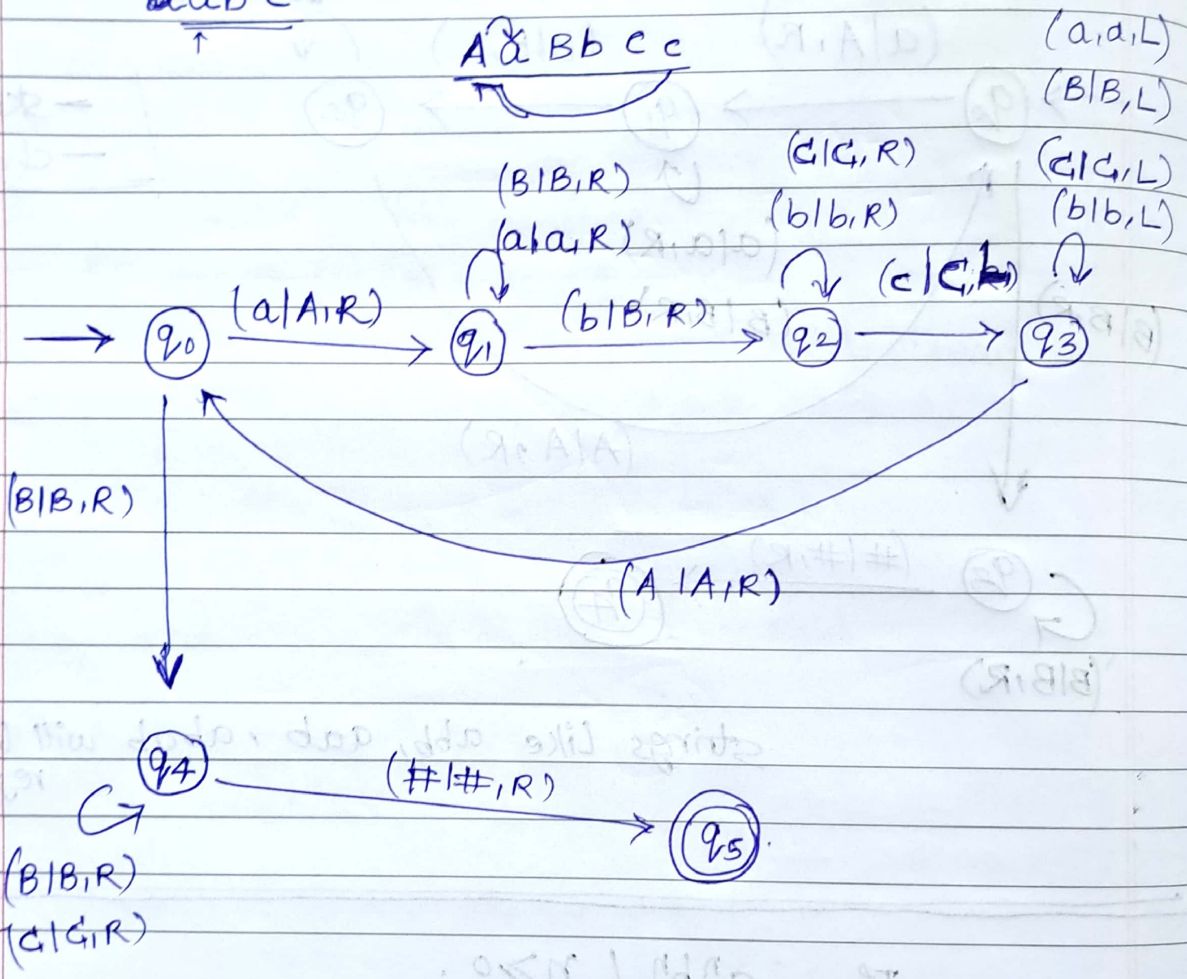


q] $a^n b^n c^n \mid n>0$

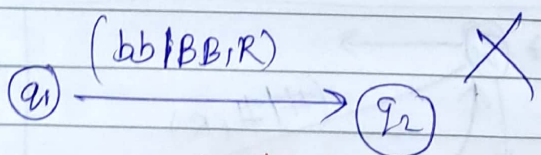
ans:- abc

aa bb cc

A a Bb c c

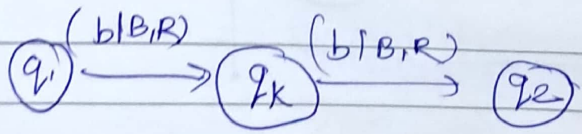


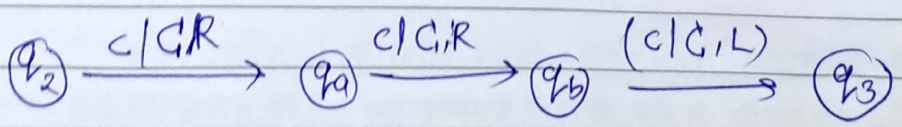
q] $a^n b^{2n} c^{3n}$ abbccc.



Not allowed.

Each time, we are reading 1 symbol.





In case of palindrome strings, we can replace by single symbol '#' instead of A, B.

(∵ # will not be in middle of string)

K is any symbol

ex:-1 a b b a

start (K ⇒ #)

move Last (K ⇒ #)

again start ⇒ skip # ⇒ (K ⇒ #)

Repeat.

WW^R Type # b b #

# # # #

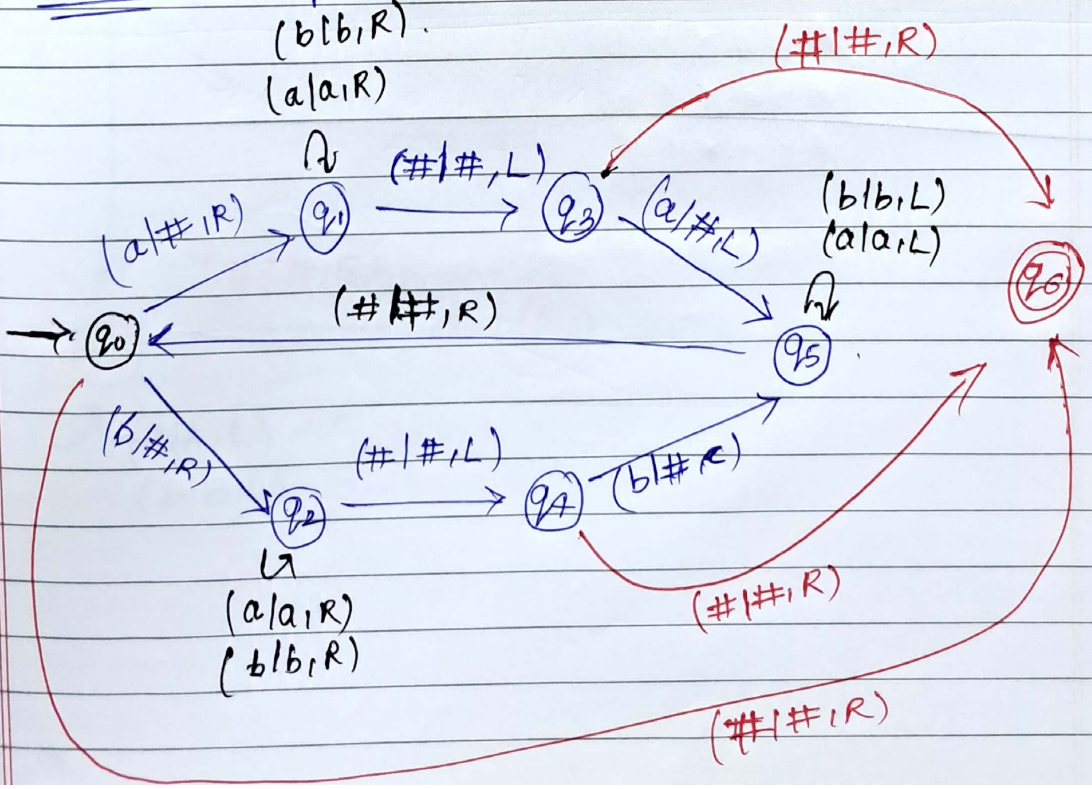
String accepted, for Even Length.

ex:-2 a b a.

WcW^R Type # b #.

accepted.

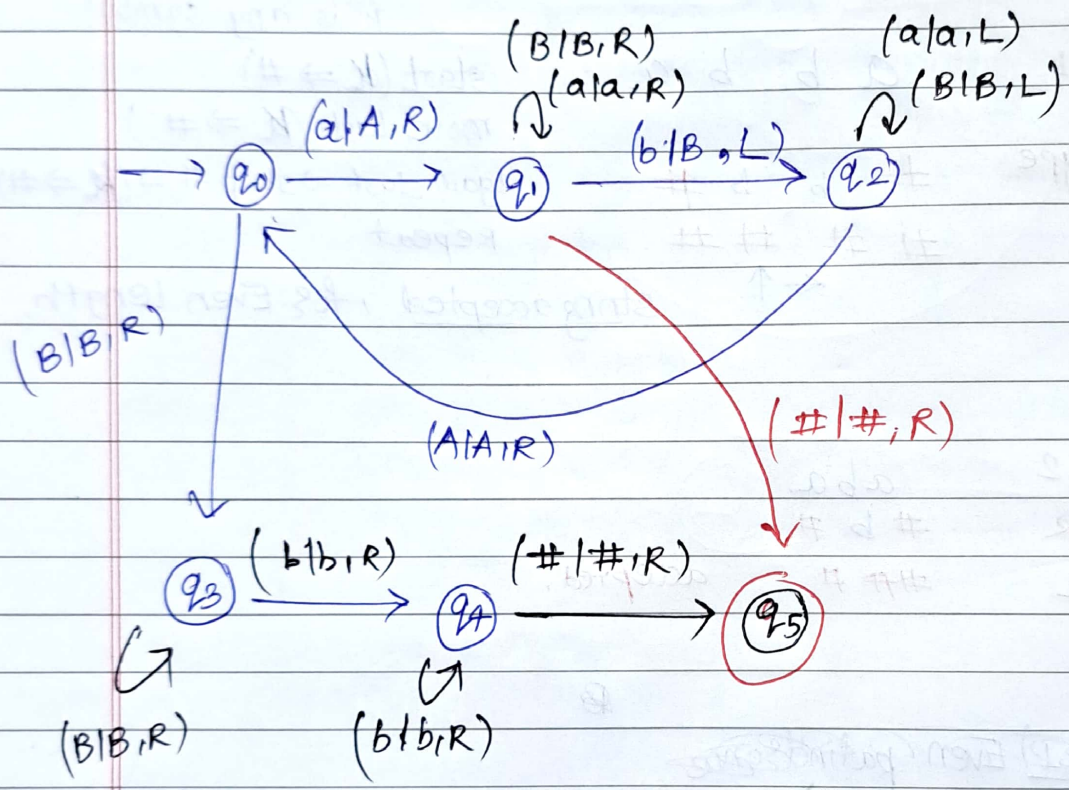
ex:-3 Even palindrome



Q) $a^n b^m \mid n \geq 1 \text{ \& } n \neq m$

ans:- Draw $a^n b^m \mid n \geq 0$.

Take ~~final state~~, add 1 a, Later ~~self loop 'a'~~ self loop 'b'.



abbba

ABBB

ABbba

ABbbA

ABBbA

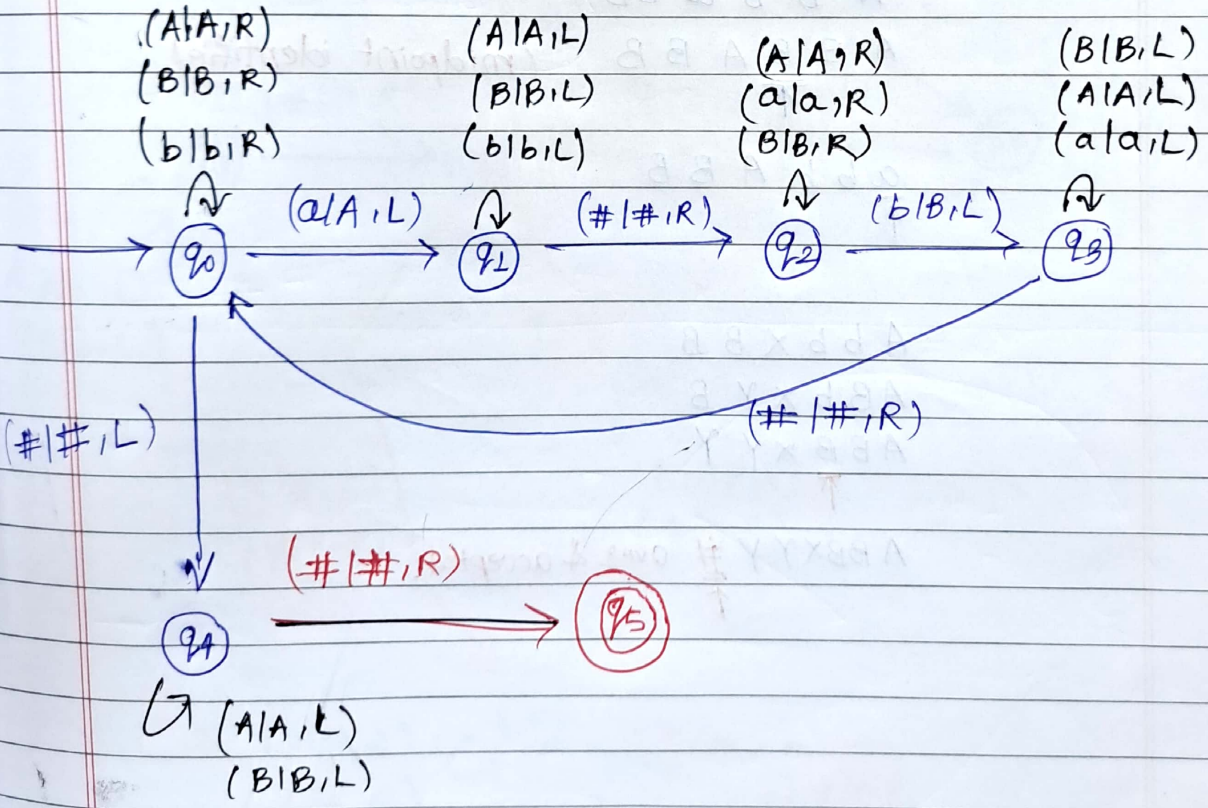
a) $L = \{ w \mid w \in (a,b)^*, n_a(w) = n_b(w) \}$.

case1) change 1a, change 1b, alternately.
skip all capitals, If # in end ⇒ accepted

ex
b b a b a b a a
b b A b a b a a
B b A b a b a a
B b A b A b a a
↑

Here we have to come again & again to starting point.

B B A B A B A A #
→ ↑



Q] $L = \{ww \mid w \in (a,b)^*\}$.

1) Find middle point by capitalising both left & right extrememe

2) Now, from middle, move left & smallise the left string.

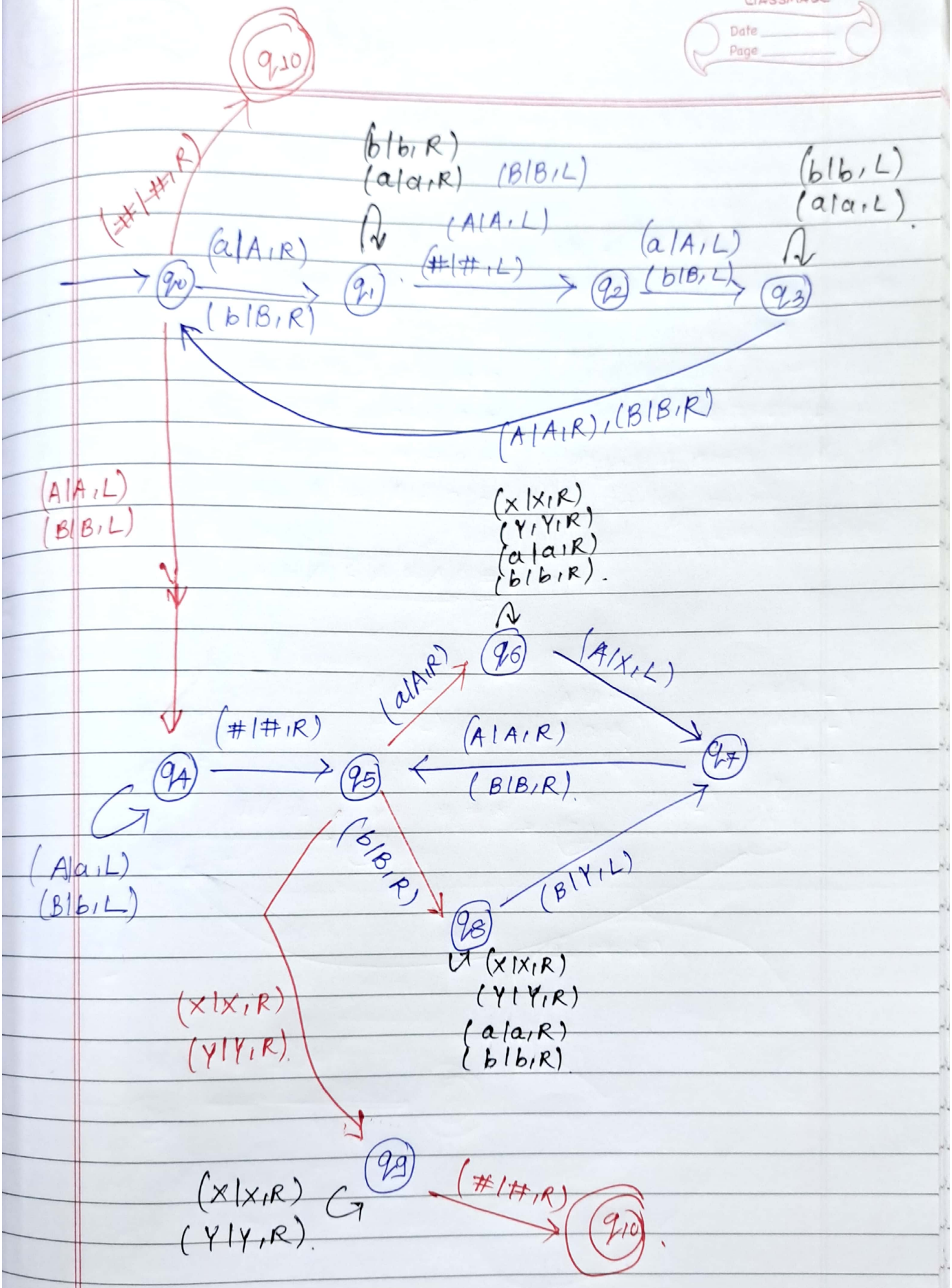
3) scan left string, change right string to new symbol.

a b b a b b
A b b a b B
A B b a B B
A B B A B B (midpoint identified)

a b b A B B

A b b X B B
A B b X Y B
A B B X Y Y

A B B X Y Y # over & accepted.


q10
(#|#,R)
(b|b,R)
(a|a,R)
(B|B,L)
(A|A,L)
(#|#,L)
(b|b,L)
(a|a,L)
(a|A,R)
(b|B,R)
q0
q1
q2
q3
(A|A,R),(B|B,R)
(A|A,L)
(B|B,L)
(x|x,R)
(y,y,R)
(a|a,R)
(b|b,R).
q6
(a|A,R)
(A|x,L)
(#|#,R)
(A|A,R)
(B|B,R).
q4
q5
q7
(A|a,L)
(B|b,L)
(b|B,R)
(B|Y,L)
q8
(x|x,R)
(y|y,R)
(a|a,R)
(b|b,R).
(x|x,R)
(y|y,R).
q9
(x|x,R)
(y|y,R).
(#|#,R)
q10.

1) Regular $(a^n b^n)$ Test answers

2) L-R = context Free,

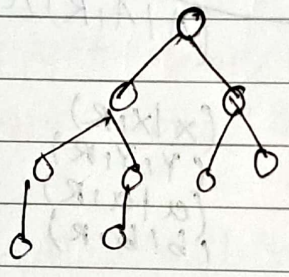
3) DFA = NFA = ϵ-NFA < DPDA < NPDA.

4) Chomsky Normal form

$S \rightarrow AB$

$A \rightarrow a$

5) This has parse tree same as binary tree



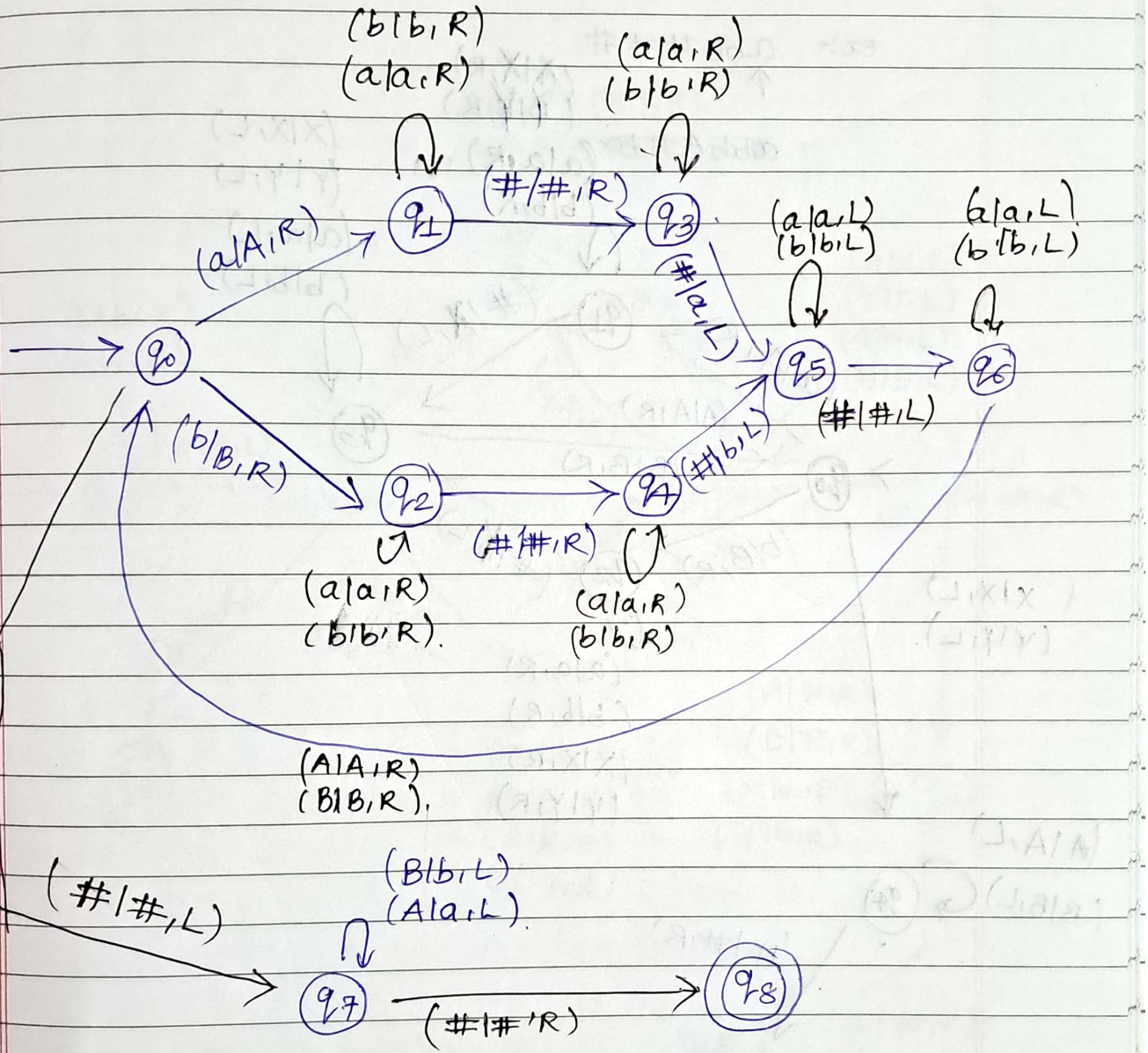
$(2l-1)$ derivations

6) $0^{n_1} m_0^{n_1} m \Rightarrow$ PDA does not exist

Copy of a string $w \in (a|b)^*$

Input :- W

output :- W # W



ex:- abb#####-

Abb#a##-

ABb#ab#-

ABB#abb-

ABb#abb-

Abb#abb-

abb#abb

Q) WW

input :- W
output :- WW.

ex :- abb###

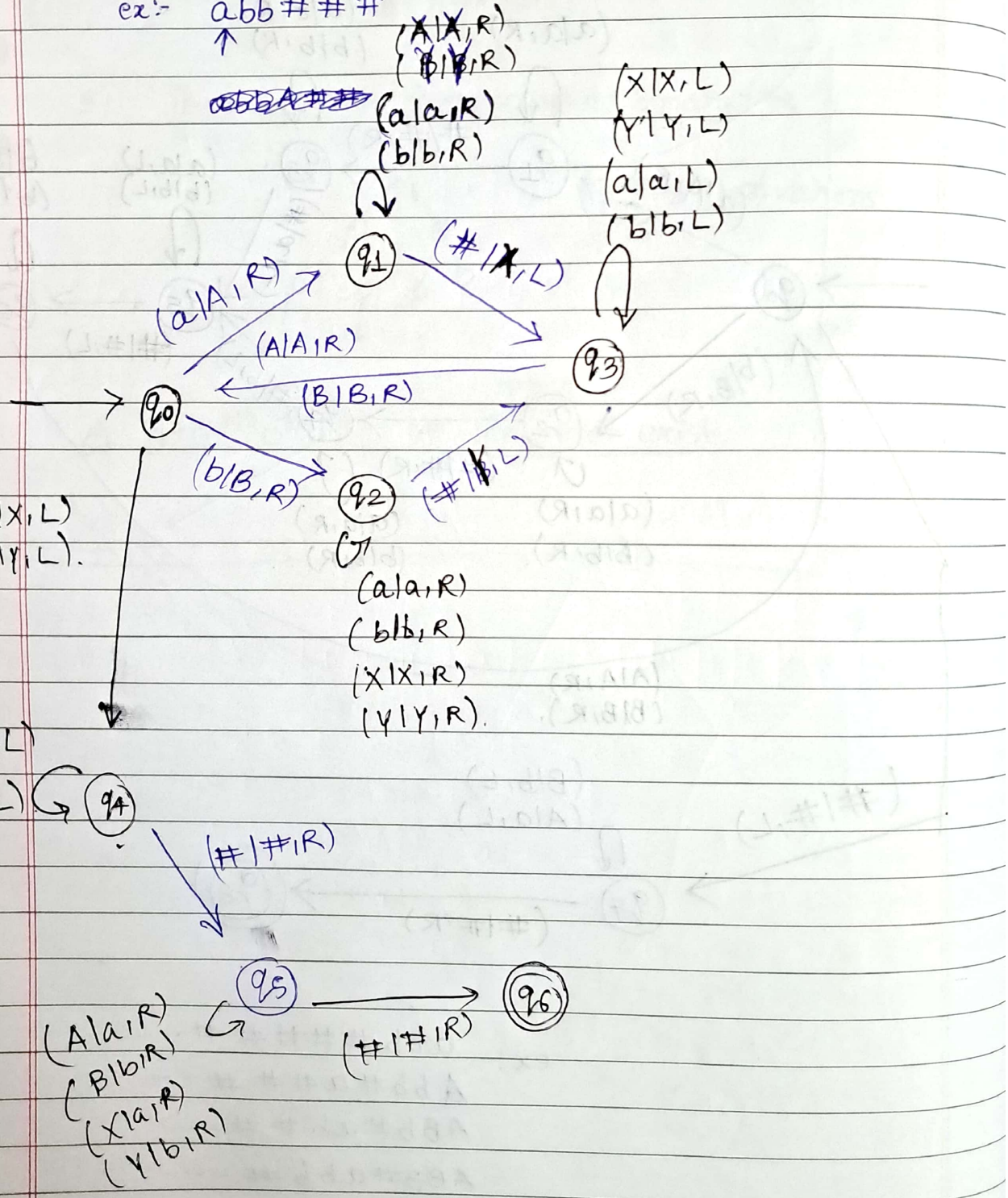
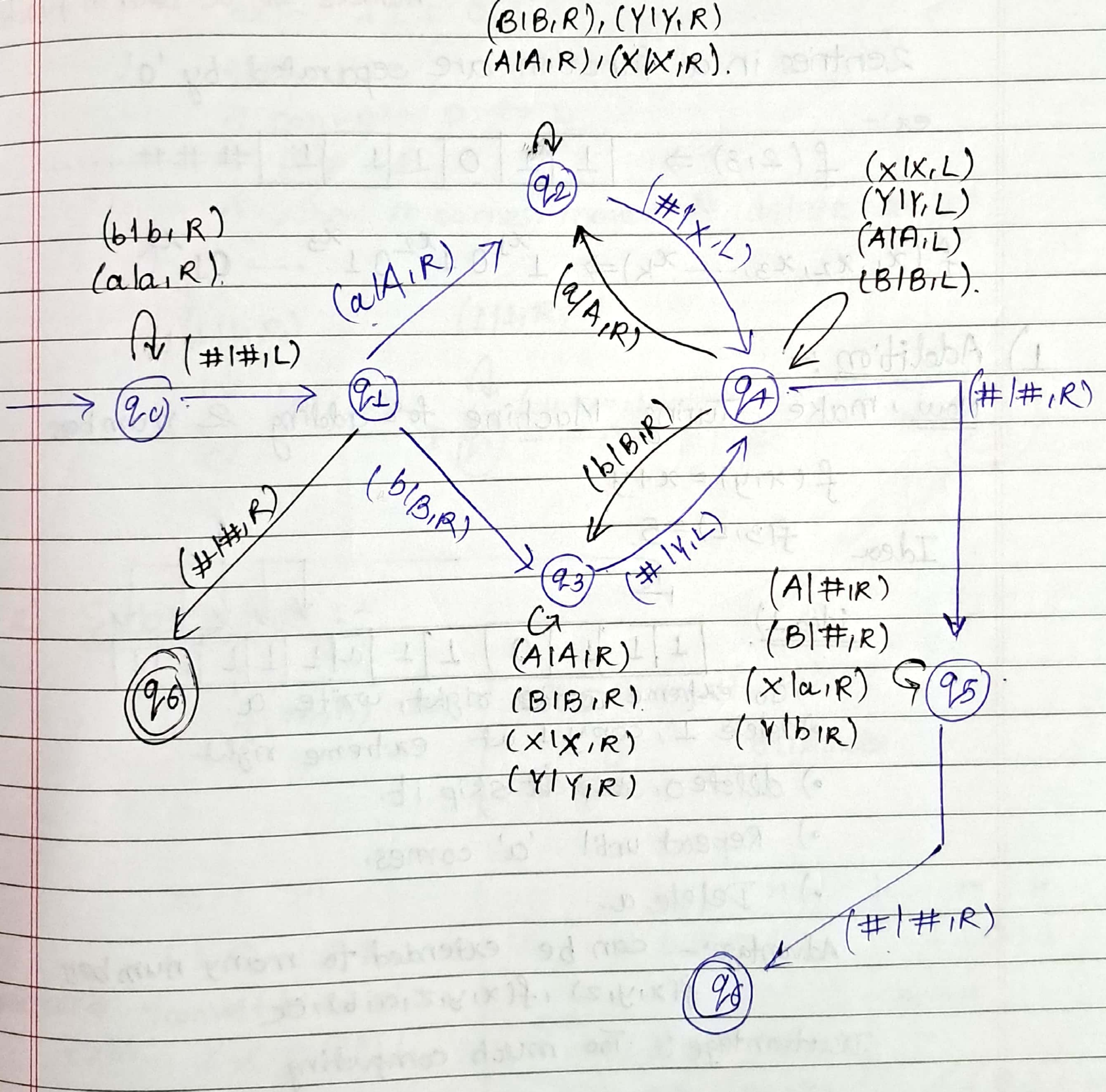


abB X##

a bb

Q] create reverse of a given string.

ans:- # abb



Arithmetic Turing Machine

Any number is represented as many times '1'.

ex:- $2 = 11$

$1 = 1$

$5 = 11111$.

$2 = 110$ invalid

↳ indicates the number will be used in function

2 entries in a function are separated by '0'.

ex:-

$f(2,3) \Rightarrow$ | 1 | 1 | 0 | 1 | 1 | 1 | # # # .....

$f(x_1, x_2, x_3, \ldots x_k) \Rightarrow 1^{x_1} 0 1^{x_2} 0 1^{x_3} \ldots 0 1^{x_k}$

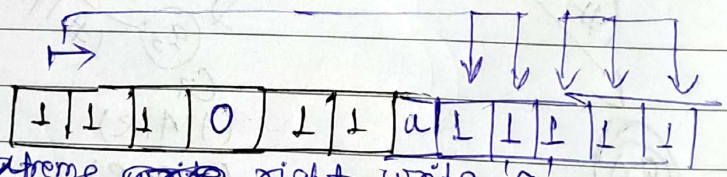
1) Addition :-

Now, make Turing Machine for adding 2 numbers.

$f(x,y) = x+y$

Idea $f(3,2) = 5$.

idea 1).



- Go extreme right, write 'a'.
- delete 1, copy 1 at extreme right
- delete 0, skip it.
- Repeat until 'a' comes.
- Delete a.

Advantage:- can be extended to many numbers

$f(x,y,z)$, $f(x,y,z,a,b)$, etc

Disadvantage :- Too much computing.

ex:-

$11\,0\,11$

$111\,0\,11\,a$

$X\,1\,X\,0\,1\,1\,a\,1\,1\,1\,1$

$\#\,1\,1\,1\,1\,1$

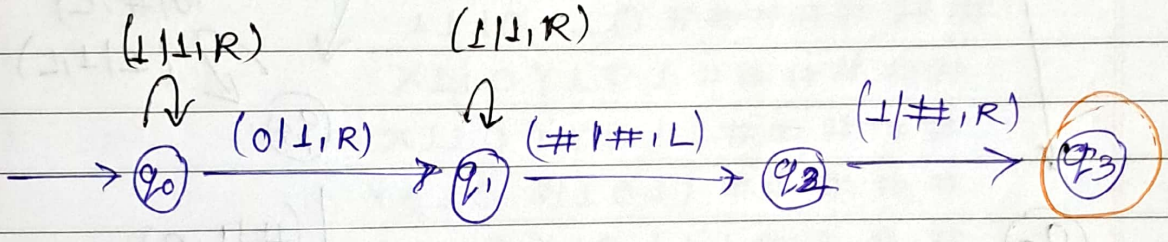
idea 2 :- •) very less computing.
•) can be extended to multiple numbers using Recursion.

Preferred solution

Eg $f(x,y,z) = f(f(x,y),z)$

ex:- 1 1 1 0 1 1

•) skip 1's.
•) convert 0 to 1
•) skip 1's
•) When # comes, move left delete a '1'.



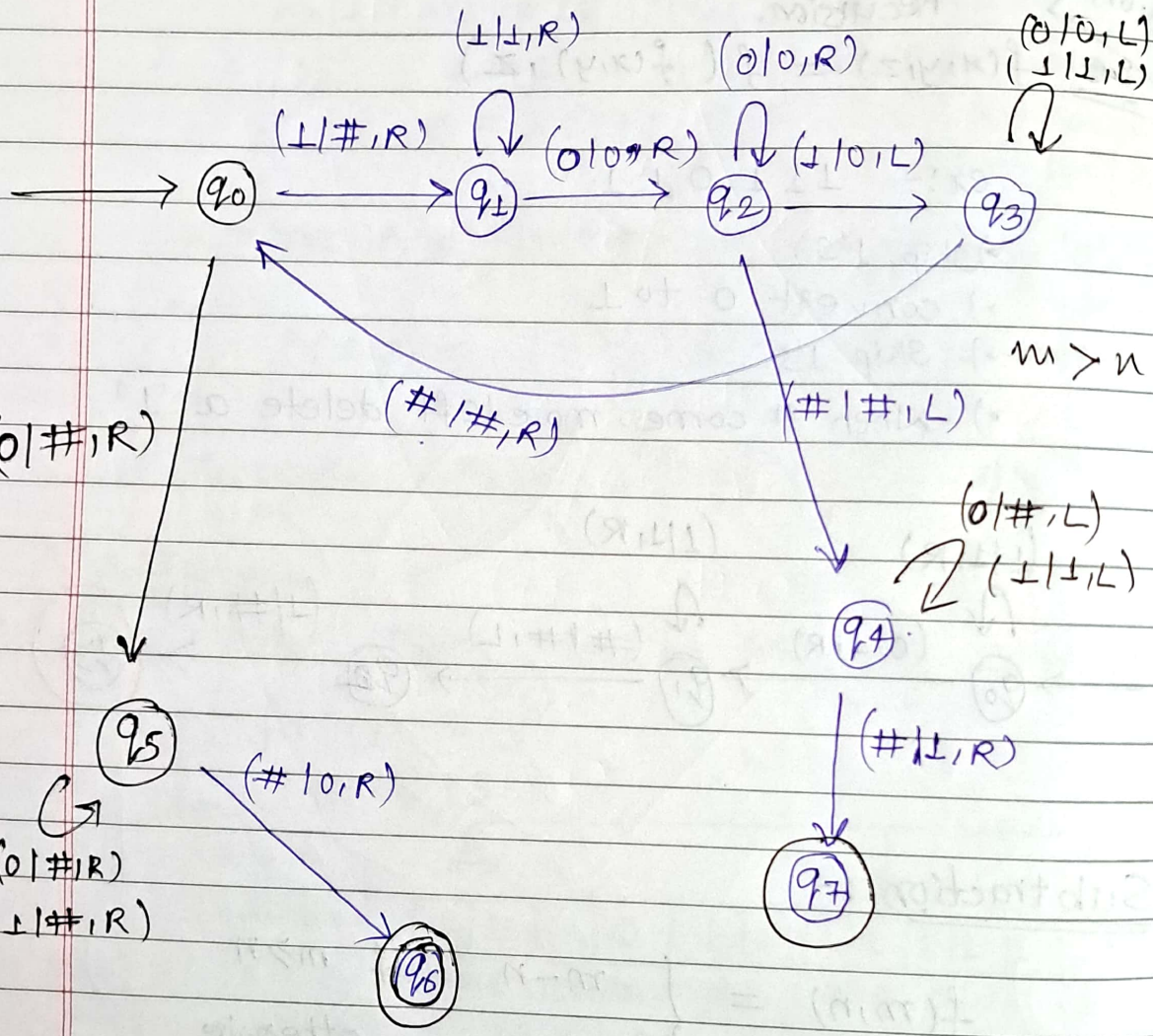
2) Subtraction :-

$$f(m,n) = \begin{cases} m-n & \text{if } m \geq n \\ 0 & \text{otherwise.} \end{cases}$$

m n
1 1 0 1 1 1
•) convert '1' to '0',
n>m X X 0 0 0 1
•) delete everything
X X 0 0 0 1
0
will use this ideal

m n
1 1 1 0 1 1
n<m
1 X 1 0 X X #
ans:- m-n
important

m n
1 1 0 1 1
m=n
X X 0 X X #
ans:- 0


(⊥|⊥,R)
(0|0,R)
(0|0,L)
(⊥|⊥,L)
(⊥|#,R)
(0|0,R)
(⊥|0,L)
q0
q1
q2
q3
m > n
(#|#,R)
(#|#,L)
(0|#,R)
(0|#,L)
(⊥|⊥,L)
q4
q5
(#|0,R)
(#|⊥,R)
(0|#,R)
(⊥|#,R)
q7
q6

Multiply using Turing machine.

$$f(x,y) = x * y$$

1 1 0 1 1 — input

1 1 1 1 1 1 — expected output :-

We know $W \longrightarrow WcW$.

will use that algorithm then

Add 0 in end of string

1 1 1 0 1 1 0 # # # # # # # #

X 1 1 0 Y 1 0 1 # # # # # #

X 1 1 0 Y Y 0 1 1 # # # # #

X 1 0 Y 1 0 1 1 # # # #

X X 1 0 Y 1 0 1 1 1 # # #

X X 1 0 Y Y 0 1 1 1 1 # #

X X 1 0 1 1 0 1 1 1 1 # #

X X X 0 Y Y 0 1 1 1 1 1 #

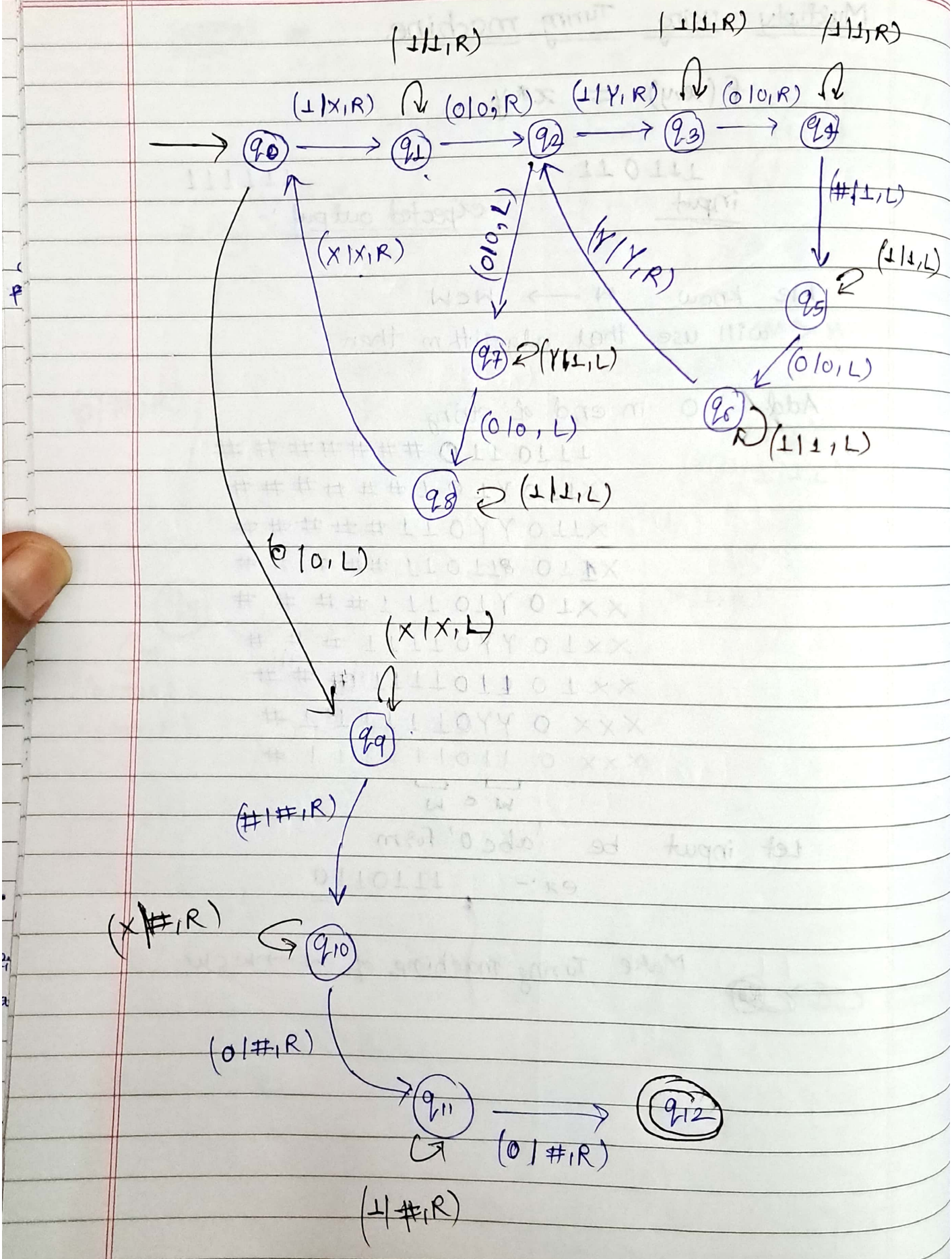
X X X 0 1 1 0 1 1 1 1 1 #

W c W

Let input be 'abc 0' form

ex:- 1 1 1 0 1 1 0

Make Turing machine of $W \rightarrow WcW$


(⊥|⊥,R)
(⊥|⊥,R)
(⊥|⊥,R)
(⊥|X,R)
(0|0,R)
(⊥|Y,R)
(0|0,R)
q0
q1
q2
q3
q4
(#|⊥,L)
(X|X,R)
(0|0,L)
(Y|Y,R)
(⊥|⊥,L)
q5
q7
(Y|⊥,L)
(0|0,L)
q6
(0|0,L)
(⊥|⊥,L)
q8
(⊥|⊥,L)
(0|0,L)
(X|X,L)
q9
(#|#,R)
(X|#,R)
q10
(0|#,R)
q11
q12
(0|#,R)
(⊥|#,R)

Division

$$f(x,y) = \begin{cases} (x/y), & \text{if } x \geq y \\ 0, & \text{otherwise} \end{cases}$$

Don't delete the last 0, it will be useful when $x < y$.

~~#~~ 1 1 0 1 1 1 0
1 0 × 1 1 0
0 × × 1 0
0 ✓
save this.

otherway (delete 0)

if 0: # # # # # # ∅ # ⇒ # # # # 0 (change # 0)

if 1's: # # # # ∅ 1 1 ⇒ stop if 1. (#)

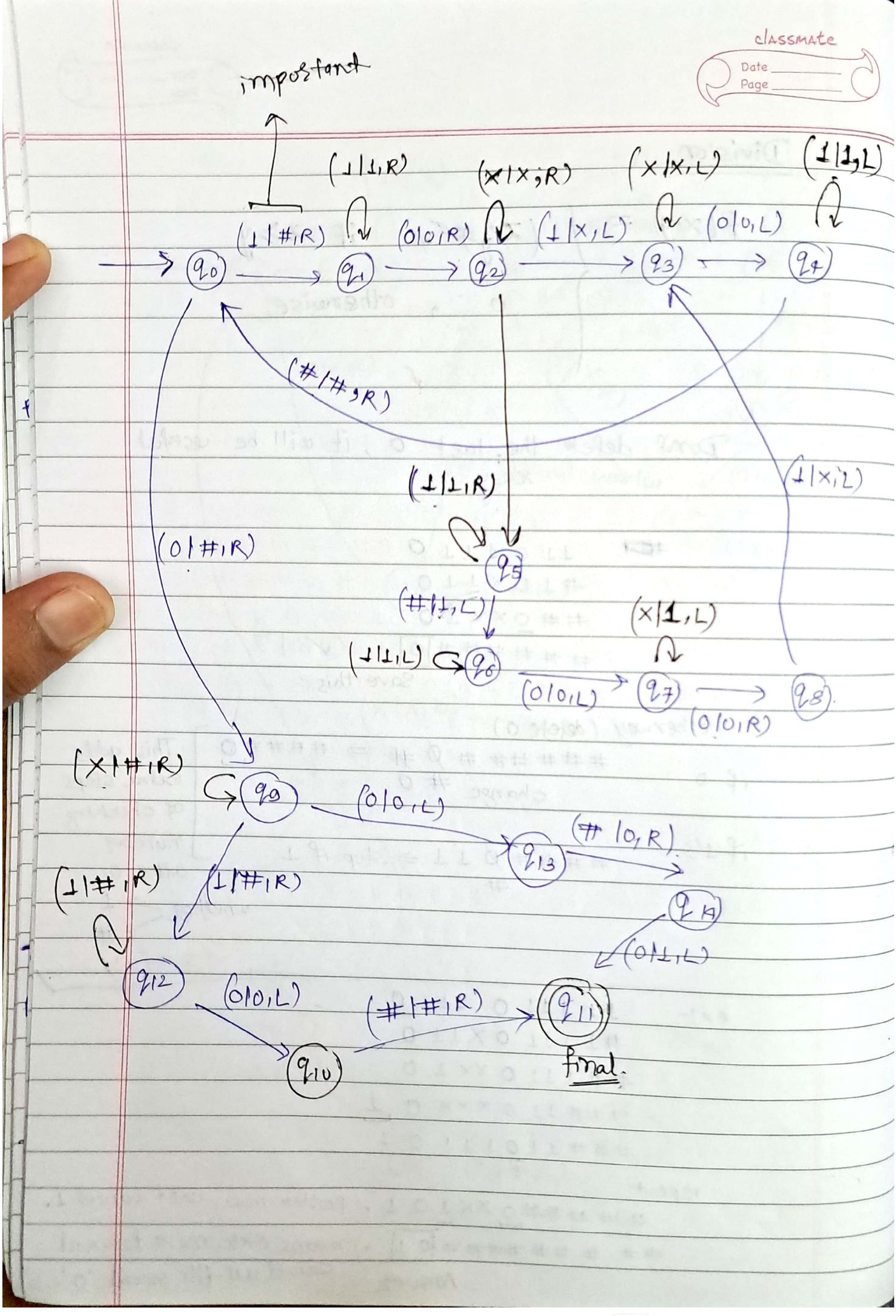
This adds extra steps of checking number after 0. whether → 1 → #

---

ex:- 1 1 1 1 1 0 1 1 1 0
1 1 1 1 0 × 1 1 0
1 1 1 0 × × 1 0
1 1 0 × × × 0 1
1 1 0 1 1 1 0 1
↑

repeat

0 × × 1 0 1 .) Problem, now, can't cancel 1.
0 1 .) means over. move forward cancel all till second '0'.
Answer


important
(1|1,R)
(1|#,R)
(0|0,R)
(x|x,R)
(1|x,L)
(x|x,L)
(0|0,L)
(1|1,L)
q0
q1
q2
q3
q4
(#|#,R)
(1|1,R)
(1|x,L)
(0|#,R)
q5
(#|1,L)
(1|1,L)
q6
(x|1,L)
(0|0,L)
q7
q8
(0|0,R)
(x|#,R)
q9
(0|0,L)
q13
(# |0,R)
(1|#,R)
(1|#,R)
q14
(0|1,L)
q12
(0|0,L)
(#|#,R)
q11
q10
final.

ceiling function for number divided by 2, i.e $\lceil \frac{x}{2} \rceil$

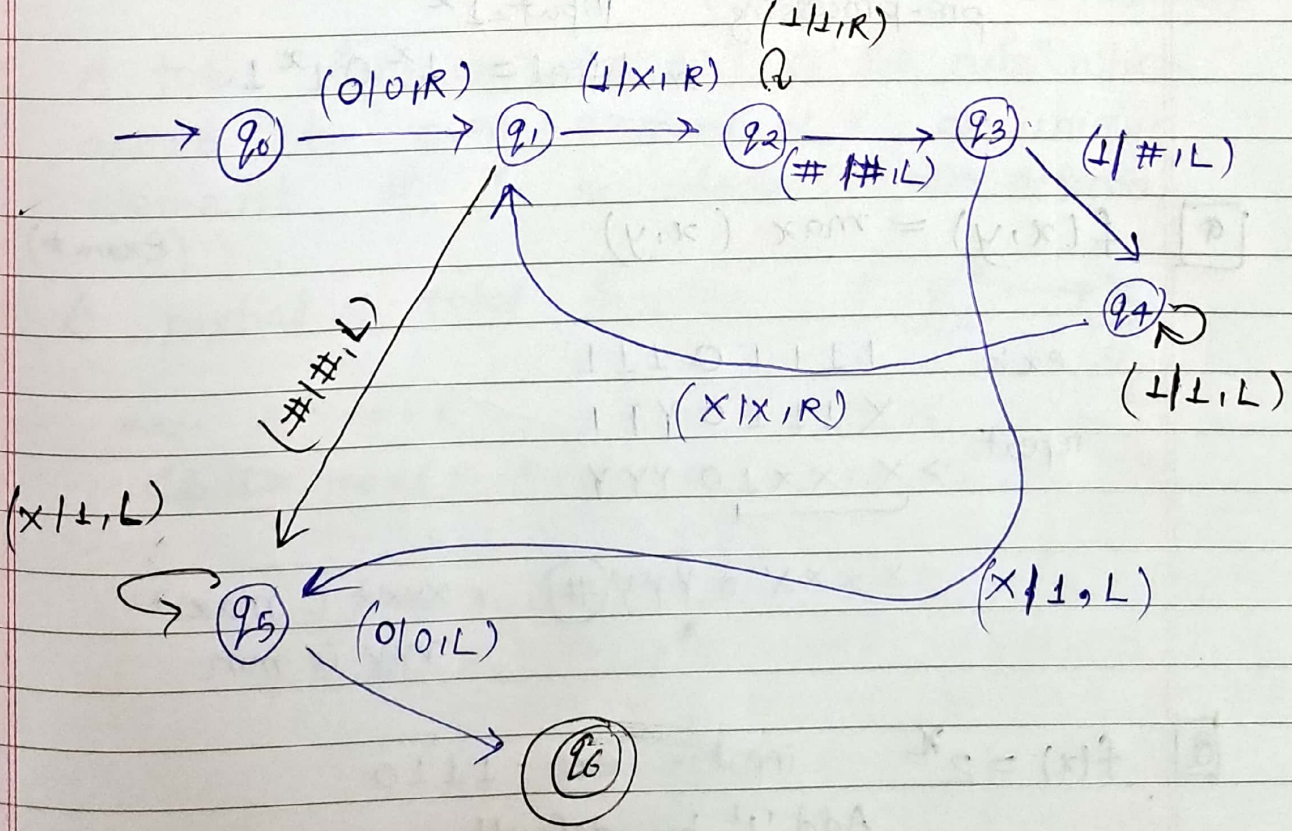
$$f(x) = \begin{cases} x/2 & \text{if } x \text{ is even} \\ \frac{x+1}{2} & \text{if } x \text{ is odd.} \end{cases}$$

input:- 0 1 1 1 1 0 1 1 1 1 1

process

0 X 1 1 #
0 X X # #.
↑
0 1 1
answer

0 X 1 1 1 #
0 X X 1 # #
↑ change all X to 1
0 1 1 1
answer



$f(x) = x^2$

input :- 1^x

pre-processing,

process :- $1^x 0$

copy :- $1^x 0 1^x$

* This is ready made ~~Input~~ Input for multiplication.

modified input :- $1^x 0 1^x$

$\downarrow f(x,y) = x * y$

over $f(x) = x^2$ achieved.

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

pre-processing input = 1^x

modified = $1^x 0 1^x 1$

Q $f(x,y) = max(x,y)$ (Exam *)

ex :- 1 1 1 1 0 1 1 1

repeat :- X 1 1 1 0 Y 1 1

$\Rightarrow$ X X X 1 0 Y Y Y

X X X X 0 Y Y Y (#) , XXXX is max.

YYY is min.

Q $f(x) = 2^x$ input :- ex :- 1 1 1 0

Add '1' by default

1 1 1 0 1 signifies 2^0

1 1 1 0 1

× 1 1 0 1 1 2^1

× × 1 0 1 1 1 2^2 copy

× × × 0 1 1 1 1 1 1 1 1 2^3 copy.

This works fine for 2^x.

does not work for $3^x, 4^x, x^y$, etc

Primitive Recursive Function :-

1) A partial function $f: X \to Y$ is a rule, which assigns to every element of X, atmost 1 element of Y. ex:- subtraction ($x-y$, if $x<y$ not defined)

2) A total function $f: X \to Y$ is the rule which assigns to every element of X, an unique element of Y. ex:- addition (all cases defined)

3) A partial or total function $f: X^K \to X$

ex:- $R^n \to R$ (K: Tuple, X: 1D)

$\max(R_1, R_2, R_3 \ldots R_n) = R_2$.

ex:- $f(x_1, x_2, \ldots x_k)$

$f(x_1, x_2) = 2x_1 + 3x_2$

Definition

A function f over N or Σ is primitive recursive when function f is defined with help of initial functions:-

1) zero function $z(x) = 0$

2) 2.1 Successor function $S(x) = x + 1$
   2.2 Predecessor function $P(x) = x - 1$

3) Projection U_i^n

$$U_i^n (x_1, x_2, x_3, \ldots x_n) = x_i$$

$$U_2^4 (18, 40, 6, 24) = 40.$$

———x———

★. Check which of the following are primitive recursive function.

$f(x,y) = x + y$ Addition.

ans:- This should be made up of zero function, successor / predecessor or projection function. Then, it will be primitive recursive.

Basis:- $f(x,0) = x$.

$$f(x, y+1) = x+y+1$$
$$= f(x,y) + 1$$
$$= S(f(x,y))$$

Yes, primitive recursive ✓

Subtraction $f(x,y) = x-y$.

ans:- If $f(x,y)$ can be converted to zero function, successor or predecessor, or projection function, then its primitive recursive.

Basis $f(x,0) = x$

$$f(x, y+1) = x-y-1$$
$$= f(x,y) - 1$$
$$= P(f(x,y))$$

If no. of tapes increase :-

space complexity ↑

Time complexity ↓.

ex :- 2 tapes can easily solve this

$$L = \langle a^{n!} \mid n > 0 \rangle$$

———x———

Universal Turing Machine

classmate
Date
Page

$TM = \langle Q, \Sigma, \tau, \delta, q_0, \#, F \rangle$

$$\delta(q_i, a_j) = (q_j, a_k, L)$$

ex:-

$$\delta(q_3, a_2) = (q_4, a_1, L)$$

Each data will be represented by 1's & separated by 0.

1 1 1 0 1 1 0 1 1 1 1 0 1 0 1 } this will be input from user.

q_3 a_2 q_4 a_1 L

$$Q = \{ q_1, q_2, q_3 \ldots\ldots, q_f \}$$

q_1: initial state. q_f: final state

$$\tau = \{ a_1, a_2, a_3 \ldots\ldots, a_n \}$$

a_1: Blank

—x—

classmate
Date
Page

:- Undecidability :-

strings accepted by some machine.

Σ^* = Accepts (M) ∪
input string — Rejects (M) ∪
Loops (M)

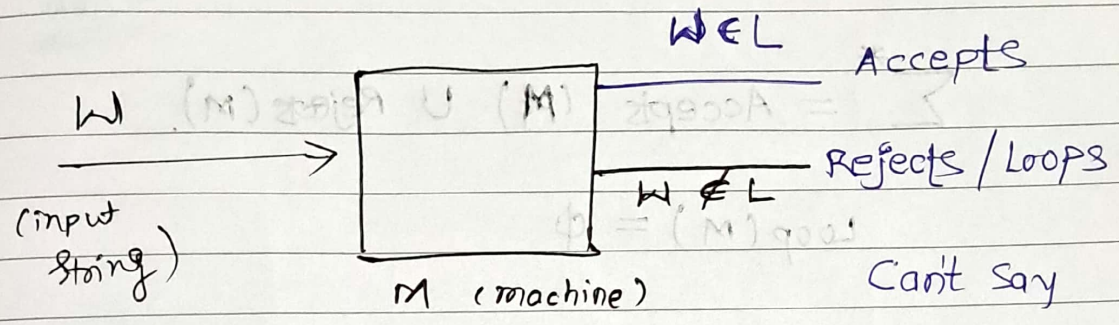
If string given to machine :-

case 1) If machine halts at → final state ⇒ Accept
→ Non-final state ⇒ Reject

case 2) string got exhausted, but machine is still in loop,
Machine is not halted.
||
Loop.

Halting problem of Turing Machine is Undecidable.

Recursive Enumerable Language :-



Q What is compliment of Recursive Enumerable ?

ans:- Recursive Enumerable language is not overall Language

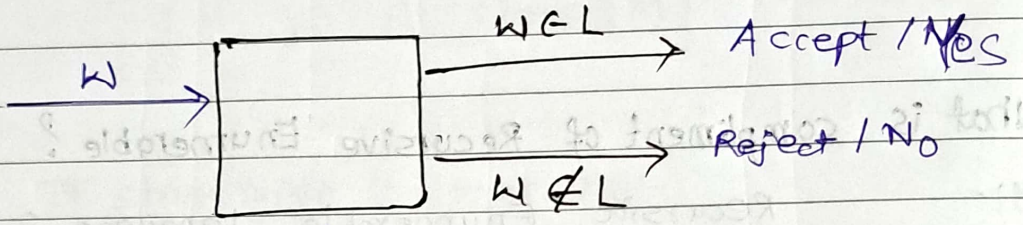
Languages exist beyond it also.

ans :- will be Revealed Later.

Recursive Language :-

Σ^* = Accepts (M) ∪ Rejects (M).

Loop (M) = ϕ



Which one is correct ?

① RE, REC

② RE, REC

③ RE, REC

④ REC, RE

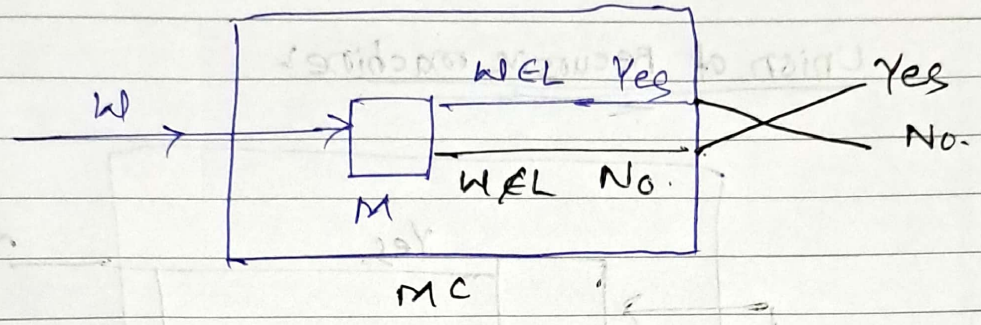
⑤ RE = REC

RE - Recursive Enumerable

REC - Recursive.

If L is Recursive, its compliment will be—?

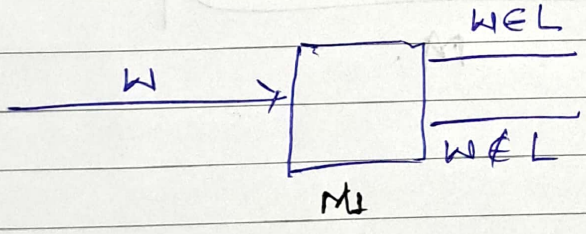
ans:-



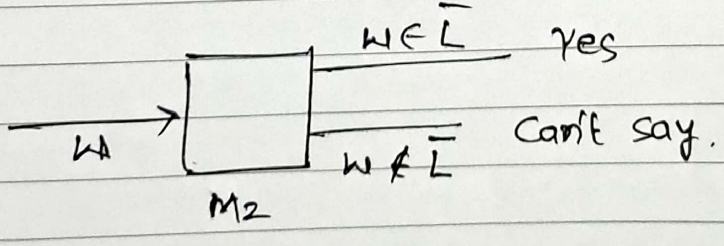
∴ $(\text{Recursive})^{\text{compliment}}$ = Recursive

Let's consider following:-

L is RE



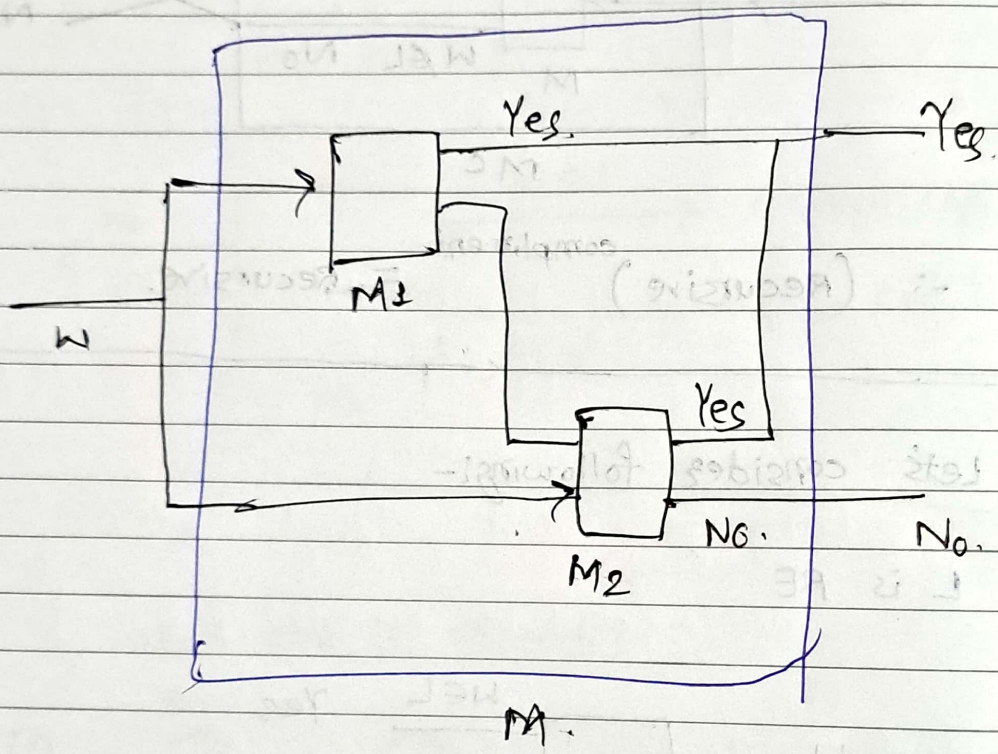
② $\bar{L}$ is RE



See, when W∈L, RE halts
when W∉L, RE halts again.
This is contradictory.

∴ Compliment of Recursive Enumerable is not Recursive Enumerable.

Union of Recursive machines



Post Correspondence Problem :-

2 convergence of n strings.

$$X = \{ x_1, x_2, \ldots\ldots, x_n \}$$
$$Y = \{ y_1, y_2, \ldots\ldots, y_n \}$$

$$x_{i_1}, x_{i_2}, \ldots\ldots, x_{i_k} = y_{i_1}, y_{i_2} \ldots\ldots, y_{i_k}$$

This problem is not Recursive.

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

$$X = \{ 01000, 0, 01 \}$$

$$Y = \{ 01, 000, 1 \}$$

x_1 x_2 x_2 x_3

01 000 000 1

0 1 0 00 0 001

y_1 y_2 y_2 y_3

TOC over